Deployment Guide

Sun™ ONE Meta-Directory

Version 5.1.1

817-3898-10
May 2004

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

Contents

Preface

The *Sun ONE Meta-Directory Deployment Guide* offers an introduction to Meta-Directory and describes how to plan and implement an Sun ONE Meta-Directory system.

This preface contains the following sections:

What You Are Expected to Know

This book is considered the “first” book in the documentation series provided with Sun ONE Meta-Directory. While it’s not essential that you understand directory technologies, you will get the most out of this guide if you are familiar with directory servers, databases, and Lightweight Directory Access Protocol (LDAP). Particularly, you should be familiar with planet Directory Server and the documentation provided with that product.

This guide is intended for use by IT professionals who want to join different directory repositories into one unified view. The functionality contained in Sun ONE Meta-Directory allows you to join the different directory information your organization maintains, thus making it easier to view and maintain the information you have spread across different directory and database applications.

Once you understand the concepts described in this guide, you will be ready to install, configure, and administer an Sun ONE Meta-Directory system, as described in the *Sun ONE Meta-Directory Installation Guide* and the *Sun ONE Meta-Directory Administration Guide*.

The Sun ONE Meta-Directory Documentation Set

The Sun ONE Meta-Directory documentation set contains the following titles:

- ***Sun ONE Meta-Directory Deployment Guide*** (this guide) describes Sun ONE Meta-Directory and details how to plan and implement a Meta-Directory system.
- *Sun ONE Meta-Directory Installation Guide* gives instructions on how to install the Sun ONE Meta-Directory software on both Solaris and Windows NT systems.
- *Sun ONE Product Brief* documents key concepts of the Sun ONE Meta-Directory.
- *Sun ONE Meta-Directory Administration Guide* documents how to configure and administer an Sun ONE Meta-Directory system once it has been installed. Configuring and administering Meta-Directory from both the Meta-Directory console and the command line are addressed.
- The *Release Notes* file gathers an assortment of information, including a description of what is new in this release, last minute installation notes, known problems and limitations, and how to report problems.

NOTE Be sure to check the Meta-Directory documentation web site for updates to the release notes and for revisions of guides. Updated documents will be marked with the revision date.

`http://docs.sun.com/prod/S1_MetaDir_511`

Organization of This Guide

Table 1 describes the organization of this book:

Table 1 Layout of This Manual

Chapter Title	Chapter Description
"Preface"	The chapter you are reading, it gives an outline of this guide and describes the Sun ONE Meta-Directory documentation set. It also provides sources for additional information related to Sun ONE Meta-Directory.
Chapter 1, "Meta-Directory Concepts"	This chapter provides a high-level overview that describes the need for Sun ONE Meta-Directory, the components provided with the software package, and the features provided by the components.
Chapter 2, "Planning the Meta-Directory System"	This chapter describes planning your Meta-Directory system. The three phases of planning are described: designing, deploying, and maintaining the system. This includes planning the topology of servers, the structuring of data, server load balancing, and so on.
Chapter 3, "Directory Server Configuration Settings for Meta-Directory"	This chapter discusses configuring the Sun ONE Meta-Directory system (as opposed to configuring the component instances which you instantiate, which is discussed in the *Sun ONE Meta-Directory Administration Guide*). Included in this chapter are the steps that need to be taken to configure Sun ONE Directory Server so that it can correctly support Meta-Directory, the schemas that need to be installed, enabling the Directory Server change log, and so on.
Chapter 4, "Meta-Directory Performance Tuning"	This chapter discusses some of the approaches you can take to increase the performance of Directory Server with regards to how it interacts with Sun ONE Meta-Directory.
Chapter 5, "Tuning Directory Server"	This chapter provides a guide to the important Sun ONE Directory Server settings a that can effect the performance of Sun ONE Meta-Directory.

Documentation Conventions

In the Sun ONE Meta-Directory documentation (such as this guide) there are certain typographic and terminology conventions used to simplify discussion and to help you better understand the material. These conventions are described below.

Typographic Conventions

This book uses the following typographic conventions:

- *Italic type* is used within text for book titles, new terminology, emphasis, and words used in the literal sense.
- `Monospace font` is used for sample code and code listings, API and language elements (such as function names and class names), filenames, pathnames, directory names, HTML tags, and any text that must be typed on the screen.
- *Italic serif font* is used within code and code fragments to indicate variable placeholders. For example, the following command uses *filename* as a variable placeholder for an argument to the gunzip command:

  ```
  gunzip -d filename .tar.gz
  ```

Terminology

Below is a list of the general terms that are used in the Sun ONE Meta-Directory documentation set:

- *Meta-Directory* refers to Sun ONE Meta-Directory and any installed instances of the Sun ONE Meta-Directory software.
- *Meta-Directory components* refers to the collective set of Sun ONE Meta-Directory components and software you have installed and running on your system, including the join engine and any connectors.
- E*xternal data source* refers to any user data that originates outside of the core Meta-Directory components, whether the data is coming from another database, directory server, data file, or other source of data.
- *Directory Server* refers to an installed instance of Sun ONE Directory Server, iPlanet Directory Server, or Netscape Directory Server.

  The term *Directory Server* refers to the instances of Sun ONE Directory Server, iPlanet Directory Server, and Netscape Directory Server that you have installed to work with Sun ONE Meta-Directory.
- Similarly the term *Administration Server* refers to an installed instance of Sun ONE Administration Server, whether it be used with the Meta-Directory components or another Sun ONE server.

- *NETSITE_ROOT* is a variable placeholder for the home directory where you have installed Sun ONE Meta-Directory and any other Sun ONE servers installed into the same server group.
- The term *flow* is used rather loosely to refer to the process of synchronizing data between an external data source and the Meta View. You “flow” data through a connector to the Connector View and then “flow” it to the Meta View. The contrary is also true; you “flow” data from the Meta View back to the Connector Views and out to the external data sources.

Related Third-Party Web Site References

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.

Where to Find Additional Information

In addition to Sun ONE Meta-Directory documentation set, you should be familiar with the documentation for products that are used in conjunction with it. Of particular interest are the Sun ONE Console and Sun ONE Directory Server documentation sets. This section lists additional sources of information you may find helpful as you use Sun ONE Meta-Directory.

Sun ONE Console Documentation

You can find the Sun ONE Console documentation at the following site:

`http://docs.sun.com/prod/s1.ipconsole.2`

Sun ONE Directory Server Documentation

You can find the Sun ONE Directory Server documentation at the following site:

`http://docs.sun.com/prod/s1dirsrv`

Directory Server Developer Information

In addition to the Directory Server documentation, you can find information on Meta-Directory, LDAP, the Sun ONE Directory Server, and associated technologies at the following Sun ONE developer sites:

http://www.sun.com/developers

Other Sun ONE Product Documentation

Documentation for all Sun ONE servers and technologies can be found at the following web site:

http://docs.sun.com/prod/sunone

Sun ONE Support

Other Useful Sun ONE information can be found at the following Internet locations:

Sun ONE Training
http://www.sun.com/training

Sun ONE Support information
http://www.sun.com/support

Sun ONE Product Information
http://www.sun.com/products

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 these Release Notes document is 817-3898-10.

Chapter 1

Meta-Directory Concepts

This chapter provides an overview of Meta-Directory and its components and describes why it is an essential application in today's directory environment.

The chapter contains the following top-level sections:

The Importance of Meta-Directory

A typical business enterprise today uses a myriad of directory system types. Often, the different systems collect information on the same entries (for example, information about a company employee), but each system will host different types of information for the entries. Think of the difference between the information contained in a Human Resource database as opposed to the employee information compiled by the Information Technologies (IT) department. As each of the systems gather data about a particular entity, there is an acute need for one product to integrate the various systems and link together their pertinent, authoritative information.

Meta-Directory joins dissimilar data repositories into a single LDAP database so you can effectively manage, modify, and search the information that is distributed among different directory and database systems. With Sun ONE Meta-Directory, you can create an "Enterprise LDAP Directory" that you can use as the central point for directory access and administration.

Historically, it has been extremely difficult to manage the data contained in differing data repositories due to dissimilar formats. Directory information stored in systems such as e-mail applications, human resources databases, various relational databases, and online directories makes it difficult to update and maintain the information for any one particular entity. The authoritative data source for a particular user attribute may exist in any one of many different user directories.

In a frequently repeated scenario, the Windows NT security accounts manager (SAM) stores account information for Windows users, the Network Information Server (NIS) stores user account information for UNIX users, a human resources database will likely store specific information for employees and contractors, the payroll database is maintained in SQL, the facilities database stores employee building information, and the PBX system is the definitive source for employee phone numbers.

The Sun ONE Meta-Directory Solution

Meta-Directory integrates the disparate directory systems within an enterprise so their joined information can be accessed and managed from a single location, the *Meta View*. The Meta View is contained in an LDAP directory that is hosted by an instance of Sun ONE Directory Server.

Meta-Directory creates an enterprise LDAP directory that contains a concentrated "view" of the external data repositories that it links. Since the Meta View is LDAPv3-compliant, the directory overcomes the limitations of system-specific directory name spaces. This integration provides functions and capabilities that give directory planners the flexibility they need to maintain and upgrade their existing enterprise directory structure.

The Meta View is a fully-functional Directory Server database. Using using the API built into LDAP, you can create applications to access the information stored in the Meta View. In fact, all features available in Sun ONE Directory Server (access control, data back-ups, and so on) are available in the Meta View.

The Meta-Directory's Meta View combines diverse details about a data entity into a single LDAP directory entry. This provides a centralized system of gathering information on an entity that may have different details scattered throughout multiple data repositories. Meta-Directory can copy data from one data repository to another data repository. During this process, Meta-Directory controls information flow, including what to copy, and where and when to copy it. You can update all of the records for a single entity either through the Meta-Directory Meta View or through the individual data repositories.

In addition to unifying systems, you can use Meta-Directory to restructure the architecture of your enterprise directory system by combining and integrating legacy systems into state-of-the-art LDAP directories. By integrating these systems, you have greater control, flexibility, and access to the information stored in each directory.

An Example Application

Suppose an organization has information for one of its employees, Micki Guzman, in a number of different data repositories (for example, a human resource database, an e-mail system, and IS directory system). Not only would there be a lot of duplicate information on Micki between the systems, but the job of managing the changes rests with multiple administrators across the organization. Changing the telephone number or job title of Micki Guzman would require administrators to change records in each system on which Micki is registered. On top of this, finding the authoritative value for one of Micki's attributes (an e-mail address, for example) can be time consuming and resource intensive.

With Meta-Directory, the details for Micki Guzman are stored in a single entry in the Meta View, which presents a single view of all her directory attributes. Using the Meta View, you can view and manage all of Micki Guzman's attribute values from a single location. Modified details are propagated back to each of the data repositories. Likewise, changing attribute values in one system will propagate, through Meta-Directory, to all connected systems.

Synchronizing and Joining Entries

Through the process of *synchronizing* and *joining*, Meta-Directory integrates and unifies directory information maintained in the different directory systems implemented across an enterprise. With Meta-Directory, directory entries are unified into a single, common LDAP directory.

Figure 1-1 illustrates the difficulty of accessing data that is stored in several different data repositories. The figure is a simplified look at a problem that faces many organizations. In this simple scenario, there are three databases that store different information about a singe object, a person in the company. In this case, if a user wants to get the phone number, the location, and the email address of an employee, they must search though an assortment of different database systems.

Figure 1-1 Manual Data Synchronization Without Meta-Directory

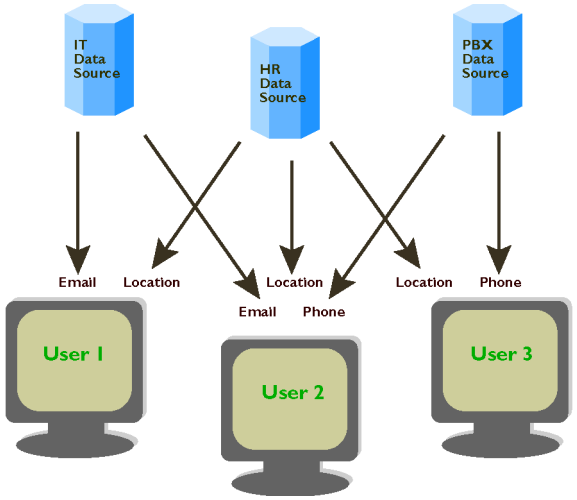
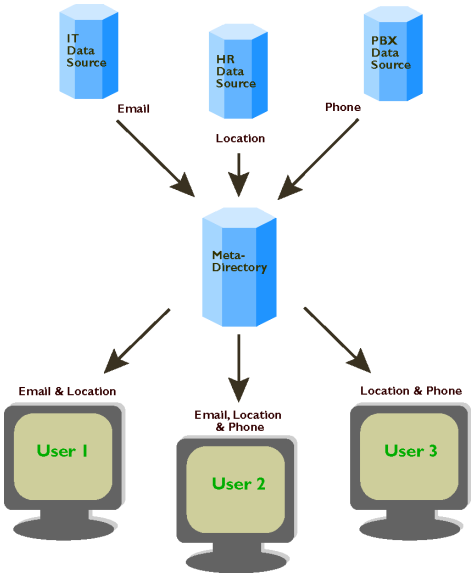


Figure 1-2 shows the highly automated data flow implemented in the Meta-Directory join process. Meta-Directory serves as the central repository and propagation facility for commonly updated user attributes. Here, users are able to query a single LDAP database (the Meta View) to obtain information associated with a user. With Meta-Directory, users are assured that the information they are retrieving is the “authoritative” information for that user. Without Meta-Directory, a user would have to query several different databases, possibly retrieving several pieces of conflicting information.

Figure 1-2 Automatic Data Synchronization Provided by Meta-Directory



Through the Meta-Directory *synchronization process*, a copy of the entries stored in an external database is provided in a LDAP format. Meta-Directory provides a "view" of the external data in what is known as a *Connector View*. The Connector View contains information from an external data source after it has been mapped into LDAP format.

The Meta-Directory *join process* completes the unification of synchronized user entries by joining the entries in the central LDAP repository (the *Meta View)*. The join process is managed by the Meta-Directory join engine which, through the join rules you write, synchronizes information from the authoritative data source for each of the joined attributes. The join engine is also able to propagate changes made in the Meta View back to the external data sources.

The join engine accomplishes the synchronization and joining by linking entries in the Meta View with those in the participating Connector Views.

