BEA WebLogic Personal Messaging API™

Administration Guide

Version 4.3
Document Revised: February 2006

Copyright

Restricted Rights Legend

Trademarks and Service Marks

Table of Contents

About This Guide

This guide is intended for administrators and software developers who want to leverage the WebLogic Personal Messaging API to integrate personal messaging and communication functionality into their enterprise applications and business process workflow.

This chapter contains the following sections:

Prerequisites

Before you install the WebLogic Personal Messaging API, perform the following steps:

- Install the Java Development Kit (JDK) 1.4 (or higher)
- Obtain a basic understanding of Microsoft Exchange, Lotus Domino, and Java
- Refer to the *BEA WebLogic Personal Messaging API Supported Configurations Guide* for information on supported versions of Microsoft Exchange and Lotus Domino

Documentation Conventions

The Windows convention of "\" as a path separator is used wherever necessary.

Also, since system software and configurations can vary from one system to another, portions of the command syntax displayed in this document may include sample parameters or variables that represent the actual command syntax you would need to enter. These entries are indicated by

parameters in uppercase placed between percent signs (`%PARAMETER%`), as shown in the following table.

Parameter	Definition
`%BEA_HOME%`	The complete directory specification for the product. For example, `d:\bea`.
`%BEA_WLPMAPI%`	The complete directory specification for the WebLogic Personal Messaging API. For example, `d:\bea\wlpmapi43`.
`%JDK_HOME%`	The complete directory specification for the Java Development Kit. For example, `d:\jdk1.4.2_09`.

CHAPTER 1

Introduction

The WebLogic Personal Messaging API is a single Application Programming Interface (API) for the integration of PIM/groupware into enterprise applications and business processes. The WebLogic Personal Messaging API offers a way to access groupware systems, such as Microsoft Exchange and Lotus Domino using a unified programming interface.The WebLogic Personal Messaging API is a collection of Java API libraries that can be integrated into Java-based applications, such as portals, web applications or standalone Java applications.

This chapter includes the following sections:

What's Included

The Personal Messaging API installation includes the files for the Java API libraries, plus other pieces to support deploying, developing, and working with the API. See Table 1-1 for a list of directories and the files.

Table 1-1 Personal Messaging Directories

Directory	Description
`%BEA_WLPMAPI%`	WebLogic Personal Messaging API files.
`%BEA_WLPMAPI%\docs`	Documentation, including JavaDoc.

Table 1-1 Personal Messaging Directories (Continued)

%BEA_WLPMAPI%\lib	The core Personal Messaging API product JAR file libraries.
%BEA_WLPMAPI%\scripts	ANT scripts.
%BEA_WLPMAPI%\domino_service	WebLogic Domino Service installer (wl_domino_service_setup-43.exe and setup.bin).
%BEA_WLPMAPI%\exchange_service	WebLogic Exchange Service installer (wl_exchange_service_setup-43.exe) and associated files.
%BEA_WLPMAPI%\example	Example code, including unit tests and Java source.

Java API

The WebLogic Personal Messaging API is segmented in different Java packages. The primary package is com.compoze.collab. See Table 1-2 for a list of Java API packages.

Table 1-2 Java API Packages

Package Name	Description
com.compoze.collab	The core package containing concepts and classes common to the entire API.
com.compoze.collab.messaging	The package containing functionality common to both groupware and messaging.
com.compoze.collab.groupware	The package containing classes common to all groupware providers.
com.compoze.collab.domino	The package containing classes for accessing groupware functionality in Lotus Domino.
com.compoze.collab.exchange	The package containing classes for accessing groupware functionality in Microsoft Exchange via MAPI or WebDAV.

The complete Java API Documentation (JavaDoc) can be found at `%BEA_WLPMAPI%\docs\api\index.html`.

CHAPTER 2

Architecture and Concepts

This chapter provides information on the architecture and concepts that are useful for understanding the high-level ideas and terminology used with the WebLogic Personal Messaging API.

