Sun Java™ System

Communications Services 6 Enterprise Deployment Planning Guide

2004Q2

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

Part No: 817-6096-10

Contents

List of Figures

List of Tables

Preface

The *Sun Java System Communications Services 6 2004Q2 Enterprise Deployment Planning Guide* contains the information you need to deploy Sun Java™ System Communications Services 6 2004Q2 into an enterprise environment. This guide helps you through the process of understanding Communications Services, evaluating and analyzing your site, and designing the kind of deployment architecture that meets your enterprise's needs.

This preface contains the following sections:

Who Should Read This Guide

This guide is for individuals who are responsible for assessing and deploying Communications Services at your site, including:

- Evaluators
- Architects
- System administrators

This guide assumes you are familiar with the following:

- How to install enterprise-level software products

- IMAP, POP, HTTP, SMTP, WCAP, and LDAP protocols
- Solaris system administration and networking

Conventions

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

Table 1 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, as contrasted with on-screen computer output.	`% su` `Password:`
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 *ms_svr_base*`/bin` directory.

The following table describes placeholder conventions used in this guide.

Table 2 Placeholder Conventions

Item	Meaning	Examples
product_base	Placeholder for the directory where the product is installed.	The *ms_svr_base*`/bin` directory might be `/opt/SUNWmgsr`.

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

In addition to this guide, you will want to refer to Sun Java™ System Calendar Server 6, Sun Java™ System Messaging Server 6, and Sun Java™ System Instant Messaging Server 6 documentation. Use the following URLs to see this documentation:

`http://docs.sun.com/db/coll/CalendarServer_04q2`

`http://docs.sun.com/db/coll/InstantMessaging_04q2`

`http://docs.sun.com/db/coll/MessagingServer_04q2`

Third-party URLs are included in this document to provide additional, related information.

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

How to Report Problems

If you have problems with Communications Services, 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 is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document. For example, the part number of this *Communications Services Enterprise Deployment Planning Guide* is 817-6096-10.

Chapter 1

Understanding Communications Services

This chapter provides an overview of Sun Java™ System Communications Services 6 2004Q2, the business reasoning behind deploying Communications Services, and the deployment process itself.

This chapter contains the following sections:

Communications Services Overview

Sun Java System Communications Services 6 2004Q2 are a secure, cost-effective communications and collaborations offering. Communications Services address enterprise customer concerns about costs, capabilities, and the security of the traditional enterprise communications infrastructure by offering a secure, scalable, lower total cost of ownership alternative to other enterprise messaging solutions.

Communications Services provide the email, calendar, and instant messaging solutions necessary to meet enterprise communications and collaboration needs. The products and services that form Communications Services provide a compelling response to common business requirements. All enterprises need communications, and many are required to provide these services across large, diverse, and geographically distributed communities of users. Traditional communications solutions are costly, and not sufficient to meet today's enterprise requirements for scalability and security. Communications Services enable enterprises to deploy solutions at a total cost of ownership they can afford.

In addition, Communications Services provide differentiated services and full-featured collaboration functionality that are required by a diverse audience. Finally, a Communications Services deployment meets an enterprise's increased security needs when extending communications outside of a corporate firewall and to mobile users through multiple devices.

The Communications Services core solution consists of:

- Sun Java™ System Messaging Server 6 (formerly Sun™ ONE Messaging Server)
- Sun Java™ System Calendar Server 6 (formerly Sun™ ONE Calendar Server)
- Sun Java™ System Instant Messaging 6 (formerly Sun™ ONE Instant Messaging)

Additional products that enhance the Communications Services solution include:

- Sun ONE™ Synchronization 1.1
- Sun Java™ System Connector for Microsoft Outlook 6 2004Q2
- Sun Java™ System Communications Express 6

Taken as a whole, Communications Services provide a standards-based, integrated communications and collaboration suite of products for enterprise deployments of many thousands of users. Communications Services deliver a robust and flexible platform meeting the diverse communications needs of all types of organizations. Communications Services are an optimal solution to connect remote offices, distributed workgroups, and global corporate locations.

NOTE Currently, this guide addresses enterprise deployments consisting of up to 5,000 enterprise users.

About Messaging Server

Sun Java System Messaging Server 6 is a high-performance, highly secure messaging platform. Scaling from thousands to millions of users, Messaging Server is suitable for enterprises interested in consolidating email servers and reducing total cost of ownership of communications infrastructure. Messaging Server provides extensive security features that help ensure the integrity of communications through user authentication, session encryption, and the appropriate content filtering to help prevent spam and viruses.

With Messaging Server, enterprises can provide secure, reliable messaging services for entire communities of employees, partners, and customers.

See the *Sun Java System Messaging Server Deployment Planning Guide* at the following location for more information on Messaging Server concepts and other deployment aspects not addressed by this guide:

http://docs.sun.com/doc/817-6440

About Calendar Server

Sun Java System Calendar Server 6 facilitates team collaboration by enabling users to manage and coordinate appointments, events, tasks, and resources. With its intuitive, Web-based interface, Calendar Server enables end users to access their personal, public, or group calendars anytime, anywhere, from any Web-enabled device. Enterprise deployments use Calendar Server, along with the Messaging Server and Instant Messaging, to offer users a comprehensive communications and collaborative environment.

See the *Sun Java System Calendar Server Administration Guide* at the following location for more information on Calendar Server concepts:

http://docs.sun.com/doc/817-5697

About Instant Messaging

Sun Java System Instant Messaging 6 enables secure, real-time communication and collaboration. Instant Messaging combines presence awareness with instant messaging capabilities such as chat, conferences, alerts, news, polls, and file transfers to create a rich collaborative environment. These features enable one-to-one as well as group collaboration through either short-lived communications or persistent venues such as conference rooms or news channels. Instant Messaging ensures the integrity of communications through its multiple authentication mechanisms and secure SSL connections. Integration with Sun Java™ System Portal Server 6 and Sun Java™ System Identity Server 6 brings additional security features, services-based provisioning access policy, user management, and secure remote access.

See the *Sun Java System Instant Messaging Deployment Planning Guide* at the following location for more information on Instant Messaging concepts and deployment considerations:

http://docs.sun.com/doc/817-5937

NOTE Though Instant Messaging is a key part of the Communications Services offering, this guide currently does not discuss how to deploy Instant Messaging. Refer to the *Sun Java System Instant Messaging Deployment Planning Guide* for such information.

About Synchronization

Sun ONE Synchronization 1.1 is a software product that runs on a Windows personal computer and enables users to synchronize Calendar Server events and tasks with mobile devices and personal information managers (PIMs) such as Microsoft Outlook.

See the Sun ONE Synchronization documentation at the following location for more information:

http://docs.sun.com/db/coll/S1_Sync_11

About Connector for Microsoft Outlook

Sun Java System Connector 6 for Microsoft Outlook enables Outlook to be used as a desktop client with Sun Java Enterprise System.

Connector for Microsoft Outlook is an Outlook plug-in that you install on end-users' desktops. Connector for Microsoft Outlook queries Messaging Server for folder hierarchies and email messages. Connector for Microsoft Outlook then converts the information into Messaging API (MAPI) properties that Outlook can display. Similarly, Connector for Microsoft Outlook uses WCAP to query Calendar Server for events and tasks, which are then converted into MAPI properties. With this model, 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.

See the Connector for Microsoft Outlook documentation at the following location for more information:

http://docs.sun.com/db/coll/MessagingServer_04q2

About Communications Express

Sun Java System Communications Express 6 provides an integrated web-based communication and collaboration client that meets enterprise users' needs.

As a web-based client, the Communications Express depends on a web server for access and a browser for presentation. The Communications Express offering consists of three client modules: Calendar, Address Book, and Mail. The Calendar, Address Book, and Mail client modules are deployed as a single application on any web container.

You can customize most of the features in Communications Express. See the Communications Express documentation at the following location for more information.

`http://docs.sun.com/db/coll/MessagingServer_04q2`

Communications Services Component Product Dependencies

Communications Services also have dependencies on other Sun Java System component products that provide infrastructure services. These component products include Sun Java™ System Directory Server and Sun Java™ System Identity Server. Additionally, Communication Services depend on a web server to serve HTML content and provide HTML connections. You can use Sun Java™ System Web Server (also known as Sun™ ONE Web Server) to fulfill this need.

Communications Services also depend on the existence of DNS. You need to have a functioning DNS server before you can install the Communications Services products.

See Chapter 3, "Understanding Product Requirements and Considerations" for more information on product dependencies.

How Communications Services Satisfy Enterprise Business Needs

Enterprises want to deploy services that simultaneously reduce cost and complexity while providing a robust set of features. The architecture of services for enterprise communications must add requirements for security and scalability that enable users to have more than just a single means of accessing information critical to their daily work. Communications Services meet these needs through providing scalable messaging, calendaring, and instant messaging at a total cost of ownership enterprises can afford.

Communications Services enable you to develop an architecture that incorporates ease of deployment and maintenance with a complete set of features and functionality. Most important, a Communications Services architecture builds security into each service element. These elements include the network infrastructure, operating environment, and the Communications Service component products themselves.

How Messaging Server Satisfies Business Needs

Messaging Server promotes superior reliability and productivity as well as reduced administrative and operational costs. Messaging Server uses committed transactions, which means that messages are not acknowledged as received until they are committed to a message store on disk. This reliability feature protects mail messages from loss and corruption. Additionally, the Message Store is built around a custom-designed database that employs a write-once data store and a two-level index to achieve excellent performance and data integrity.

How Calendar Server Satisfies Business Needs

Calendar Server provides one of the industry's most open, interoperable, and high-performance time and resource management solutions. Calendar Server provides the features you need at a lower total cost of ownership than alternative solutions. Through its flexible and extensible architecture, Calendar Server scales both vertically (by increasing the number of CPUs per system) and horizontally (by adding more servers to the network).

How Instant Messaging Satisfies Business Needs

Instant Messaging software is closely integrated with Java Enterprise System, helping you to shorten the project lifecycle and deploy new services affordably. In addition, Instant Messaging is fully integrated with Portal Server, Identity Server, Messaging Server, and Calendar Server. This integration provides enterprise users with a full-featured, secure, scalable communications and collaboration services platform from a single vendor. The well-documented Java APIs included in Instant Messaging provide open standards for ease of integration, as well as multiple platform support, platform extensibility, and customization of real-time communications and collaboration features. These features can thus be embedded in existing enterprise applications or become the basis of new applications.

Summary of Communications Services Benefits

The Communications Services components have been traditionally deployed in large-scale, carrier-class deployments. As discussed in this guide, the same dependability required for the large-scale deployments can be used in the enterprise.

The following table summarizes the benefits to the enterprise provided by Communications Services.

Table 1-1 How Communications Services Benefit the Enterprise

Key Feature	Benefit to the Enterprise
High performance and scalability	Enables efficient communications and improves quality of service for both enterprises and ISPs.
Extensive security features	Protects the integrity of communications and data and the privacy of employees, customers and partners, and enables compliance with industry regulations.
Virtual domain hosting and delegated administration	Messaging Server enables you to host messaging for several companies on one server or corporate IT to host multiple departments within the enterprise, reducing number of servers needed, and lowering TCO.
Scalable, robust and extensible components	Enables deployment of unified communication services, bringing together telephone services with email notification, faxing, paging, and other technologies.

Table 1-1 How Communications Services Benefit the Enterprise *(Continued)*

Key Feature	Benefit to the Enterprise
Extensible collaboration platform for scheduling events, and for managing tasks and resources	Calendar Server improves time and resource management, and enhances user productivity.
Group scheduling for meetings and events	Calendar Server improves team collaboration and communication across the enterprise.
Information sharing through hyperlinks in events or tasks	Calendar Server facilitates collaboration through exchange of information relevant to tasks or events.
Multiple client support	Provides integrated web-based client and support for multiple rich clients including Ximian Evolution and Microsoft Outlook.
Open, modular, and standards-based architecture	Enables customers to deploy customized and personalized solutions.

Making the Communications Services Deployment Highly Available

Messaging Server and Calendar Server both provide high-availability options that support both the Sun™ Cluster services and Veritas® clustering solutions. With this option, a secondary Message Server or Calendar Server host provides services to users if the primary system is taken offline for maintenance or is down due to a problem.

Even without the use of Sun Cluster, Messaging Server has built-in monitoring capabilities that continuously check the status of server processes and service availability. Messaging Server can restart process and services automatically, if necessary. Messaging Server logs failures and recovery operations, which you can use for reporting and analysis.

Additionally, you can deploy the Communications Services products in a highly available configuration through use of redundant components. This kind of deployment gives services a high level of uptime. A highly available deployment of this sort requires the redundancy of every component in the service architecture. These components include a duplicate data store server, duplicate network interface cards, and duplicate system storage.

NOTE This guide does not discuss the details of using Sun Cluster in highly available deployments for Communications Services. See the Sun Cluster, Messaging Server, and Calendar Server documentation for more information on this topic.

Using Portal Server with Communications Services

You can install Communications Services products with Portal Server to provide access to messaging and calendar portlets in a portal page. These portlets provide a summary of messaging information, calendar schedules, and address book information. The integration of Portal Server includes single sign-on capabilities between Portal Server, Calendar Express web client, Messaging Express web client, and the Communications Express client.

NOTE You can run Communications Express in both Sun Java™ System Schema 1 and Schema 2 environments. If you are using Schema 2, then you can use Identity Server authentication and single sign-on for Communications Express.

Portal Server also supports message archiving for Instant Messaging. In addition, the Messaging Express, Calendar Express, and Instant Messaging clients are made available to users through the Portal Server Desktop.

The following two components of Portal Server provide additional functionality to a basic Communications Services deployment:

- **Portal Server Desktop.** Instant Messenger installed on Portal Server enables the Instant Messaging client to be launched from the Instant Messaging channel.
- **Sun Java™ System Portal Server 6, Secure Remote Access.** Enables remote end users to securely connect to Secure Remote Access organization's network and its services over the Internet. End users access Secure Remote Access by logging in to the web-based Portal Server Desktop through the Secure Remote Access gateway. The authentication module configured for Portal Server authenticates the end user. The end-user session is established with Portal Server and the access is enabled to the end user's Portal Server Desktop.

NOTE This guide does not discuss portal deploying Communications Services in a portal environment. See the Portal Server documentation for more information.

Understanding the Deployment Process

The Communications Services 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.

For detailed information on the deployment process for Communications Services, see the *Sun Java Systems Messaging Server Deployment Planning Guide*:

`http://docs.sun.com/doc/817-6440`

Designing the Deployment and Architecture

In general, during 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 (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.

Development and Customization

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.

Additional work might be necessary, either in extending services through the use of APIs, or in customizing look and feel, for example, introducing a corporate branding.

For some solutions, development and customization might be quite extensive, requiring you to develop new business and presentation services. In other cases, it might be sufficient to customize existing graphical user interfaces, such as the Portal Server desktop, to achieve the functionality required.

For more information on using product APIs and customizing product functionality, see the appropriate component product documentation, including:

- *Sun Java System Messaging Server Developer's Reference*
- *Sun Java System Messaging Server Messenger Express Customization Guide*
- *Sun Java System Communications Services Event Notification Service Guide*
- *Sun Java System Calendar Server Developer's Guide*

Prototyping and Testing

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 and Customization" on page 25), 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.

Rolling Out the Production System

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. Verifying the deployment and performing capacity planning are also part of this phase. Capacity planning, of which monitoring the system plays an important role, is necessary for meeting the long-term needs of system growth.

Chapter 2

Analyzing Your Enterprise's Requirements

Planning your Communications Services deployment requires that you first analyze your enterprise's business and technical requirements. This chapter helps you to gather and assess your requirements, which you then use to determine your Communications Services design.

This chapter contains the following sections:

Identifying Deployment Goals

Before you purchase or deploy Communications Services hardware or software, you need to identify your deployment goals. Deployment requirements can come from various sources within an enterprise. In many cases, requirements are expressed in vague terms, requiring you to clarify them towards determining a specific goal.

The outcome of your requirements analysis should be a clear, succinct, and measurable set of goals by which to gauge the deployment's success. Proceeding without clear goals accepted by the stake holders of the project is precarious at best.