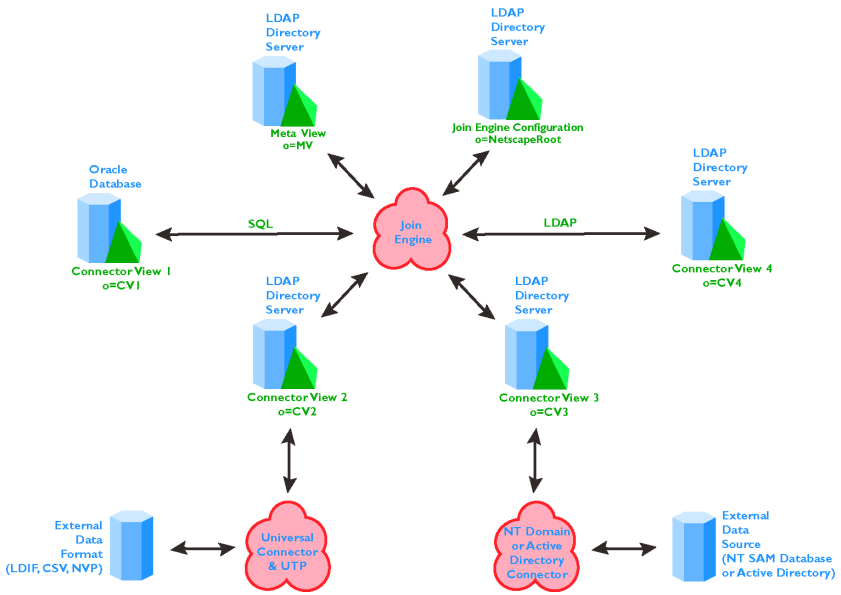
The specific information contained in the Meta View (the attribute values for each entry) is configured when you set up the join process. You control the information contained in the Meta View by specifying which attributes to accept or reject from each external data source. In addition, you configure which external data sources can accept changes made in (or propagated through) the Meta View.

A Meta-Directory Deployment

Successfully deployed, Meta-Directory provides a consistent view of all directory databases in the enterprise. As shown in Figure 1-3, Meta-Directory leverages the Sun ONE Directory Server, which it uses to store component configuration data and the Meta View itself. The Directory Server also provides user authentication, access control to directory data, and replication and referral services.

In this figure, the join engine and the Meta-Directory console configuration are stored in one directory information table (DIT), while the Meta View (the unified LDAP directory) is stored in another DIT. Connector views (LDAP copies of foreign directory data) may be combined in one directory instance or maintained in separate instances, depending on the directory requirements. All Meta-Directory components, including the Connector Views that participate in the Meta View, are administered as one server group through the Sun ONE Console.

Figure 1-3 The Meta-Directory Solution



The following sections explain in more detail how the Meta-Directory components interact and work with each other. The Meta-Directory configuration process (setting up the Meta View, the join engine, the directory connectors, and Connector Views) is fully explained in the *Sun ONE Meta-Directory Administration Guide*.

Sun ONE Meta-Directory Components

Sun ONE Meta-Directory works in conjunction with Sun ONE Console, Sun ONE Administration Server, and Sun ONE Directory Server. You use Sun ONE Console to manage the different Sun ONE server groups that you have installed in your Meta-Directory system. You use Sun ONE Administration Server to administer instances of Sun ONE Directory Server and any installed Meta-Directory components. You use Sun ONE Directory Server to provide support for the Meta-Directory configuration files and to provide the directory instance for the populated Meta View.

With the mixing and matching of these software products and components, it can be difficult to draw a distinct line where one product ends and another begins. However, simply put, Sun ONE Meta-Directory consists of the following components:

- The Meta-Directory console
- The join engine (and its associated Meta View)
- The supported connectors (and their associated Connector Views)

Sun ONE Meta-Directory Console

The *Meta-Directory console* provides a graphical user interface (GUI) that you use to configure Meta-Directory components and manage the flow of information to and from the Meta View and Connector Views.

The Meta-Directory console contains both left and right panes. The left pane displays a *navigation tree*, which uses icons to represent the components installed in your Meta-Directory system. When you select a particular Meta-Directory component in the navigation tree, an interface to a set of administration tasks displays in the right pane of the console. Refer to the *Sun ONE Meta-Directory Administration Guide* for detailed information on managing the Meta-Directory components through the Meta-Directory console.

Sun ONE Console and Sun ONE Directory Server Console

In addition to the Meta-Directory console, you will also be working with the Sun ONE Console and the Sun ONE Directory Server console.

The Sun ONE Console

The *Sun ONE Console* is used to administer the Sun ONE server groups that you have installed. A *server group* is an installed set of Sun ONE servers that are administered by a single Sun ONE Administration Server. It is likely that you will have more than a single Sun ONE server group, with each server group being installed into a separate directory structure. For example, if you install Sun ONE Directory Server into one directory structure and Sun ONE Meta-Directory into another, they will each be represented in the Sun ONE Console as different server groups.

An Sun ONE Administration Server is installed into each server group. Sun ONE Administration Server allows remote administration through the Sun ONE Console, including the ability to view and modify Meta-Directory component configurations, start and stop agents, and view component status. For more information on the Sun ONE Console, refer to the *Sun ONE Console and Administration Server 5.2 Server Management Guide*, which can be found at the following Sun ONE web site:

`http://docs.sun.com/db/prod/s1.ipconsole.2`

In addition to administering server groups, you also use the Sun ONE Console to access both the Meta-Directory console and the Directory Server console. From the Sun ONE console, double-click the Directory Server icon or an icon for a Meta-Directory component to display the associated product console.

The Directory Server Console

Before you install Meta-Directory, you must first install Sun ONE Directory Server. Directory Server has its own console, the *Sun ONE Directory Server console*. Centralized LDAP directory administration services—including access control, authentication, backup, and replication—are implemented using the Sun ONE Directory Server console. For more information about the Directory Server, the Directory Server console, and LDAP, refer to the *Sun ONE Directory Server Administrator's Guide* and the other documents located at the Sun ONE Directory Server web site:

`http://docs.sun.com/db/prod/s1dirsrv`

The Join Engine and the Meta View

The core service of Meta-Directory is the join engine, responsible for managing and controlling the flow of information into and out of the Meta View. Its job is to maintain the Meta View by synchronizing its information with the external data repositories.

The Join Engine

The join engine uses a configurable sequence of searches to automatically join (or link) entries in external data repositories with corresponding entries in the Meta View. These searches are called *rule sets*, which consist of ordered sets of rules that you configure. When information from an external data repository is propagated to the Connector View, the join engine "links" that information with the corresponding entry that exists in the Meta View, if it is configured to do so.

In addition to joining participating Connector Views into the Meta View, the join engine is also responsible for controlling the flow of information from the unified Meta View to external data sources.

Meta View Services

In addition to containing the joined data of the participating directories and databases, the Meta View provides the following centralized directory services:

- **Incremental updates**. External directories and databases can be updated incrementally as they are modified in the Meta View.
- **Object modification**. The Meta View can reflect modifications made to an entry in an external directory or database.
- **Object creation**. Objects in external directories and databases can be created from the Meta View.
- **Object deletion**. Objects can be deleted from the Meta View and from the external directory or database, depending on entity ownership.

Connectors and Connector Views

Connectors provide to the join engine the data contained in an external data source. The data is provided in the form of a physical directory, called the *Connector View*. Once a connector populates a Connector View, the join engine can flow the data to the Meta View. There are two types of connectors: indirect and direct.

Indirect Connectors

An *indirect connector* provides an LDAP-based copy of the entries that reside in an external data repository. The indirect connector presents the information in a corresponding *Connector View*. The join engine communicates with a Connector View over LDAP and consolidates this information into the Meta View.

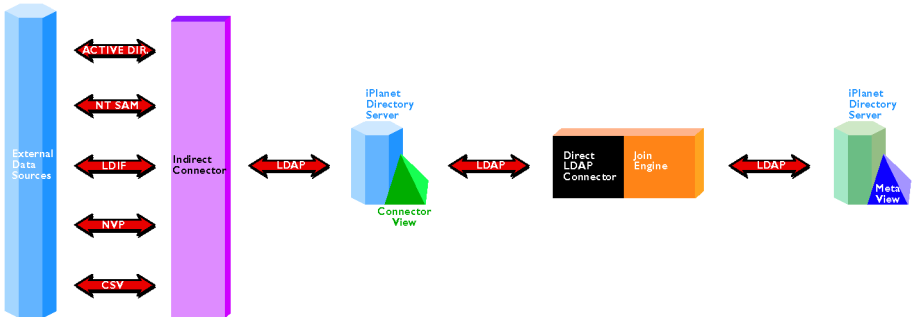
Using the Meta-Directory Universal connector, indirect connectors convert data contained in an external data repository into LDAP. The LDAP copy of this data is stored in a Connector View that is hosted by Sun ONE Directory Server. The join engine can then link the entries in the Connector View with the entries contained in the Meta View.

Meta-Directory 5.1.1 provides the following indirect connectors:

- The Universal Text Parser (UTP) is a text-file processor that parses and generates ASCII text files. Using the Universal Text Parser, you can build indirect connectors that interface with text-based data sources, such as an application that outputs a comma-separated value text file.
- The NT Domain connector provides bidirectional synchronization of user and group data from a Windows NT SAM database into a Connector View.
- The Active Directory connector provides bidirectional synchronization of user and group data from an Active Directory database into a Connector View.
- The Microsoft Exchange connector provides bidirectional synchronization of user and group data from a Microsoft Exchange database into a Connector View.
- The Novell Directory Connector provides bidirectional synchronization of user and group data between a Novell directory server and a Connector View in Sun ONE Directory Server.
- The Lotus Notes Connector provides bidirectional synchronization of user and group data from a Lotus Notes address book into a Connector View.

An indirect connector asynchronously copies entries between the external data repository and its corresponding indirect Connector View, as shown in Figure 1-4.

Figure 1-4 Indirect Connector Views



Directory Change Notification System

The *Directory Change Notification System* (DCNS) is a service provided by Meta-Directory as a shared library. The join engine and Universal Connector use DCNS to detect changes in indirect Connector Views and in the Meta View. The Meta-Directory components also uses this service to provide change notification to the connected data sources that can accept notifications.

DCNS relies on the change log files that are generated by the Directory Server instances that host your Meta-Directory views. Without the DCNS service, the join engine and your connectors would not be able to react to the changes made to any of the Meta-Directory views. If the change log is not enabled, the DCNS will be unable to function and the error message `mds_controller_unavailable` will be written to the Meta-Directory logs.

Direct Connectors

The join engine communicates directly with external data sources that format their data using the LDAP protocol (such as Sun ONE Directory Server). Because of this, the join engine forms a *direct connection* with applications that provide data in LDAP format. In addition to LDAP, the join engine also “speaks” SQL. With this, the join engine can also form a direct connection with SQL-based data sources.

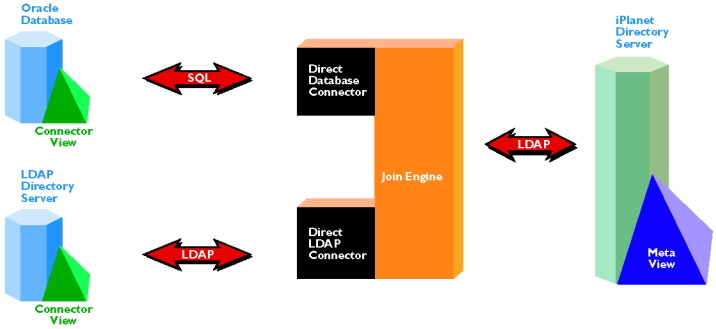
NOTE Meta-Directory 5.1.1 does not support all LDAP and SQL directories and databases. Sun ONE Meta-Directory 5.1.1 should support all versions of the Sun ONE Directory Server 5.x. However, for versions of the Sun ONE Directory Server 4.x the Connector View and not the Meta View is supported.

In addition, Meta-Directory can make SQL connections only through the Oracle Call Interface (OCI). Specifically, Meta-Directory supports connections to Oracle version 8.1.5 and 8.1.7 databases using the OCI programming interface.

The LDAP and SQL *direct connectors* are actually a part of the join engine. Because the join engine can directly connect to LDAP and SQL data repositories, Meta-Directory does not require a separate connector component to copy the data into an LDAP-based Connector View. Instead, the join engine provides a direct "view" of the data that resides in the external data repository. With the LDAP and SQL direct connections, the join engine is able to synchronously access entries in both the Meta View and the Connector View that resides on the external data source. Note: Oracle connector uses the o=netscaperoot to act as an object cache, so deployments using Oracle direct connector may consider special attention to the location of this directory instance.

The Meta-Directory direct connectors are shown in Figure 1-5.

Figure 1-5 Direct Connectors



How Meta-Directory Works

Two types of services integrate information from different data repository systems into and out of Meta-Directory:

- The connectors, which transfer entries to and from disparate external data repositories into the Meta-Directory views.
- The join engine, which unifies entries contained in Connector Views with those in the Meta View.

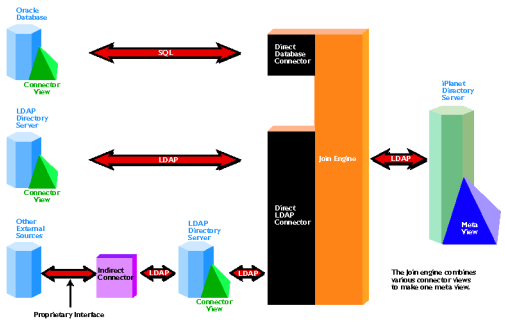
These services act upon data created and stored within two types of Meta-Directory views: the Meta View and the Connector Views.

How Connectors Work

While there are inherent differences between direct and indirect connectors, they both perform the same basic function: connectors allow the join engine to synchronize entries between the Meta View and the disparate Connector Views. With direct connectors, the associated Connector View is located on the external data source. On the contrary, indirect connectors must first flow data to an LDAP-based Connector View before the join engine can synchronize data.

Figure 1-6 shows the process of transferring information between external data repositories, the connectors, and the Meta View. Note that LDAP and SQL data repositories do not use a connector component to populate the associated Connector View.

Figure 1-6 Meta-Directory Component Overview



With direct connectors, the join process rules that you define are enough to synchronize data between the external data source and the Meta View. However, with indirect connectors, you need to configure a separate set of *indirect connector rules* to define what attributes should flow between the external data source and the indirect Connector View. Once the entries are in the Connector View, the join engine can link them to the Meta View.

Indirect Connector Rules

When you configure indirect connectors, you specify the entries and attributes that should flow between an external data source and its associated Connector View using indirect connector rules. *Indirect connector rules* consist of the following rule sets:

- **Attribute flow rules** specify which attributes should be synchronized between the external data source and the Connector View. With attribute flow rules, you specify attribute name mapping, attribute filtering, and modification propagation.
- **Default value rules** specify attribute values to use for a destination value when the value of the source attribute is empty or when the source attribute itself is not present.
- **Filtering rules** specify particular domains, containers, or user entries to include or exclude from the synchronization process.

However, these rules are not supported by the Novell Directory Connector and Lotus Notes Connector. It is recommended that the same features are achieved via the "Attribute Construction" and "Filters" features of the Join-Engine configuration when the Connector View corresponding to either a Novell Directory Connector or Lotus Notes Connector is enabled as a predicating view and configured wrt the Join-Engine.

Indirect connector rules are similar to the join process rules used by the join engine (which are described on “The Join Process Rules” on page 30). However, while the join engine uses its rule sets to determine what flows between the Meta View and the participating Connector Views, connectors use attribute flow rules to determine what should be synchronized between an external data source and the indirect Connector View.