This chapter contains the following sections:

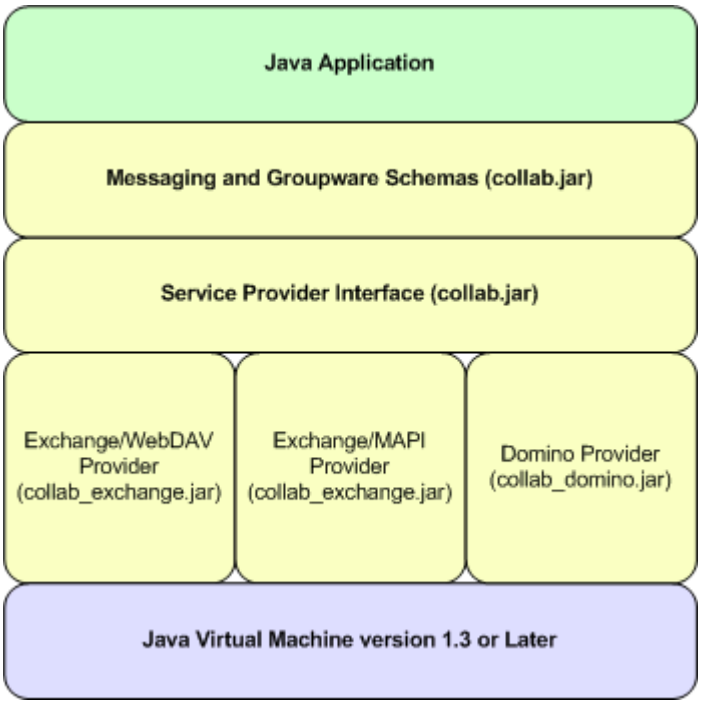
Software Components

Figure 2-1 shows the various components of the WebLogic Personal Messaging API and what JAR file the classes for these components are located in. When setting up the WebLogic Personal Messaging API to run within your Java application, the appropriate JAR files must be placed on the CLASSPATH as shown in the diagram. For example, for an application that requires connectivity to Exchange, the `collab.jar` and `collab_exchange.jar` files should both be in the CLASSPATH. In lieu of selecting multiple individual JAR files, there is a JAR file called `collab_all.jar` that contains all schemas, providers, and supporting classes.

Note: If the `collab_all.jar` file is used, no other WebLogic Personal Messaging API JAR files need to be placed on the `CLASSPATH`.

Note: Include the `license.bea` file in the `CLASSPATH`.

Figure 2-1 Personal Messaging API Software Components



Schemas

The schema is an insulating layer between the application and a specific provider implementation. For example, because an application uses the `groupware` schema, rather than an Exchange implementation directly, the same application can be used with Domino simply by switching provider implementations.

Your Java application uses a schema to access functionality and can leverage the `groupware` schema, `exchange` schema, or `domino` schemas—or some combination of all of them. The schema is backed by a provider implementation, which is chosen when the programmer creates

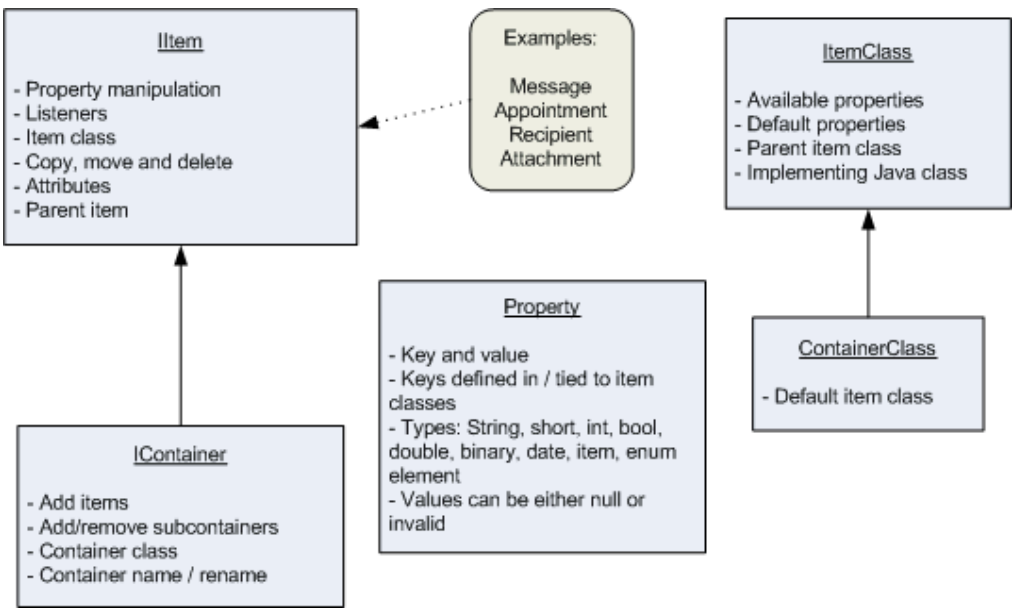
a `com.compoze.collab.ISession` using the API. For more information on providers, see "Providers" on page 2-4.