Some of the requirements you need to examine before you can plan your deployment include:

- Business requirements
- Technical requirements

- Financial requirements
- Service Level Agreements (SLAs)

Business Requirements

Your business objectives affect deployment decisions. Specifically, you need to understand your users' behavior, your site distribution, and the potential political issues that could affect your deployment. If you do not understand these business requirements, you can easily make wrong assumptions that affect the accuracy of your deployment design.

Operational Requirements

Express operational requirements as a set of functional requirements with straightforward goals. Typically, you might come across informal specifications for:

- End-user functionality
- End-user response times
- Availability / uptime
- Information archival and retention

For example, translate a requirement for "adequate end-user response time" into measurable terms such that all stake holders understand what is "adequate" and how the response time is measured.

Culture and Politics

A deployment needs to take into account your corporate culture and politics. Demands can arise from areas that end up representing a business requirement. For example:

- Some sites might require their own management of the deployed solution. Such demands can raise the project's training costs, complexities, and so forth.
- Given that the LDAP directory contains personnel data, the Human Resources department might want to own and control the directory.

Technical Requirements

Technical requirements (or functional requirements) are the details of your organization's system needs.

Supporting Existing Usage Patterns

Express existing usage patterns as clearly measurable goals for the deployment to achieve. The following questions will help you determine such goals.

- How are current services utilized?
- Can your users be categorized (for example, as sporadic, frequent, or heavy users)?
- What size messages do users commonly send?
- How many invites are usually on calendar appointments?
- How many messages do users send?
- How many calendar events and tasks do users typically create per day or per hour?
- To which sites in your company do your users send messages?

Study the users who will access your services. Factors such as when they will use existing services are keys to identifying your deployment requirements and therefore goals. If your organization's experience cannot provide these patterns, study the experience of other organizations to estimate your own.

Regions in organizations that have heavy usage might need their own servers. Generally, if your users are far away from the actual servers, they will experience slower response times. Consider whether the response times will be acceptable.

Site Distribution

Use these questions to understand how site distribution impacts your deployment goals:

- How are your sites geographically distributed?
- What is the bandwidth between the sites? Centralized approaches will require greater bandwidth than de-centralized. Mission critical sites might need their own servers.

Network

The following questions help you understand your network requirements:

- Do you want to obfuscate internal network information?
- Do you want to provide redundancy of network services?
- Do you want to limit available data on access layer hosts?
- Do you want to simplify end-user settings, for example, have end users enter a single mail host that does not have to change?
- Do you want to reduce network HTTP traffic?

NOTE Answering yes to these questions suggests a two-tier architecture.

Existing Infrastructure

You might be able to centralize servers if you have more reliable and higher available bandwidth.

- Will the existing infrastructure and facilities prove adequate to enable this deployment?
- Can the DNS server cope with the extra load? Directory server? Network? Routers? Switches? Firewall?

Support Personnel

24-hour, seven-day-a-week (24 x 7) support might only be available at certain sites. A simpler architecture with fewer servers will be easier to support.

- Is there sufficient capacity in operations and technical support groups to facilitate this deployment?
- Can operations and technical support groups cope with the increased load during deployment phase?

Financial Requirements

Financial restrictions impact how you construct your deployment. Financial requirements tend to be clearly defined from an overall perspective providing a limit or target of the deployment.

Beyond the obvious hardware, software, and maintenance costs, a number of other costs can impact the overall project cost, including:

- Training
- Upgrade of other services and facilities, for example, network bandwidth or routers
- Deployment costs, such as personnel and resources required to prove the deployment concept
- Operational costs, such as personnel to administer the deployed solution

You can avoid financial issues associated with the project by applying sufficient attention and analysis to the many factors associated with the project requirements.

Service Level Agreements (SLAs)

You should develop SLAs for your deployment around such areas as uptime, response time, message delivery time, and disaster recovery. An SLA itself should account for such items as an overview of the system, the roles and responsibilities of support organizations, response times, how to measure service levels, change requests, and so forth.

Identifying your organization's expectations around system availability is key in determining the scope of your SLAs. System availability is often expressed as a percentage of the system uptime. A basic equation to calculate system availability is:

```
Availability = uptime / (uptime + downtime) * 100
```

For instance, a service level agreement uptime of four nines (99.99 percent) means that in a month the system can be unavailable for about four minutes.

Furthermore, system downtime is the total time the system is not available for use. This total includes not only unplanned downtime, such as hardware failures and network outages, but also planned downtime, preventive maintenance, software upgrade, patches, and so on. If the system is supposed to be available 7x24 (seven days a week, 24 hours a day), the architecture needs to include redundancy to avoid planned and unplanned downtime to ensure high availability.

Determining Project Goals

Your investigation and analysis should reveal your project's requirements. Next, you should be able to determine a clearly measurable set of goals. Specify these goals in such a manner that personnel not directly associated with the project can understand the goals and how to measure the project against them.

Stake holders need to accept the project goals. The projects goals need to be measured in a post-implementation review to determine the success of the project.

Planning for Growth

In addition to determining what capacity you need today, assess what capacity you need in the future, within a timeframe that you can plan for. Typically, a growth timeline is in the range of six to twelve months. Growth expectations and changes in usage characteristics are factors that you need to take into account to accommodate growth.

As the number of users and messages increase, you should outline successful guidelines for capacity planning. You need to plan for increases in message traffic for the various servers, a larger volume of users, larger mailbox sizes, more calendar appointments, and so forth. As growth occurs in the user population, usage characteristics change over time. Your deployment goals (and therefore deployment design) must respond accordingly to be viable into the future.

Ideally, you should design your architecture to easily accommodate future growth. For example, use logical names for the Communications Services themselves. See "Using Logical Service Names" on page 73 for more information. Monitoring the deployment, once it enters its production phase, is also crucial to being able to understand when and by how much a deployment needs to grow.

Understanding Total Cost of Ownership

Total Cost of Ownership (TCO) is another factor that affects capacity planning. This includes choosing the hardware upon which to deploy your Communications Services. Table 2-1 on page 33 presents some factors to consider as to whether to deploy more smaller hardware systems or fewer larger hardware systems.

Table 2-1 Considerations for Total Cost of Ownership

Hardware Choice	Pros	Cons
More, smaller hardware systems	• Smaller hardware systems generally cost less. • More, smaller hardware systems can be deployed across many locations to support a distributed business environment. • More, smaller hardware systems can mean less down time for system maintenance, upgrade, and migration because traffic can be routed to other servers that are still online while others are being maintained.	• Smaller hardware systems have a more limited capacity, so more of them are needed. Management, administration, and maintenance costs go up as the number of hardware systems goes up. • More, smaller hardware systems require more system maintenance because there are more of them to maintain.
Fewer, larger hardware systems	• Fewer hardware systems means fewer fixed management costs per server. If your management costs are a recurring monthly bill, whether internal or from an ISP, costs will be lower, because you have fewer hardware systems to manage. • Fewer hardware systems can also mean easier system maintenance, upgrade, and migration because there are fewer systems to maintain.	• Larger hardware systems generally cost more initially. • Fewer hardware systems can also mean a greater system down-time for maintenance, upgrade and migration.

Chapter 3

Understanding Product Requirements and Considerations

This chapter describes requirements and considerations that impact the design of your deployment. You need to understand these requirements and considerations to accurately determine your Communications Services architecture.

This chapter contains the following sections:

Planning for Various Components

When designing your deployment architecture, take into account the requirements of the various component products of your deployment. For example, if you have a technical requirement to integrate Communications Services with other Java System products, you need to choose your schema accordingly. Inter-product dependencies, for example, how Communications Services access and place load on Directory Server, present deployment choices as well.

Understanding the individual components of each product enables you to plan for the type of architecture to best suit your requirements. Depending on your deployment, you need to potentially understand and plan for the following components:

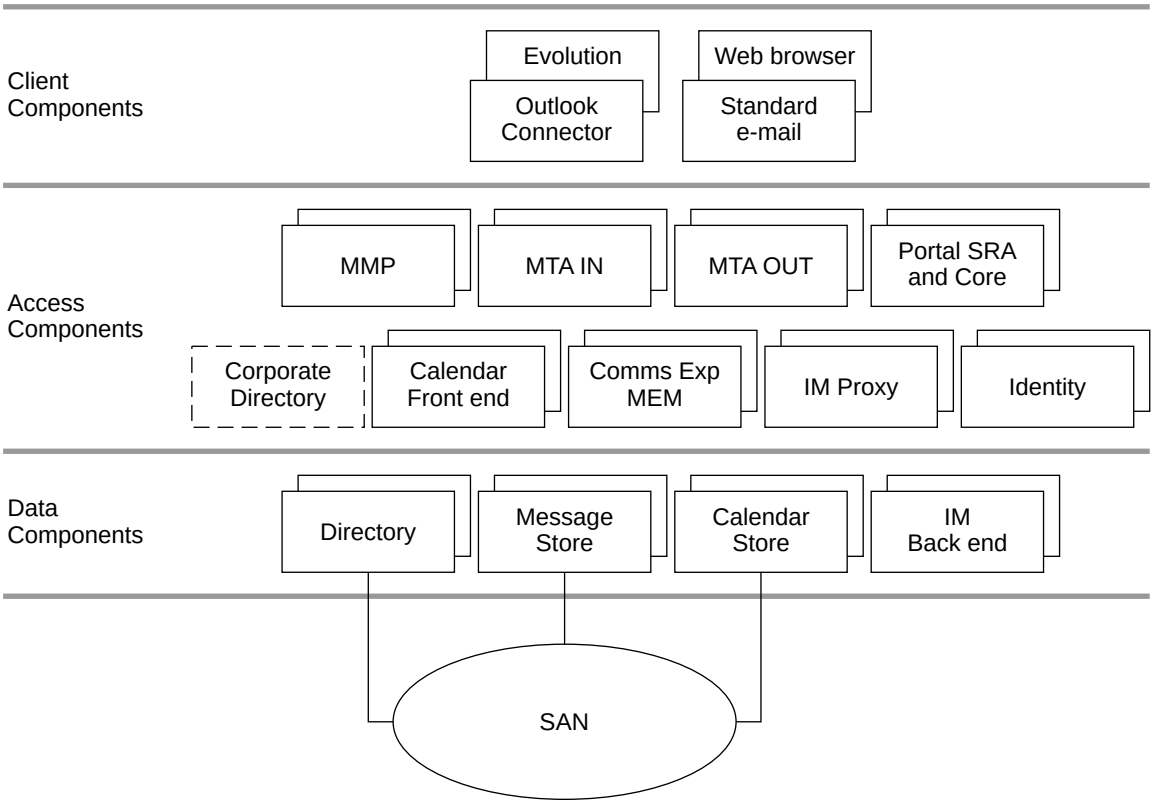
- LDAP Directory Information Tree
- Schema
- Directory Server (Identity Server)
- Messaging Server
  - Message Transfer Agent (MTA)
  - Message Store
  - MMP (Messaging Multiplexor)
  - MEM (Messaging Express Multiplexor)
- Calendar Server
  - Front End
  - Calendar Store
- Instant Messaging
  - Instant Messaging Proxy
  - Instant Messaging Back End
- Portal Server
- Connector for Microsoft Outlook
- Communications Express

Understanding Service Components and Service Tiers

When planning a Communications Services deployment for multiple component products or services, you need to understand the composition of each component product (or service) itself.

Figure 3-1 on page 38 illustrates how you can separate each service into components that can be deployed on separate hosts, and the particular tier each component occupies. Though you can deploy all components on a single host, or deploy a particular service's components on the same host, consider moving to a tiered architecture. A tiered architecture, whether it be single-tiered, or two-tiered, provides a number of benefits. See "Benefits of a Single Tier Architecture" on page 68 and "Benefits of a Two Tier Architecture" on page 68 for more information.

Figure 3-1 Communications Services Components



In the preceding figure, the client components consist of the Outlook Connector plugin, thick clients such as Evolution, browsers, and standard email applications. These components reside on end users' client computers. The access layer components consist of front-end services from Messaging Server (MMP, MTA, and MEM); Calendar Server; Communications Express (which must be collocated with MEM); Instant Messaging (Instant Messaging Proxy); Portal Server (SRA and Core); Identity Server (for authentication) and a corporate directory, which provides address book lookup. The data layer components consist of back-end services from Directory Server (which, in itself, can consist of front-end and back-end components); Messaging Server (Message Store); Calendar Server (Calendar Store); and Instant Messaging. A Storage Area Network (SAN) "cloud" represents the physical data storage.

NOTE The corporate directory shown in this figure is not a component product in itself. It represents a “copy” of the corporate directory that enterprises typically deploy in the access layer for clients to perform address-book type lookups.

The following sections explain these various components in more detail.

LDAP Directory Information Tree Requirements

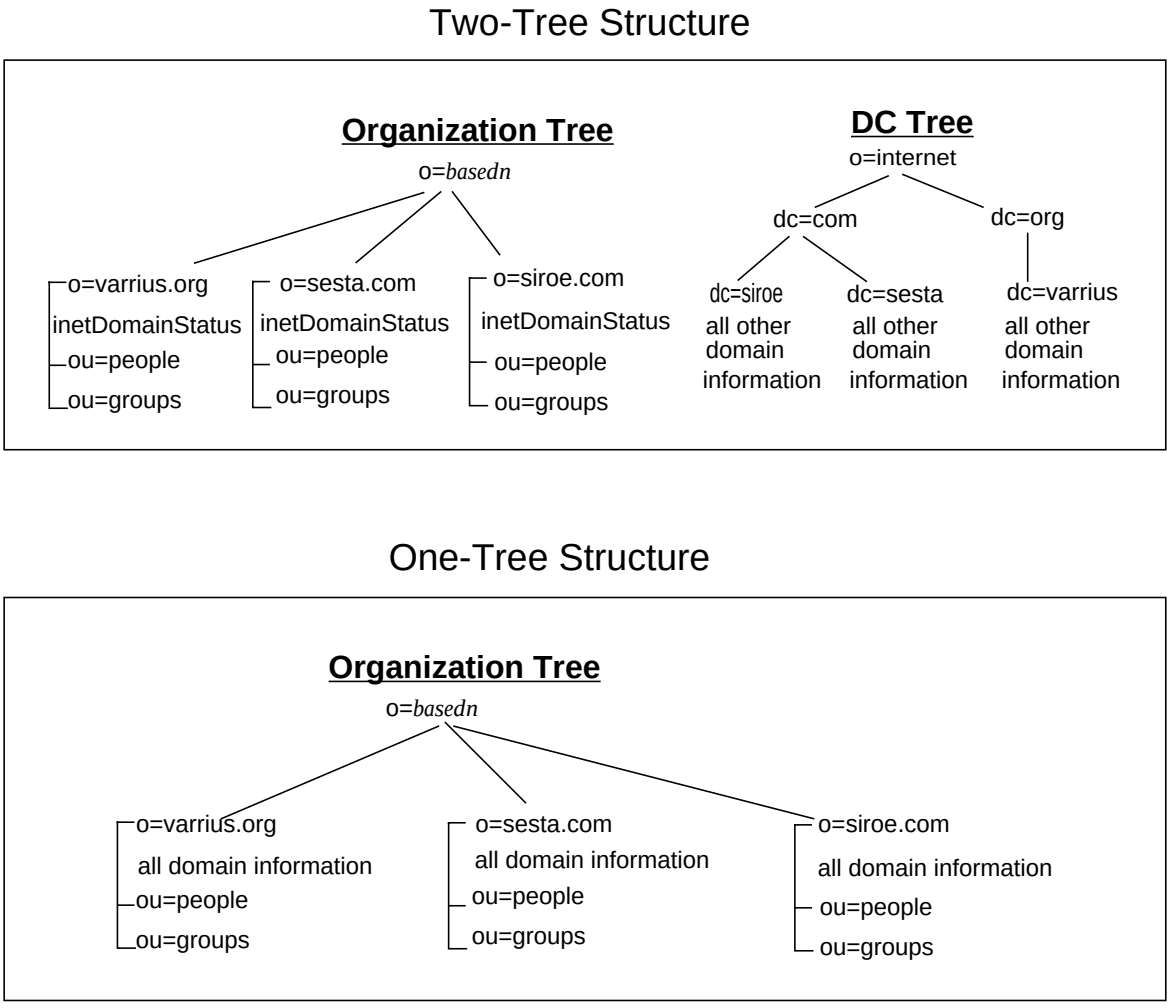
The Directory Information Tree (DIT) is a way to organize directory entries in a tree structure, or schema, with nodes representing domains, subdomains, users, and groups. Sun Java Enterprise System introduces a fundamental change to how the directory is structured by implementing a one-tree structure.

