Technical Overview

Sun Java™ Enterprise System

Version 2003Q4

817-5085-10
December 2003

Contents

List of Figures 5

List of Tables 7

Preface 9
Audience 10
Using the Documentation 10
Conventions 11
Resources on the Web 12
How to Report Problems 13
Sun Welcomes Your Comments 13

Chapter 1 Conceptual Foundations 15
Java Enterprise System Component Products 16
Distributed Enterprise Application Architecture 18
Architectural Model 18
Java Implementation of the Architectural Model 20
Client Tier 21
Presentation Tier 21
Business Logic Tier 21
Data Tier 22
Distributed Service Infrastructure 22
The Java Enterprise System 26
System Servers 27
Dependencies Between System Servers 27
Anatomy of a System Server 28
External Interfaces 30
System Server Configuration 31
Sun Cluster 32

Application Life-Cycle Concepts 33
Requirements Analysis 34
Use Cases and Best-Fit Solution 34
System Requirements 34
Deployment 35
Overview of the Deployment Process 35
Deployment Architectures 37
Production Execution 38

Java Enterprise System Key Terms 39

Index 43

List of Figures

Figure 1-1 Architectural Model for Distributed Enterprise Applications 19
Figure 1-2 Java Implementation of Distributed Enterprise Applications 20
Figure 1-3 Support Infrastructure for Distributed Enterprise Applications 22
Figure 1-4 Distributed Service Infrastructure 23
Figure 1-5 The Java Enterprise System 26
Figure 1-6 Java Enterprise System Server Anatomy 29
Figure 1-7 Application Life-Cycle Stages 33
Figure 1-8 Requirements Analysis Specifies a Deployment Scenario 34
Figure 1-9 Deployment Architecture Maps Logical Architecture to Physical Topology 37

List of Tables

Table 1 Java Enterprise System Documentation 10
Table 2 Typeface Conventions 11
Table 3 Symbol Conventions 12
Table 1-1 Java Enterprise System Components 16
Table 1-2 System Server Dependencies 27
Table 1-3 System Server Interface Protocols and API Support 30

Preface

The *Sun Java™ Enterprise System Technical Overview* introduces the conceptual foundations of the Java Enterprise System. This overview describes the Java Enterprise System, its components, and role in supporting distributed enterprise applications. The overview also provides an introduction to system deployment.

This preface contains the following sections:

- Audience
- Using the Documentation
- Conventions
- Resources on the Web
- How to Report Problems
- Sun Welcomes Your Comments

Audience

The *Java Enterprise System Technical Overview* is meant for system analysts, administrators, developers, and other individuals who will be installing, configuring, deploying, or architecting solutions based on the Java Enterprise System.

Individuals reading the *Java Enterprise System Technical Overview* should be familiar with the following technologies:

- The Java language, Java 2 Standard Edition components, and Java 2 Enterprise Edition components
- Network fundamentals, such as network protocols and Internet terminology
- Directory concepts and standards, such as Lightweight Directory Access Protocol (LDAP) and LDAP directory structures
- Security fundamentals relating to authentication, authorization, and encryption, as well as familiarity with security standards and libraries

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 manuals in the Java Enterprise System documentation set. The left column provides the name of each document and the right column describes the document's general contents.

Table 1 Java Enterprise System Documentation

Document	Contents
Java Enterprise System Release Notes `http://docs.sun.com/doc/816-6876`	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 Roadmap http://docs.sun.com/doc/817-4715	Provides task-oriented guidelines for using documentation that relates to the Java Enterprise System environment. Includes pointers to all documentation associated with the component products.
Java Enterprise System Technical Overview http://docs.sun.com/doc/817-5085	Introduces the 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 Installation Guide http://docs.sun.com/doc/816-6874	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. Describes how to perform basic administration tasks, including provisioning users and setting up single sign-on.
Java Enterprise System Glossary http://docs.sun.com/doc/816-6873	Defines terms that are used in the Java Enterprise System documentation.

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 (Monospace bold)	What you type, when contrasted with on-screen computer output.	`% su` `Password:`

Table 2 Typeface Conventions *(Continued)*

Typeface	Meaning	Examples
AaBbCc123	Book titles.	Read Chapter 6 in the *User's Guide*.
(Italic)	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.

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

Table 3 Symbol Conventions

Symbol	Meaning	Notation	Example
`[ ]`	Contain optional command options.	`0[`*n*`]`	`-04, -0`
`{ }`	Contain a set of choices for a required command option.	`d{y|n}`	`-dy`
`|`	Separates command option choices.		
+	Joins simultaneous keystrokes in keyboard shortcuts that are used in a graphical user interface.		Ctrl+A
-	Joins consecutive keystrokes in keyboard shortcuts that are used in a graphical user interface.		Esc-S
>	Indicates menu selection in a graphical user interface.		File > New File > New > Templates

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 *Java Enterprise System Technical Overview* is 817-5085-10.

Chapter 1

Conceptual Foundations

This description of the Sun Java™ Enterprise System is an effort 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 39, which defines the terms and clarifies how they are used in the Java Enterprise System context.

The Java Enterprise System is a software infrastructure that provides the *service* needed to support enterprise-strength applications distributed across a network or Internet environment. These applications are referred to as *distributed enterprise application*. The structure of these distributed enterprise applications and the infrastructure services needed to support them are described in subsequent sections of this chapter.

The Java Enterprise System is also a Sun software release and delivery methodology and a new business and pricing strategy. The focus of this document, however, is on the Java Enterprise System as a software system.