Schemas consist of a hierarchy of abstract items and containers. Each item contains one or more properties, which are key/value pairs. Containers hold both items and other containers. Root containers are obtained available from `ISession`, which is instantiated with a particular service provider. Specific item classes exist for the different entities in the schema. For example, the `groupware` schema contains item classes for tasks, appointments, contacts and mail messages. Regardless of what item class you are using, the programming interface for manipulating its properties it is the same.

The main schemas are the `groupware`, `exchange` and `domino` schemas. All inherit from the `messaging` schema. To say that a schema inherits from another schema means that it has all of the same item classes and container classes, plus additional ones. Providers may also contain their own schema extensions. For example, the `exchange` schema extends the `groupware` schema to add Exchange-specific groupware functionality. The `domino` schema extends the `groupware` schema to add Domino-specific groupware functionality.

Figure 2-2 shows several of the schema-related API classes and their functions.

Figure 2-2 Schema-Related API Classes



The packages and classes for the `groupware` and `messaging` schemas are found in the `collab.jar` file. Schema definitions are stored in the `collab.xml` file. For more information, see "The collab.xml File" on page 2-5.

Providers

A provider is an implementation of one or more schemas that makes that schema interact with a specific back-end system. For example, the Exchange/MAPI Provider is an implementation of the `groupware` schema that connects to a MS Exchange server.

Providers often implement extended schemas for access to the higher-level schemas that the extended schema implements. For example, the exchange schema contains additional item classes that are MS Exchange specific and thus not supported in the groupware schema. However, because the exchange schema extends the groupware schema, the Exchange/MAPI Provider still fully supports the groupware schema.

Table 2-1 lists the available providers. For more information, see "Exchange Providers" on page 3-1.

Table 2-1 Providers

Provider	Description
Domino Provider	Groupware system: Lotus Domino Implements schema: `domino` Extends schema: `groupware` Internal Name: `domino` For more information, see "Domino Provider" on page 3-8

Table 2-1 Providers (Continued)

Exchange/MAPI Provider	Groupware system: Microsoft Exchange Implements schema: `mapi` Extends schema: `exchange`, `groupware` Internal Name: `exchange_mapi` For more information, see "Exchange/MAPI Provider" on page 3-2.
Exchange/WebDAV Provider	Groupware system: MS Exchange Implements schema: `webdav` Extends schema: `exchange`, `groupware` Internal Name: `exchange_webdav` For more information, see "Exchange/WebDAV Provider" on page 3-5.

Note: There are multiple providers that support connectivity to MS Exchange, see "Exchange Providers" on page 3-1.

Each provider has an internal name, which is a string representation of that provider. When viewing the output log and the `collab.xml` file, you will encounter references to the provider by internal name.

The collab.xml File

Each WebLogic Personal Messaging API `JAR` file that contains providers and schemas contains a `collab.xml` file in the `META-INF` directory. The `collab.xml` file contains sections for schemas and providers depending on what the `JAR` file contains.

For the accurate information on what configuration information is stored in the `collab.xml` file, see the `collab.dtd` file in the `META-INF` directory of `collab.jar`. This Document Type Definition (DTD) is used for `collab.xml` validation and has comments that describe each of the nodes in a `collab.xml` file.

Note: Editing the `collab.xml` file is not required to run the WebLogic Personal Messaging API.

Determining the Version Number

The WebLogic Personal Messaging API version may be determined by executing any of the delivered `JAR` files, such as `collab.jar` or `collab_all.jar`. Alternatively, run the class `com.compoze.collab.version.Version`. The version of the product in the form `X.Y.Z` will be printed to `System.out` and displayed in a Swing dialog. For maximum compatibility, all versions of the `JAR` files should be the same or `collab_all.jar` should be used.

Minimal Java Security Policy

In production environments, the WebLogic Personal Messaging API can run in a virtual machine with the Java security manager enabled. The code in Listing 2-1 contains a section for `java.policy` (or similar) is the minimum needed to run.