Changes in the DIT Structure

Messaging Server and Calendar Server have introduced a one-tree structure, where there is no Domain Component (DC) Tree. All domain information is held in domain nodes in the Organization Tree. Aliasing is handled entirely differently in the new one-DIT structure.

The bottom half of Figure 3-2 illustrates a one-tree LDAP structure.

Figure 3-2 Two-Tree LDAP Structure Compared With One-Tree Structure



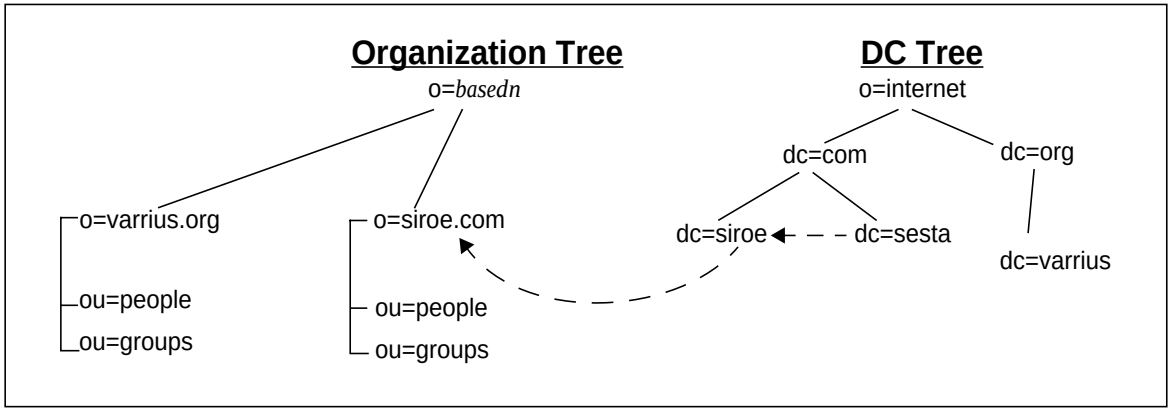
Benefits of a One-Tree DIT Structure

The main advantages to using the one-tree structure Schema 2 native mode are:

- The structure is integrated with Identity Server.
- The structure is more closely aligned with industry standards.
- The structure is significantly less complex than the two-tree structure.

As illustrated in the following figure, in the two-tree structure, some nodes point directly to a node in the Organization Tree (using the attribute `inetDomainBaseDN`). Other nodes are aliased nodes, which instead of pointing directly to an Organization Tree node, point to another DC Tree node, using the `aliasedObjectName` attribute.

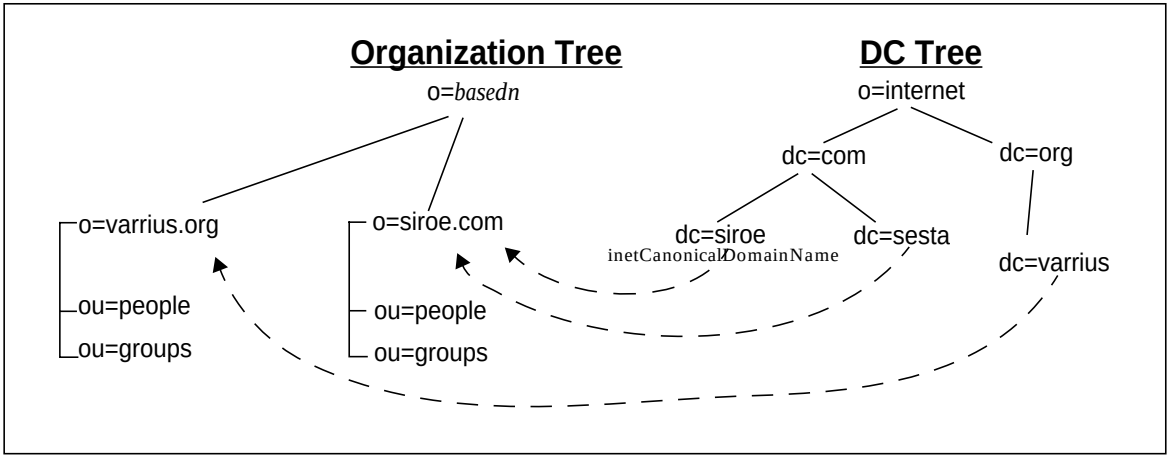
Figure 3-3 Two-Tree Aliasing With `aliasedDomainName` and `inetDomainBaseDN`



In the previous figure, `sesta.com` in the DC Tree points to `siroe.com` in the DC Tree using `aliasedObjectName`, and `siroe.com` points to the like named node in the Organization Tree, using `inetDomainBaseDN`.

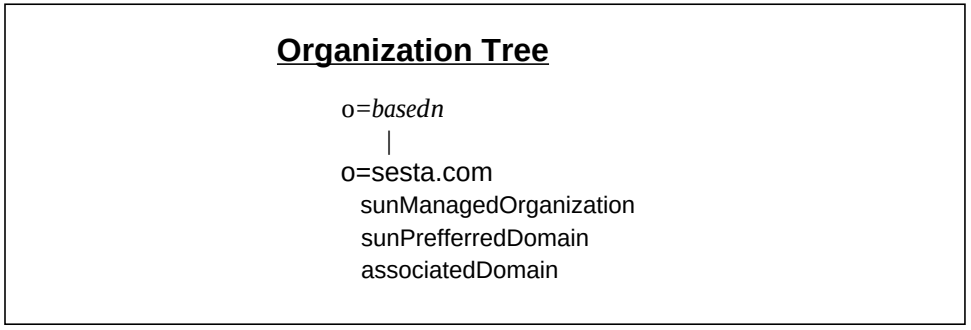
Furthermore, as shown in Figure 3-4 on page 42, there could be one or more nodes in the DC Tree using `inetDomainBaseDN` to point directly to the same node in the Organization Tree. In this case, a “tie-breaker” attribute, `inetCanonicalDomainName`, is necessary on one of the DC Tree nodes to designate which is the “real” domain name (the domain where the mail actually resides and where the mail is routed).

Figure 3-4 Two-Tree Aliasing With `inetCanonicalDomainName`



By contrast, a one-tree structure contains only an Organization Tree, as shown in the following figure.

Figure 3-5 One-Tree Aliasing With `associatedDomain`



In the one-tree structure, domain nodes in the Organization Tree contain all the domain attributes formerly found on the DC Tree. Each domain node is identified by the `sunManagedOrganization` object class and `sunPreferredDomain` attribute, which contains the DNS domain name. A domain node can also have one or more `associatedDomain` attributes, which list the alias names this domain is known by. Contrary to the two-tree structure, there are no duplicate nodes for the alias names.

A one-tree DIT structure is beneficial in how you partition data for organization-specific access control. That is, each organization can have a separate subtree in the DIT where user and group entries are located. Access to that data can be limited to users in that part of the subtree. This allows localized applications to operate securely.

In addition, for new deployments of Calendar Server or Messaging Server, a one-tree structure maps better to existing single-DIT LDAP applications.

Schema Requirements

Before you install any of the Communications Services products, you need to understand which schema you will use. The schema is the set of definitions describing what types of information can be stored as entries in the directory. Two schema choices, Sun Java™ System LDAP Schema 1 and Sun Java™ System LDAP Schema 2, are available and supported with Communications Services. Your choice of schema depends on the following criteria:

Use Schema 2 if you:

- Want to use Identity Server 6 2004Q2 (formerly Sun ONE Identity Server 6.1), either for user provisioning or single sign-on (SSO)
- Are installing Communications Services components for the first time
- Are integrating Communications Services with other Java Enterprise System products, such as Portal Server

NOTE You do not have to use Identity Server 6 to provide SSO. If you choose, you can still use the trusted circle type of SSO, which does not rely on Identity Server 6.

Use Schema 1 if you:

- Have an existing version of any of the Communications Services components, for example, if you are upgrading from Messaging Server 5.2
- Do not need to provision users through Identity Server (unless Identity Server SSO is also a requirement)
- Want to use Delegated Administrator to do your user provisioning

See the *Sun Java System Messaging Server Deployment Planning Guide* for more information on schema choices:

http://docs.sun.com/doc/817-6440

Directory Server Considerations

Sun Java System Directory Server provides flexible, multi-tiered data storage for 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.

All custom LDAP schemas, such as those from Portal Server, Identity Server, Messaging Server, Calendar Server, and Instant Messaging Server, install into a single Directory.

There are many ways to architect your data environment and many factors to consider that depend on your business objective and expected usage patterns. Your directory design should address the following areas:

- Directory schema and object class definitions
- The Directory Information Tree (DIT)
- Groups and membership definitions
- A strategy for Access Control Lists (ACLs) including static and dynamic groups
- Directory architecture for high availability and performance, including failover, replication, and referrals
- Management of the directory

See the *Sun Java System Directory Server Deployment Planning Guide* for a complete description of these factors and suggestions about how to architect your data environment:

http://docs.sun.com/doc/817-5218

Directory Server and Tiered Architecture Considerations

In moving from a single tier architecture to a multiple tier architecture, Directory Server should be the first component that you “split out” onto its own machine. At a certain point of load, Directory Server and Messaging Server on the same host have inherent performance impacts. This is due to the way Messaging Server is architected to work with Directory Server. Separating Directory Server onto its own machine is the first step to improve performance for a deployment.

See the *Sun Java System Messaging Server Deployment Planning Guide* for more information on tiered architectures:

`http://docs.sun.com/doc/817-6440`

Directory Server Topology Considerations

Though it is feasible to build a deployment around an instance of Directory Server installed on a single machine, the other Communications Services components depend upon the directory as a core service to function. Thus, beyond trivial deployments, you should plan to deploy Directory Server in a redundant or highly availability configuration.

The first step toward making Directory Server more available is to establish a pair of master directory servers. Next, multi-master replication can be used to improve the LDAP write throughput and availability. If Sun Cluster is used for a high-availability deployment, then the two LDAP masters are clustered together. See “Making the Directory Highly Available” on page 78 for more information.

Directory Server Capacity Planning

While there are no hard and fast rules for Directory Server capacity planning, actively monitoring the directory is essential to ensure that performance metrics are met. When the system is not meeting these metrics, then it is time to add an additional directory consumer. Typically, you will want to monitor:

- Hit load
- Cache load
- Requests per second

Evaluate the above metrics against a target response time of 10 Milliseconds. The IOWAIT should not exceed 10 Milliseconds, and the sum of the CPU utilization in this tier should not exceed 70 percent.

Directory Server and Calendar Server Interaction Considerations

Calendar Server performs multiple writes to user entries stored in Directory Server. The bulk of these writes occur when the user logs into Calendar Server for the first time and when the user performs certain actions. These actions include creating a calendar, subscribing to a calendar, changing a preference, and so on. If you do not take these actions into consideration, the Directory Master Server can experience heavy loads.

If you use Directory replication, the LDAP Master Server is replicating entries to the LDAP Replica servers. As Calendar users perform one of these actions, Calendar Server will only be able to write changes to the Master Directory Server. This is because the Replicas are read-only.

The second interaction consideration exists in these replicated Directory structures. As users make preference changes, their changes mighty not be rendered successful until the change is successfully replicated from the Master Directory Server to the Directory Replica, which is in use by the Calendar Server. A workaround is available, in which you configure Calendar Express (`cshttpd`) attempts to cache the change locally to avoid this latency delay.

Directory Server and Personal Address Book Considerations

The Messenger Express Client supports the concept of a Personal Address Book (PAB). This enables users to store personal contacts (for example, business contacts, friends, and family) in the Directory Server. Each time a new personal contact is added to the user's PAB, a write is made on the Directory Server. If you do not take these actions into consideration, the LDAP Master Server can face heavy loads (regardless of the Directory replication strategy).

One method to avoid performance issues on the User and Group Directory Server is to place the PAB information on a separate Directory Server. This enables PAB interactions to avoid placing a load on the LDAP Master Server.

Directory Server and Communications Express Considerations

The Communications Express client requires some configuration changes on the Directory Server. The Communications Express configuration requires the `comms_dssetup.pl` script, which ships with Java Enterprise System 2004Q2. This script is different from the one that shipped with Sun ONE Calendar Server 6.0 and Sun ONE Messaging Server 6.0 as part of Java Enterprise System 2003Q4.

Messaging Server Considerations

In developing your Communications Services architecture, you need to evaluate the performance aspects of the following Messaging Server components:

- Message Store
- Message Transfer Agent (MTA)
- Mail Message Proxy (MMP)
- Messaging Express Multiplexor (MEM)

The following sections provide an overview of the considerations for each of these components. For a complete discussion of the performance aspects of these components, and potential hardware solutions, see the *Sun Java System Messaging Server Deployment Planning Guide*:

`http://docs.sun.com/doc/817-6440`

Message Store Considerations

Message store performance is affected by a variety of factors, including:

1. Disk I/O
2. Inbound message rate (also known as message insertion rate)
3. Message sizes
4. Login rate (POP/IMAP/HTTP)
5. Transaction rate (IMAP/HTTP)

6. Concurrent number of connections for the various protocols
7. Network I/O

The preceding factors list the approximate order of impact to the Message Store. Most performance issues with the Message Store arise from insufficient disk I/O capacity. Additionally, the way in which you lay out the store on the physical disks can also have a performance impact. For smaller standalone systems, it is possible to use a simple stripe of disks to provide sufficient I/O. For most larger systems, segregate the file system and provide I/O to the various parts of store.

Message Transfer Agent (MTA) Considerations

MTA performance is affected by a number of factors including, but not limited to:

- Disk performance
- Use of SSL
- The number of messages/connections inbound and outbound
- The size of messages
- The number of target destinations/messages
- The speed and latency of connections to and from the MTA
- The need to do spam or virus filtering
- The use of SIEVE language filtering rules and the need to do other message parsing (like use of the conversion channel)

The MTA router is both CPU and I/O intensive. The MTA uses two different file systems for the queue directory and the logging directory. For a small host (four processors or less) functioning as an MTA router, you do not need to separate these directories on different file systems. The queue directory is written to synchronously with fairly large writes. The logging directory is a series of smaller asynchronous and sequential writes.

In most cases, you will want to plan for redundancy in the MTA in the disk subsystem to avoid permanent loss of mail in the event of a spindle failure. (A spindle failure is by far the single most likely hardware failure.) This implies that either an external disk array or a system with many internal spindles is optimal.

In addition, with respect to placement of MTAs, you should always deploy the MTA at your firewall.

MTA and LMTP Considerations

The Messaging Server's LMTP service as implemented is not designed to work with other LMTP servers or other LMTP clients. See the *Sun Java System Messaging Server Release Notes* for information on setting up a mix of LMTP and SMTP servers in your deployment:

http://docs.sun.com/db/doc/817-6363

Mail Message Proxy (MMP) Considerations

The MMP uses no disk I/O other than for logging. The MMP is completely CPU and network bound. Unlike all the other Messaging Server components, the MMP is not multiprocess and multithreaded. The primary execution code is single process and multithreaded. Thus, because the MMP is not sufficiently a multiprocess, it does not scale as well as the other components.

The MMP does not scale beyond four processors, and scales less than linearly from two to four processors. Two processor, rack mounted machines are good candidates for MMPs.

In deployments where you choose to put other component software on the same machine as the MMP (MEM, Calendar Server front end, Communications Express Web Client, LDAP proxy, and so on), look at deploying a larger, four processor SPARC machine. Such a configuration reduces the total number of machines that need to be managed, patched, monitored, and so forth.

Messaging Express Multiplexor (MEM) Considerations

The MEM provides a middle-tier proxy for the Webmail client. This client enables users to access mail and to compose messages through a browser. The benefit of the MEM is that end users only connect to the MEM to access email, regardless of which back-end server is storing their mail. MEM accomplishes this by managing the HTTP session information and user profiles via the user's LDAP information. The second benefit is that all static files and LDAP authentication states are located on the Messaging Server front end. This benefit offsets some of the additional CPU requirements associated with web page rendering from the Message Store back end.