Attribute Flow Rules

With attribute flow rules, you specify the actual attributes that you want to synchronize between the external data source and the Connector View. In addition, attribute flow rules specify how modifications flow between the views.

Attribute flow rules specify the following:

- **Attribute name mapping** specifies how attribute names are mapped between the external data source and the Connector View. You control both how the names are mapped from the external data source to the Connector View and how they are mapped from the Connector View back to the external data source.
- **Attribute filtering** specifies which attributes will flow from the external data source into the Connector View. Attribute filtering works one way in that Meta-Directory does not filter any attributes output by the Connector View to the external data source; all attributes contained in the Connector View will be flowed out to the external data source.
- **Modification propagation** specifies how modifications to attributes are handled. Depending on the attribute flow granularity, modifications to existing data can be restricted to the owner of the object or they can be allowed by either the external data source or the Connector View.

Attribute Name Mapping

Attribute mappings specify attribute names and they specify how attributes should be matched between two different data repositories. When you define attribute flow rules, you *map* the attributes from one data repository to another. For example, you might specify that an external database field `LastName` will map into the indirect Connector View attribute `sn`. If you are flowing data to an external data source, you can also specify how the attribute names will map from the Connector View to the external data source.

Because an indirect Connector View is an LDAP DIT, information stored in a Connector View must conform to a declared LDAP schema. Because of this, all attribute names must be conform with the schema that is loaded into that view. In contrast, an external database will likely contain data fields that do not conform to an LDAP schema. Attribute name mapping ensures that the attribute names are correctly named between the two data sources.

For example, Table 1-1 shows possible external database field names and the names that they could be mapped to in the Connector View, (the Connector View attribute names are part of the LDAP schema `inetOrgPerson`):

Table 1-1 Sample Name Mapping

External Database Field Name	Mapped Connector View Attribute Name
`LastName`	`sn`
`Firstname`	`givenname`

Table 1-1 Sample Name Mapping

External Database Field Name	Mapped Connector View Attribute Name
FullName	cn
Position	title
PhoneNumber	telephoneNumber

In this example, when flowing entries between an external data source and the Connector View, the attribute LastName in the external data source maps to the attribute name sn in the Meta-Directory Connector View. Once mapped into an LDAP schema, the attribute sn can be propagated to the Meta View, which is also stored in LDAP format.

Attribute Filtering

Attribute filtering controls the flow of object attributes from an external data source to its associated Connector View. When you map the attribute names from the external data source to the Connector View, you specify only those attributes whose values you want to flow to the Connector View. External attributes whose names you do not specify will not be flowed to the Connector View.

Attribute filtering is a one-way filter, it filters only the attributes that will be flowed to the Connector View from the external data source. If you are flowing data out of the Connector View back to the external data source, no filtering will take place; the Connector View will flow all the data that it contains. You control the filtering at this stage through the import function of your external data source.

Modification Propagation

Modification propagation specifies how modifications to existing data will be propagated between the Connector View and its associated external data source.

Modification propagation is determined by the *attribute flow granularity*, which is defined as follows:

- **Entry-level granularity** is the default. If you do not specify any attribute flow rules, Meta-Directory will use this level of granularity when if flows data that has been modified.
- **Attribute-level granularity** used if you define *any* attribute flow rules.

Meta-Directory uses the defined attribute flow granularity to determine how it will propagate entries that contain modified attribute values. If a Connector View has entry-level granularity defined, then Meta-Directory will only flow the entries whose data was modified by the source that owns the entry (Meta-Directory will

not flow a modified entry if the entry was modified by a data source other than the one that created it). However, if you have defined any attribute flow rules for the Connector View, then attribute-level granularity prevails and Meta-Directory will flow all modified entries between the Connector View and its associated external data source, regardless of who owns the entries.

The Novell Directory Connector and Lotus Notes Connector require the user to always select one of the attribute flow rules (the preset rules or user-created custom rules). Hence, there is no support for entry-level granularity.

Object Class Flow Rules

The Novell Directory Connector and Lotus Notes Connector use object class flow rules to specify the mapping between external data source object classes and the corresponding Connector View object classes. Novell Directory Connector and Lotus Notes Connector provide a single preset configuration for Object Class Flow:

- Object Class Set for Default Schema, that represents mappings for the default user and group object classes present in both Novell Directory Server or Lotus Notes and Sun ONE Directory Server (external data source and Connector View).

By default "Object Class Set for Default Schema" is selected as the "Object Class Flow Configuration". In addition to the preset object class flow configuration, you can also create new / custom object class flow rules manually. This allows you to flow entries belonging to any object class (not just those corresponding to user and group) in both directions.

For more details, see the sections - "Object Class Flow", "To add Object Classes for Novell Directory connectors", "To add Object Classes for Lotus Notes connectors" and "To Configure an Object Class Flow Rule" in the Sun ONE Meta-Directory Administration guide.

Default Value Rules

A *default attribute value* is a pre-configured value that Meta-Directory will use in a Connector View if an entry in the external data source has no value for the corresponding attribute, or if the attribute has no value.

When updating a Connector View or external data source, Meta-Directory updates all attributes or fields in the external directory that have associated values. Normally, if an field has no value in the external directory, it is not added to the Connector View, and vice versa. To make sure that the Connector View or external directory contains appropriate attributes and values, you can define a default attribute value to use with a particular attribute or field.

For example, if the `PhoneNumber` field in an external data source has no value, but you want the `telephoneNumber` attribute in the Connector View to contain a valid phone number, you can specify a default attribute value (such as the main office telephone number) to be used in the Connector View whenever the field is empty or missing in the external data source.

Filtering Rules

By default, all entries in both the Connector View and the external data source are included in the synchronization process. However, you can set up filters to designate that particular entries are to be included or excluded from the synchronization. With filtering rules, you have the control to specify which level of a directory tree you want to include (or exclude) from the synchronization process. Within that tree, you can then exclude (or include) specific groups of entries.

How the Join Engine Works

The Meta-Directory join process, which the join engine performs, is the key to unifying directory and database entries into the Meta View. The join process *links* entries that exist in each of the Connector Views with an entry in the Meta View. A link between an entry in the Meta View and the entries in one or more Connector Views allows attributes to flow between the views.

The outcome of the join process depends on the join engine applying the sets of *join process rules* that you define. After the join process occurs, entries between the views are either linked or not linked, depending on how well they match the criteria defined in your join process rules.

When creating the join process rules, you have a lot of flexibility in determining which entries become linked between the different views. In addition, many elements of the join engine configuration are shared between views, making it easy to administer the rules that are shared between Connector Views.

The Join Process Rules

When the join engine synchronizes one view with another, it uses a set configuration settings, or join process rules, to determine the correct join and data flow. The following join engine settings are configurable from the Meta-Directory console:

Join rules enable you to specify the rules for joining entries between a Connector View and the Meta View.

Attribute flow rules enable you to specify which attributes should be propagated between a Connector View and the Meta View.

Filters enable you to include or exclude entries from a Connector View by a subtree or a specific entry in a subtree. With filters, you specify groups of entries that you want to include or exclude from the Connector View.

Distinguished Name (DN) mapping rules let you specify where you will map entries between a Connector View and a Meta View. Using DN mapping rules, you create an RDN under a Connector View or Meta View which you can populate with data. You can also create a DN mapping rule to calculate the primary key of an Oracle table.

Constructed attributes enable you to create attributes in entries stored in the Meta View or Connector View by combining and manipulating entry information.

A general description of these items is given in the sections below; for complete information on how to configure these items, refer to the *Sun ONE Meta-Directory Administration Guide*.

Join Rules

Join rules specify which entries will be linked between the participating Connector Views and the Meta View. When Meta-Directory applies the join rules, the result will be a one-to-one correspondence between the entries that are linked the respective views. If a join rule is applied against a view, and the rule results in more than one candidate entry being selected from a view, no entries will be joined (links between views cannot be one-to-many).

Filters can be built to select an entry, or set of entries, from the opposing view. A *filter* is an LDAP search rule (or an SQL query) that when applied against a meta or Connector View, result in a set of entries being selected to flow to the opposing view.

Join rules are applied in sets. When creating your join rules, you will add individual rules in a particular sequence to form a join rule set. A *join rule set* specifies an ordered list of join rules that are sequentially tested until either an individual join rule identifies a single entry or until all rules fail.

Each rule in the join rule is fully evaluated, one rule at a time, until a rule returns one or more entry matches. If the evaluation of a join rule results in more than a single candidate match, then no entries will be joined by that rule. In addition, no entries will be joined if the evaluation of the entire join rule set fails to match any entry. In each of these cases, you can manually join specific entries using the Fix-It Tool. See “Manually Joining Entries” on page 34 for more information.

Attribute Flow Rules

Passing data from an external data source, through a connector to a Connector View, and to the Meta View is known as *flowing data*. You specify the authoritative attributes, those that flow to the Meta View, using *attribute flow rules*. Attribute flow rules enable you to specify the specific data source for each of the attributes represented in the Meta View. They also let you specify which external data repositories can be modified when an entry or attribute value changes.

Attribute flow defines the flow of attributes between Connector View and Meta View entries. An *attribute flow rule* lists the attributes that flow between two linked entries and provides pairs of source / target attribute names.

Multiple attribute flow rules may be grouped together into an *attribute flow set*. The join engine scans through the attribute flow set and selects the first attribute flow rule that matches the selection criteria. The join engine then applies all the attribute flow pairs from the selected rule.

Attribute flow sets provide ordered sets of rules that the join engine scans until it finds a match in the Connector View or Meta View it is searching. Each rule set is applied in order until a valid rule is found.

Types of Attribute Flow

The selection criteria for attribute flow is based on two types of attributes: entry attributes and context attributes. While *entry attributes* deal strictly with the source entry and the target entry, *context attributes* are involved with the context of the operation the join engine is performing. Context attributes are based on a selection of items from the following list:

- Ownership of entry
- Membership of entry
- Operation (add, update, or delete)
- Flow direction (Connector View to Meta View, or vice-versa)

Entry Ownership and Membership

The ownership and membership of entries are key concepts when working with the propagation of changes through Meta-Directory. The join engine updates entries based on entry ownership. Entry membership allows the join engine to work with sets of entries.

Entry ownership dictates the way the join engine decides which data source has ultimate control over an entry. Entry ownership is configurable from the Meta-Directory console—using the console, you specify who will own the entry when the entry is created in one of the Meta-Directory views.

The ability to delete an entry from a view (either the Meta View or a Connector View) depends on who "owns" the entry. If a view does not "own" an entry, and you delete the entry from that view, the entry will be added back to the view during the next synchronization process.

NOTE It's important to note that data contained in indirect Connector Views is an LDAP representation of the data contained in the corresponding external data source. In most cases, the owner of the data is the external data source. As such, you should always use the services of the external data repository to modify (add, delete, or modify) the objects that it owns; you should not modify the objects in the corresponding Connector View. Making sure that modifications are made by the owner of the data ensures that the modifications will properly flow through the Meta-Directory system.

In Meta-Directory, it is possible to have a Connector View that contains data from a variety of data sources. *Membership* identifies an entry within a Connector View that is native to the data source represented by that Connector View. Rules can then be configured and applied based on the attributes that are already present in the data source. Membership can be used to control the flow of attributes to and from a view. Membership can also be used when you create selection criteria rules. For example, you can specify that only members of a view can flow from that view to other views, or that a view will take modifications only if the object being modified is a member of that view.

Filters

Filters are used to explicitly include or exclude entries within specific directory subtrees during synchronization. Filters enable you to configure the Meta View or Connector View to exclude specific subtrees or entries from the unification process.

Distinguished Name (DN) Mapping Rules

Distinguished Name (DN) mapping rules identify where to create a new entry in the destination view (either the Meta View or a Connector View) given a source entry. The input into a DN mapping rule is the source entry, and the result is a partial DN, which the join engine concatenates with the view location DN to produce a fully qualified DN. The DN mapping rule actually creates an RDN for a specific entry, placing it in the appropriate place under the root of the associated Meta View or Connector View.

Multiple DN mapping rules may be grouped into a *DN Mapping Rule Set* to allow for an ordered testing of rules. These rules can be defined differently for entries originating from each different Connector View connected to Meta-Directory.

Constructed Attributes

The term *constructed attribute* refers to the broader scope of any additional attribute that is not explicitly part of a source entry, but that may be derived in a target entry given source data. A simple example would be as follows: if a user has an attribute "state equals New York," a constructed country attribute of "country equals USA" can be created.

The real functionality of constructed attributes comes into play when you use them in the creation of attribute flow settings. You can use constructed attributes in the join rules you write, giving you flexibility when flowing data between two data sources.

Manually Joining Entries

When you add, delete, or modify an entry in a Connector View, a search is performed on the Meta View for a matching user entry using the configured join rules. If a match is found, a link is made between the two views and any changes made to the entry in either view will be reflected in the other view (of course, depending on the attribute flow configurations).

For example, if an entry for the user David Lewis appears in the Human Resource's database Connector View and the join engine links the David Lewis entry in the Meta View, user information for David Lewis may be modified in the Meta View. Likewise, any Human Resource database attribute changes for David Lewis that are made in the Meta View will flow to the Human Resource database.

However, if while evaluating a join rule, the join engine returns more than a single candidate entry, the join will fail (joins can only exist between a single entry in the Meta View and an associated Connector View). For example, if a join rule links entries based on the sn attribute, and there happens to be two entries in the Connector View that have the same sn attribute, the join rule will return multiple candidate entries after processing the rule. Because of this, it is possible for an entry to be left out of the join process. Using the Query Fix-It Tool, you can manually link entries that your join rules miss.

It's good practice to examine the Meta-Directory logs after each synchronization cycle to ensure that no entries were left out of the synchronization cycle. If you find that the cycle missed an entry, you can use the Query Fix-It Tool. The Query Fix-It Tool enables you to manually control the joining process. With this tool, you can link, exclude, unlink, or create entries on an individual basis. For information on using the Query Fix-It Tool, refer to the *Sun ONE Meta-Directory Administration Guide.*

Chapter 2

Planning the Meta-Directory System

The Meta-Directory solution involves planning, installing, and configuring the Sun ONE Directory Server and Sun ONE Meta-Directory systems so they can produce the Meta View you have designed. In addition to these software components, you might also need to consider the installation and configuration of any additional services that your external directory data sources might need to operate.

Successful integration of Meta-Directory in the enterprise occurs in three phases: Design, Deployment, and Maintenance:

- In the **design phase**, you analyze your existing system and establish the architecture and requirements of the Meta-Directory system you will deploy. In this phase, you must take a careful look at your existing directory data sources and plan how to incorporate these into a single Meta View.
- In the **deployment phase**, you devise a systematic approach to the configuration of components and the implementation of servers and services in the enterprise. This phase deals with how to set up a Meta-Directory system, the different servers and services you need to install, and the topology of the hardware that you will be employing.
- In the **maintenance phase**, you provide a plan for maintaining the Meta-directory system once it has been deployed. This third and last phase deals with what needs to be done to keep your Meta-Directory system properly tuned and maintained.