This document explores the conceptual foundations of the Java Enterprise System, including the following topics:

- Java Enterprise System Component Products
- Distributed Enterprise Application Architecture
- Distributed Service Infrastructure
- The Java Enterprise System
- Application Life-Cycle Concepts

Java Enterprise System Component Products

The 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: they are compatible with one another and are synchronized on a common set of shared libraries. The components also share a common installation and upgrade technology.

The components of Java Enterprise System and the infrastructure services they provide are described in the following table.

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 ONE 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 ONE 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 ONE 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 ONE 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 profiles, roles, groups, and policies.
Sun ONE 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 ONE 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 ONE 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 ONE 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 ONE 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.

Distributed Enterprise Application Architecture

The Java Enterprise System provides infrastructure services to support applications that have both distributed and enterprise application aspects:

Distributed The application consists of interacting software components deployed across an intranet or Internet environment that might include geographically remote sites. These *distributed component*s, running on the various *computing node*s in the environment, work together to deliver specific business functions to *end user*s.

Enterprise 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.

In short, a distributed enterprise application has one or more of the following functional characteristics:

- Supports high transaction throughput
- Serves large populations of end users
- Provides high availability
- Performs complex functions
- Supports secure access and communications
- Supports dynamic upgrades of features or functionality

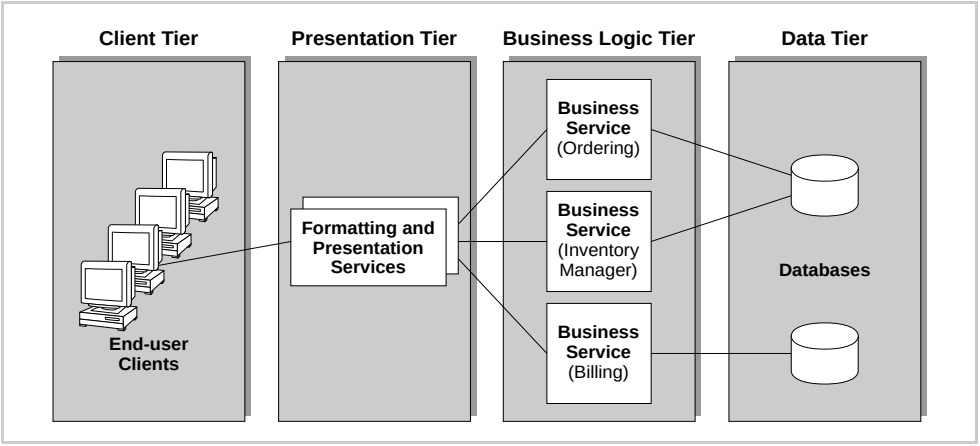
Architectural Model

The standard *architecture* for distributed enterprise applications separates application logic into a number of tiers:

- **Client tier** – the front-end user interface
- **Presentation tier** – the formatting and presentation of data
- **Business logic tier** – the logic that manipulates data
- **Data tier** – the back-end data storage

The architectural model is illustrated in the following figure.

Figure 1-1 Architectural Model for Distributed Enterprise Applications



The services shown in the presentation and business logic tiers of Figure 1-1 are the centerpiece of this model. These services are multi-threaded software processes capable of supporting large numbers of *client*s. These clients can be either end-user clients or other services.

The architectural model illustrated in Figure 1-1, 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:

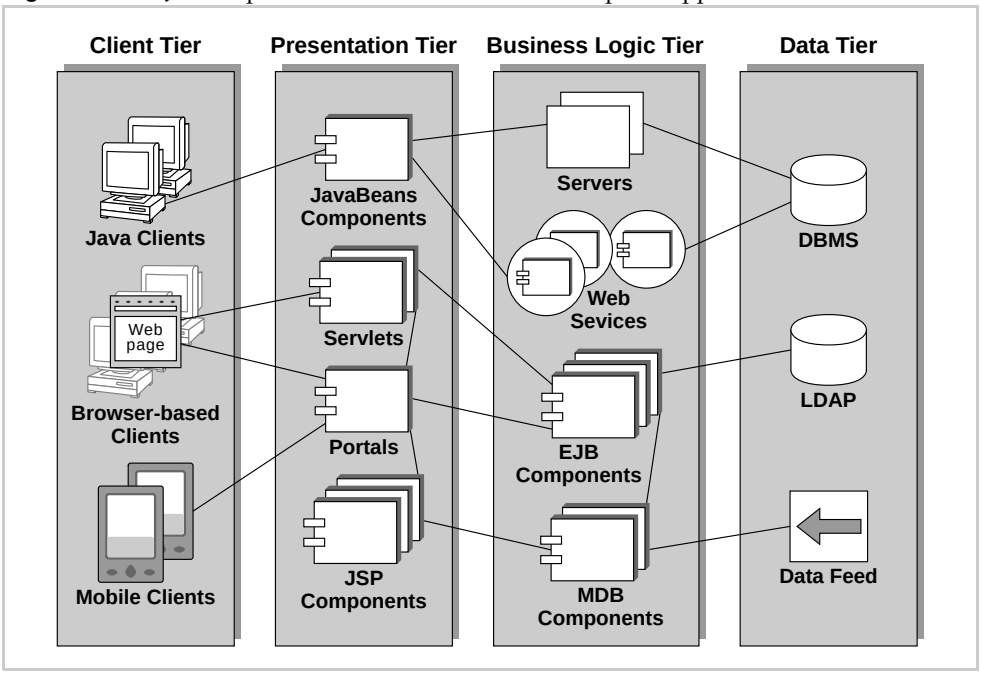
- **Logical independence**. The four tiers in the architectural model represent logical independence: you can modify application logic in one tier (for example, in the business logic tier) independently of the logic in other tiers. You can change your implementation of business logic without having to change or upgrade logic in the presentation tier or client tier. This independence means 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 idea is to run distributed application components on the computing nodes best suited to their individual computing requirements and best suited to maximizing network bandwidth.