Listing 2-1 Sample Code for java.policy

```
// WebDAV, MAPI Remote and Domino providers all need either or both of these,
depending on whether SSL is used or not
permission java.net.SocketPermission "*:80", "connect";
permission java.net.SocketPermission "*:443", "connect";
```

CHAPTER 3

Providers

The following providers are included with the WebLogic Personal Messaging API:

- Exchange/MAPI Provider
- Exchange/WebDAV Provider
- Domino Provider

This chapter contains the following sections:

Exchange Providers

The Personal Messaging API offers two separate providers for connectivity to MS Exchange. Your decision on which provider to use may be affected by the need to support MS Exchange 5.5. In addition, consider the performance, latency and functionality requirements for your application. With a few exceptions, the `exchange` schema can be used in the same way regardless of which provider you choose. See Table 3-1 for more detail on both Exchange Providers.

Table 3-1 Exchange Providers

Provider / MS Exchange	5.5	2000	2003	Comments
Exchange/MAPI Provider	X	X	X	Better overall performance vs. WebDAV Compatibility with 5.5 Support for task requests Less load on Exchange server vs. WebDAV Requires installation and configuration of an WebLogic Exchange Service
Exchange/WebDAV Provider		X	X	Less stringent network latency requirements Easier installation with no MAPI subsystem or WebLogic Exchange Service required Support for multiple public folder trees Requires installation of IIS and Outlook Web Access (OWA) on Exchange servers

Note: In some cases, the optimal way to decide which provider is better in your environment is to test both with your application and evaluate the IT requirements of your deployment against the results of that test.

This section contains the following topics:

- Exchange/MAPI Provider
- Exchange/WebDAV Provider
- Reporting Problems with Exchange Messages

Exchange/MAPI Provider

The Exchange/MAPI Provider offers an implementation of the `groupware` schema (and an extended `exchange` schema) for interacting with MS Exchange. The Exchange/MAPI Provider

uses the low-level MAPI interfaces to MS Exchange to provide groupware functionality. These low level interfaces offer the best potential performance when interacting with MS Exchange. Much of the complexity of Exchange groupware functionality is exposed in MAPI but is implemented in the Outlook client. The Exchange/MAPI Provider hides these details while making the Java application appear as though it were an Outlook client to the Exchange server.

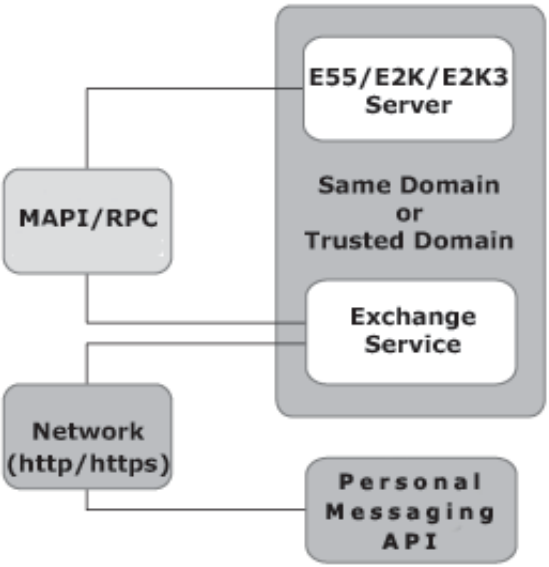
This section contains the following topics:

WebLogic Exchange Service

To use the Exchange/MAPI Provider, you must install and configure the WebLogic Exchange Service. The WebLogic Exchange Service provides connectivity to Exchange as an intermediary between Java and the MAPI subsystem.

Refer to the *BEA WebLogic Exchange Service Setup Guide* for more information on installing and configuring the WebLogic Exchange Service.

Figure 3-1 Configure the WebLogic Exchange Service to Use the Exchange/MAPI Provider



Connectivity to Exchange

The WebLogic Exchange Service is implemented using the MAPI subsytem to communicate with an Exchange server. Refer to the *BEA WebLogic Exchange Service Setup Guide* for more information on installing and configuring the WebLogic Exchange Service and the MAPI subsytem.

Network/Firewall Requirements

HTTP or HTTPS traffic must be able to pass between the Java application machine (where the API is running) and the WebLogic Exchange Service machine. This could be port 80, port 443, or whatever port you have chosen for your WebLogic Exchange Service. The network connection for the traffic between the application and service machine does not require as low a latency as the MSRPC connection. 50-100 ms ping times should be tolerable given reasonable bandwidth.

The network link between the Java application API and theWebLogic Exchange Service machine can be over wide area networks to traverse larger distances between your application and the Exchange server. However, the WebLogic Exchange Service machine should be as close to Exchange as possible. If you have Exchange servers in different locations, try to group them together and place the WebLogic Exchange Service machines as close as possible on the network to each cluster of Exchange servers.