This chapter contains the following sections:

Designing Your Meta-Directory System

Ideally, all of your directory data would be contained in one integrated Sun ONE Directory Server system, and the system would be spread evenly through the entire enterprise. We know, however, that such an ideal system is rarely in place. More often than not, many different types of directory systems are implemented across a large corporate network. Although the ideal of a single LDAP directory system is often out of reach, you can use Meta-Directory to join the heterogeneous types of directory systems running across your organization, and thus create a single LDAP directory of the information.

This section addresses the different tasks that you need to perform to design a robust Meta-Directory system. Included in this section are the following subsections:

Beginning the System Design

When reviewing the systems that are implemented across your organization, it's likely that you will see Sun ONE Directory Server already being used as part of the directory system mix. When you add Meta-Directory to the directory architecture, you will be adding at least another instance of Sun ONE Directory Server to the mixture of directory systems already in place. When you begin to address Meta-Directory system design, it's a good idea to separate the Directory Server instances that host organizational data and the Directory Server instances that host any of the Meta-Directory views. Its also advised to store configuration data c=netscaperoot in a separate directory server.

Before you can begin to design your Meta-Directory system, you must clearly define and design the Directory Server system(s) you will use to host your organization's user directories. To help with this task, the *Sun ONE Directory Server Deployment Guide* details the planning steps that you should follow to successfully deploy Directory Server in your organization.

Although the entire *Sun ONE Directory Server Deployment Guide* is devoted to designing and implementing your user directory system, most of the principles and concepts described in that guide are also valid for designing and deploying a Meta-Directory system (Meta-Directory is built on Sun ONE Directory Server). This guide assumes that you understand the concepts presented in the *Sun ONE Directory Server Deployment Guide* and that you are familiar with configuring and running Directory Server, as explained in the *Sun ONE Directory Server Administrator's Guide*.

The design of Meta-Directory begins with a site survey to gather information about the existing data sources and how you want to incorporate them into a single LDAP directory (the Meta View). The next steps are to determine the data flow (selecting the attributes you want to use from each database) and plan the physical design of the system.

When designing the system, keep in mind the impact the new system will have on existing directory structures.

Installing Directory Server to Support Meta-Directory

Meta-Directory stores its configuration data in a Directory Server directory information tree (DIT). Like other Sun ONE servers and components, Meta-Directory stores its configuration under the suffix `o=netscaperoot`.

When you install both Sun ONE Directory Server and Sun ONE Meta-Directory, the following components store their configuration data under `o=netscaperoot`:

- Sun ONE Console
- Sun ONE Directory Server
- Sun ONE Meta-Directory
- Sun ONE Administration Server

To optimize system performance, it is best to use a dedicated Directory Server instance for the Meta-Directory configuration database and a separate dedicated Directory Server instance for the Meta View (which contains the user data that you will be synchronizing with Meta-Directory). Often, the Directory Server instance that hosts the Meta-Directory configuration is called the "config server."

NOTE If you are synchronizing data contained in an Oracle database, Meta-Directory will use `o=netscaperoot` to store link information generated for the Oracle entries. In this case, it is possible that `o=netscaperoot` will house a large number of entries.

In addition to using separate Directory Server instances, it is also recommended that you use a dedicated disk partition to store the directory information tree that hosts the Meta View. In production scenarios where you plan to synchronize large amounts of user data in the Meta View, using dedicated systems for the individual Directory Server instances can greatly increase system performance. For more information on setting up your Meta-Directory components and services, refer to "System Topology" on page 50.

Directory Server Design and Deployment

Detailed information describing the design of LDAP directory services abounds. In addition to the Sun ONE Directory Server documentation, refer to the following references for more information on deploying Directory Server in your organization:

- *Understanding and Deploying LDAP Directory Services* by Mark Smith, Tim Howes, and Gordon Good (Macmillan Technical Publishing, January 1999)
- *Implementing LDAP*, by Mark Wilcox (Wrox Press, March, 1999)

Performing a Site Survey

The first stage of planning your Meta-Directory system is to perform a site survey.

Determining Data Sources and Authoritative Attributes

In the *site survey*, you will, among other things, determine the following:

- The data sources you want to join in the Meta View.
- Which data sources contain the "authoritative" user attributes you want to flow to the connector and Meta Views.

The first stage of planning your Meta-Directory system design is to determine the different data sources that you wish to link with Meta-Directory. While this might at first seem obvious, it is a good idea to poll the different organizations in your enterprise to see if they have implemented any directories or databases that contain information that you want to join with the Meta View.

Once you have determined each of the data sources you want to join in the Meta View, you will need to compile a list of the attributes or fields contained in each directory or database. From this list, you can begin the process of defining which attributes or fields you will use from each of the directories or databases in your organization.

Authoritative attributes are the attributes that you will use to populate your Meta View. For example, suppose you have two external data sources that flow attributes to the Meta View: a Human Resources database and an Information Systems (IS) directory DIT. Consider that both data sources contain information on the company's employees. You will need to decide which attributes (fields) will be the authoritative source for each of the attributes contained in the Meta View. It is likely that both data sources will contain the first and last names of the employees. However, it's likely that the Human Resources database will contain the most up-to-date and correct names. On the other hand, it's likely that the IS directory will contain the correct network IDs for each of the users.

Determining the authoritative attributes in your Meta-Directory system is a bit like going through the normalization process when you design a relational database. You must make sure to keep track of the authoritative attributes as you flow them from different data sources and join them in the Meta View. Be especially attentive when you design how modified data flows back out to the various external data repositories. When designing your system, anticipate events that could affect system performance and stability. Minimize problems that could occur if a database goes down and keep enough flexibility in your plan to allow for the changes that could occur when a new data source comes on line. When planning, be sure that modifications made to one data source will not adversely affect the data contained in another.

In the example shown in Table 2-1, several data sources provide the authoritative attribute data.

Table 2-1 Data Sources From the Site Survey

Data Source	Purpose	Provide Changes?	Accept Changes?	Connector Type/Protocol
Human Resources database (Oracle)	Maintains all the people entries for regular employees and contractors, including location, employment status, and so on.	Yes	Yes	Direct Oracle connector
PBX	Stores employee and contractor telephone numbers. Depends on HR to provide information about changes in employment status.	Yes	Yes	Customized text connector (UTP)
IS database 1 -- NT Domain	NT Domain stores user information for Windows users	Yes	No	NT Domain connector

Table 2-1 Data Sources From the Site Survey *(Continued)*

Data Source	Purpose	Provide Changes?	Accept Changes?	Connector Type/Protocol
IS database 2 -- NIS (Network Information Server)	NIS contains UNIX user account information.	Yes	No	Customized text connector (UTP)
Netscape Phone book	Contains location, phone number, and email address information. This is a Directory Server application that enables the searching, browsing, and editing, of user information.	No	Yes	Direct LDAP connector

Determining Data Flow

In addition to determining the data sources and attributes you will use to populate the connector and Meta Views, you will also need to define the flow of the attributes into and out of the different Meta-Directory views. To help you determine this, you should collect all of the following information during the Meta-Directory planning phase:

- The data sources you want to join into your Meta View.
- The user attributes (or database fields) contained in each of the external data repositories.
- The user attributes that need to flow to and from the unified directory (the Meta View). These are often the authoritative attributes from each data source.
- The frequency that attributes need to flow to and from the Meta View.

  The following questions should be answered during your site survey:

  - How often are updates made to each external data repository?
  - How often will modifications be made to the attributes contained in the Meta View?

  Determining the update frequency will help you schedule the frequency of the synchronization cycles for your connector and Meta Views.

- The mapping of user attributes from the external data source to the Connector View, and vice-versa. Mapping specifies which attributes from the external data sources flow to the specific attributes in the Connector Views, and vice-versa.
- The direction of attribute flow. Attributes may flow from the external data source to the Connector View, from the Meta View to the Connector View, or both.
- The data ownership. Data ownership defines who owns a data object: the Meta View or the external data source that populates a Connector View. Data objects can be deleted only by the "owner" of the object. While you might be able to temporarily delete an object from a Meta-Directory view own the object, the object will reappear when the data is refreshed.

Data Flow Worksheet

Once you have collected the data needed to determine the data flow, you should create a data flow worksheet. The worksheet will list all the data sources you plan to include in your Meta-Directory system, the attributes in each data source, the attributes that you will flow from each data source and their name mappings, any constructed attributes that you plan to incorporate in your join rules, and so forth.

Determining the Data Design for Your Views

You should determine the design of the directory information trees (DITs) that host your Meta-Directory views during the planning phase. Whether you use one or more Directory Server instances, best performance will be achieved if you dedicate a directory suffix (namespace) for each view that you create. This means that the Meta View will have its own directory suffix, as well as the individual Connector Views that you instantiate in your Meta-Directory system.

If you use individual Directory Server instances for your meta and Connector Views, this recommendation is moot. However, if you are instantiating different views on a single instance of Directory Server, then you will want to pay close attention to how you set up the DIT.

System performance is not influenced much by whether you create a relatively flat tree structure or a more vertical tree. However, a flat tree structure is easier to manipulate as you administer the different branches in your directory tree. In addition, should the need arise, troubleshooting the system will be a bit easier if you have a flat tree structure rather than a vertical tree structure.

For example, consider the following flat tree structures for hosting a Meta View and three Connector Views:

```
o=metaview o=notesCV4
o=oracleCV1 o=ndsCV5
o=ldapCV2 o=excCV6
o=adCV3
```

Compare this flat tree structure to a tree structure that has more levels, as the one shown below:

```
ou=metaview, ou=metadir, o=siroe

ou=oracleCV1, ou=connectorviews, ou=metadir, o=siroe

ou=ldapCV2, ou=connectorviews, ou=metadir, o=siroe

ou=adCV3, ou=connectorviews, ou=metadir, o=siroe

ou=notesCV4, ou=connectorviews, ou=metadir, o=siroe

ou=ndsCV5, ou=connectorviews, ou=metadir, o=siroe

ou=excCV6, ou=connectorviews, ou=metadir, o=siroe
```

When you determining the structure of the directory trees that host your different Meta-Directory views, need to determine the base dn for your Meta View as well as the base dn for each of the Connector Views in your system.

Setting Up the Meta-Directory DIT

You set up the namespaces for your Meta-Directory views through the Directory Server console that hosts the views.

In Directory Server 5.x, do the following:

1. From the Directory Server console, select the Configuration tab.
2. In the navigation tree in the left pane, right-click the Data icon and choose Create New Root Suffix.
3. Enter the dn for the root entries for the directory tree. In the dn specified, make sure there are no leading spaces or spaces between commas and attribute types.

Planning the LDAP Schema Used in Your Views

In the system you design, Directory Server and Meta-Directory will likely exist to support one or more directory-enabled applications. These applications have requirements concerning the data contained in the directory, its format, and how the data is interpreted.

It is always best to use an existing LDAP schema to represent the data you are hosting in your Meta-Directory views. Using an existing schema will allow your data to be more portable to other applications if the need arises. If you do need to customize an existing schema, be sure to consider carefully any custom attributes you will add. Once you move forward with a custom schema, it is difficult to undo your modifications and return to a standard set of LDAP attributes.

Planning System Security

Many Meta-Directory design decisions have security implications. Just as with all applications that deal with a company's proprietary data, you want to ensure that the Meta-Directory system you deploy provides a secure environment for the data it manipulates. System security for Meta-Directory includes (but is not limited to) physical access to the systems used by Meta-Directory, network access to those systems, binding under commonly used accounts (for example, `cn=directory manager`), non-Meta-Directory client access to the directory server instances, and so on.

You can use the user authentication and access control mechanisms that Directory Server provides to protect the data that you host in the different Meta-Directory views. Using Directory Server services, you should be able to meet the security and privacy requirements of your system administrators, as well as the users of the applications that access the Meta-Directory views.

Between instances of Directory Server, you can use LDAPS for secure communication. However, it's important to note that Meta-Directory components do not use LDAPS to communicate between the components and views.

Allocating the Necessary Resources

Allocating resources includes obtaining the hardware and software necessary to support Meta-Directory and the other services you will use with your Meta-Directory system (this might include messaging systems, relational databases, network monitoring tools, and any special programs or utilities). Choosing hardware and software that satisfies design requirements for scaling and performance is crucial to successful deployment. Special attention to fast disk subsystems is key to overall system performance as often Sun ONE Meta-Directory systems suffer from slow overloaded disks.

In addition to hardware and software, consider the human resources that you might need to get your system up and running, and the resources that you will need to maintain the system that you design.

Deploying Your System and Creating a Maintenance Plan

At this point, the design phase is over and it is time that you begin to deploy the system you have planned. The remaining sections in this chapter cover deployment and what you need to plan for once your system is up and running.

As part of the Planning phase, you should create a maintenance plan. While some of the items to consider are discussed on "Maintaining Your Meta-Directory System", you should make sure to properly plan for the growth and expansion of your Meta-Directory system.

Deploying Your Meta-Directory System

Deploying your Meta-Directory system consists of implementing the system you mapped out in the design phase, as described in the preceding section,"Designing Your Meta-Directory System."

Effective Meta-Directory deployment centers around the use of system resources (your system hardware) to set up the instances of Sun ONE Directory Server needed to host the different Meta-Directory components. When sizing system needs, you must also consider the services provided by any external data repositories and the resources they use.

A careful, systematic approach to your Meta-Directory deployment is strongly advised. By deploying your system incrementally, you can make sure each discrete portion of the system works before moving on to the next part of the deployment phase. For example, with indirect connectors, first prove flow rules work before installing the Join Engine and defining Joins.

The deployment phase of Meta-Directory consists of properly sizing your host systems and effectively locating them on the network. Once your systems are up and running, tuning issues, system maintenance, and scaling the system for growth become the system administrator's biggest concern.

Data Entry Size and RAM Considerations

The size of the data objects that you will be flowing through your Meta-Directory system plays an important part in the overall design of your system setup. For example, you might be flowing entries that are relatively small (for example, 1024 Kb per entry) or you may be flowing larger entries that contain binaries, such

as photos or digital signatures. Entry size plays an important role in the resources that the join engine uses to synchronize entries between views. Throughput of the system depends on having the available system resources to meet the demands of your setup.

In general, you can calculate the size of an entry by exporting it to LDIF and counting the number of characters it takes to represent the entry in text form. Add two characters for each line in the entry to account for the carriage return and line feed.

If your data contains binary data, you can use a different method to calculate the size of entries. Binary data is base64 encoded, meaning that binary data is stored in the Directory Server database as text characters. To calculate the size of an entry, export the entry to LDIF, then check the size of the LDIF file. On a UNIX system, you can use the following command to report the size of an LDIF file:

`ls -l` *LDIF_file*

On Windows NT systems, use the `DIR` command to view file size.

If you are estimating the size of entries for sizing information, you will likely want to average the size of several different entries to get a better idea of your database needs.