The mapping of application components to a hardware environment 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.

Java Implementation of the Architectural Model

The architectural model of Figure 1-1 can be implemented using the various J2EE platform distributed component types and service implementations shown in the following figure. (The links shown between components are suggestive, and are not meant to reflect all possibilities.)

Figure 1-2 Java Implementation of Distributed Enterprise Applications



This section provides a brief description of the four tiers of the architectural model as implemented in the Java programming language.

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 service*s in the business logic tier.

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

Business Logic Tier

The business logic 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 higher-level business services, such as an inventory service or tax calculation service. Individual components and higher-level service assemblies can be encapsulated as loosely-coupled *web service*s that conform to Simple Object Access Protocol (SOAP) interface standards. Business services can also be built as standalone *server*s, 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 load balancing and failover, and you can map these service providers to computing nodes in a way that optimizes network bandwidth and computing resources.

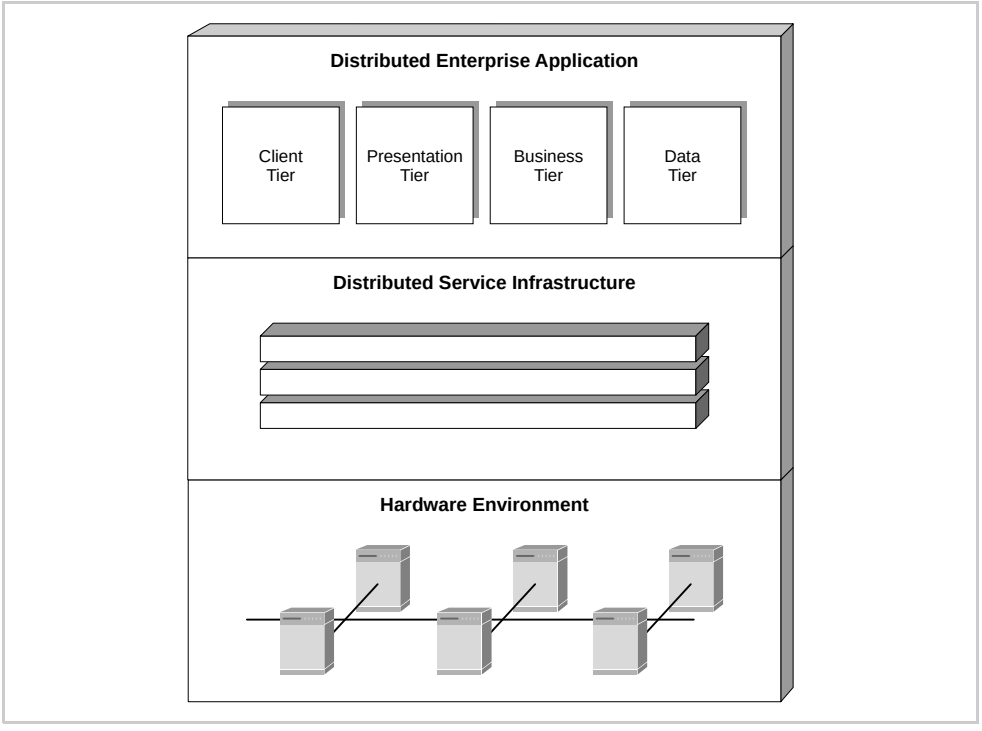
Data Tier

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 Data Access Protocol (LDAP) data store. The data can also include real-time data feeds from external sources or data accessible from legacy computing systems.

Distributed Service Infrastructure

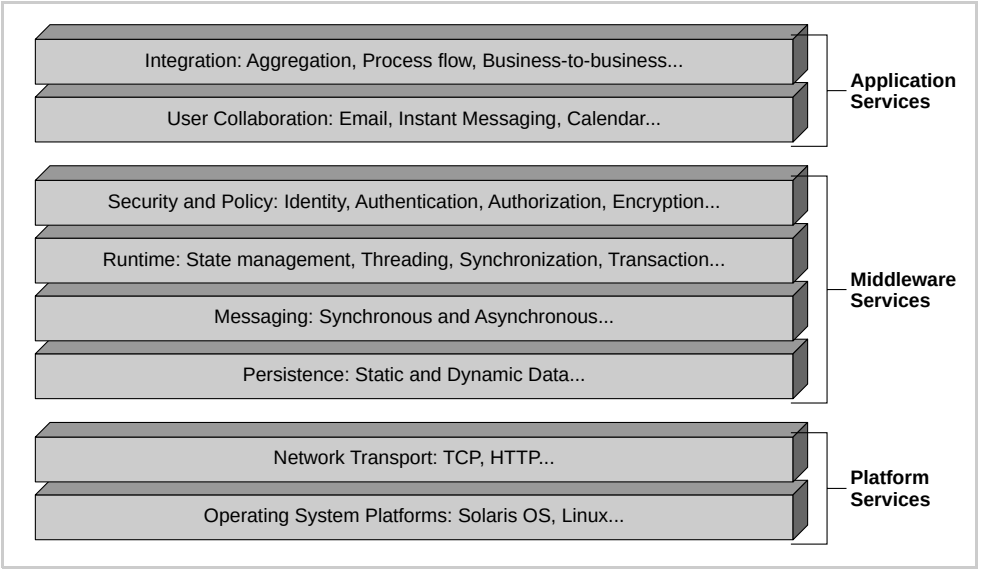
Applications based on the architectural model described in "Distributed Enterprise Application Architecture" on page 18 depend upon many software services at many levels. The distributed components of such an application require an underlying distributed service infrastructure to communicate with each other, to coordinate their work, to implement secure access, and so forth. This service infrastructure, in turn, is supported by a hardware environment of computing nodes and network links. The general scheme is illustrated in the following figure.