Refer to the *BEA WebLogic Exchange Service Setup Guide* for more information on installing and configuring the WebLogic Exchange Service.

Sizing Information

A guideline is approximately 250 active users per machine running the WebLogic Exchange Service. Potentially more users than that could have sessions open on the machine, but a limit of around 250 in-use sessions exists for the process running MAPI. If more than 250 sessions are open, the Exchange/MAPI Provider closes and opens any underlying sessions to Exchange as needed, but performance will suffer if many of these opens and closes need to occur.

Additional processors on the WebLogic Exchange Service machine will minimize response time degradation as more simultaneous active users are added to the machine, but there will be a point of diminishing returns. Testing suggests that a dual processor machine is the best combination of scalability vs. cost when running the WebLogic Exchange Service. Using a faster machine for the WebLogic Exchange Service will simply allow the MAPI/RPCs with Exchange to occur faster and the WebLogic Exchange Service to run faster; however at a certain point you will be limited by the speed with which the Exchange server can respond to your MAPI/RPC requests.

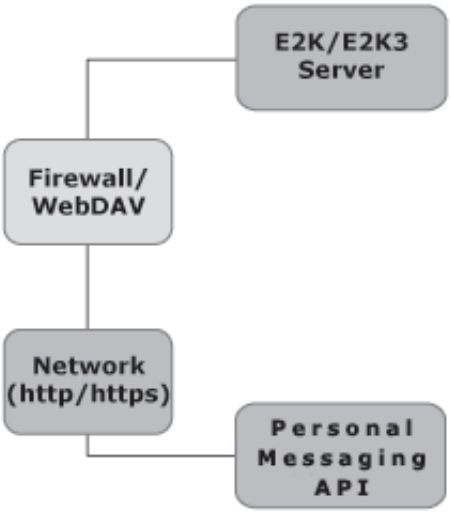
A recommended configuration for a fully loaded WebLogic Exchange Service machine is one GByte of RAM. Minimum configuration for low-load configurations is 128 MBytes of RAM.

Refer to the *BEA WebLogic Exchange Service Setup Guide* for more information on installing and configuring the WebLogic Exchange Service.

Exchange/WebDAV Provider

As shown in Figure 3-2, the Exchange/WebDAV Provider uses the WebDAV protocol directly against the Exchange server to provide groupware functionality. This allows the Java application and no MAPI subsystem installation or intermediary service machine, regardless of which operating system your applications runs on. Much of the complexity of Exchange groupware functionality is actually exposed in the published WebDAV protocol, but is implemented in the Outlook client. The Exchange/WebDAV Provider hides these details while interacting with Exchange using the same protocol as Outlook Web Access 2000 or 2003.

Figure 3-2 The Exchange/WebDAV Provider



Network/Firewall Requirements

The Java application connects directly to the OWA-enabled Exchange server using the HTTP or HTTPS protocol over any TCP port (typically the standard 80 or 443). The only firewall requirement is that traffic on whichever port is to be used may pass between the application and the Exchange server.

For more information on network requirements you may wish to see the Outlook Web Access (OWA) Planning Chapter available at: http://www.microsoft.com/technet/prodtechnol/exchange/2000/deploy/upgrademigrate/series/planningguide/p_10_tt1.mspx. Additionally, there are many other resources available on Microsoft's site regarding OWA deployment and scalability.

One important note regarding WebDAV performance and networking is that when connecting to Exchange 2000 you will experience a delay associated with TCP delayed acknowledgements that can hinder the performance of your application. Windows waits 200 milliseconds to acknowledge

the small TCP packets that come from Exchange 2000 server. This can be fixed by applying the workaround in the following Microsoft article:
http://support.microsoft.com/default.aspx?scid=kb;en-us;321098.

Sizing Information

A guideline is approximately 250 active WebDAV users on a CPU running the Java application (an active user defined not as a thread, but as a user using an application with typical user delays between each request). Unlike MAPI, there is no per-process limit. Adding additional CPUs to the hardware will increase the scalability of the application, with diminishing returns. A potential guideline is going from one to two CPUs increases the number of users that could be supported by about 75%. Going from two to four CPUs increased the two CPU number by about 50%. However, each application and environment may be different.

A fully loaded machine should need no more than one GByte of RAM to run the virtual machine with the WebDAV provider. Low load configurations can get away with 128 MBytes of RAM.

Reporting Problems with Exchange Messages

You might find that a particular message or folder is causing problems. In this case, it is possible to export the original messages for support to import in order to reproduce the problem.