RAM Sizing

Not surprisingly, join engine performance is directly related to system resources: the more system memory (RAM), the faster the join engine will be able to synchronize entries between the Meta-Directory views. While you can supplement system RAM with virtual memory (swap space), you must keep in mind that overloading virtual memory will impact the performance of other system processes, such as other applications running on the system.

This section offers some simple examples to help you judge the amount of system RAM you will need in the units you deploy in your system. The database entries used in the examples below are relatively small, around 1024 Kb per entry. While this is a normal size for most data contained in a DIT of object class `inetorgPerson`, the size of the entries synchronized in the system you are designing might be vastly different, especially if you are flowing binary data.

When sizing the RAM requirements for your system units, consider the following data points observed from system testing:

- When synchronizing 300,000 entries, the join engine used roughly 400 Mb of memory.

  Note that in this scenario, Directory Server was actually hosting 600,000 entries: 300,000 in the Meta View and 300,000 in the Connector View. Here, both the Meta View and the Connector View were hosted by a single Directory Server instance located on a single machine, however, the join engine was hosted on a different machine.

- When synchronizing 500,000 entries, the join engine used roughly 665 Mb of memory. In this scenario, there were 1 million data entries hosted by Directory Server, with each entry being 1024 Kb. Both the Meta View and the Connector View were hosted by a single Directory Server instance located on a single machine, however, the join engine was hosted on a different machine.

When you start the join engine, the memory consumption will increase rapidly until it reaches the load where it begins to level off.

The minimum hardware requirement for Meta-Directory is 256 Mb of RAM (or 512 Mb if you are hosting Directory Server on the same machine). However, keep in mind that a larger production system will require more RAM, and perhaps more system units, than the established minimum requirements. Also, it is recommended that you hose the Meta-Directory components (the join engine and any components you have installed) on a different machine that the ones that host your Directory Server instances.

Table 2-2 gives additional examples on the memory used to synchronize entries between a Connector View and a Meta View. In this table, the average entry size is 1024 Kb, or 1 mega-byte of data per entry, per view.

Table 2-2 Estimating System RAM Size

System RAM	Number of Entries Synchronized	Notes
512 Mb	1000 -> 10,000	Sun ONE Console, join engine, and Directory Server hosted on the same machine.
512 Mb	10,000 -> 100,000	Sun ONE Console, join engine, and Directory Server hosted on the same machine. Directory Server heavily tuned for 100,000 entries (a total of 200,000 entries). Synchronization might be slow and may time out. You must adjust the time-outs of the Directory Server that hosts the join engine.

Table 2-2 Estimating System RAM Size

System RAM	Number of Entries Synchronized	Notes
512 Mb	100,000 to 300,000	Use this to test-pilot a system only! Directory Server would have to be heavily tuned in favor of 300,000 Kb of data. Synchronization times are long and time-outs are likely.
1024 Mb	> 100,000	Sun ONE Console, join engine, and Directory Server hosted on the same machine. Comfortable performance, Directory Server settings need to be tuned, but time-outs are unlikely.
1024 Mb	100,000	Sun ONE Console, join engine, and Directory Server hosted on the same machine. To sync 100,000 entries, Directory Server must be heavily tuned for 100 Kb of data. This would be much better than a 256 Kb system, but the join engine would still need longer than the default time-outs.
2 Gb	500,000 (1 million hosted data entries)	Sun ONE Console and join engine hosted on the same machine; Directory Server hosted on a separate system. The directory server needs to be heavily tuned in favor of this high volume of data entries.

There is a delicate balance that needs to be achieved when you have large amounts of data. For details on the adjustments that you can make to customize your Meta-Directory setup, see Chapter 4, “Meta-Directory Performance Tuning.” You should also refer to the Directory Server Deployment guide for further details about hardware requirements.

Directory Sizing for Windows NT

Calculating the number of Windows NT Domain entries that will be synchronized is useful for determining the RAM and disk space required for the directory service (for optimal performance, the Directory Server needs to hold the entire directory data in RAM).

In this case, entry size is not the only consideration. It is equally important to determine how often the data will be changing in each of the supported Windows NT Domains, how many systems the changed data will have to flow to, and the complexity of the data transformation (for example, attribute mapping, DN mapping, and the filter and rule sets that need to be established).

System Topology

The Meta-Directory system *topology* is the physical layout and locations on the network of the machines that you use to host all the servers and services in your Meta-Directory system. When designing the system topology, you are concerned with the physical implementation of the Meta-Directory components and where you place them on the network. In designing the system, you want to strike a balance between resource use and growth potential.

When you perform a site survey during the Meta-Directory planning phase, you will determine (among other things) the network location of the applications that will provide user information to your Connector Views and, ultimately, the Meta View. Into this network layout, you will add the Meta-Directory components and services (the join engine, the Meta View, connectors, and the associated Connector Views; plus the servers and services that support these components).

When deploying Meta-Directory, you must plan the topology of the system by looking at the types and amounts of data you plan to flow through the Meta-Directory connectors and join engine. Best performance is achieved when you design the topology to ensure that the set of hardware hosting the Meta-Directory components is properly load balanced. Network latency can also have an impact on synchronization times. It is possible for a smaller number of system units to outperform a larger set of system units if the smaller system is sized correctly.

Operating System Configuration

The configurations of the operating systems involved in your deployment is an important factor to consider when designing the Meta-Directory topology. When designing the system, you must take into account the physical memory (RAM) of the systems used, operating system settings (such as the amount of swap space used), and hard disk partitioning and configuration.

The slapd Processes

In Meta-Directory, individual processes are used to manage the main Meta-Directory components:

- nsmds runs the join engine.
- nsperlconn runs the Universal connector (UTC).
- Perl scripts are used by the Active Directory and NT Domain connectors.
- javaserver runs the Novell Directory Connector and the Lotus Notes Connector

Depending on your needs, you can host some of these processes on separate machines.

Directory Server Instances

When you are synchronizing large databases, it is best to create separate Sun ONE Directory Server instances for the Connector Views, as well as using a different Directory Server instance for the Meta View. Because of this, it is likely that your Meta-Directory system will contain a number of different instances of Sun ONE Directory Server.

Hosting Connector Views on separate Directory Server instances allows you to balance the load of processing across different servers and machines. When designing the layout of the Directory Sever instances, distinguish between Connector Views that are populated by indirect connectors and the Connector Views that are directly updated by the join engine through the LDAP direct connector. Making this distinction is important when you tune the different Directory Server instances incorporated in your Meta-Directory system—Directory Server instances that host indirect Connector Views should be tuned for write performance while those that host “external” data should be tuned for read performance.

In determining the number and location of the Directory Server instances, keep in mind the Meta-Directory configuration is normally stored in the same Directory Server instance and DIT as the configuration for all other Sun ONE servers used in your Meta-Directory setup.

When creating different Directory Server instances to be used in a single Meta-Directory system, be sure to give the different instances different names as you create them.

Data Servers

Within the Meta-Directory console, each Directory Server instance is configured through the Data Server tab. To view the currently configured Directory Server instances:

1. Open the Meta-Directory console and highlight Meta-Directory in the navigation tree. The right pane displays six tabs.
2. Select the Data Servers tab to view the currently configured Directory Server instances.

Refer to *Sun ONE Meta-Directory Administration Guide* for instructions on how to add a new Directory Server instance to the Meta-Directory configuration.

Planning the Layout of Your System

There are three basic scenarios for deploying the Sun One Meta-Directory system:

- The ideal, separate Meta-Directory views (either a Meta View or a Connector Views) on dedicated Directory Server instances ideally on separate machines (or Solaris domains).
- The compromise, separate the Meta View from the other Meta-Directory views and connectors. The Meta View and its host system will typically require different performance characteristics than the other meta components. Combine the Connector Views and configuration directory on the same machine in separate Directory instances.
- The consolidated view, is where all Connector Views, Meta View and configuration all stored in one Directory Server instance using separate namespaces.

In a large deployment, it is likely you will employ a combination of these setups.

Locating Meta-directory views on separate machines can increase the performance of the services, but you must balance any gains with the loss incurred through the added network latency. Through system analysis and comparisons with projected loads, you must decide on the setup that is needed for your particular implementation. However, using separate systems for each Directory Server instance on separate hosts is the best choice. If your system resources are limited, consider hosting at least the Meta View on a dedicated system.

Whether more than one Meta-Directory view should be hosted on a single machine is ultimately dependent on the machine's processing capacity, the memory allocated to each of the system units, availability of disk space, the load on disk controllers and file systems, and the volume of data flowing between Connector Views and the Meta View. System profiling and piloting swill provide the data required to judge the performance of your particular Meta-Directory set up.

Requirements for Hosting the Meta View

The Meta View presents the final "image" of the collected data to end users. Most likely, client applications will retrieve the information stored in the Meta View. Because of this, the Meta View normally has a higher performance requirement than does the typical Connector View. While search indexes must be built for the Meta View to speed up the different types of search operations performed by the applications accessing the Meta View, the same level of searches are not generally used in Connector Views that are dedicated to the Sun ONE Meta-Directory system.

See "Determining Which Indexes to Create" on page 84 for a list of the different indexes you should create for the different views in your deployment.

Requirements for Hosting Connector Views

The task of any Connector View is to host an LDAP copy of the data contained in an external data source. Connector views should not be used for dual purpose tasks. The primary operations performed on the Connector View are more heavily weighted on add, delete, and modify; searching through Connector Views is not a priority as it is with the Meta View. With a lighter demand for searches, there is a lesser need for indexes in the Connector View. This directly affects your design for system performance: the fewer indexes that need to be updated in the Connector View directory, the better the performance will be.

Reasons for Choosing a Multiple-Instance Configuration

Hosting the Meta View and Connector Views on different Directory Server instances will result in better performance and diminish the likelihood that the failure of one component will greatly affect the performance of another.

A multiple Directory Server configuration makes it easier to track log file data for a component. This also makes it easier to maximize the overall Meta-Directory performance since you will be able to optimize components individually. For more information about Directory Server optimization for Meta-Directory, refer to Chapter 4, “Meta-Directory Performance Tuning.”

Using Different Directory Server Instances for the Configuration Database and the Meta View

It is often desirable to use one Directory Server instances for the Meta-Directory configuration database and a different one for the Meta View. By default, Meta-Directory expects these two DITs to exists on the same Directory Server instance. To use separate Directory Server instances for these databases, do the following:

1. Install two Sun ONE Directory Server instances: one for the Meta View and one for the configuration database. These instances can be on the same machine or they can be on different machines. Be sure to give the different instances different names.
2. Install Sun ONE Meta-Directory. Make sure you reference the Directory Server instance that you want to host the configuration database during the Meta-Directory installation.
3. Start the Sun ONE Console from the Directory Server instance that will host the configuration database. Notice that the Sun ONE Console shows several Server Groups.
4. Create an instance of the join engine:
    a. Select the Server Group that represents Meta-Directory.

b. Right click the Server Group, then choose Create Instance of the Join Engine.

c. When supplying information for the join engine, *type* in the information for the Directory Server instance that you want to host the join engine and the Meta View (this Directory Server instance will not be in the drop down list of available Data Servers until you specify it when creating an instance of a Meta-Directory component).

Once you have configured several Directory Server instances, you can view them in the Data Servers tab from the Meta-Directory console.

NOTE It is recommended that the join engine bind to the Directory Server as a Directory Manager client. This will give the join engine the permissions it needs to perform its tasks.

Server Groups

Sun ONE Console uses “server groups” to represents the different sets of Sun ONE servers and services installed in your Meta-Directory system.

When you install different instances of Administration Server, each one is represented as a *server group* within Sun ONE Console. In your Meta-Directory system, you will have at least one server group displayed in your Sun ONE Console. However, it is possible to have several, depending on your Meta-Directory system setup.

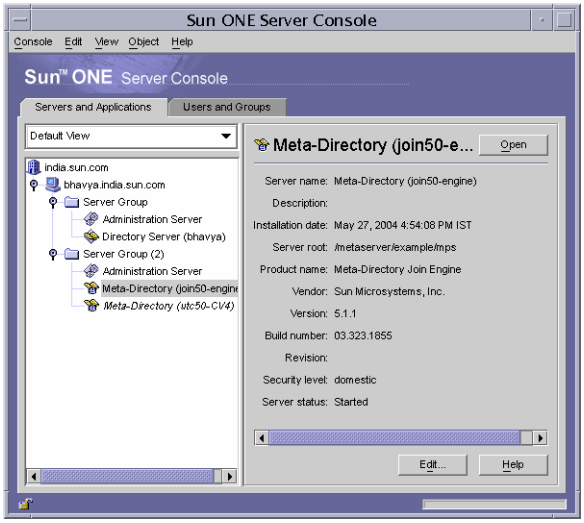
For example, suppose you are using Sun ONE Directory Server 5.x to host your Meta-Directory views. In this case, you will first install Sun ONE Directory Server then Sun ONE Meta-Directory. Sun One Meta-Directory is supported by Sun ONE Administration Server 5.2. Sun ONE Directory Server is managed by Sun ONE AdministrationServer 5.0.

Suppose you then use Netscape Directory Server 4.15 to host a direct Connector View. Netscape Directory Server 4.15 is supported by Netscape Administration Server 4.2.

In this scenario, there are three versions of Administration Server. When you open Sun ONE Console to administer your Meta-Directory system, you will see three server groups, one for each Administration Server.

The figure below shows three server groups. In this setup, the configuration database and the Meta View are hosted by different Directory Server instances, with each instance supported by its own Administration Server. A third server group was created for the Meta-Directory components because a third Administration Server was installed when Meta-Directory was installed. Here, the join engine uses the Directory Server instance in Server Group (2) to host the Meta View.

Figure 2-1 Multiple Server Groups in Sun ONE Console



Different versions of Administration Server must be installed into different file paths—you cannot install Sun ONE software into the same file path if the components use different versions of Administration Server.

NOTE Do not attempt to start the Meta-Directory console or any of its components using Netscape Console 4.*x* or Netscape Administration Server. Also, do not use Netscape Administration Server to administer Meta-Directory components. However, it is possible (and recommended) to administer Netscape components using Sun ONE Console 5.x.

Sun ONE Meta-Directory 5.1 must be managed using its own Administration Server and not the 5.0 or 4.x versions installed with iPlanet Directory Server 4.x or 5.1

Non-Participating Connector Views

In Meta-Directory, it is possible to populate a Connector View, but you may choose to not synchronize the data in that view with the Meta View. For example, you might wish to flow entries from an Active Directory data repository to those contained in an LDAP Connector View for the sake of having the entries available in an LDAP directory. In this configuration, you would not configure the Connector View to be a *Participating View* of the Meta View, since you would not be flowing entries from this view to the Meta View.

If you find a need to populate a Connector View, but do not wish to make it a Participating View of the Meta View, you should use a Directory Server instance other than the one that hosts the Meta View as the host for the non-participating Connector View.

The reason for this recommendation is that the join engine processes all the change log entries of the Connector Views that participate in the Meta View. If you have a non-participating Connector View sharing a Directory Server instance with the Meta View, the join engine will waste time processing the change log entries generated by the non-Participating View when there is no need to do so.