Figure 1-3 Support Infrastructure for Distributed Enterprise Applications



The distributed service infrastructure shown in Figure 1-3 provides the software platform upon which a distributed enterprise application runs. This platform consists of a set of distributed services as shown in the following conceptual illustration.

Figure 1-4 Distributed Service Infrastructure



The levels in Figure 1-4 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 interacting with intermediate services. 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 1-4 are not strictly prescribed. Other service levels, such as a monitoring or management service, might also be included.

In general, the services shown within the distributed service infrastructure fall into three broad groupings: low-level platform services, high-level application services, and a group of middleware services, so named for their location in the infrastructure.

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 1-4:

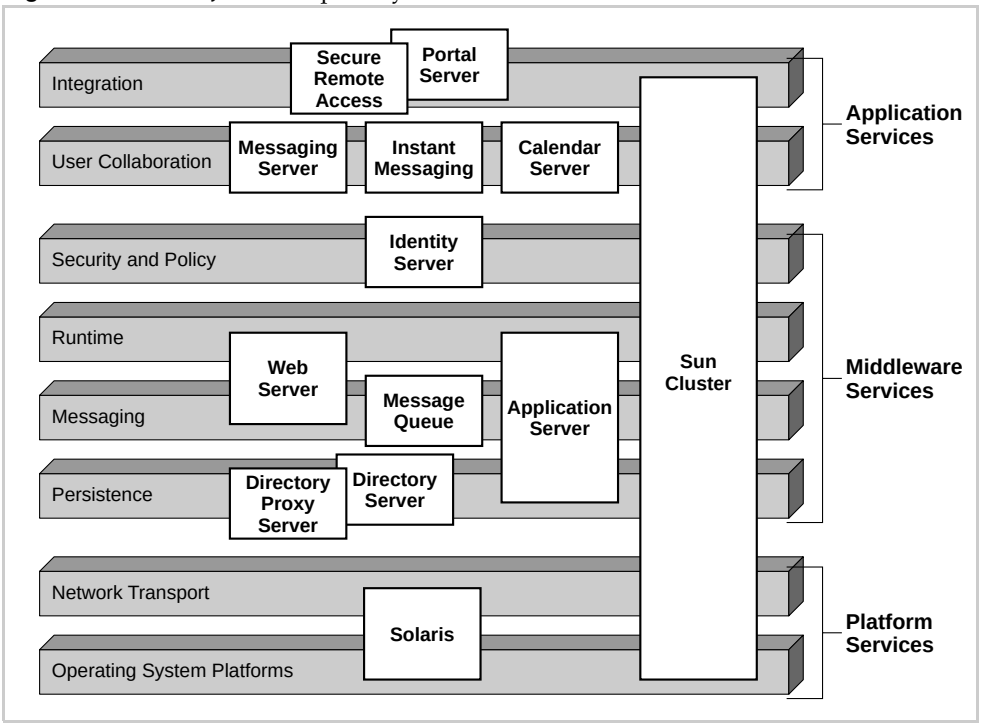
- **Operating system platform.** Provides the basic support for any process running on a computing node. The operating system (such as Solaris™ Operating System, Linux, or Windows) manages physical devices as well as memory, threads, and other resources necessary to support the Java Virtual Machine (JVM™ machine).
- **Network transport.** Provides the basic networking support for communication between distributed application components running on different computing nodes. this layer includes 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 access to and storage of both static data (such as user, directory, or configuration information) and dynamic application data (information that is frequently being updated).
- **Messaging.** Provides the 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 the support required by any distributed component model, such as the CORBA or J2EE 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 MDB containers in an application server or web server.

- **Security and policy.** Provides the support for secure transmission of data and secure access to application resources. These services include support for policies that govern group access to distributed resources, as well as single sign-on capabilities. Single sign-on allows a user's identity to be known by all application components (J2EE components, business services, and web services) in a distributed 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 email 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.

The Java Enterprise System

The Java Enterprise System is an implementation of the distributed service infrastructure (Figure 1-4). The positioning of the different Java Enterprise System components within the distributed service infrastructure is shown in the following illustration:

Figure 1-5 The Java Enterprise System



The Java Enterprise System includes two main component types:

- Discrete software servers (*system servers*) that provide services at various levels within the distributed service infrastructure. These service providers are based on the client-server model (a multi-threaded server process that supports a large number of clients). Some of the system components in this category include their own clients.

- Sun Cluster, which provides high availability and scalability for the system as a whole. Cluster software is based on a peer-to-peer service model (the software is both server and client to other peers). As shown in Figure 1-5, Cluster services span all the infrastructure services provided by the Java Enterprise System.

The following sections describe Java Enterprise System servers and Sun Cluster in greater detail.