The MEM has many of the same characteristics as the MMP. The MEM will scale beyond four processors, but in most environments, there is no particular value in doing so. Also, in the future, the Webmail component will be offloaded from the Message Store and onto access layer machines that are running the XML rendering as Java servlets under the web server. Java servlets do not presently scale well beyond two processors. Thus, plan your hardware choice around either SPARC or Intel two-processor machines for the MEM, or assume that you will repurpose your current two-processor MEM hardware to be replaced by smaller machines when the next generation solution becomes available.

You can put the MMP and MEM on the same set of servers. The advantage to doing so is if a small number of either MMPs or MEMs are required, the amount of extra hardware for redundancy is minimized. The only possible downside to collocating the MMP and MEM on the same set of servers is that a denial of service attack on one protocol can impact the others.

Calendar Server Considerations

Calendar Server consists of five major services:

- HTTP Service (`cshttpd`) listens for HTTP requests. It receives user requests and returns data to the caller.
- Administration Service (`csadmind`) is required for each instance of Calendar Server. It provides a single point of authentication and administration for the Calendar Servers and provides most of the administration tools.
- Notification Service (`csnotify`) sends notifications of events and to-dos using either email or the Event Notification Service.
- Event Notification Service (`enpd`) acts as the broker for event alarms.
- Distributed Database Service (`csdwpd`) links multiple database servers together within the same Calendar Server system to form a distributed calendar store.

In a scalable Calendar Server deployment, you deploy an instance of HTTP Service and Administration Service together as a Calendar front-end system. (You would deploy one instance of the `cshttpd` per machine. On each machine, you would configure one `cshttpd` process per CPU on that machine.) An instance of Notification Service, Event Notification Service, Distributed Database Service and Administration Service are deployed together as the Calendar back-end system.

Authentication and XML / XSLT transformation are two Calendar Service activities that generate heavy load. Additional CPUs can be added to meet quality of service requirements.

Calendar back-end services usually require half the number of CPUs sized for the Calendar front-end services. To support quality of service by the Calendar front-end system, the Calendar back-end system should use around two-thirds of the front-end CPUs.

You will want to consider early on in a deployment separating the Calendar Service into front-end and back-end services.

The Calendar Server HTTP process that is typically a component of the front-end services is a dominant user of CPU time. This suggests care should be taken in accounting for peak calendar usage and choosing sufficient front-end processing power to accommodate the expected peak HTTP sessions. Typically, you would make the Calendar Server front end more available through redundancy, that is, by deploying multiple front-end hosts. As the front-end systems do not maintain any persistent calendar data, they are not good candidates for HA solutions like Sun Cluster or Veritas. Moreover, the additional hardware and administrative overhead of such solutions make deploying HA for Calendar Server front ends both expensive and time-consuming.

NOTE The only configuration for Calendar front ends that might warrant a true HA solution is where you have deployed the Calendar front end on the same host that contains a Messaging Server MTA router. Even in this configuration, however, the overhead of such a solution should be carefully weighed against the slight benefit.

A good choice of hardware for the Calendar Server front ends is a single or dual processor SPARC or Intel server. You would deploy one instance of the Calendar Server `cshttpd` process per machine. Such a deployment affords a cost-effective solution, enabling you to start with some level of initial client concurrency capability and add client session capacity as you discover peak usage levels on your existing configuration.

When you deploy multiple front ends, a load balancer (with sticky / persistent connections) is necessary to distribute the load across the front-end services.

NOTE Communications Express does not scale beyond two processors. The same hardware choices explained previously for Calendar Server apply to Communications Express deployments.

The Calendar Server back-end services are well balanced in resource consumption and show no evidence of bottleneck formation either in CPU or I/O (disk or network). Thus, a good choice of hardware for the back end would be a SPARC server with a single striped volume. Such a machine presents considerable capacity for large-peak calendar loads.

If your requirements include high availability, it makes sense to deploy the Calendar Server back end with either Sun Cluster or Veritas cluster, as the back end does contain persistent data.

NOTE In a configuration with both front-end and back-end Calender Server hosts, all hosts must be running:

- The same operating system
- The same releases of Calendar Server, including patch or hotfix releases.

Instant Messaging Considerations

As with other Communications Services components, you can create an architecture in which you separate Instant Messaging into front-end (Instant Messaging multiplexor) and back-end components (server and store).

See the *Sun Java System Instant Messaging Deployment Planning Guide* for more information regarding Instant Messaging considerations:

`http://docs.sun.com/doc/817-5937`

Portal Server Considerations

You can install Communications Services products with Portal Server to provide an “umbrella” front end to access messaging, calendar, and instant messaging applications. The integration of Portal Server includes single sign-on capabilities between Portal Server, Calendar Express web client, Messaging Express web client and Communications Express client. In addition, the Messaging Express, Calendar Express, and Instant Messaging clients are made available to users through the Portal Server desktop.

See the *Sun Java System Portal Server Deployment Guide* and the *Sun Java System Identity Server Deployment Planning Guide* for more information:

http://docs.sun.com/doc/817-5321

http://docs.sun.com/doc/817-5707

Connector for Microsoft Outlook Considerations

This section describes some deployment issues you will encounter deploying Connector for Microsoft Outlook. For complete information, see the Connector for Microsoft Outlook documentation:

http://docs.sun.com/db/coll/ConnectorMSO_60

Sun Java System Connector for Microsoft Outlook enables Outlook to be used as a desktop client with Sun Java Enterprise System. Connector for Microsoft Outlook is an Outlook plug-in that must be installed on the end-user's desktop. Connector for Microsoft Outlook queries Messaging Server for folder hierarchies and email messages. It converts the information into Messaging API (MAPI) properties that Outlook can display. Similarly, it uses the WCAP protocol to query Calendar Server for events and tasks which are then converted into MAPI properties. With this model, 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.

Connector for Microsoft Outlook Component Product Dependencies

To use Connector for Microsoft Outlook, both Sun Java System Messaging Server and Sun Java System Calendar Server are required in the same deployment. See those products' release notes for information on supported versions.

For Connector for Microsoft Outlook to function correctly, the following LDAP attributes in the Sun Java System Directory Server should be indexed for at least presence and equality to improve the overall performance:

- `icsCalendar`
- `mail`
- `mailalternateaddress`

See the *Sun Java System Connector for Microsoft Outlook Release Notes* for a complete list of product dependencies.

http://docs.sun.com/db/coll/MessagingServer_04q2

Migrating Sun ONE Calendar Server Data

If you are using a version of Calendar Server prior to Sun ONE Calendar Server 6.0, you need to engage Sun Professional Services℠ to convert and migrate the data to the new format. This Calendar Server data migration is required for the use of Outlook, and is necessary because of the underlying changes in the storage and management of recurring events. No migration service is required for new customers of Sun Java System Calendar Server.

Migrating Exchange Server Data

Connector for Microsoft Outlook does not convert Microsoft Exchange Server messages on the Exchange Server. You need to engage with Sun Professional Services to convert the data.

Communications Express Considerations

Sun Java System Communications Express is a webserver-based application that can be accessed by using a browser. The Communications Express offering consists of three client modules: Calendar, Address Book, and Mail. The Calendar, Address Book, and Mail client modules are deployed as a single application in web containers and are collectively referred to as the Communications Express client.

Communications Express depends upon the following Sun Java System component products:

- Directory Server
- (Optional) Identity Server
- Calendar Server
- Messaging Server

You install Communications Express as a front-end server (in a multi-tier environment). You must install the complete set of Messaging Server packages on the same host that Communications Express is running on. The Messaging Server is used to provide the Messenger Express interface directly to the Message Store, or to provide the MEM, which connects to a back-end store running Messenger Express. In addition, you can configure the AddressBook server of Communications Express on the front-end machine to store its data either in the LDAP directory infrastructure or on an LDAP server other than the Communications Express machine. See the *Sun Java Communications Express Administration Guide* for more information:

`http://docs.sun.com/doc/817-5416`

Communications Express communicates with the local (typically on the same machine) `cshttp` daemons for Calendar Server. In turn, these daemons communicate to the Calendar Server back end by using DWP. Communications Express utilizes Messenger Express (Webmail) to communicate with the Message Store.

When using a load balancer or port director type device, make sure to utilize “sticky” (persistent) connections such that users are continually routed to the same front-end server for the duration of their session.

Security Considerations

Security for a Communications Services enterprise deployment is managed by taking a “defense in depth” approach. By individually securing the network, hardware platform, operating system, and applications themselves, you make each layer of the architecture secure. Security includes hardening each layer by closing unnecessary network ports and access mechanisms. You also minimize the number of installed software packages so that only those packages required by the system are available. Finally, you secure and insulate the layers from unintended access within the network.

You can implement a Messaging Server proxy server to augment data security. A proxy server placed on the firewall with the Messaging Server behind it prevents attacks on the information on the Messaging Server.

Calendar Server provides a number of security levels to protect users against eavesdropping, unsanctioned usage, or external attack. The basic level of security is through authentication. Calendar Server uses LDAP authentication by default, but also supports the use of an authentication plugin for cases where an alternate means of authentication is desired. Integration with Identity Server enables Calendar Server to take advantage of its single sign-on capability.

Instant Messaging ensures the integrity of communications through its multiple authentication mechanisms and secure SSL connections. Integration with Portal Server and Identity Server bring additional security features, services-based provisioning access policy, user management, and secure remote access.

For more information on security, see the *Sun Java System Messaging Server Deployment Planning Guide*, the *Sun Java System Calendar Server Administration Guide*, and the *Sun Java System Instant Messaging Deployment Planning Guide*.

NOTE To ensure a completely security environment, the deployment needs a time server to synchronize the internal clocks of the hosts being secured.

Network Security

The recommended deployment configuration, to support both horizontal scalability and service security, is to place the access layer of the architecture behind a firewall. In a two-tier architecture, use two firewalls, creating a DMZ. This enables access to the information delivery elements, the calendar and messaging front ends, while protecting the main service elements on the internal network behind a second firewall. Such a configuration also enables the access layer and data layer elements to be scaled independently, accommodating traffic and storage elements.

Operating System Security

Reduce potential risk of security breaches in the operating environment by performing the following, often termed “system hardening:”

- **Minimize the size of the operating environment installation.** When installing a Sun server in an environment that is exposed to the Internet, or any untrusted network, reduce the Solaris installation to the minimum number of packages necessary to support the applications to be hosted. Achieving minimization in services, libraries, and applications helps increase security by reducing the number of subsystems that must be maintained.

  The Solaris™ Security Toolkit provides a flexible and extensible mechanism to minimize, harden, and secure Solaris Operating Environment systems. The primary goal behind the development of this toolkit is to simplify and automate the process of securing Solaris systems. For more information see:

http://wwws.sun.com/software/security/jass/

- **Track and monitor file system changes.** Within systems that require inclusion of security, a file change control and audit tool is indispensable as it tracks changes in files and detects possible intrusion. You can use a product such as Tripwire for Servers, or Solaris Fingerprint Database (available from SunSolve Online).

Application Security

The Communications Services product portfolio provides features that ensure the security and integrity of business communications. Communications Services offer extensive “built-in” security features, such as:

- Authentication
- Message and session encryption
- Virus and spam protection
- Archiving and auditing of communications
- End-user configurable privacy options

Implementing Secure Connections

Communications Services support security standards such as SSL / TLS, S / MIME, and SAML. SSL / TLS enables all communication between clients and servers to take place inside an encrypted session.Through integration with Portal Server, additional authentication mechanisms are available out-of-the-box, as well as single sign-on capabilities across the applications.

If you want to implement public key data security, you must select mail clients that support public key infrastructure and key choice. Communications Services products are capable, out of the box, of participating in the transmission and storage of messages so encrypted. However, the Webmail client is not capable of generating or decoding the messages.

A more commonly used mechanism for data security protects the data across the wire only (that is, from client to server) by using SSL encryption on the connections used to transmit data between various messaging agents. This solution is not as complete as public key encryption, but is far easier to implement and is supported by many more products and service providers.

What problem does this solve? The enterprise assumes that it controls its own corporate network and that data transmitted on that network is safe from non-employees. Mail sent to anyone from outside the corporate network using the corporation's infrastructure transmits the data over an encrypted connection to the corporation's network. Likewise, all mail picked up by a corporate user from outside the corporate network will be transmitted over an encrypted connection. Thus, if the enterprise's assumption about the safety of the internal network is true, and its employees use only sanctioned servers for transmission between themselves and other employees, mail between employees is safe from external attack.

What doesn't this solution solve? First of all, this approach does not protect the data from unintended viewing by non-intended recipients of the data who have access to the enterprise's internal network. Secondly, there is no protection offered for data being transmitted between employees and their external partners, customers, or suppliers. The data travels across the public Internet in a completely insecure fashion.

However, this problem can be remedied by configuring SSL encryption between MTA routers at both the enterprise's and customer's network. This type of solution requires setup for each private connection you want to use. In so doing, you add an important additional layer of security with customer or partner data being sent or received via email. Using Communications Services' MTA and SSL, companies can save money by using the public Internet as the transport, but force the MTAs to use SSL for their partners. This solution does not take into account other network traffic to and from partners. Nevertheless, mail is usually a large proportion of the traffic, and because companies can pay based on data transmitted, using the public Internet is usually cheaper.

Implementing Secure Connections Using Two Different Certificate Authorities (CAs)

You can implement SSL connections between server and clients, for example, from Messaging Server, and to other servers in your deployment as well (Web Server, Calendar Server, Directory Server). If desired, you can use two different Certificate Authorities (CAs), one for the server and one for the client.

In such a scenario, you can use one CA to issue server certificates, and another CA to issue client certificates. If you want the client to accept the server's certificate as genuine, you will need to load the CA certificate for the server into the client's cert DB. If you want the server to accept the client as genuine, you will need to load the CA certificate for the client into the server's cert DB.

Chapter 4

Developing a Communications Services Logical Architecture

This chapter describes how to develop your Communications Services logical architecture.

This chapter contains the following sections:

Communications Services Enterprise Deployment Logical Architectures Overview

You can deploy Communications Services in either a *single tier* or *two tier* logical architecture. Deciding on your logical architecture is crucial, as it determines which machine types you will need, as well as how many.

In general, enterprise corporate deployments tend towards the single tier, whereas ISP and Telco deployments are two tier. However, as with all generalities, the exceptions prove the rule. Small ISPs might just as well deploy on a single machine, and larger, centralized enterprises might deploy in a two-tier architecture for many of the same reasons that ISPs do. As more and more corporations look to offer ease of access to employees working remotely, their deployments will begin to look more and more like an ISP.

This section discuss the following Communications Services logical architectures:

- **Single Tier.** All services are either located on a single host sufficiently provisioned with memory and CPUs, or deployed on multiple servers, with each server hosting all of a particular component product's services.
- **Two Tier.** The access and data layers for the component products are separated onto different servers.
- **Edge.** Builds on the two-tier architecture by also providing for secure connections over the Internet to a mobile workforce.

Whether you deploy Communications Services in a single tier or multiple tiers, you need to understand the advantages and disadvantages of both models.

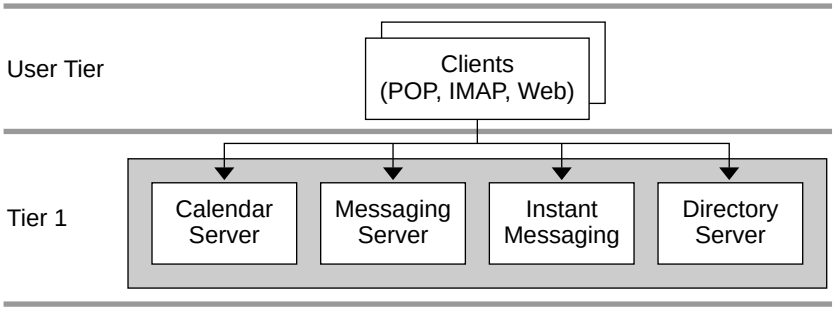
Single Tier, One Host Logical Architecture

As its name implies, the single tier, one host logical architecture locates all services onto a single machine. In general, such an architecture is best suited for enterprises that are:

- Composed of 500 users or less
- Not geographically distributed
- Served by few administrators
- Seeking an entry-level configuration

The following figure represents the single tier, one host logical architecture.