Locating the Log Files

During the initial phases of deployment, you will likely have verbose logging turned on to help you ensure that things are running smoothly. In these stages of deployment, Meta-Directory can generate copious log files. Setting up your system so that it writes log files to different file systems on separate partitions, can improve performance when system logging is tuned to a high level.

It's important to note that in addition to increased controller traffic on a partition, increasing the logging level of your Meta-Directory components can result in a performance hit on your synchronization times. A higher level of logging will generate more disk write operations. These extra disk write operations use processor cycles that could otherwise be spent on synchronization. Once your system is running smoothly, you might see a gain in performance when you reduce the level of system logging.

Changelog

Sun ONE Directory Server version 5.x implements two different change log mechanisms. One is used for multi-master replication and the other tracks updates made to the Directory Server database.

- Enabling the Directory Server retro change log (cn=changelog) allows Meta-Directory to participate in change notifications with other components. If it is not enabled, Meta-Directory will not function. If the Directory Server retro change log is not enabled, Meta-Directory will display a dialog that prompts you to enable the change log when you create an instance of a Meta-Directory component.
- RetroChangelog (cn=changelog) Directory Server plug-in must be installed on all Directory Server instances in the Meta-Directory System. This includes the configuration directory too. If the changelog on this configuration instance is not enabled then the Join Engine does not start properly.

Locating the retro change log on dedicated disk partitions can improve overall performance.

Multi-Master Replication

Meta-Directory supports Sun ONE Directory Server 5.x in an MMR configuration. However, failover via MMR is not supported.That is, the Meta-View can reside on one and only one master. The join-engine is configured to point to this master. Then via MMR the master meta-view data can be replicated to another master and, if required, 1-n consumers. When MMR is configured, two kinds of change log must be utilized in order to allow Meta-Directory and MMR to function correctly:

- retro changelog

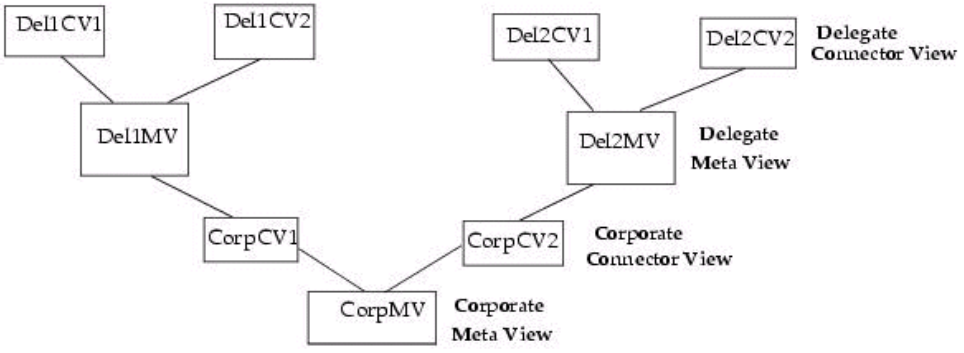
The join engine in Meta-Directory depends on the architecture of the change log facility that was implemented in Netscape Directory Server 4.1x. This is known as the retro change log plug-in within Sun ONE Directory Server 5.x. See the Sun ONE Meta-Directory Administration Guide for details of how to configure this.

- changelog

To synchronize two master directory servers, the new change log in Sun ONE Directory Server 5.x architecture is also utilized. See the Directory Server Administration Guide on how to configure this.

Cascading Meta Views

The idea of cascading Meta View comes into use, when there is a need to synchronize data from various Sun ONE Meta-Directory installations into one single repository. The diagram below illustrates a typical setup with three Meta-Directory installations.Two subordinate or delegate Meta installations connected to one main Corporate Meta installation.



Each entry in a Meta-Directory installation is owned by a particular view. This ownership information is stored in the attribute mdsEntityOwner. Any modifications made to an entry in any of the views flows to all the views. However, only the owner view has the right to delete an entry. If an entry is deleted from any other view, it should get added back. A non-owner, however, can modify an entry. In a cascading Meta-View setup, such as the one shown above in the figure, deletion of entries even by owner views was not happening in the previous release.

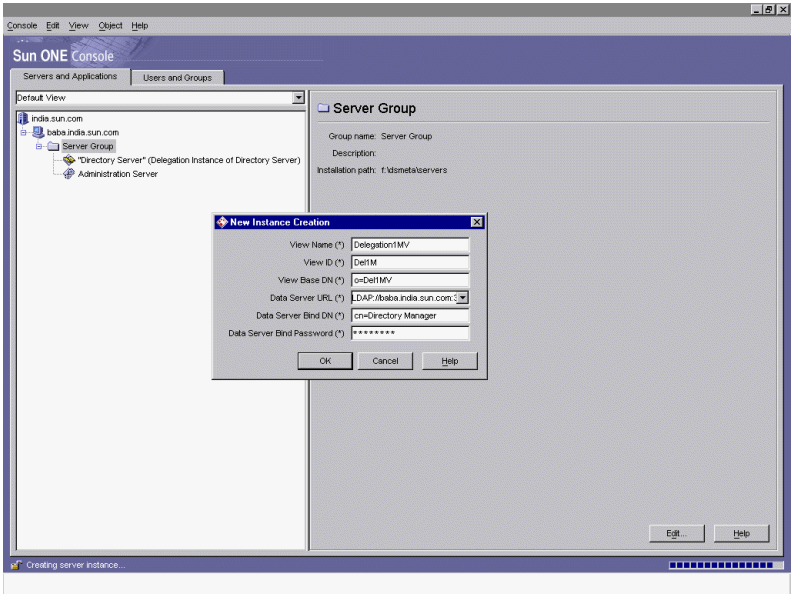
This was reported in the problem (#4636032) "Entries failed to delete from cascade Meta View setting". This problem has now been fixed. Now entries can be deleted from the owner views, across all the Meta-Directory installations. If an entry is deleted by a non-owner, it gets added back.

Steps for setting up Cascading Meta views

This set up consists of 2 Delegation Installations and 1 Corporate Installation. The idea is to synchronize the data from various views to Corporate Meta View. All these 3 installations have separate Sun ONE Directory Server installations.

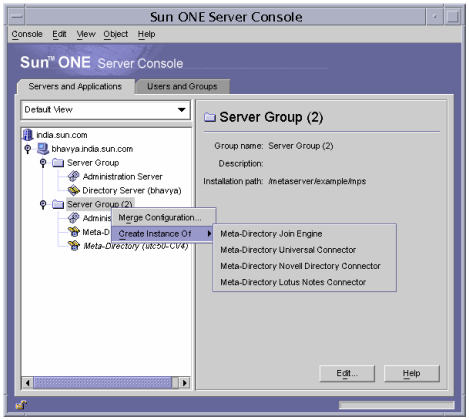
1. Install Sun ONE Meta-Directory on the first machine (refer this installation as delegation 1 installation) and create the Join Engine instance and give the view name as Del1MV.

Figure 2-2 Del1MV



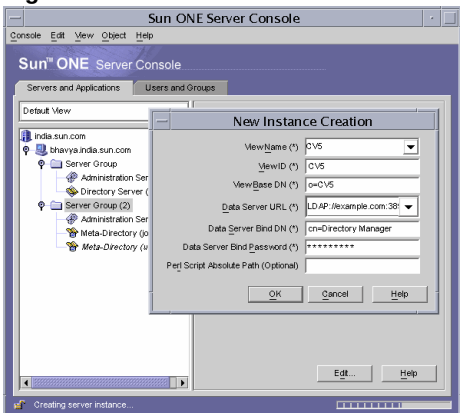
2. Create a Connector view 1 (refer this as Del1CV1 or UTC CV1, use any UTC connector or Oracle connector) and populate data into Connector View.

Figure 2-3 Del1CV1



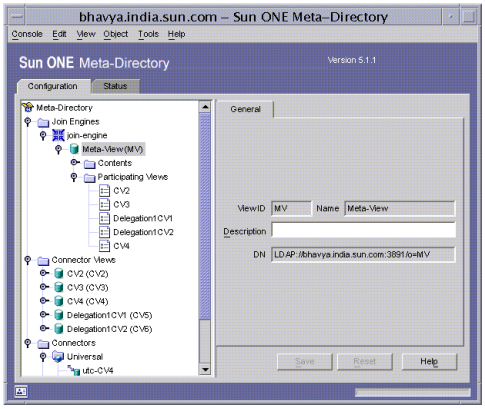
3. Create a Connector view 2 (refer this as Del1CV2 or UTC CV2, use any UTC connector or Oracle connector) and populate data into Connector View.

Figure 2-4 Del1CV2



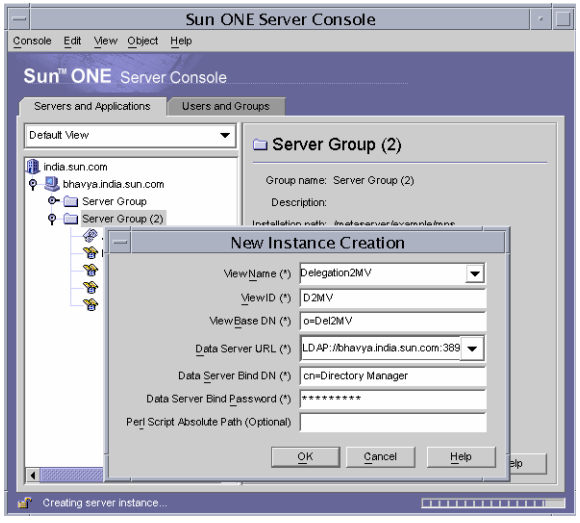
4. Add Del1CV1 & Del1CV2 as JoinEngine's Participating Views and enable the Participating Views. Assume contents of Del1MV is present under suffix o=Del1MV.

Figure 2-5 Del1 Predicating View



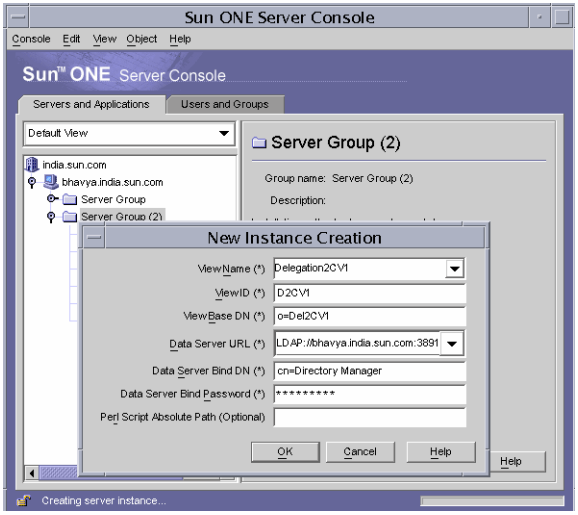
5. Do another Delegation installation of Meta-Directory on the second machine (refer this as Delegation 2 installation) and create the Join Engine instance and give the view name as Del2MV.

Figure 2-6 Del2MV



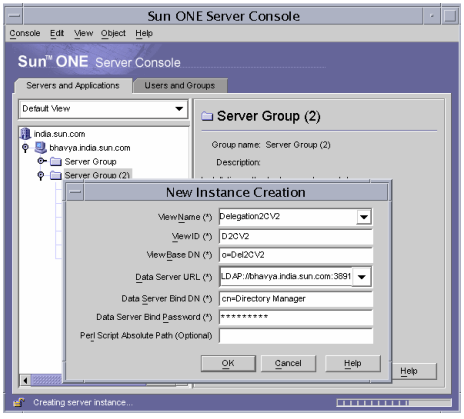
6. Create a Connector view 3 (refer this as Del2CV1 or UTC CV3, use any connector and populate data into Connector View.

Figure 2-7 Del2CV1



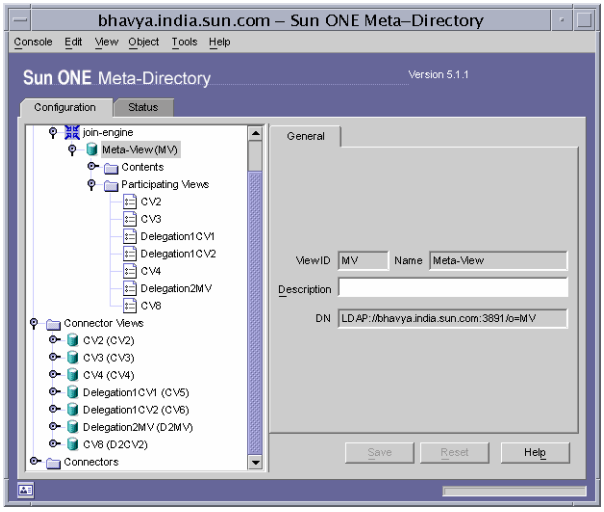
7. Create a Connector view 3 (refer this as Del2CV2 or UTC CV4, use any connector and populate data into Connector View.

Figure 2-8 Del2CV2



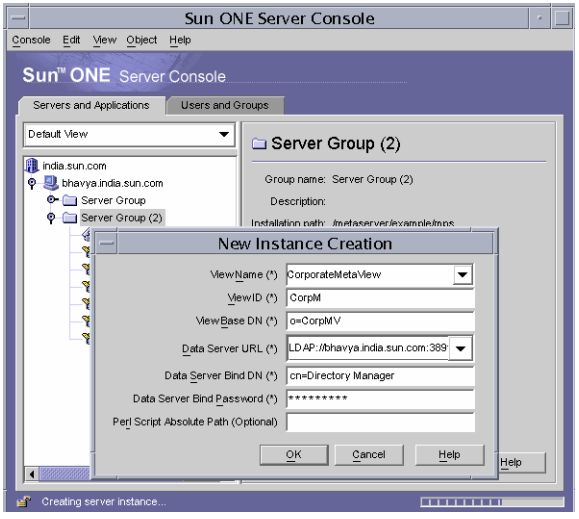
8. Add Del2CV1 & Del2CV2 as Join Engine's Participating Views and enable the Participating Views. Assume contents of Del2MV is present under suffix o=Del2MV.

Figure 2-9 Del2 Participating View



9. Now, do Corporate Installation of Sun ONE Meta-Directory on the third machine (refer this as Corporate installation) and create the Join Engine instance and give the view name as CorpMV.

Figure 2-10 CorpMV



10. Create a Corporate Connector view1 (refer this as CorpCV1) and populate data into Connector View. To create the Connector View, you have two options:

    Either (a) Create a LDAP data server from Sun ONE Meta-Directory - Data Servers, and use this URL while creating the CorpCV1.