System Servers

Java Enterprise System servers implement most of the layers within the distributed service infrastructure, as shown in Figure 1-5. Each system server provides a particular service or set of services in support of distributed enterprise applications. These *system service*s are what uniquely characterize each server (see Table 1-1 on page 16 for a brief description of the services provided by each system server).

Dependencies Between System Servers

In general, each system server depends on system servers below it in the infrastructure and supports system servers above it. Some servers, however, implement lower-level services internally and are therefore not dependent on other lower-level system servers.

Table 1-2 shows the specific dependencies between the different Java Enterprise System servers, listed from top to bottom, as shown in Figure 1-5.

Table 1-2 System Server Dependencies

System Server	Depends On
Portal Server	Web Server or Application Server Directory Server Identity Server Messaging Server (optional: channel) Calendar Server (optional: channel) Instant Messaging (optional: channel)
Portal Server, Secure Remote Access	Portal Server Identity Server (optional: SDK)
Messaging Server	Directory Server Web Server (optional) Identity Server (optional: single sign-on)

Table 1-2 System Server Dependencies *(Continued)*

System Server	Depends On
Instant Messaging	Directory Server Identity Server (optional: SDK) Messaging Server (optional)
Calendar Server	Directory Server Identity Server (optional: single sign-on) Messaging Server (optional: email notices)
Identity Server	Web Server or Application Server Directory Server
Web Server	Directory Server (optional: access control)
Application Server	Message Queue Directory Server (optional: access control)
Message Queue	Directory Server (optional: directory services)
Directory Server	
Directory Proxy Server	Directory Server

Anatomy of a System Server

Despite the different system 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

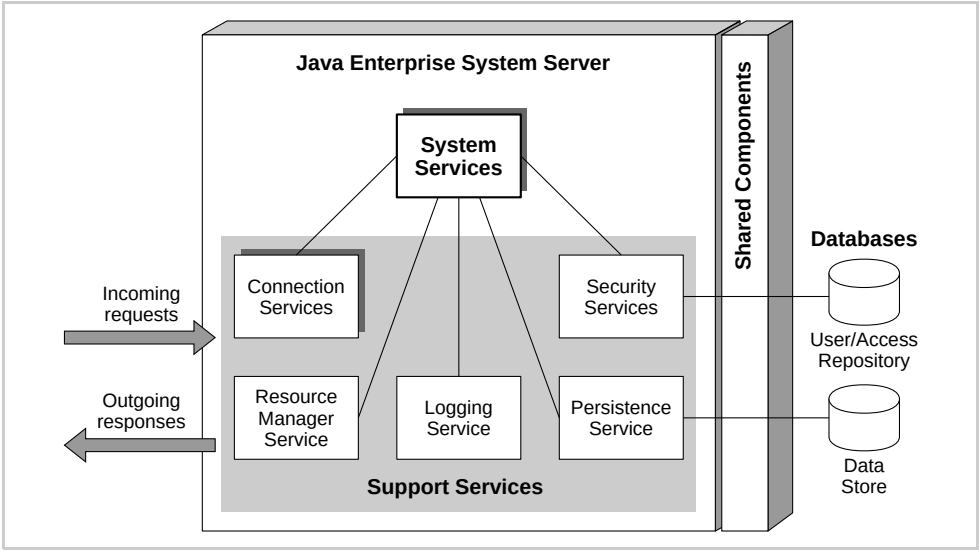
These subcomponents are shown schematically in Figure 1-6 and described briefly in the sections that follow.

System Services Subcomponents

Table 1-1 on page 16 summarizes the main system services provided by each of the Java Enterprise System servers.

Each system server implements the services it provides differently. 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.

Figure 1-6 Java Enterprise System Server Anatomy



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 1-6 typically correspond to various services in the distributed software service infrastructure. 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 minimize the use of internal support services in favor of external system services, extracting common internal services and implementing them as new system services, such as a new logger service or new 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.

External Interfaces

Each Java Enterprise System server interacts with clients through an interface. The clients might be other servers, business service implementations, simple distributed components, or they might be end-user clients such as the client programs delivered with Calendar Server and with Instant Messaging.

A system server's external interfaces include two kinds:

- **A messaging protocol** (normally conforming to an industry standard) by which clients access the server (normally conforms to an industry standard)
- **An application programming interface** (API) by which you can customize the server). Customization can be an important aspect of deploying a business solution.

The following table summarizes the protocols by which clients interact with each system server and indicates which system servers provide a customization API. The servers are listed in alphabetical order.

Table 1-3 System Server Interface Protocols and API Support

System Server	Protocols	Customization API
Application Server	RMI (J2EE), HTTP/HTTPS	No
Calendar Server	HTTP/HTTPS, Web Calendar Access Protocol (WCAP)	Yes
Directory Proxy Server	LDAP, Directory Service Markup Language (DSML), version 2	Yes
Directory Server	LDAP	No
Identity Server	HTTP/HTTPS	Yes
Instant Messaging	HTTP/HTTPS, TCP	Yes
Message Queue	JMS, HTTP/HTTPS, TCP	No

Table 1-3 System Server Interface Protocols and API Support *(Continued)*

System Server	Protocols	Customization API
Messaging Server	SMTP, IMAP, POP3, HTTP, LMTP	Yes
Portal Server	HTTP/HTTPS	Yes
Portal Server, Secure Remote Access	HTTP/HTTPS	Yes
Web Server	HTTP/HTTPS	Yes

Each Java Enterprise System server also has an administration interface for configuring and managing server functions. This administration interface provides access to the server for server-specific administration clients. These administration clients might be GUI based, command line, or both.