Figure 4-1 Single Tier, One Host Logical Architecture



End-user client programs, such as Outlook and Messenger Express, form the User Tier. Tier 1 is a single machine running all services, including messaging, calendar, instant messaging, and directory. The distinction of the single tier deployment is that end users communicate directly to the stores, and not through proxies or other agents.

The single tier, one host logical architecture requires a machine that provides sufficient CPU, memory, and storage. You should work with your Sun representative to determine the machine that best meets your enterprise's needs for this type of deployment.

When implementing the single tier, one host architecture, you can position the deployment for growth into a tiered architecture by assigning logical names to services. Such a configuration makes use of DNS mapping to direct users to the same front-end process (machine). If, in the future, you need to make changes to accommodate growth, such as splitting out services in a tiered fashion, users do not need to reconfigure their client applications. See "Using Logical Service Names" on page 73 for more information.

Single Tier, Multiple Hosts Logical Architecture

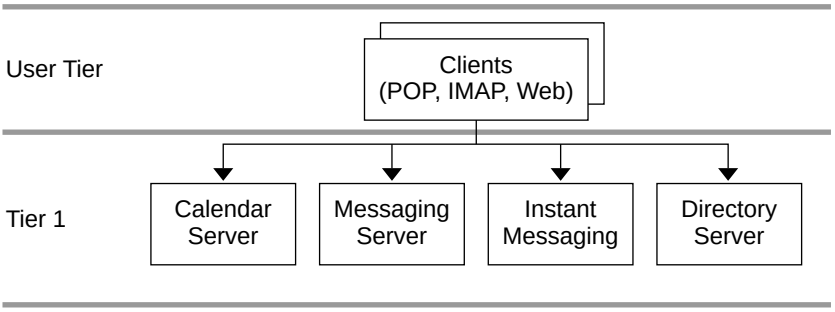
The single tier, multiple hosts logical architecture is a set of servers that each run the services particular to a component product. For example, the Messaging Server host is installed and configured to run all the Messaging Server services, the Calendar Server host is installed and configured to run all the Calendar Server services, and so on. This architecture might also be configured for high availability.

The distinction of the single tier logical architecture is that end users communicate directly to the data stores, and not through proxies or other agents. For example, in Messaging Server, users would not be routed through MMPs or MTAs. The single tier logical architecture might have standalone MTA routers for routing mail between servers, or in and out of the corporate network, but end users submit mail to the MTA on their message stores. No MMPs are involved in intranet connection to the message stores.

The same idea applies to both Calendar Server and Instant Messaging. In the single tier logical architecture, no front-end processes are located on separate machines.

Figure 4-2 on page 62 represents the one-tier, multiple hosts logical architecture for Communications Services.

Figure 4-2 Single Tier Multiple Hosts Logical Architecture



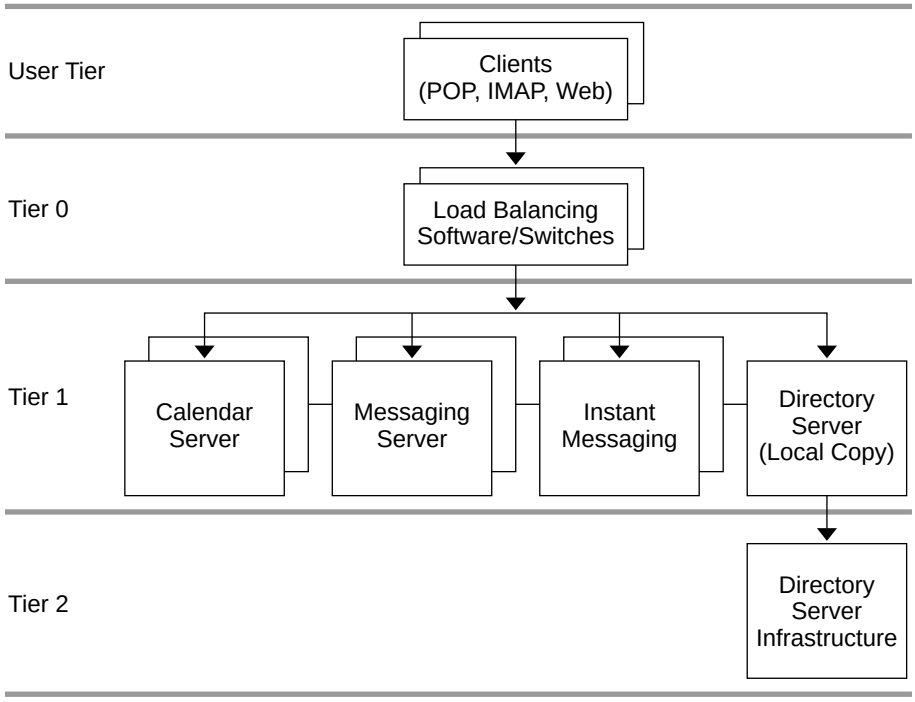
In the preceding figure, end-user client programs, such as Outlook and Messenger Express, form the User Tier. Tier 1 is a set of four servers. One server runs the Calendar Server processes, the second runs the Messaging Server processes, the third runs the Instant Messaging processes, and the fourth runs the Directory Server process.

Single Tier Distributed Logical Architecture

The single tier distributed logical architecture is a variant of the single tier in that the Directory Server is deployed in two tiers. Such a deployment works well for enterprises with small departments or organizations that are geographically distributed. Each department or office has its own services (mail, calendar, instant messaging) and a local directory instance (consumer). All the local directory instances are cached, but are synchronized with the centralized, master corporate repository. This is a fairly common scenario for offices with low bandwidth connectivity. The directory is architected in a two-tier fashion and replicated over the low-bandwidth to keep data local.

Figure 4-3 on page 63 represents the single tier distributed logical architecture for Communications Services.

Figure 4-3 Single Tier Distributed Logical Architecture



In the preceding figure, end-user client programs, such as Outlook and Messenger Express, form the User Tier. Tier 0 consists of load balancers that distribute load across the Tier 1 layer. Tier 1 is a set of multiple servers for the Communications Services processes. Multiple servers run the Calendar Server processes, multiple servers run the Messaging Server processes, and multiple servers run the Instant Messaging processes. Directory Server processes are split between a consumer server in Tier 1 running a local, replicated copy of the directory, and another server situated in Tier 2, which contains the master copy of the directory. Notice that in this kind of deployment, client queries are directed to the local directory copy, not to the master copy. Only the local Directory Server communicates to the master Directory Server.

NOTE When deploying a single tier with Internet connectivity, use a separate access layer. For example, you direct access to the data stores from inside the intranet without having to use SSL. However, you direct access to the data stores from the Internet through an access layer over SSL. This offloads much of the SSL load on the data stores to the access layer that separates it from the Internet.

The downside to this type of deployment is that users who make use of the server from a system that is sometimes on the corporate intranet and sometimes accessing the server from the Internet must configure their client applications to use SSL all the time. This is because it is too much trouble to switch back and forth. Therefore, there will still be a substantial percentage of SSL traffic being put on the stores directly. By using an access layer inside the intranet, you can remove that problem and by limiting connection directions further protect the intranet from illegal access.

Two Tier Logical Architecture

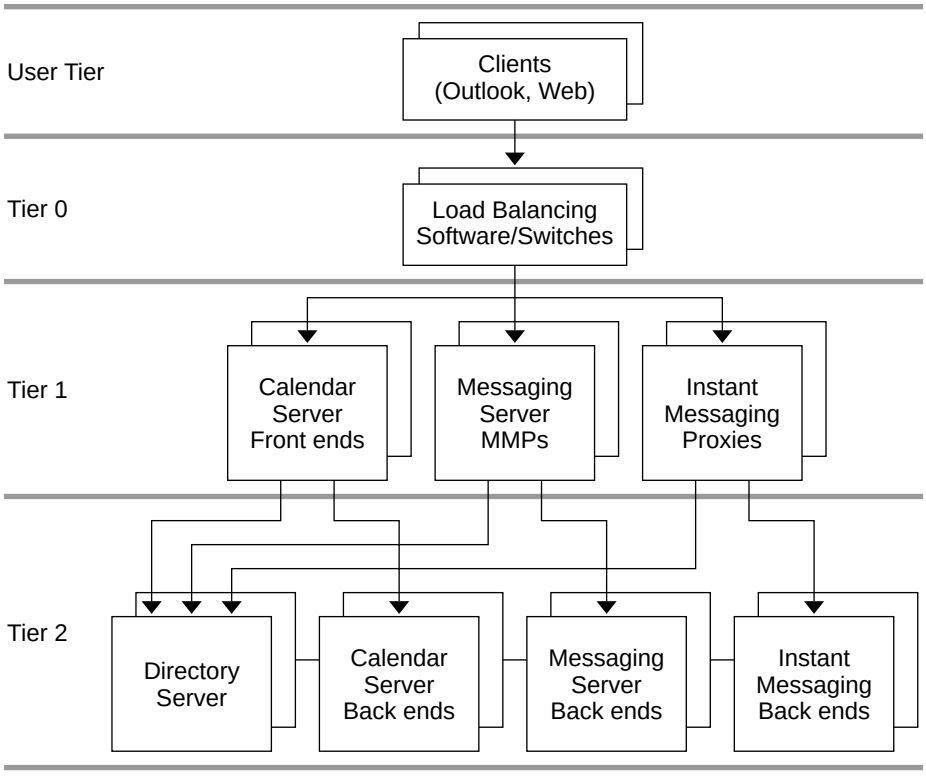
In a two tier logical architecture, the data stores communicate through front-end processes. In the case of Messaging Server, this means MMPs, MEMs, and MTAs are residing on separate machines from the data store processes. This enables the mail store to offload important and common tasks and focus on receiving and delivering mail. In the case of Calendar Server, this means the HTTP service and Administration service reside on a separate machine from the store processes. In the case of Instant Messaging, this means the proxy service is residing on a separate machine from the back-end processes.

There might be some level of cohabitation with other services. For example, you could have the Calendar store and the Message Store on the same machine. Similarly, you could have the calendar front end on the MMP machine.

In a two tier logical architecture, Directory Server is usually a complex deployment in its own right, with multi-master and replication to a set of load-balanced consumer directories.

Figure 4-4 on page 65 represents the two tier logical architecture for Communications Services.

Figure 4-4 Two Tier Logical Architecture



In the preceding figure, end-user client programs, such as Outlook and Messenger Express, form the User Tier. The load balancers form Tier 0. The Calendar Server, Messaging Server, and Instant Messaging front ends form Tier 1. Finally, the Directory Server, Calendar Server, Messaging Server, and Instant Messaging back ends form Tier 2. This architecture enables you to deploy Tier 1 and Tier 2 elements as separate instances, increasing overall flexibility of design. Additionally, you enhance system security by assigning discrete functions to individual instances.

For typical deployments, place the messaging and calendar front ends within the network Demilitarized Zone (DMZ), connecting to the main messaging and calendar services through a firewall. This configuration enables you to scale the system horizontally, as the Tier 1 elements can be scaled independently. Do not scale these elements beyond the capacity of the back-end servers.

When the front-end elements have reached the capacity of the back-end servers, you can scale the back-end Tier 2 elements to support more users. In general, the front end should scale as a function of the traffic. The back end should be scaled as a function of the number of users.

NOTE For specific instructions on sizing components in single-tier or two-tier architectures, contact your Professional Services representative.

Edge Logical Architecture

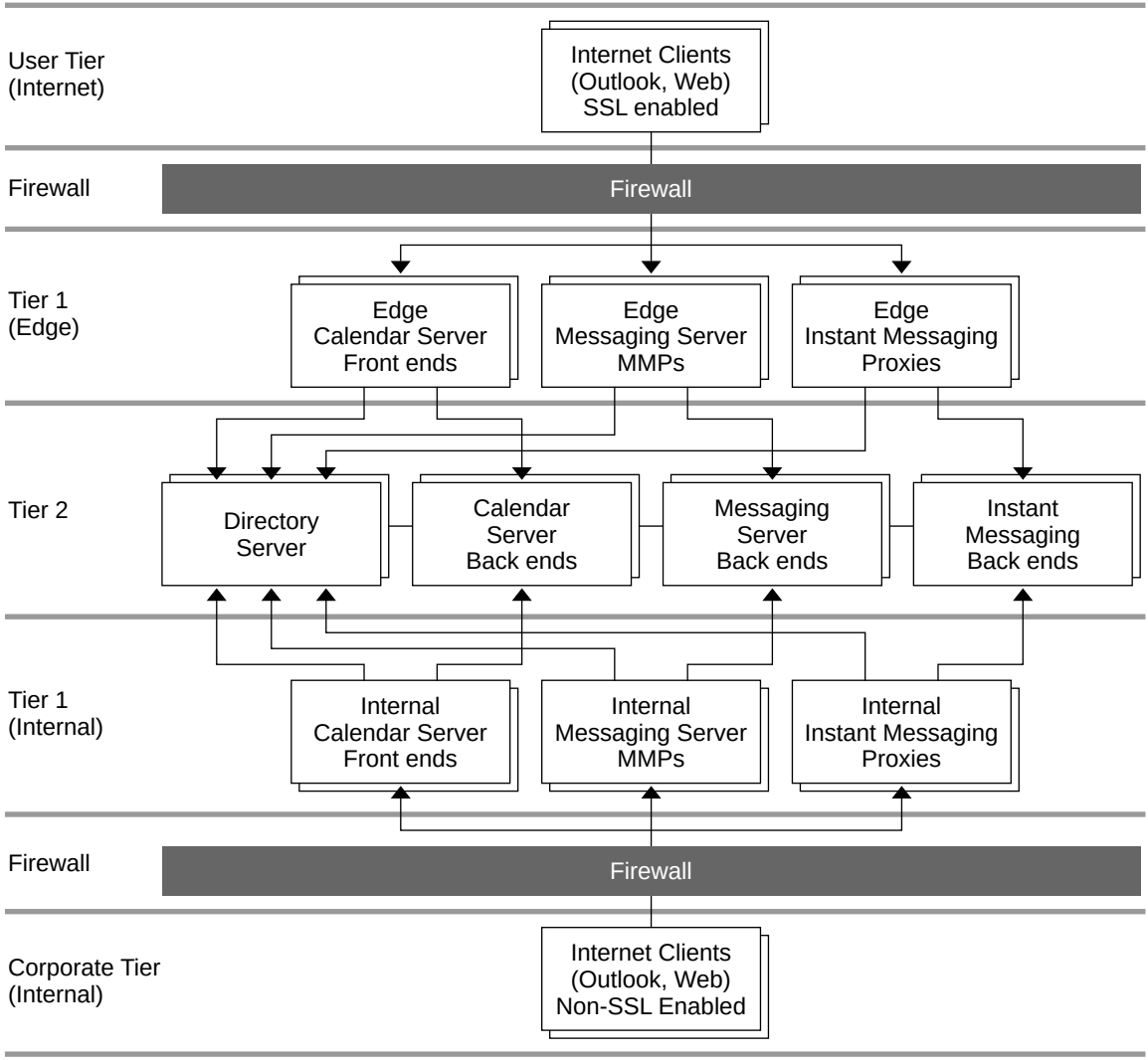
The edge logical architecture adds security for remote access to the two tier logical architecture. An edge deployment grants access to a remote, mobile workforce over the public Internet by using only name/password authentication (`SMTPAuth`). As messages travel to and from the corporate network over the public Internet, they are encrypted through the use of SSL. No virtual private network is involved. The internal side of the communications transmission is “in the clear” for maximum performance. Access is contained on the “edge” of the deployment, protecting the data stores from unauthorized intrusion.

Business reasons for an edge deployment include:

- Your workforce consists of mobile, remote workers
- You don’t want to install and maintain Communications Services servers at every remote site.

Figure 4-5 on page 67 represents the edge logical architecture for Communications Services.

Figure 4-5 Edge Logical Architecture



In the preceding figure, the data stores are located in Tier 2, which is a secure, private network, connected only to the “edge” and “internal” front-end servers. Remote clients connect to front-end servers by using SSL. Internal clients do not need to use SSL to connect, as the assumption is made that internal access is inherently secure.