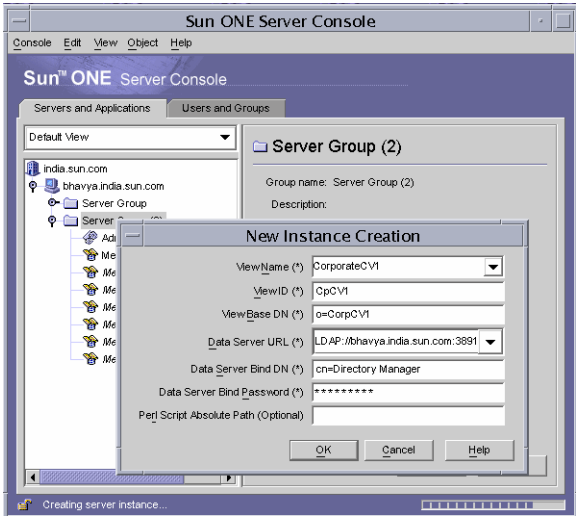
    host name -- host name of the system where delegation mds installation(Del1MV) is running.

    port -- port number that stores contents of o=Del1MV.

    Bind dn and password for that ids instance. (Assume that data server created as a result of this operation as baba.india.sun.com:389)

    or (b) Create a new connector (refer this cv as CorpCV1) on corporate mds installation which actually points to delegation 1 installation's MV (For example, give baba.india.sun.com:389 as data server url. enter o=Del1MV as base view dn).

Figure 2-11 CorpCV1



11. Create a Corporate Connector view1 (refer this as CorpCV2) and populate data into Connector View. To create the Connector View, you have two options

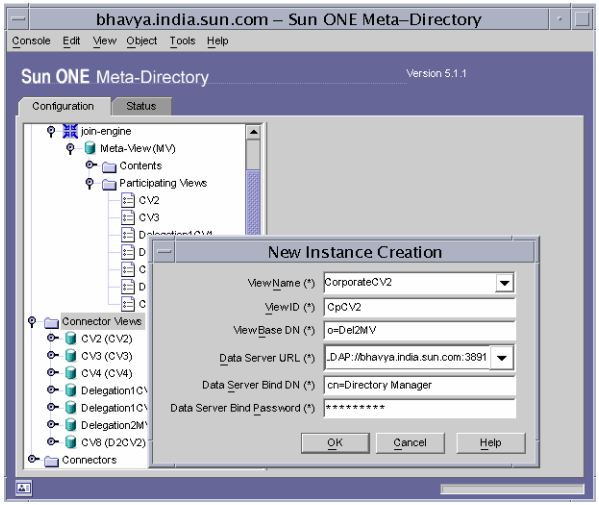
    Either (a) Create a LDAP data server from Sun ONE Meta-Directory - Data Servers, and use this URL while creating the CorpCV2

host name -- host name of the system where delegation mds installation(Del2MV) is running.

port -- port number that stores contents of o=Del2MV.

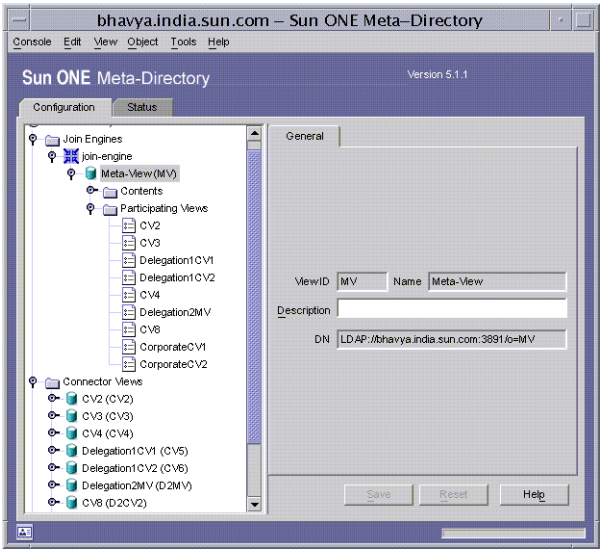
Bind dn and password for that ids instance. (Assume that data server created as a result of this operation as baba.india.sun.com:389) or (b) Create a new connector (refer this cv as CorpCV2) on corporate Sun ONE Meta-Directory installation which actually points to delegation 2 installation's MV (For example, give baba.india.sun.com:389 as data server url. enter o=Del2MV as base view dn).

Figure 2-12 CorpCV2



12. Add CorpCV1 & CorpCV2 as Participating Views of corporate Sun ONE Meta-Directory installation and enable the Participating Views. Assume contents of CorpMV is present under suffix o=CorpMV

Figure 2-13 Corporate Participating View



With this configuration, data flows properly between Del1MV, Del1CV1, Del1CV2, Del2MV, Del2CV1, Del2CV2, CorpMV, CorpCV1 and CorpCV2.

Recommendations

1. It is recommended that the view IDs of all the Connector-Views be unique across all the Meta-Directory installations.
2. The cascaded view functionality requires that the capabilities to be on in both sides; i.e. the Connector View as well as the Meta View. This is required for the add backs to happen when an entry, owned by a view outside this Meta-Directory installation, is deleted.

Limitations

1. Deletion of entries owned by delegation Meta-Views or Corporate Connector-Views results in inconsistent behavior. However, by default, the entries are owned by the data source Connector Views and as such this limitation should not normally affect use.
2. All the View IDs of the Meta-Views of all the Meta-Directory installation in the cascaded setup must be unique.

Bringing the System On Line

At this point you have gone through the process of planning and physically deploying your Meta-Directory system. You have finished buying system resources (for the time being, at least) and have installed all the software needed to bring the system on line. To open the Sun ONE Console so you can begin instantiating and configuring the various components in your Meta-Directory system, you will need to start the following servers and services:

1. Start the slapd process that will host your Meta-Directory configuration, then start the Administration Server that administers the slapd process.
2. Start the Sun ONE Console that will host your Meta-Directory components.

At this point, you can use Sun ONE Console to configure your Meta-Directory system, as described in the *Sun ONE Meta-Directory Administration Guide*. However, before you begin the configuration process, consider piloting and tuning your system before going into production.

Piloting

The best way to determine how Meta-Directory will perform in your particular scenario is to run a pilot test of the proposed deployment and measure the system performance. By piloting the system you have designed, you will gain the information needed to properly gauge the production system you will need to put in place. In a pilot deployment, you set up a smaller scale version of the Meta-Directory system you have designed.

During the pilot run, you not only test system topology configurations, but you can also test the different sets of rules that you will use to synchronize data from the disparate data sources. Running and tuning the rules you set up is a key ingredient to achieving the best possible system performance in your production system.

During the pilot run, test one connector at a time before attempting to merge additional connectors into the mix. The purpose is to test your Meta-Directory design, including the data flow, the application of join rules and DN mapping rules, the attribute flow and filtering of entries, and the functionality of constructed attributes. Ideally, your pilot run will give you the confidence to move the system into the complete system that you will bring on line.

Tuning

By monitoring network status and join engine performance, you can look for bottlenecks and other inefficiencies in system performance. Are the system links to external directories working consistently? Are the links between Directory Server instances that host Connector Views operating optimally?

For more information about tuning Directory Server performance for Meta-Directory, refer to Chapter 4, “Meta-Directory Performance Tuning.”

Going Production

When it is time to finally go into production, the best course of action is to begin by synchronizing your main source of data, first into a Connector View, then into the Meta View. From there, flow other external data sources into Connector Views, one at a time. Once you have populated a second Connector View, go ahead and link the entries with the ones contained in the Meta View.

Check your logs to ensure things are flowing correctly by looking for failed join rules. A join rule failure will result in an entry being orphaned; entries are missed when a join rule fails, when a join rule returns more than a single object, or when a join rule encounters bad data. Use the Query Fix-It Tool to search for any unlinked entries and fix them whenever needed. Also, you can search the logs files with using the following commands:

```
grep -i fail *
grep -i error *
```

If you move one view at a time, running the system and checking the logs after each incremental step, you can more easily identify the location of performance bottlenecks. More importantly, you can address and fix any potential problems as you bring up the system.

Maintaining Your Meta-Directory System

During the system design phase, a maintenance plan should be created for maintaining the system once it has been deployed. Meta-Directory system maintenance includes (but is not limited to) the following items:

- Performing Data Backups
- Monitoring the System
- Planning for System Expansion

Performing Data Backups

It should almost go without saying: Perform regular backups of Meta-Directory data. At a minimum, plans should be in place to back up the Meta View on a regular basis. In a more elaborate system, you should also consider backing up any Connector Views that you have populated.

In Meta-Directory, all the views are hosted by Directory Server (except when you use Oracle to host a Direct Connector View). The Directory Server provides inherent replication capabilities that can be used to fulfill the task of data backup. Refer to Directory Server documentation for details on this feature.

Monitoring the System

How will you assure reliability and availability in your Meta-Directory system? You should design into the maintenance plan the structure needed to insure that system resources are available and the throughput of the system can be maintained at the needed level. Your enterprise may have an existing structure in place that monitors system resources such as the network traffic and individual system loads. If possible, use this existing infrastructure to monitor your Meta-Directory deployment. In other cases, you may need to set up an individual monitoring system as part of the Meta-Directory maintenance plan.

Monitoring your system resources also encompasses the task of trimming old log files to free up disk space. Provide a schedule for monitoring the log file directories and a procedure for deleting the old, outdated files.

Network security might also be a concern. The maintenance plan should have an eye toward future needs and include a section on how the system security will be kept at required levels.

Planning for System Expansion

In the course of monitoring system loads, you might find that faster performance is needed. Plans should be in place to add more resources in case you need to scale the system.

When does CPU utilization reach the point where additional processing power is required? Should one consider installing additional/faster performing disks to offset performance degradation? When does network bandwidth utilization reach the point where additional bandwidth is required? Through the analysis of the system load and performance, you should be able to gauge the resources that are necessary to support the needs of your system.

While it is the job of the system administrator to make sure the system is available, it is the job of the system planners to ensure that the proper resources are on line to fulfill the needs of the system they design. For more information on the requirements of Directory Server, refer to the *Sun ONE Directory Server Deployment Guide*.

Chapter 3

Directory Server Configuration Settings for Meta-Directory

Once the Sun ONE Directory Server and Meta-Directory software is installed, you must modify several Directory Server configuration settings to support the Meta-Directory system. This chapter describes the various configuration settings you need to make before you can begin using Meta-Directory.

Although the Meta-Directory console provides automatic support for making some of the configuration settings, this chapter explains in detail what is needed to configure Directory Server instances hosting Meta-Directory components, and it describes how to manually configure these settings.

The topics covered in this chapter are:

Installing and Configuring Directory Server

Directory Server must be installed and configured before you can run Meta-Directory. Detailed documentation for the Sun ONE Directory Servers, including installation and configuration instructions, can be found at the following Sun ONE web site:

`http://docs.sun.com/db/prod/s1dirsrv`

To support the Sun ONE Meta-Directory components, you must modify several Sun ONE Directory Server server settings. In particular, you must:

- Enable the retro change log plugin
- Load the Meta-Directory schema
- Adjust the Directory Server plug-ins, if required
- Modify / create necessary Directory Server indexes

NOTE The Meta-Directory console will automatically make several of these configuration settings when you create instances of the Meta-Directory components. However, the sections in this chapter explain the details of these configuration settings and show how to manually make the settings.

Directory Server Configuration Steps

The following steps describe what is needed to correctly install and configure Sun ONE Directory Server so it can properly support Meta-Directory:

1. Install and start the Directory Server instance that will host the Meta-Directory configuration database (`o=netscaperoot`) or one of its components.

   See Chapter 2, "Planning the Meta-Directory System"for considerations on planning your Meta-Directory system setup.

2. Configure the Directory Server instance for use with Meta-Directory by performing the following tasks (these tasks are described in the sections that follow):

   a. Enable the Retro change log plugin

   b. Load the Meta-Directory schema. This is done automatically if prompted when creating Sun ONE Meta-Directory Meta View or Connector View

   c. Adjust the Directory Server plug-ins, if required.

   d. Modify / Create necessary Directory Server indexes.

3. Shut down and restart the Directory Server instance.

   The Directory Server instance is now ready to support Meta-Directory components. Depending on your system setup, you might locate a Meta View, or one or more Connector Views, on this instance. In addition, you will have a Directory Server instance supporting the Meta-Directory configuration files.

   The configuration for all Sun ONE servers and components are stored in the directory tree under *NETSITE_ROOT*. This directory tree is configured when you install the Meta-Directory software.

4. Edit the configuration parameters of the user Directory Server instance to tune its performance.

   Once it is configured, you can fine tune the Directory Server settings to optimize its performance with Meta-Directory. Fine tuning Directory Server is described in Chapter 4, "Meta-Directory Performance Tuning."

Modifying Directory Server Settings

There are three ways in which you can edit the Directory Server configuration settings. The following list ranks these three methods from the easiest to the technically most difficult:

- Edit the configuration settings from the Directory Server console.
- Export the configuration settings to an LDIF file and modify the configuration settings using a text editor. Save the new configuration, reload the LDIF, and restart the server.
- Issue commands to modify the configuration database using the `ldapmodify` utility.

Configuring UTF8 Support

Earlier versions of the Meta-Directory only consistently supported ASCII characters. Sun ONE Meta-Directory 5.1 synchronizes attribute values which contain UTF-8 encoded Unicode characters as well. Please consult the *Sun ONE Meta-Directory Administration Guide* for more information about this.

Enabling the Retro Change Log

Enabling the Directory Server change log (cn=changelog) allows Meta-Directory to participate in change notifications with other components. If it is not enabled, Meta-Directory will not function.

If the Directory Server Retro change log is not enabled, Meta-Directory will display a dialog that prompts you to enable the change log when you create an instance of a Meta-Directory component. The procedures in the section describe how to manually enable this Directory Server feature.

Enabling the Sun ONE Directory Server Change Log

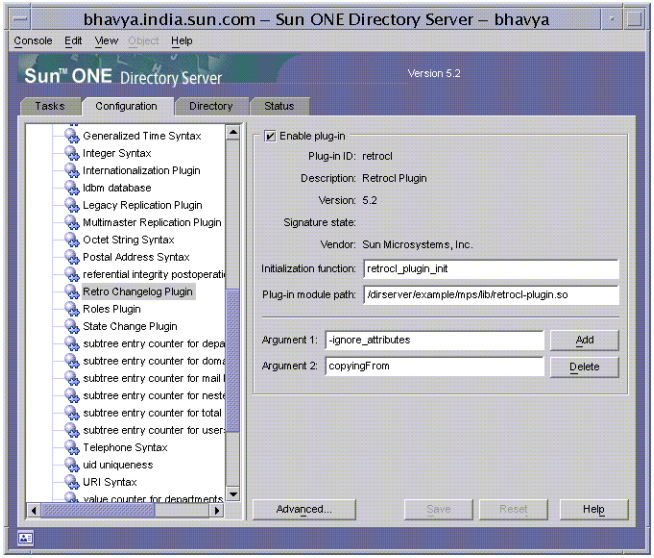
Sun ONE Directory Server version 5.x implements two different change log mechanisms. One is used for multi-master replication and the other tracks updates made to the Directory Server database. The latter, the *retro change log*, must be enabled to support Meta-Directory processing.

In Sun ONE Directory Server, the retro change log is implemented as a plug-in. To enable it, do the following:

1. Open the Sun ONE Directory Server console.
2. Select the Configuration tab; the Configuration window appears.
3. In the navigation tree, expand the Plug-Ins node.
4. In the list of plug-ins, select the Retro Changelog Plug-In.