System Server Configuration

For each Java Enterprise System server, the internal components can be tuned to maximize server performance. In fact, each *instance (server instance)* of a system server is specified by properties that affect the behavior of its various system service and support service subcomponents.

In general, there can be several instances of a system server on a single computing node, single instances of a server on one or more nodes, or a combination of the two, depending on the needs of a deployed application. You can configure each of these instances separately, though multiple instances on a single node might share a common set of configuration properties.

Most system servers have an administration console (a graphical administration interface) that you can use to start, stop and configure each instance of a server. Often this is a browser-based interface that allows you to administer remote servers.

NOTE The Java Enterprise System includes, as a separately installable component, the Sun ONE Administration Server (and Server Console) used to configure and manage server instances, as well as perform user and group administration, for Directory Server, Directory Proxy Server, and Messaging Server.

Sun Cluster

Sun Cluster software provides high availability and scalability services for the Java Enterprise System as well as for applications that run on top of the 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.

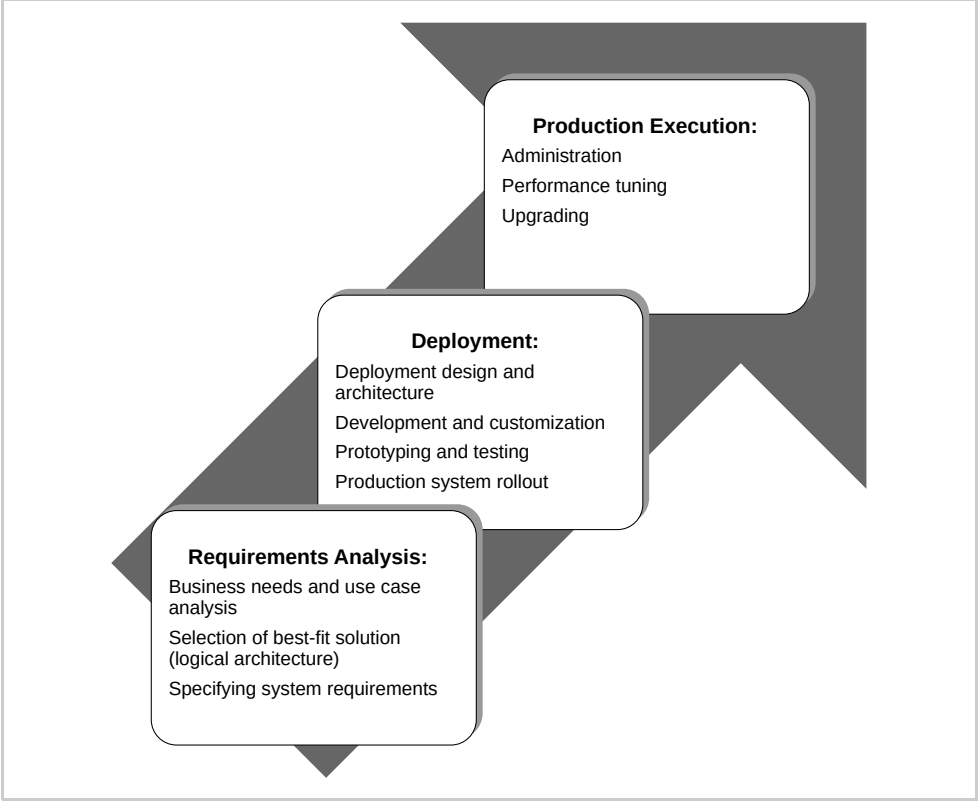
Unlike the services provided by Java Enterprise System servers, Cluster services are distributed peer-to-peer services. Cluster software therefore needs to be installed on every computing node in a cluster.

Application Life-Cycle Concepts

The following figure shows the three main life-cycle stages of a distributed enterprise application:

- Requirements Analysis
- Deployment
- Production Execution

Figure 1-7 Application Life-Cycle Stages

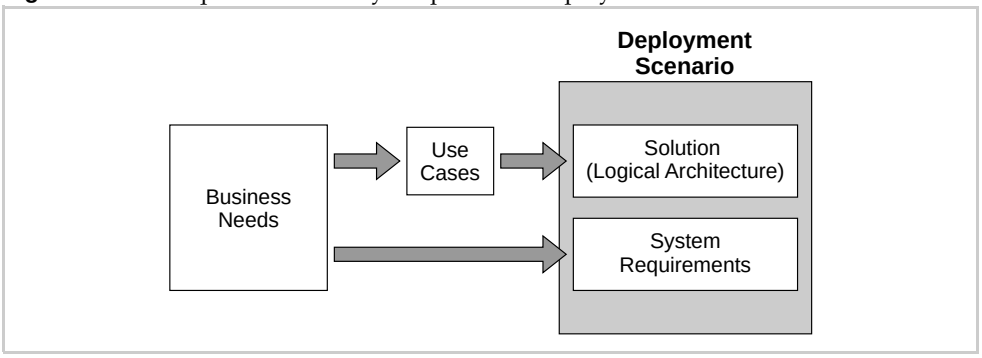


The following sections describe the concepts and terminology relevant to each of these life-cycle stages.

Requirements Analysis

In the *requirements analysis* stage of the life cycle, you translate information about business needs into *use case*s and other technical requirements, which you use to evaluate possible solutions. You then specify a best-fit solution and the system requirements it must meet in a *deployment scenario*, as shown in the following figure.