Edge Architecture Design Recommendations

- Capacity planning for Edge tier is difficult to generalize. You should work with the hardware and software vendors who are supplying equipment for your deployment to develop a capacity plan. Nevertheless, you should implement the Realtime Blackhole List (RBL) at your site at the Edge tier. The RBL is a list of IP addresses whose owners refuse to stop the proliferation of spam.
- Design the Edge tier for minimal latency (less that one millisecond through the entire Edge tier).
- Use load balancing algorithms that are load-aware by CPU utilization or by the number of active connections. Round-robin is not an acceptable load-balancing model. With the exception of MTAs (stateless), use sticky bit load balancing.
- Webmail clients need load balancers that can manage sticky bits, because Webmail interfaces do not share state across Webmail servers.

Benefits of a Single Tier Architecture

The benefits of the single tier approach to your logical architecture come down to cost savings, as you do not have to purchase nor maintain additional hardware.

Answer the following questions to help decide if the single tier logical architecture is best for your enterprise:

- Is the threat of denial of service minimal?
- Is there no requirement for SSL?
- Do you have significant maintenance windows available?

Answering yes to these questions suggests that your enterprise could use a single tier logical architecture.

Benefits of a Two Tier Architecture

All services within the Communications Services offering rely on network capabilities. A two tier architecture provides for a network design with two separate networks: the public (user-facing) network, and the private (data center) network. The separation of these two networks is made possible by the Access-Layer Application hosts.

Separating your network into two tiers provides the following benefits:

- **Hides Internal Networks.** By separating the public (user-facing) network and the private (data center) network, you provide security by hiding the data center information. This information includes network information, such as IP addresses and host names, as well as user data, such as mailboxes and calendar information.
- **Provides Redundancy of Network Services.** By provisioning service access across multiple front-end machines, you create redundancy for the system. By adding redundant messaging front-end servers, you improve service uptime by balancing SMTP requests to the available messaging front-end hosts.
- **Limits Available Data on Access Layer Hosts.** Should the access layer hosts be compromised, the attackers cannot get to critical data from the access hosts.
- **Offloads Tasks to the Access Layer.** By enabling the access layer to take complete ownership of a number of tasks, the number of user mailboxes on a message store increases. This is useful because the costs of both purchase and maintenance are much higher for store servers than for access layer machines (the second tier). Access layer machines are usually smaller, do not require large amounts of disk (see "Message Transfer Agent (MTA) Considerations" on page 48), and are rarely backed up. A partial list of features that are offloaded by the second tier includes:
  - Denial of Service protection
  - SSL
  - Reverse DNS
  - UBE (spam) and virus scanning
  - Initial authentication - Authentications to the Message Store should always succeed and the directory servers are more likely to have cached the entry recently.
  - LMTP - With support for LMTP between the MTA relays and the message stores, SMTP processing is offloaded and the need to do an additional write of the message into the MTA queues on the message stores is eliminated.
- **Simplifies End-user Settings in Client Applications.** By using a two-tier architecture, end users do not have to remember the physical name of hosts that their messaging and calendar applications connect to. The Access-Layer Application hosts provide proxies to connect end users to their assigned messaging or calendar data center host. Services such as IMAP are connected

to the back-end service using LDAP information to identify the name of the user's mailbox host. For calendar services, the calendar front-end hosts provide a calendar lookup using the directory server to create a back-end connection to the user's assigned calendar store host.

This capability enables all end users to share the same host names for their client settings. For example, instead of remembering that their message store is `host-a`, the user simply uses the setting of `mail`. The MMP provides the proxy service to the user's assigned message store. You need to provide the DNS and load balancing settings to point all incoming connections for mail to one (or more) MMPs.

By placing Calendar Server into two tiers, more than one Calendar back-end server can be used. Calendar Server's group scheduling engine enables users to schedule appointments with users whose calendars are on any of the back-end Calendar hosts.

An additional benefit of this proxy capability provides geographically dispersed users to leverage the same client application settings regardless of their physical location. Should a user from Europe visit California, the user will be able to connect to the immediate access server in California. The user's LDAP information will tell the access server to create a separate connection on the user's behalf to the user's message store located in Europe.

Lastly, this enables you to run a large environment without having to configure user browsers differently, simplifying user support. You can move a user's mailbox from one mail store to another without contacting the user or changing the desktop.

- **Reduces Network HTTP Traffic on the Data Center.** Both the Messaging Express Multiplexor (MEM) and the calendar front-end greatly reduce HTTP traffic to the data center network. HTTP provides a connectionless service. For each HTML element, a separate HTTP request must be sent to the mail or calendar service. These requests can be for static data, such as an image, style sheets, JavaScript™ files, or HTML files. By placing these elements closer to the end user, you reduce network traffic on the back-end data center.

Horizontal Scalability Strategy

Scalability is critical to enterprises needing to make the most cost-effective use of their computing resources, handle peak workloads, and grow their infrastructure as rapidly as their business grows. Keep these points in mind:

- How the system responds to increasing workloads: what performance it provides, and as the workload increases, whether it crashes or enables performance to gracefully degrade.
- How easy it is to add processors, CPUs, storage, and I/O resources to a system or network to serve increasing demands from users.
- Whether the same environment can support applications as they grow from low-end systems to mid-range servers and mainframe-class systems.

When deployed in a two-tier architecture, the Communications Services offering is meant to scale very effectively in a horizontal manner. Each functional element can support increased load by adding additional machines to a given tier.

In practice, the method for scaling the front-end and back-end services differs slightly.

For Tier 1 elements, you start the scaling process when traffic to the front end grows beyond current capacity. You add relatively low cost machines to the tier and load balance across these machines. Thus, load balancers can precede each of the Tier 1 service functions as overall system load, service distribution, and scalability requirements dictate.

For Tier 2 elements, you start the scaling process when the back-end services have exceeded user or data capacity. As a general rule, design the Tier 2 services to accommodate just under double the load capacity of the Tier 1 services.

For example, for an architecture designed for 5,000 users, the Tier 1 front-end services are designed to support 5,000 users. The back-end services are then doubled, and designed to accommodate 10,000 users. If the system capacity exceeds 5,000 users, the front-end services can be horizontally scaled. If the overall capacity reaches 5,000 users, then the back-end services can be scaled to accommodate. Such design enables flexibility for growing enterprises, whether the growth is in terms of users or throughput.

Other Deployment Issues

This section describes some common Communications Services deployment best practices and other deployment considerations.

Implementing LMTP for Messaging Server

Best practices say you should implement LMTP to replace SMTP for message insertion. An LMTP architecture removes some of the demand for storage IOPs while providing a more robust and efficient solution that will increase system reliability and position your deployment for implementing new services.

The MTA's LMTP server is more efficient for delivering to the back-end Message Store because it:

- Reduces the load on the stores. Relays are horizontally scalable while stores are not, thus it is a good practice to make the relays perform as much of the processing as possible.
- Pulling the MTA queues from the stores could reduce IOPS by as much as 30 percent.
- Reduces the load on LDAP servers.
- The LDAP infrastructure is often the limiting factor in large messaging deployments.
- Reduces the number of message queues.

You need a two-tier architecture to implement LMTP. See the *Sun Java System Messaging Server Administration Guide* for instructions on configuring LMTP:

`http://docs.sun.com/doc/817-6266`

Implementing Realtime Blackhole List (RBL)

The Mail Abuse Protection System's Realtime Blackhole List (MAPS RBL) is a dynamically updated list of known UBE sources identified by source IP address. The Messaging Server SMTP server supports use of the RBL and can reflect mail coming from sources identified by the RBL as originators of unsolicited bulk email (UBE), or spam.

Implementing an RBL should be a consideration of every deployment. In general, a good RBL deployed in front of MTAs reduces traffic to the MTAs by a minimum of 10 percent, and in some cases, much higher.

RBL and anti-spam and anti-virus servers, such as BrightMail, can work together. For example, if your anti-spam server rejects 95 to 99 out of 100 emails from a particular IP address, you can add that IP address to the RBL. You can also adjust the RBL for BrightMail's false positives when you conduct your BrightMail analysis. Thus, you make the RBL much more proactive in handling a specific wave of UBE.

See the *Sun Java System Messaging Server Administration Reference* for information on configuring the `ENABLE_RBL` option of the MTA Dispatcher:

`http://docs.sun.com/doc/817-6267`

Using Logical Service Names

Design your deployment around the use of logical names for the Communications Services. You should use logical names even on a single-system deployment, to position it for ease of future growth and expansion. Using logical names does not impose any additional deployment setup costs other than populating your DNS.

You can think of these logical names as falling into two categories: those that affect end users, such as settings in email client programs; and those affecting back-end administration, such as inbound SMTP servers.

The following table describes these logical entities.

Table 4-1 User Facing Logical Names

Example	Description
`mail.siroe.com`	Name of the server from which end users collect their email.
`imap.siroe.com`	Name of the IMAP server from which end users collect their email.
`pop.siroe.com`	Name of the POP server from which the end users collect their email.
`smtp.siroe.com`	Name of the SMTP server users set as outgoing mail server.
`webcal.siroe.com` or `ce.siroe.com`	Name of the Communications Express (formerly Calendar Express) server.

Table 4-2 Maintenance Level Logical Names

Example	Description
relay-in.siroe.com	Corresponds to a bank of inbound SMTP servers.
relay-out.siroe.com	Corresponds to a bank of outbound SMTP servers.
mmp.siroe.com	Corresponds to a bank of MMP servers.
mem.siroe.com	Corresponds to a bank of MEM servers.
storeAA.siroe.com	Back-end message store. Select a naming scheme to work with your topology, for example, storeAA.siroe.com through storeZZ.siroe.com.
calstoreAA.siroe.com	Back-end calendar store. Select naming scheme to work with topology, for example, calstoreAA.siroe.com through calstoreZZ.siroe.com.

Table 4-3 Mapping of User Level to Maintenance Level Logical Names

Maintenance Level	User Level
relay-in.siroe.com	N/A
relay-out.siroe.com	smtp.siroe.com
mmp.siroe.com	Any one or more of mmp.siroe.com, pop.siroe.com, and imap.siroe.com
mem.siroe.com	webmail.siroe.com
storeAA.siroe.com - storeZZ.siroe.com	N/A, hidden from end users
calstore_aa.siroe.com - calstore_az.siroe.com	N/A, hidden from end users

Chapter 5

Designing for Service Availability

Once you have decided on your logical architecture, the next step is deciding what level of service availability is right for your site. The level of service availability you can expect is related to hardware chosen as well as the software infrastructure and maintenance practices you use. This chapter discusses several choices, their value, and their costs.

This chapter contains the following sections:

High Availability Solutions Overview

The Communications Services offering supports two different high availability solutions, Sun Cluster and Veritas Cluster Server (VCS). Messaging Server and Calendar Server provide agents for each of these solutions.

Messaging Server and Calendar Server support both asymmetric and symmetric HA. In asymmetric HA, you use a hot standby server. In symmetric HA, you run multiple instances of the software on a single server in the event of a failure. There are advantages to both solutions. However, in most cases, choose to run asymmetric HA. The following sections discuss the advantages and disadvantages of each solution.

See the *Sun Java System Messaging Server Deployment Planning Guide* for more information on HA deployments:

`http://docs.sun.com/doc/817-6440`

Symmetric HA

In symmetric HA, every server has a partner server that will take its entire processing load if the currently active server fails. Normally you deploy symmetric HA in pairs of servers, although this is not a requirement. However, when you deploy more than a pair of servers, the deployment becomes more complex to manage.

When deploying symmetric HA solutions, you need to size servers such that they are capable of running their own load as well as their partner's load. This does not necessarily mean that servers must be sized to handle 200 percent of typical peak load with fast response times. It does mean that the server must not fall over under 200 percent of typical peak load. If you deploy to just survive under 200 percent load, and there is a failure during peak hour, then you must understand that user response times will increase (possibly dramatically). Also, there is a distinct possibility that mail delivery into the store will backlog. To many enterprises (and ISPs), this is an acceptable risk.

Asymmetric HA

Asymmetric HA can be deployed as 1/1, meaning each live server has a corresponding hot standby. Again, it is not necessary that the failover server is as powerful as the main server, but using a smaller box means that if a failure occurs during peak hours you will experience increased service times and possibly backlogs of mail delivery. Asymmetric HA can also be deployed in N/1, N+1, or N+M modes. Sun Cluster is essentially limited to a total of four nodes in a standard HA arrangement, so N+M is impractical, but 3+1 is a reasonable and supportable solution.

Veritas supports a much larger number of nodes, so the limitations are driven by customer comfort and not by the software hard limits or supported limits.

Automatic System Reconfiguration (ASR)

In addition to evaluating an HA solution, you should consider deploying hardware that is capable of ASR.

ASR is a process by which hardware failure related downtime can be minimized. If a server is capable of ASR, it is possible that individual component failures in the hardware result in only minimal downtime.

As a general rule, the more ASR capabilities a server has, the more it costs. In the absence of high availability software, choose machines with a significant amount of hardware redundancy and ASR capability for your data stores, assuming that it is not cost prohibitive.

Various Sun SPARC servers support various levels of ASR. Refer to each product's data sheet to understand its ASR capabilities.

Using Enabling Techniques and Technologies

In addition to the high availability solutions discussed in the previous section, you can use enabling techniques and technologies to improve both availability and performance. These techniques and technologies include load balancers, Sun Java System Directory Proxy Server, and replica role promotion.

Using Load Balancers

You can use load balancers to ensure the functional availability of each tier in your architecture, providing high availability of the entire end-to-end system. Load balancers can be either a dedicated hardware appliance, or a strictly software solution.

Load balancing is the best way to avoid a single instance, server, or network as a single point of failure while at the same time improving the performance of the service. One of the primary goals of load balancing is to increase horizontal capacity of a service. For example, with a directory service, load balancers increase the aggregate number of simultaneous LDAP connections and LDAP operations per second that the directory service can handle.

Using Directory Proxy Server

Sun Java System Directory Proxy Server (formerly Sun ONE Directory Proxy Server) provides many proxy type features. One of these features is LDAP load balancing. Though Directory Proxy Server mighty not perform as well as dedicated load balancers, consider using it for failover, referral following, security, and mapping features.

See the Directory Proxy Server documentation for more information:

`http://docs.sun.com/coll/DirectoryProxyServer_04q2`

Using Replica Role Promotion

Directory Server includes a way of promoting and demoting the replica role of a directory instance. This feature enables you to promote a replica hub to a multi-master supplier or vice versa. You can also promote a consumer to the role of replica hub and vice versa. However, you cannot promote a consumer directly to a multi-master supplier or vice versa. In this case, the consumer must first become a replica hub and then it can be promoted from a hub to a multi-master replica. The same is true in the reverse direction.

Replica role promotion is useful in distributed deployments. Consider the case when you have six geographically dispersed sites.You would like to have a multi-master supplier at each site but are limited to only one per site for up to four sites. If you put at least one hub at each of the other two sites, you could promote them if one of the other multi-master suppliers is taken offline or decommissioned for some reason.

See the *Sun Java System Directory Server Administration Guide* for more information:

`http://docs.sun.com/doc/817-5221`

High Availability Solutions for Communications Services

Making the Directory Highly Available

From the Communications Services standpoint, the most important factor in planning your directory service is availability. As an infrastructure service, the directory must provide as near-continuous service as possible to the higher-level applications for authorization, access, email routing, and so forth.

A key feature of Directory Server that provides for high availability is replication. Replication is the mechanism that automatically copies directory data from one Directory Server to another. Replication enables you to provide a highly available directory service, and to geographically distribute your data. In practical terms, replication brings the following benefits:

- Fault tolerance / Failover
- Load balancing
- Higher performance and reduced response times

- Local data management

The following table shows how you can design your directory for availability.