Perform the following steps to export the messages:

1. Open Outlook 2000 or above to the account with the problem messages.
2. Choose **File > Import and Export**.
3. In the Import and Export Wizard screen, **choose Export to a file**.
4. In the Export to a File screen, choose **Personal Folder File (.pst)**.
5. In the Export Personal Folders screen, choose the folder that you wish to export.

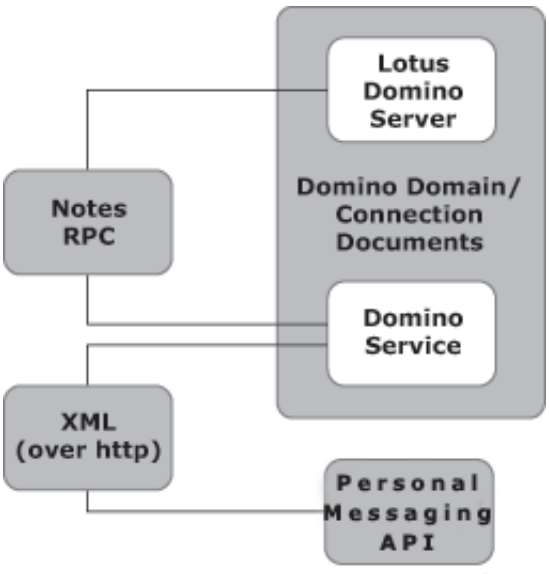
It is easiest to export an entire folder (such as the calendar folder), but you may also click **Filter** and restrict what is exported by date range, subject, attendees, created time, etc. Be sure that the offending messages get included by the filter you have chosen. Provide this file to support along with a small `readme.txt` file that explains the problem with the message/s and any filter that was used for the export in step 5 .

Domino Provider

The WebLogic Domino Service leverages native Domino Service to expose Lotus Domino groupware functionality from the Domino mail database. The WebLogic Domino Service machine acts as an intermediary between the Java API and Lotus Domino, as shown in Figure 3-3.

See the *BEA WebLogic Domino Service Setup Guide* for more information on installing and configuring the WebLogic Domino Service.

Figure 3-3 The WebLogic Domino Service Machine Acts as an Intermediary Between the Java API and Lotus Domino



This section contains the following topics:

- Connectivity to Domino
- Network/Firewall Requirements
- Sizing Information

Connectivity to Domino

The WebLogic Domino Service is implemented using a combination of Notes RPC and Notes DSAPI and runs within Lotus Domino as part of the HTTP task. The details of this do not need to be understood by the application programmer, but this is the reason the Lotus Domino install is a prerequisite of the installation. This is where Notes RPC and the Notes DSAPI filter for Domino are obtained.

Network/Firewall Requirements

The WebLogic Domino Service must be located on a machine running Lotus Domino that is part of the *same Notes Domain* as the Domino server to access. It is possible to put the WebLogic Domino Service on an existing Domino server, but be aware of the additional processor and memory burden that will be placed on Domino.

HTTP traffic must be able to pass between the Java client and the WebLogic Domino Service. Traversing an HTTP proxy is OK as long as it is able to pass the POST requests used by the XML protocol. Although a high bandwidth, low-latency connection will improve performance, the protocol has been designed to reduce the number of round trips made on the network. Therefore, packet round trip times of 50-100ms should be tolerable for the application. The amount of bandwidth required will depend on the number of users simultaneously using the application. Each user may consume roughly 1K/sec. on average, with this number increasingly dramatically if users do a lot of work with large file attachments.

Notes RPC traffic must be able to pass between the WebLogic Domino Service and Lotus Domino. Notes RPC requires TCP port 1352 to be open. The network connection for this Notes RPC traffic must have a low latency (less than 10 milliseconds, and preferably a 100 megabit LAN with less then one millisecond response times). Round trips are made over the network for each Notes RPC, therefore the WebLogic Domino Service machine must be located as close as possible to Domino on the network.

Sizing Information

A guideline is approximately 250 active Domino users on a CPU running the Java application (an active user defined not as a thread, but as a user using an application with typical user delays between each request). Adding additional CPUs to the hardware will increase the scalability of the application, with diminishing returns. A potential guideline is going from one to two CPUs increases the number of users that could be supported by about 75%. Going from two to four CPUs increased the two CPU number by about 50%. However, each application and environment may be different.

A fully loaded machine should need no more than one GByte of RAM to run the virtual machine with the Domino provider. Low load configurations can get away with 128 MBytes of RAM.