Figure 1-8 Requirements Analysis Specifies a Deployment Scenario



Use Cases and Best-Fit Solution

Use cases are the specific end-user tasks or set of tasks that a distributed enterprise application must perform to meet business needs. Use cases are therefore the basis for arriving at a software solution. The solution is a high-level *logical architecture* that specifies the application logic required to implement the use cases as well as the infrastructure services needed to support that logic.

Use cases are not only instrumental in choosing a solution, but also, in the *deployment* stage, in testing that solution to verify that it meets the system requirements identified by your requirements analysis.

System Requirements

A requirements analysis, in addition to specifying a set of use cases that reflect business functional needs, also specifies other technical and performance requirements that a solution must meet. Hence, a deployment scenario must not only specify a software solution, but also a set of system requirements that the solution must meet. These system requirements include the following:

- The operating system platforms that must be supported
- Load condition requirements (the number of concurrent end users that the application must support, and the level of performance required)

- Security requirements
- Application availability requirements
- Seviceability requirements (system upgrade requirements)
- Latent capacity requirements

The system requirements you specify in the deployment scenario reflect the usage patterns and growth you expect for a deployed application. A clearly specified deployment scenario allows you to create a deployment design (in the deployment stage) that meets your business needs.

Deployment

In general, the deployment stage of the life style is a complex process that involves a number of iterative phases and depends not only on an solution's logical architecture, but also on designing a *deployment architecture* that meets the application's performance, availability, security, serviceability, and other requirements.

The deployment process generally involves software components in all the tiers (shown in Figure 1-1 on page 19) and in all the service layers (shown in Figure 1-4 on page 23) required to support an application. Thus, in addition to any specific distributed application components (J2EE components, web services, or other servers) you might deploy, you must also deploy any Java Enterprise System components (*system components*) needed to support the application.

This section covers a number of topics related to the deployment stage of the life-cycle process:

- Overview of the deployment process
- Deployment architectures
- Reference deployment architectures

Overview of the Deployment Process

The deployment process consists of the following general phases:

- Deployment design
- Development
- Prototype testing
- Production rollout

The deployment phases are not rigid: the deployment process is iterative in nature. Nevertheless, the following subsections discuss each of the deployment phases independently.

Deployment Design

In the deployment design phase, you construct a deployment architecture based on the deployment scenario specified in the requirements analysis phase. The objective is to map the logical building blocks of an application (the logical architecture) to a physical environment (a physical topology) in a way that meets the system requirements specified in the deployment scenario.

One aspect of this design is sizing the physical environment to meet load, availability, and performance requirements. The deployment architecture takes into account details of the physical topology, such as the capabilities of different computing nodes and network bandwidth, in assigning system servers and application components to the computing nodes in the environment. For more information, see "Deployment Architectures" on page 37.

Development

The logical architecture specified in the requirements analysis stage of the life cycle determines the scope of the *development* work needed to implement a solution.

For some 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 environment or Web Server environment. In other cases, it might be sufficient to customize existing system servers, such as the Portal Server, to achieve the functionality required.

In the case of solutions requiring extensive development, Java Enterprise System 2003Q4 does not provide the tools you need to program distributed components or business services. These tools are provided in Sun Java Studio, which simplifies the programming and testing of applications supported by the Java Enterprise System infrastructure.

Prototyping

In the prototyping phase, you prototype your deployment design by implementing the deployment architecture in a test environment. You use new application logic and server customizations from the development effort, as described above (see "Development"), to perform proof-of-concept deployment testing. This phase involves installing, configuring, and starting up distributed applications and any required infrastructure services in your test environment.

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

Production Rollout

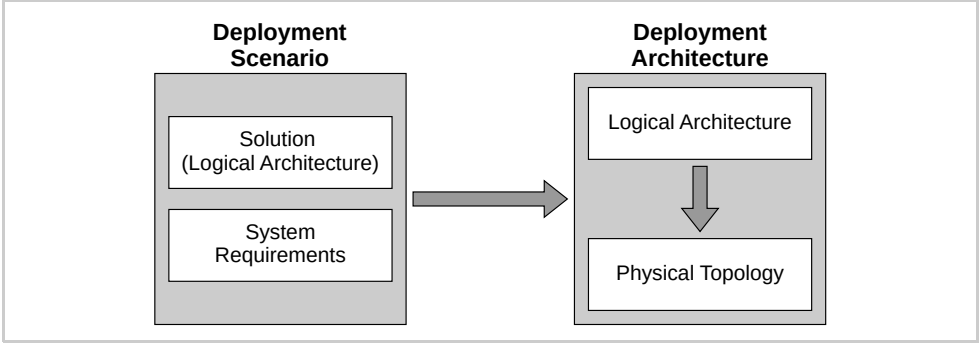
In the production rollout phase, you implement your deployment architecture in a production environment. This phase involves installing, configuring, and starting up distributed applications and any required infrastructure services in a production environment. You normally start with a limited deployment and move to organization-wide implementation. In this process, you perform trial runs, in which you apply increasing loads and stress test the system.

As part of the rollout phase you might need to perform administrative tasks such as provisioning users, implementing single sign-on, and tuning the system to meet performance objectives.

Deployment Architectures

A deployment architecture maps a logical architecture to a computing environment. The system requirements specified in the deployment scenario (as well as restrictions on the total cost of ownership or total cost of availability) affect the design of a deployment architecture, as illustrated in the following figure.