Table 5-1 Designing Directory Server for High Availability

Method	Description
Single-master replication	A server acting as a supplier copies a master replica directly to one or more consumer servers. In this configuration, all directory modifications are made to the master replica stored on the supplier, and the consumers contain read-only copies of the data.
Two-way, multi-master replication	In a multi-master environment between two suppliers that share responsibility for the same data, you create two replication agreements. Supplier A and Supplier B each hold a master replica of the same data and there are two replication agreements governing the replication flow of this multi-master configuration.
Four-way multi-master	Provides a pair of Directory Server masters, usually in two separate data centers. This configuration uses four-way Multi-Master Replication (MMR) for replication. Thanks to its four-way master failover configuration, this fully-connected topology provides a highly-available solution that guarantees data integrity. When used with hubs in the replication topology, load distribution is facilitated, and the four consumers in each data center allow this topology to scale for read (lookup) operations.
Sun Cluster Agent for Directory Server	Using Sun Cluster software provides the highest level of availability for your directory implementation. In the case of failure of an active Directory Server node, Sun Cluster provides for transparent failover of services to a backup node. However, the administrative (and hardware) costs of installing, configuring, and maintaining this kind of environment are typically higher than the Directory Server replication methods.

See the *Sun Java System Directory Service Deployment Planning Guide* for more information.

`http://docs.sun.com/doc/817-5218`

Making Messaging Server and Calendar Server Highly Available

Both Messaging Server and Calendar Server can be configured to be highly available by using Sun Cluster and Veritas technology. Messaging Server and Calendar Server support asymmetric and symmetric configurations. In the asymmetric (“hot standby”) configuration, services run only on the primary node, and a standby secondary node remains idle. Detection of a fault in any of the resources (such as storage, host system, or process itself) causes the services to be stopped on the primary node and started on the secondary node. In the symmetric configuration, multiple nodes are concurrently running active services, and the nodes serve as backup for each other. Upon failover, the services on the failing node are shut down and restarted on a designated backup node. In this configuration, the backup node is also already running other active services.

In a tiered Communications Services architecture, where front-end and back-end components are distributed onto separate machines, you would want to make the back-end components highly available through cluster technology. This is because the back ends are the “stores” that maintain persistent data. You would want to make the Messaging Server MTA front end highly available by protecting its disk subsystems. It does not make sense to use cluster technology on the Calendar Server front end. Typically, you would make the Calendar Server front end more available through redundancy, that is, by deploying multiple front-end hosts.

For more information, see the *Sun Java System Messaging Server Deployment Planning Guide*:

http://docs.sun.com/doc/817-6440

Chapter 6

Communications Services Software Features

This chapter provides an overview of specific features of Sun Java System components that affect your deployment planning.

This chapter contains the following sections:

Communications Services Component Features

Communications Services consist of three core components:

- Messaging Server
- Calendar Server
- Instant Messaging

Communications Services components have dependencies on three additional components:

- Directory Server
- Identity Server
- A web server (Sun Java System Web Server can fulfill the need)

This section provides an architectural overview of the core components.

Messaging Server Software Overview

In general, a messaging service is an open standards-based client-server solution that meets the email needs of enterprises and messaging hosts of all sizes. A messaging service provides superior directory services, administration, scalability, performance, encryption, and remote connectivity.

In particular, a messaging service quickly delivers email with embedded sound, graphics, video files, HTML forms, Java applets, and desktop applications, while providing for future upgrade and scalability.

At a simplistic level, a messaging service:

- Accepts mail from external sites
- Determines the user mailbox to deliver that message to and route it accordingly
- Accepts mail from internal hosts
- Determines the destination system to deliver that message to and route it accordingly

In addition, a messaging service might provide Guaranteed Mail Delivery (GMD), anti-virus solutions, as well as spam control.

Of crucial importance to a good messaging service is a messaging server that follows the SHARP (scalability, high availability, reliability, and good performance) standard.

A messaging server is an electronic mail delivery system that supports email storage and delivery from one system to another system. In general, messaging servers, including Sun Java System Messaging Server, contain the following components:

- **Mail User Agent.** A client email program for end users to store, compose, and reply to messages.
- **Message Access/Transfer Protocols.** POP3, IMAP4, HTTP, and SMTP protocols. These protocols are further described in the sections on Message Access Protocols and Message Transfer Protocols.
- **Message Store.** A data store that holds mail until it is retrieved or deleted by an access server.
- **Message Transfer Agent (MTA).** Forwards messages from one mail server to another mail server. In addition, the MTA transfers messages to its own Message Store.

- **LDAP server.** A “lightweight” version of the Directory Access Protocol (DAP). While typically not a messaging server component, the LDAP server is integral for user maintenance and management. It usually serves the needs of a wide variety of applications in an organization.

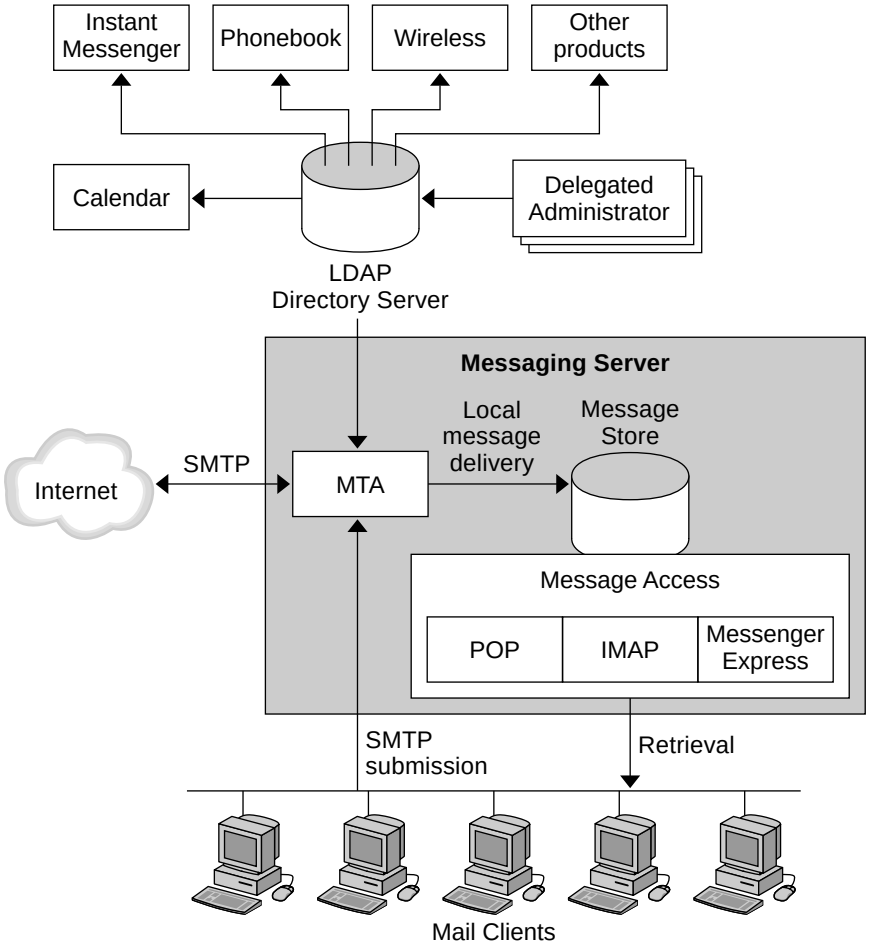
NOTE Since a proper Directory Server implementation is instrumental to a successful Messaging Server deployment, read the *Sun Java System Directory Server Deployment Planning Guide* in addition to this guide.

Messaging Server Architectural Overview

Sun Java System Messaging Server includes the Message Store, the MTA, and the server-side software that implements the Message Transfer and Message Access protocols. It also contains the HTTP message access client, Messenger Express. In addition, it is fully configurable with Directory Server software.

Figure 6-1 on page 84 shows the basic out-of-the-box Messaging Server architecture.

Figure 6-1 Messaging Server Basic Architecture



In the preceding figure, incoming messages from the Internet or local clients are received by the MTA via SMTP. If the address is within the Messaging Server domain, the MTA delivers the message to the Message Store. If the message is addressed to a domain outside of Messaging Server control, the MTA relays the message to another MTA on the Internet.

Although it is possible to deliver mail to an old UNIX `/var/mail` file system, local messages are typically delivered to a more optimized Messaging Server Message Store. These messages are then retrieved by any IMAP4 or POP3 client, or via the included Messenger Express interface.

The Directory Server stores and retrieves local user and group delivery information such as addresses, alternate mail addresses, and mail host information. When the MTA receives a message, it uses this address information to determine where and how the message should be delivered.

In addition to storing messages, the Message Store uses the Directory Server to verify user login name and passwords for mail clients that access mail. The Directory also stores information about quota limits, default message store type, and so on.

Outgoing messages from mail clients go directly to the MTA, which sends the message to the appropriate server on the Internet or, if the address is local, sends it to the Message Store.

New users and groups are created by adding user and group entries to the directory. Entries can be created or modified by using the Delegated Administrator, or by modifying the directory using LDAP.

Messaging Server components are administered by the Messaging Server Administration Console (not pictured in Figure 6-1 on page 84). In addition, a set of command-line interfaces and text-based configuration files are provided. Any machine connected to the Messaging Server host can perform administrative tasks (assuming, of course, the administrator knows the password). Some of the more common administrative tasks are adding, modifying, and removing users and groups from the mail system. In addition, an administrator might configure the operation of the MTA and Message Store.

Web-based Mail Client Service (HTTP) Through Messenger Express

The Messaging Express Multiplexor (MEM) provides a middle-tier for the Sun Java System Messenger Express mail client. This client enables users to access their mailbox data and to compose email messages through a web browser (WUA). The benefit of the MEM is that end users connect only to the MEM to access their email regardless of which back-end server is storing their mail. The MEM manages the HTTP session information and profiles the user via the users's LDAP information stored in Directory Server. The second benefit is that static files and LDAP authentication states are located on the Messaging Server front-end server. This benefit helps to offload the disk requirements on the Messaging Server store back end.

Where to Go For More Information on Messaging Server

See the following documentation for more information on Messaging Server:

http://docs.sun.com/db/coll/MessagingServer_04q2

Calendar Server Software Overview

Sun Java System Calendar Server is built from the ground up on standards. Its data store is based on iCalendar technology. It supports iMIP Publish, Request, Reply, and Cancel messages for events and tasks. Native support for iCalendar standards enables users to schedule events in a format that is easily shared across the Internet. Calendar Server ships with a full- featured Web client, Sun Java System Calendar Express, which is based on XML and eXtensible Stylesheet Language Transformations (XSLT) templates, and can be customized for a particular enterprise or ISP. Calendar Server also provides connectors to other popular calendar clients, thereby enabling a variety of clients and devices to communicate with Calendar Server.

In addition, Calendar Server provides an open protocol, known as the Web Calendar Access Protocol (WCAP), for developing clients that communicate with Calendar Server. WCAP is extremely useful in cases where calendar information must be provided to nonbrowser interfaces such as phones, PDAs, or other devices.

For information about WCAP commands, see the *Sun Java System Calendar Server Developer's Guide*:

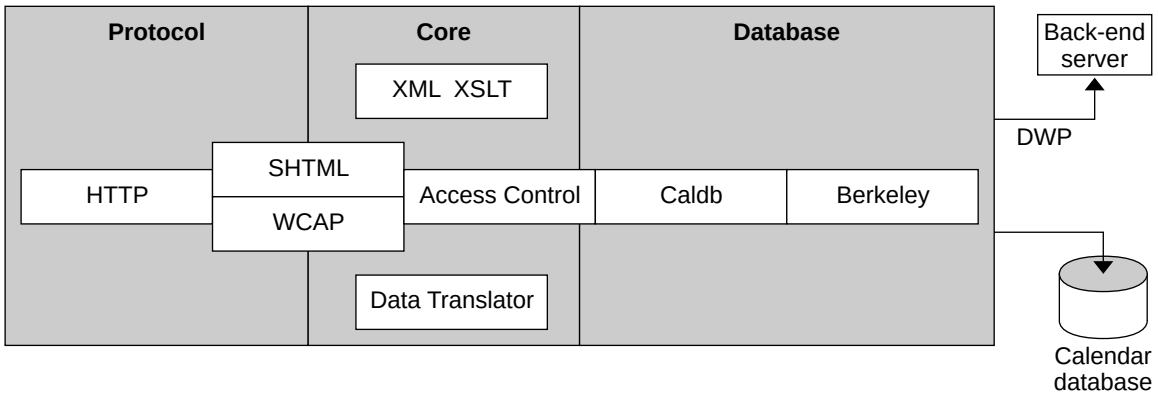
`http://docs.sun.com/doc/817-5698`

Built with flexibility in mind, Calendar Server is capable of running in LDAP-based or identity-based environments. The LDAP-based environment uses the LDAP directory for authentication and Calendar Server administration tools for provisioning. The identity- based deployment uses Identity Server for authentication and Identity Server administration tools for provisioning.

Calendar Server Architectural Overview

Figure 6-2 on page 87 illustrates the Calendar Server internal architecture.

Figure 6-2 Calendar Server Internal Subsystems Logical Flow



Calendar Server is built, in part, on proven components from Messaging Server and Directory Server. The design is highly modular. Calendar Server is implemented by way of a collection of shared libraries that fall into three main components:

- **Protocol Subsystem.** Commands and requests enter through the HTTP protocol layer. This is a minimal HTTP server implementation, streamlined to support calendar requests. Clients use SHTML or Web Calendar Access Protocol (WCAP) commands to submit requests.
- **Core Subsystem.** The Core subsystem includes the access control subsystem, user interface (UI) generator subsystem (either SHTML using XML and XSLT or WCAP using data translators), calendar database subsystem, and any CSAPI plugins. The Core subsystem processes calendar requests and generates the desired UI output. The Core subsystem also handles user authentication, including Calendar Server API (CSAPI) and Proxy Authentication SDK (authSDK).
- **Database Subsystem.** The Database subsystem uses the Berkeley DB from Sleepycat Software (the database API is not public). The Database subsystem stores and retrieves calendar data to and from the database, including events, todos (tasks), and alarms. Calendar data is based on iCalendar format, and the schema used for Calendar Server data is a superset of the iCalendar standard. The Database subsystem returns data in a low-level format, and the Core UI generator (either SHTML or WCAP) then translate the low-level data into the desired output.

The shared libraries of the protocol, core, and database subsystems are bound in various combinations to produce the executable daemons `cshttpd`, `csdwpd`, `csadmind`, and `csnotifyd`. In addition, the daemon `enpd` provides an event notification service that is shipped with Calendar Server.

Commands or requests enter the server through the Protocol subsystem and are passed to the Core subsystem for processing. Database accesses go through the Database subsystem.

Where to Go For More Information on Calendar Server

See the following documentation for more information on Calendar Server:

`http://docs.sun.com/db/coll/CalendarServer_04q2`

Instant Messaging Software Overview

An instant messaging service is an open standards-based client-server solution that meets the instant messaging needs of enterprises and hosts of all sizes. It provides superior administration, scalability, performance, security, and connectivity throughout the enterprise and across the Internet.

At a simplistic level, an instant messaging service:

- Accepts instant messages from external sites
- Determines the user to which the message should be delivered, and routes it accordingly
- Accepts instant messages from internal hosts
- Determines the destination system to which the message should be delivered, and routes it accordingly

In addition, an instant messaging service can provide real-time conferencing, news and calendar alerts, and for offline users, email message forwarding.

Of crucial importance to a good instant messaging service is that the service follows the SHARP (scalability, high availability, reliability, and good performance) standard.