Figure 1-9 Deployment Architecture Maps Logical Architecture to Physical Topology



The larger the number of system servers in the deployment scenario, and the more demanding your system requirements (the higher the end-user load, the more stringent the security and firewall requirements, the higher the required availability, and so forth), the more demanding your design is on high-power

computing nodes and high network bandwidth. Where hardware is limited, or prohibitively expensive, you will have to make trade-offs between different system requirements or increase the sophistication of your design.

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

Production Execution

In the *production execution* 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 2003Q4 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.

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 services*, and *servers* running on the various computing nodes.

deployment A stage of the application life-cycle process in which a logical architecture is mapped to a physical topology, 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 design that depicts the mapping of a *logical architecture* to a physical topology. The physical topology includes the *computing nodes* in an intranet or Internet environment, and the network links between them.

deployment scenario A software solution (*logical architecture*) and the system requirements that the solution must satisfy to meet business needs. The system requirements include: load (number of concurrent users), availability, security, serviceability (dynamic upgrade requirements), and latent capacity (growth potential).

development A phase in the deployment process, in which a logical architecture is implemented (either through programming or customization) and tested.

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.

production execution 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.

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 solution (*logical architecture*) and a set of system requirements the solution must meet.

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).

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.

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

A

Application Server 16, 26
application services 23
applications
  architecture of 18
  distributed, *See* distributed enterprise applications
  enterprise, *See* distributed enterprise applications
architecture
  deployment 35, 37
  distributed enterprise application 18
  logical 34
availability
  requirements 35
  services 32

B

business services 21

C

Calendar Server 16, 26
capacity requirements 35
clients 19
clusters, *See* Sun Cluster
component products
  and infrastructure services 26
  as system components 16
  dependencies 27
  descriptions of 16
components
  distributed 18
  EJB 21
  J2EE 20, 21
  JSP 21
  Servlet 21
  shared 30
  system, *See* system components
computing nodes 18
configuration 31

D

dependencies 27
deployment
  architecture 35, 37
  design phase of 36
  development phase of 36
  life-cycle stage 34, 35
  process 35
  production rollout phase of 37
  prototyping phase of 36
  scenarios, *See* deployment scenarios

deployment scenarios
  best-fit solution 34
  logical architecture 34
  system requirements 34
development 36
Directory Proxy Server 16, 26, 31
Directory Server 16, 26, 31
distributed
  applications, *See* distributed enterprise applications
  components 18
  services, *See* distributed services
distributed enterprise applications
  about 15
  architecture 18
  characteristics 18
  infrastructure for 23
  Java implementation of 20
distributed services
  application level 23
  integration 25
  messaging 24
  middleware 23
  network transport 24
  operating system platform 24
  persistence 24
  platform 23
  runtime 24
  security and policy 25
  user collaboration 25
documentation set 10

E

EJB components 21
end users 18, 34

I

Identity Server 17, 26
infrastructure
  for distributed enterprise applications 23
  services, *See* distributed services
instances 31
Instant Messaging 17, 26
integration services 25
interfaces 30

J

J2EE
  components 21
  container service 16
  distributed component model 21
  platform 17
J2ME platform 21
J2SE Platform 30
Java Servlet components 21
JMS (Java Message Service) 17
JSP components 21
JSS 30

L

LDAP 22
life-cycle stages
  deployment 34
  production execution 38
  requirements analysis 34
logical architecture 34

M

Message Queue 17, 26
Messaging Server 17, 26, 31
messaging services 24
middleware services 23

N

network transport services 24
nodes, *See* computing nodes
NSPR 30
NSS 30

O

operating system services 24
operating systems 34

P

performance requirements 34
persistence services 24
platform services 23
Portal Server 17, 26
Portal Server, Secure Remote Access 17, 26
production execution stage 38
production rollout 37
protocols 30
prototyping 36

R

requirements analysis life-cycle stage 34
requirements, *See* system requirements
runtime services 24

S

scalability services 32
security and policy services 25
security requirements 35
servers
  instances of 31
  standalone 21
  system, *See* system servers
serviceability requirements 35
services 15
  business 21
  distributed, *See* distributed services
  high availability 32
  scalability 32
  support 29
  system, *See* system services
  web 21
shared components 30
Solaris 26
Sun Cluster 16, 26, 27, 32
Sun ONE Administration Server 31
Sun ONE products
  Application Server, *See* Application Server
  Calendar Server, *See* Calendar Server
  Directory Proxy Server, *See* Directory Proxy Server
  Directory Server, *See* Directory Server
  Identity Server, *See* Identity Server
  Instant Messaging *See* Instant Messaging
  Message Queue, *See* Message Queue
  Messaging Server, *See* Messaging Server
  Portal Server, Secure Remote Access, *See* Portal Server, Secure Remote Access
  Portal Server, *See* Portal Server
  Web Server, *See* Web Server
support services 29
system
  components, *See* system components
  requirements, *See* system requirements
  servers, *See* system servers
  services, *See* system services
system components
  component products 16
  shared components 30
  Sun Cluster 27
  system servers 26

system requirements
availability 35
latent capacity 35
performance 34
security 35
serviceability 35
system servers
about 26
anatomy of 28
configuration of 31
customization APIs 30
dependencies 27
protocols 30
subcomponents of 28
system services 27

T

tiers
application architecture, and 18
business logic 21
client 21
data 22
presentation 21

U

use cases 34
user collaboration services 25
users, *See* end users

W

Web Server 17, 26
web services 21