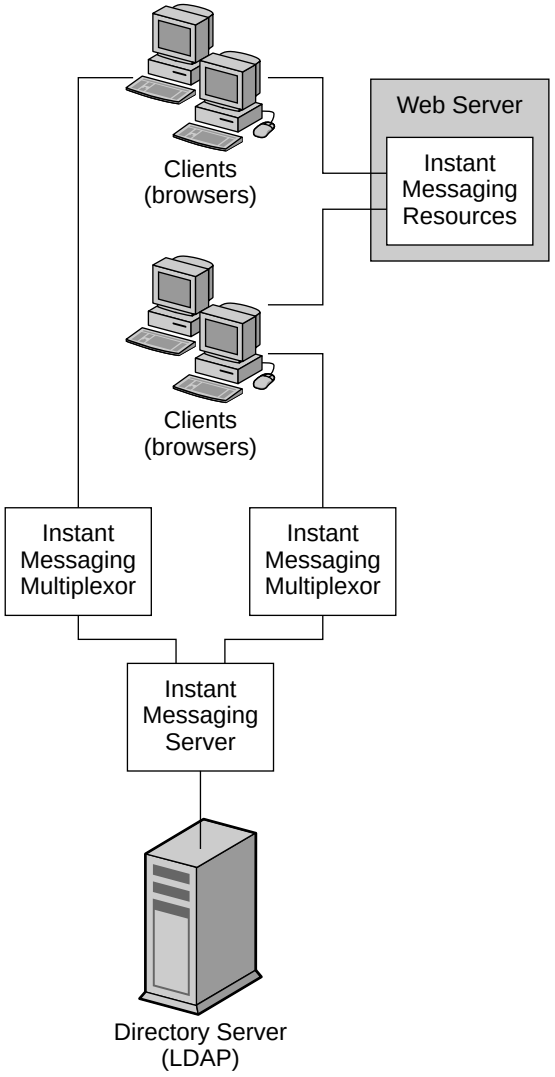
Instant Messaging contains the following core components:

- **Instant Messenger resources (client).** A set of files that make up the client program for end users to initiate, compose, and reply to messages. Typically users also use the client to participate in conferences. The client is also called Instant Messenger.
- **Instant Messaging server.** An electronic message delivery system that supports instant message delivery from one system to another system. The server serves the presence information to Instant Messenger clients, allows end users to establish sessions, and enforces policies.
- **Instant Messaging multiplexor.** A scalability component that consolidates messenger connections. In order to support large deployments, for example 40,000 concurrent connections, Instant Messaging uses a connection multiplexor to improve server scalability. This component opens a single connection to the Instant Messaging server. In addition to scalability, you can install the multiplexor outside the firewall while leaving the server inside the firewall in order to protect from unauthorized external access. The Instant Messaging multiplexor is also referred to as the multiplexor.
- **Access, Communication, and Transfer Protocols.** LDAP, HTTP, TCP/IP, and SMTP protocols.
- **Identity Server Instant Messaging Service Definition.** A service used by Identity Server and the Identity Server SDK to provide policy management and single sign-on capabilities for Instant Messaging.
- **Instant Messaging API.** Enables you to create custom Instant Messaging clients.
- **LDAP server.** A “lightweight” version of the Directory Access Protocol (DAP). While typically not an instant messaging server component, the LDAP server is integral for user maintenance and management. It usually serves the needs of a wide variety of applications in an organization.

Instant Messaging Architectural Overview

Figure 6-3 on page 90 shows the basic out-of-the-box Instant Messaging architecture.

Figure 6-3 Instant Messaging Basic Architecture



Web Server (or an application server with a Web service embedded) downloads the Instant Messaging resources via a browser to the clients. The resource files make up the client. Clients sends messages to one another through a multiplexor which forwards the messages on to the Instant Messaging server.

Directory Server stores and retrieves local user and group delivery information such as preferences, location, and to which multiplexor to route messages for this user. When the Instant Messaging server receives a message, it uses this information to determine where and how the message should be delivered. In addition, Directory Server can contain user information such as contact lists and subscriptions.

In this basic configuration, Instant Messaging directly accesses Directory Server to verify user login name and passwords for mail clients that use Instant Messaging.

Outgoing instant messages from clients go directly to the multiplexor. The multiplexor sends the message to the appropriate Instant Messaging server, which in turn forwards the message to another Instant Messaging server, or if the message is local, to the multiplexor with which the recipient is associated.

New users are created by adding user entries to the directory. Entries can be created or modified by modifying the directory using the tools provided with the Directory server.

Instant Messaging components are administered using a set of command-line interfaces and text-based configuration files. Any machine connected to the Instant Messaging host can perform administrative tasks (assuming, of course, the administrator has the required privileges).

Where to Go For More Information on Instant Messaging

See the following documentation for more information on Instant Messaging:

http://docs.sun.com/db/coll/InstantMessaging_04q2

Infrastructure Component Features

This section describes the Directory Server, Identity Server, and DNS infrastructure components upon which Communications Services components depend.

Directory Server Software Overview

A directory service is the collection of software and processes that store information about your enterprise, subscribers, or both. In the context of this documentation, a directory service consists of at least one Directory Server and one or more directory client programs. Client programs can access names, phone numbers, addresses, and other data stored in the directory, depending on the permissions that have been set.

Directory Server includes the directory itself, the server-side software that implements the LDAP protocol, and a graphical user interface that allows users to search and change entries in the directory. Other LDAP clients are also available, including the directory managers in the Server Console. In addition, you can purchase other LDAP client programs or write your own using the LDAP client SDK included with the Directory Server product.

Without adding other client programs, Directory Server can provide the foundation for an intranet or extranet. Every Sun Java System component server uses the directory as a central repository for shared server information, such as employee, customer, supplier, and partner data.

You can use Directory Server to manage extranet user-authentication, create access control, set up user preferences, and centralize user management. In hosted environments, partners, customers, and suppliers can manage their own areas of the directory, reducing administrative costs.

Directory Server consists of the following components:

- An LDAP server with a plug-in interface
- Administration Server
- Server Console to manage the servers
- Command-line tools for starting and stopping the server, importing and exporting data in the database, database reindexing, account inactivation and deactivation, LDIF merges, kernel tuning, and replication management
- An SNMP agent
- The Directory Services Markup Language (DSML)

Directory Server Architectural Overview

The Directory Server basic architecture consists of the following:

- **Server Front Ends Responsible for Network Communications.** The server front ends of Directory Server manage communications with directory client programs. The Directory Server functions as a daemon. Multiple client programs can communicate with the server using LDAP over TCP/IP or DSML over HTTP. The connection can be protected using Secure Socket Layer over Transport Layer Security (SSL/TLS), depending on whether the client negotiates the use of TLS for the connection.

- **Plugins for Server Functions.** Directory Server relies on plugins for such aspects as access control and replication. A plugin is a way to add functionality to the core server. For example, the `Uid Uniqueness` plugin can be used to ensure that values given to the user id (uid) attribute are unique in the suffix configured when installing the directory.
- **Basic Directory Tree Containing Server-related Data.** The directory tree, also known as a directory information tree or DIT, mirrors the tree model used by most file systems, with the tree's root, or first entry, appearing at the top of the hierarchy.

Where to Go For More Information on Directory Server

See the following documentation for more information on Directory Server.

`http://docs.sun.com/db/coll/DirectoryServer_04q2`

Identity Server Software Overview

Identity Server is an identity management solution designed to meet the needs of rapidly expanding enterprises. Identity Server enables you to get identities for your employees, your partners and suppliers into one online directory. Then it provides a means for establishing policies and permissions regarding who has access to which information in your enterprise. Identity Server is the key to all your data, your services, and who has access to what.

When an enterprise user or an external application tries to access content stored on a company's web server, the policy agent intercepts the request and directs it to Identity Server. Identity Server asks the user to present credentials such as a username and password. If the credentials match those stored in the central Directory Server, Identity Server verifies the user's identity. Next, Identity Server evaluates the policies associated with the user's identity, and then determines whether the user is allowed to view the requested information. Finally, Identity Server either grants or denies the user access to the information.

Identity Server consolidates four major features into a single product that can be viewed in a single administration console:

- Identity Administration
- Access Management
- Service Management
- Federation Management

Identity Server Architectural Overview

Identity Server uses a Java technology-based architecture for scalability, performance, and ease of development. It leverages industry standards including the following:

- HyperText Transfer Protocol (HTTP)
- eXtensible Markup Language (XML)
- Simple Object Access Protocol (SOAP)
- Security Assertions markup Language (SAML) specification

Where to Go For More Information on Identity Server

See the following documentation for more information on Identity Server:

`http://docs.sun.com/db/coll/IdentityServer_04q2`

DNS Overview

A high quality caching Domain Name System (DNS) server on the local network is a requirement for a production deployment of Communications Services.

DNS is an application-layer protocol that is part of the standard TCP/IP protocol suite. This protocol implements the DNS name service, which is the name service used on the Internet.

Though it supports the complex, world-wide hierarchy of computers on the Internet, the basic function of DNS is actually very simple: providing name-to-address resolution for TCP/IP-based networks. Name-to-address resolution, also referred to as “mapping,” is the process of finding the IP address of a computer in a database by using its host name as an index.

DNS provides two principal services. It performs name to address mapping (and also maps addresses to host names). It also helps mail delivery agents, such as `sendmail` and POP, deliver mail along the Internet.

To deliver mail across the Internet, DNS uses mail exchange records (MX records). Many organizations do not allow direct delivery of mail that comes across the Internet for hosts within the organization. Instead, they use a central mail host (or a set of mail hosts) to intercept incoming mail messages and route them to their recipients.

The mail exchange record identifies the mail host that services each machine in a domain. Therefore, a mail exchange record lists the DNS domain names of remote organizations and either the IP address or the host name of its corresponding mail host. Consider the following table.

DNS Domain	Mail Host
International.com	129.44.1.1
sales.example.com	SalesExampleMailer
eng.example.com	EngExampleMailer
siroe.com	SiroeMailer

When the mail agent receives a request to send mail to another domain, it parses the address of the recipient from right to left and looks for a match in the table.

If it receives a request to send mail to neverhome.sales.example.com, it first extracts the topmost label, com. It examines the mail exchange record to see if there is an entry for com. Since there is none, it continues parsing. It extracts the next label and looks for an entry for example.com. Because there is none, it continues looking. The next entry it looks for is sales.example.com. As you can see in the preceding table, the mail host for that domain is SalesExampleMailer. Because that is a host name, the mail agent asks DNS to resolve it. When DNS provides that mail host's IP address, the mail agent sends the message.

If, instead of the mail host name, the mail exchange record had specified an IP address, the mail agent would have sent the message directly to that address, since it would have needed no name resolution from DNS.

Chapter 7

Communications Services Deployment Example

This chapter contains a Communications Services enterprise deployment example. Depending on the features you want to implement in your deployment, you will need to install different sets of hosts and other networking infrastructure.

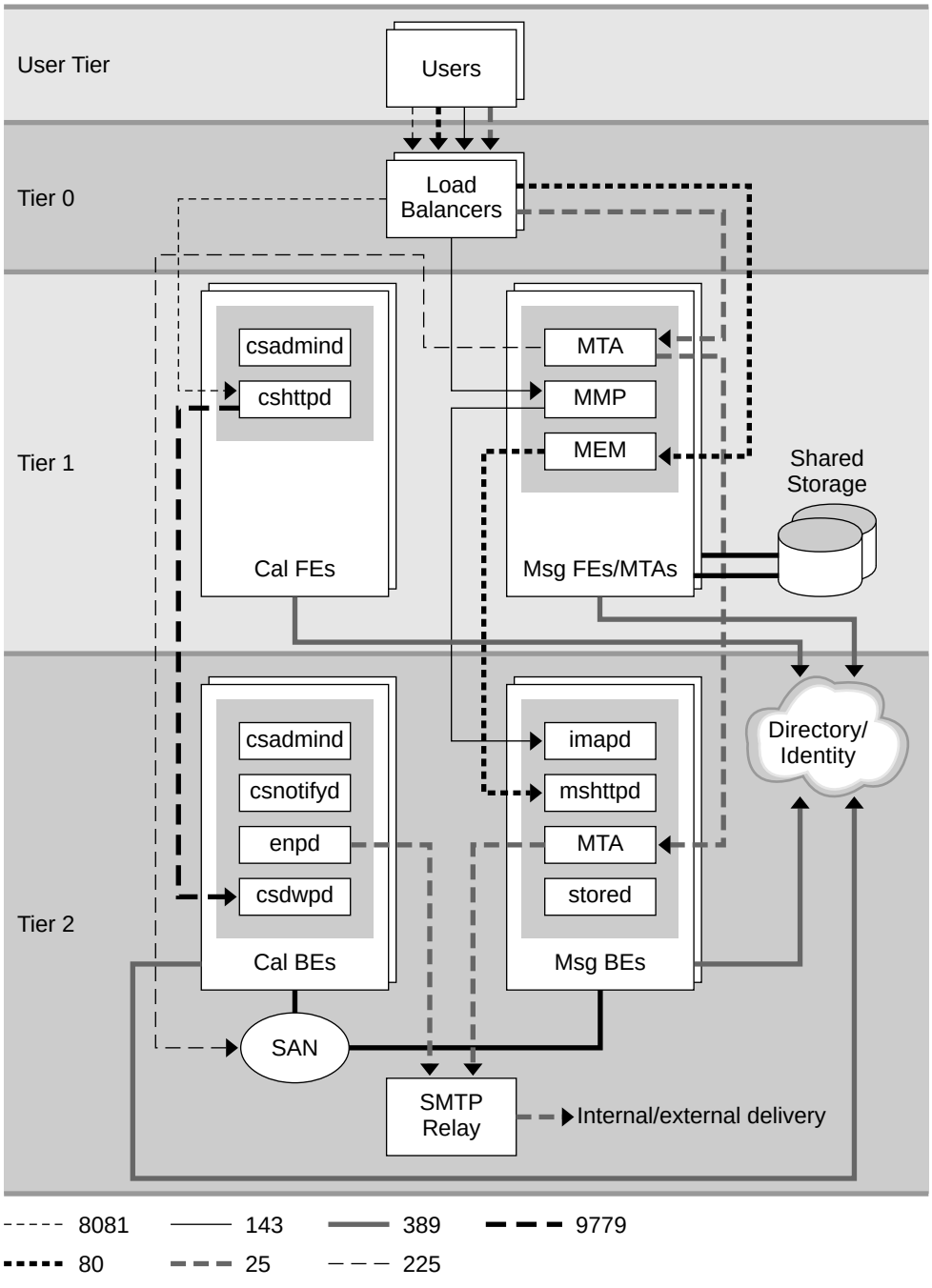
This chapter contains the following section:

Two Tier Deployment Example

Figure 7-1 on page 98 shows a two tier deployment for Messaging Server and Calendar Server. Tier 0 consists of load balancers. Tier 1 consists of Calendar Server and Messaging Server front ends. The Calendar Server and Messaging Server back-end stores form Tier 2.

Directory Server and Identity Server are a complex deployment on their own. This figure represents those components by a “cloud.”

Figure 7-1 Communications Services Two Tier Deployment Example



In the preceding example, load balancers form Tier 0, and direct user access to the front-end services.

The front-end services consist of four machines. Two machines are installed with Calendar Server front-end components. These Calendar Server front-end machines consist of one or two CPU servers and their own internal disk storage. Two other machines are configured as Messaging Server proxies and MTAs, and share an external disk array. These Messaging Server machines consist of four CPU servers.

The back end also consists of four machines. Two machines serve as mail stores and run the Messaging Server processes. Two other machines serve as the calendar stores and run Calendar Server process. The store machines are attached to a Storage Area Network (SAN). These back-end machines can be deployed in a variety of ways, based upon your CPU needs. Once you determine the total number of CPUs, you can opt for a vertical or horizontal configuration. For example, if your architecture called for a total of twelve CPUs, you could use three four-way servers, two six-way servers, or even one 12-way server.

Another machine serves as an SMTP relay for both Calendar Server notifications and Messaging Server emails.

The following table explains the protocols and port numbers used by this deployment.

Table 7-1 Protocols And Ports Used by Two Tier Deployment Example

Protocol	Port	Used By
HTTP	80	Clients to communicate with the Messaging Server MEM and Webmail (`httpd`) components
SMTP	25	Clients to communicate with the Messaging Server MTA component, MTA components on the front end and back end, and Calendar Server (`csendp`) components for email notifications
IMAP	143	Clients to communicate with the Messaging Server MMP and `imapd` components
LMTP	225	MTA routes email directly from the front end to the Message Store on the back end, bypassing the back-end MTA
LDAP	389	Front ends and back ends to communicate with LDAP directory
HTTP	8081	Clients to communicate with Calendar Front End (`cshttpd`)
DWP	9779	Calendar front end (`cshttpd`) to communicate with Calendar back end (`csdwpd`)

Glossary

Refer to the *Java Enterprise System Glossary* (http://docs.sun.com/doc/816-6873) for a complete list of terms that are used in this document.

Index

D

E

F

H

I

L

M

O

P

R

S

T

U

W