   The settings for the retro change log display in the right pane, as shown in Figure 3-1.
5. Check the Enable Plug-In box and choose Save to save the setting.
6. Once you enable the change log, you must restart the Directory Server for the settings to take effect.

Figure 3-1 Enabling the Change Log in Sun ONE Directory Server

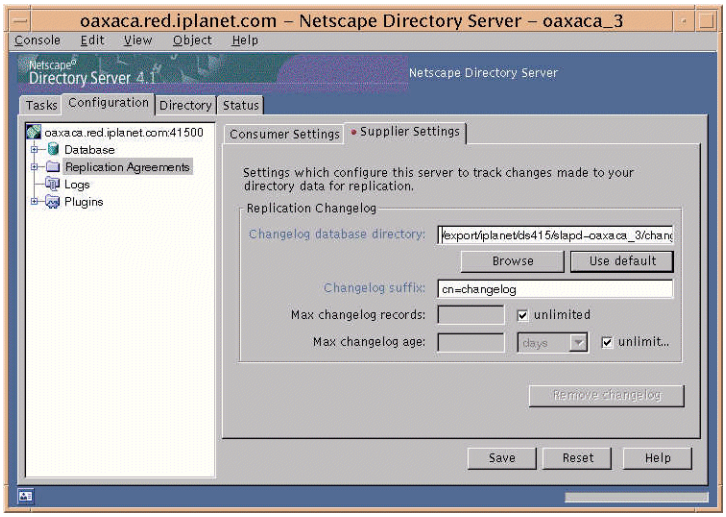


Enabling the Netscape Directory Server Change Log

To enable the change log for Netscape Directory Server version 4.1*x*:

1. Open the Directory Server console.
2. Select the Configuration tab; the Configuration window appears.
3. Select the Replication Agreements node in the navigation tree, then select the Supplier Settings tab in the right pane (shown in Figure 3-2).
4. Click the Use Default button to specify the default file path for the change log files.
5. Check the Max Changelog Records and Max Changelog Age settings so these parameters are set to “unlimited.”
6. Once you enable the change log, you must restart the Directory Server for the settings to take effect.

Figure 3-2 Enabling the Change Log in Netscape Directory Server



Change Log Location

The location of the change log is important with regard to system performance. Since the change log can potentially produce many writes, you do not want activity to the change log to conflict with activity to other Directory Server databases. You might consider configuring Directory Server so that it creates and stores the change log in a disk partition that is different from where your LDAP databases are stored. Specifying a separate disk partition for the change log will reduce the seek time and disk latency of the disk(s) housing other LDAP databases.

In iPlanet Directory Server 5.x, the default path for the retro change log is related to the directory tree where the Directory Server instance is located: `DS_Instance/db/changelog`. You can modify this default path by editing the `nsslapd-changelogdir` attribute under `dn: cn=Retro Changelog Plugin,cn=plugins,cn=config`.

Alternately, you can have Meta-Directory configure the path of the retro change log. If the retro change log is not enabled when you create an instance of a Meta-Directory component (such as the join engine), the Meta-Directory console will enable it for you and will prompt you for the directory where you want the log to be located.

In Directory Server 4.1*x*, you can set the change log location when you enable the feature.

Setting Write Permissions on Solaris Systems

On Solaris systems, Directory Server is normally installed by a user with root privileges. Because of this, the directory containing Directory Server and all its subdirectories will contain a file permission mask of 755. If you create a special directory for the change log, you must ensure that Meta-Directory can write to this directory. It is recommended that the directory containing the change log has a file permission mask of 777. The following UNIX command will change the file permission of a directory to the desired 777:

```
chmod -R 777
```

Loading Meta-Directory Schema

To process Meta-Directory requests, Directory Server must recognize the extended schema (LDAP object classes and attributes) used by Meta-Directory components. Loading the Meta-Directory schema into Directory Server allows the Meta-Directory components to communicate with the Directory Server over LDAP.

Whenever you create an new instance of a Meta-Directory component, the Meta-Directory console will prompt you to load the extended schema into the Directory Server hosting the component.

The schema needs to be loaded into each Directory Server instance only once; after it is loaded, you need not reload it. However, because Sun ONE Console cannot verify that the schema has been loaded, it will prompt you to do so, even if it has already been loaded.

Manually Loading the Meta-Directory Schema

The Meta-Directory schema is provided in the LDIF file `md-schema.ldif`, which is located in the `NETSITE_ROOT/bin/join50/install/templates` subdirectory of your Meta-Directory installation. You can manually add the Meta-Directory schema to the Directory Server configuration directory (*NETSITE_ROOT*\config) using either `ldapmodify` or through the Directory Server console.

Use the following command as an example if you want to add the Meta-Directory schema to a Directory Server instance using `ldapmodify`:

```
ldapmodify -h hostname -p 389 -D "cn=directory manager"
    -w password -a -c -v -f md-schema.ldif
```

The Meta-Directory schema contained in `md-schema.ldif` can also be imported using the Directory Server console. For more information on importing schema, refer to the *Sun ONE Directory Server Administrator's Guide.*

NOTE While possible, it is not necessary to add the Meta-Directory schema to an instance of Directory Server that does not host a Meta-Directory component. (For example, if you directly connect to a Directory Server instance to populate a Connector View, that Directory Server instance does not host a Meta-Directory join engine or indirect connector component). If you do load the Meta-Directory schema into such an instance of Directory Server, you will get a string of error messages stating `No Such Attribute ... Cannot delete`. These messages do not indicate a problem—they are generated because the ldapmodify tool is attempting to delete the Meta-Directory attribute before it adds a new copy of the attribute.

Adjusting Write Permissions (Solaris Only)

On Solaris, the Directory Server instance may run with an identity different from its managing Administration Server. To grant Directory Server the permissions necessary to modify the Directory Server schema, issue the following commands from the UNIX command line:

```
% cd NETSITE_ROOT
% chmod ugo+w slapd-*/config/slapd-user_*.conf
```

Adjusting Directory Server Plug-Ins

There are two Directory Server plug-ins that need to be disabled in most Meta-Directory deployments: uid-uniqueness and referential integrity postoperation.

Setting uid-uniqueness Plug-In

If the data for the Connector View and the Meta View are in the same Directory Server instance, it may be required to turn off the Directory Server uid-uniqueness plug-in. You can turn off the plug-in from the Directory Server console.

The setting of the uid-uniqueness plug-in depends on the data you are hosting on a particular Directory Server instance. The uid-uniqueness plug-in applies a check to a particular suffix in a Directory Server instance. If you are flowing entries that have identical uid attributes in the same suffix, then you must turn off the uid-uniqueness plug-in in the Directory Server. Turning off the plug-in prevents errors arising from the check. Errors arising from a uid-uniqueness violation will be written to the join engine logs with an `OBJECT_VIOLATION` message.

For example, if you have the entry `uid=x,ou=cv1` in a Connector View containing the suffix `o=siroe.com`, and you flow the entry to the Meta View, `o=mv`, the uid-uniqueness can remain enabled because the uid-uniqueness applies to a particular suffix and there is no conflict. You will be creating `uid=x,o=mv` for the meta-view entry and `uid=x,o=sunone` for the Connector View.

However, if the Meta View has `ou=mv1,o=siroe.com`, then there will be a conflict with the Connector View under the same `o=siroe.com` suffix: `uid=x,ou=cv1,o=siroe.com`. In this case, you must disable the uid-uniqueness plug-in if the data contains a `uid` attribute.

Disabling the uid-uniqueness Plug-In

1. Open the Directory Server console.
2. Choose the Configuration tab and select Plug-Ins in the navigation tree.

   The list of available Directory Server plug-ins displays in the navigation tree.
3. Select the uid-uniqueness plug-in.

   Details for the plug-in display in the right pane.
4. Deselect the Enable Plug-In check box.
5. Click Save and restart the Directory Server.

Setting the Referential Integrity Postoperation Plug-In

Normally, you will need to disable the Directory Server referential integrity plug-in in the Directory Server instance(s) that host Meta-Directory components.

When referential integrity is enabled, Directory Server will not write the changes that it makes to the change log. As a result, changes made to data cannot be detected by the Meta-Directory components and the Meta-Directory views will not be properly updated.

CAUTION In cases where there are users and groups, you must watch out for a side effect when you disable the referential integrity plug-in.

If you delete users belonging to a group (the user entries, as opposed to their group memberships), the group will still list the memberships of the users you have deleted. In these cases, you must manually delete the respective memberships from any groups listing the deleted users.

In very special circumstances, it is possible to keep the referential integrity plug-in enabled. Specifically, you can enable the referential integrity plug-in if all changes to data occur in one Meta-Directory view (for example, if all modifications are made in the Meta View or if they are made in an external data source that populates a Connector View). In this scenario, data modifications made in one view will be synchronized to the other Meta-Directory views when that view is refreshed. Here, data modifications do not rely on the change log to be synchronized to other views since all changes will flow from a single out to the other views. Note, however, that there may be a significant lag time between the refresh and the clean up done by the plug-in. Because of this, it is best to manually refresh the view after you make any data modifications.

Disabling the Referential Integrity Postoperation Plug-In

1. Open the Directory Server console.
2. Choose the Configuration tab and select Plug-Ins in the navigation tree.

   The list of available Directory Server plug-ins displays in the navigation tree.
3. Select the referential integrity postoperation plug-in.

   Details for the plug-in display in the right pane.
4. Deselect the Enable Plug-In check box.
5. Click Save and restart the Directory Server.

Enabling Retro Change Log Trimming

The modification process of Add, Delete, or Modify is stored in the *Directory Server Retro Change Log* suffix database. After a period of time, depending upon the number of updates, the `cn=changelog` suffix could reach its maximum limit. Thus, it is recommended to minimize the number of old entries to enhance database

performance. Once the Join Engine processes the changelog entries, then, the entries can be deleted. The time period between recorded changelog entry and the Join Engine processing the entry depends upon the site configuration. Typically, if the Join Engine is running the default DCNS scheduling for every 15 seconds, then, the Join Engine would check the changelog suffix in the Directory Server for the entries. You can use the “Retro change log trimming” feature of the Directory Server to limit the size of this database.

For more information, see Chapter 8, “Using the Retro Change Log Plug-in”, “Trimming the Retro Change Log” section in the *Sun ONE Directory Server 5.2 Administration Guide.*

Chapter 4

Meta-Directory Performance Tuning

There are several ways to tune the processing performance of Meta-Directory. In tuning, the overall goal is always to speed the flow of data through the Meta-Directory system. However, there are several systems to consider when you tune performance. For example, there are various configuration settings that affect the flow of data through the Meta-Directory components. Also, there are configuration settings that you can adjust to fine tune the performance of the Directory Server instances that you use in conjunction with your Meta-Directory system.

This chapter focuses specifically on the settings you can configure to speed the flow of data through the Meta-Directory components. The next chapter, Chapter 5, "Tuning Directory Server,"discusses the particular things you can do to tune the performance of the different Directory Sever instances running in your Meta-Directory system.

When configuring the Meta-Directory components and the Directory Server instances that host them, keep in mind that, compared to other Directory Server clients, Meta-Directory is particularly write intensive (especially to the Meta View). Because of this, it's particularly important to optimize the write performance of the Directory Server instance that's hosting the Meta-Directory components. This is very different than tuning the performance of a Directory Server instance that is hosting a user directory. These are normally optimized for read performance.

This chapter contains the following topics:

Adding Meta-Directory Indexes

To improve the performance of the Meta-Directory system, you should create indexes for certain attributes in the Meta-Directory object class. The Meta-Directory object class attributes are contained in the Meta-Directory schema—you must first load the Meta-Directory schema before you can index these attributes. This is discussed in the section "Loading Meta-Directory Schema" on page 77.

The Meta-Directory indexes are created and maintained by Directory Server. You create the indexes on each Directory Server instance based on the Meta-Directory views that are hosted by that Directory Server instance. In all, you can create indexes on the following five Meta-Directory object class attributes:

- `hasSubordinates`
- `mdsLinkToCV`
- `mdsLinkToMV`
- `mdsExcludedCVs`
- `mdsExcludedMVs`

You can add these indexes through either the Directory Server console or from the command line.

Determining Which Indexes to Create

You create the indexes only on Directory Server instances that host Meta-Directory components. For example, if a particular Directory Server instance hosts only a Meta View, then you should create and maintain the following Meta-Directory indexes on that Directory Server instance:

Table 4-1 Indexes for Directory Server Instances That Host a Meta View

Index Attribute	Index Settings
`hasSubordinates`	Equality
`mdsLinkToCV`	Presence and equality
`mdsExcludedCVs`	Presence and equality

Likewise, if a Directory Server instance hosts only Meta-Directory Connector Views, then you need to create the following indexes on that Directory Server instance:

Table 4-2 Indexes for Directory Server Instances That Host Connector Views

Index Attribute	Index Settings
`hasSubordinates`	Equality
`mdsLinkToMV`	Presence and equality
`mdsExcludedMVs`	Presence and equality

In addition to the above guidelines, you should also add the Connector View indexes listed in Table 4-2 to a Directory Server instance that hosts the Meta-Directory configuration (`o=NetscapeRoot`) if that Directory Server instance also hosts a database (Oracle) Connector View.

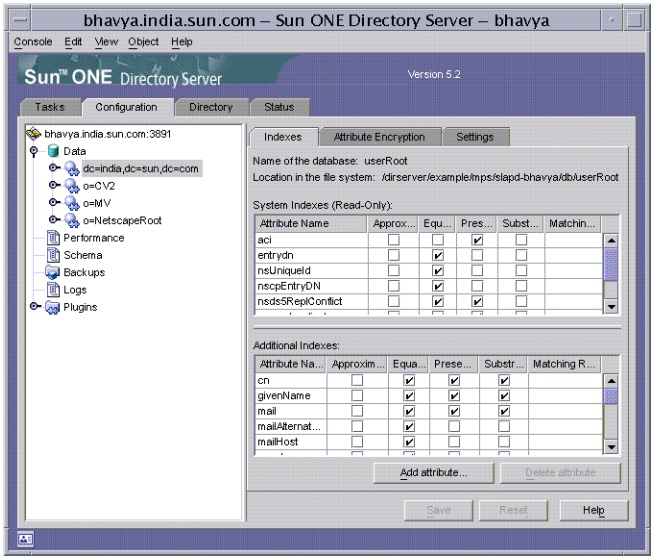
To Add Indexes From the Directory Server Console

To index attributes from the Directory Server 4.1*x* console, do the following:

1. Open the Directory Server console.
2. Navigate to the database index pane:
   - In the Sun ONE Directory Server 5.x console, choose the Configuration tab, expand the Data node, then select Database Settings in the navigation tree.
   - In the Netscape Directory Server 4.1*x* console, choose the Configuration tab and select the Database node in the navigation tree.

   The right-hand pane will display the attributes that are currently indexed.
3. In the right-hand pane, click Add Attribute.
4. Select the attribute name you want to index and click OK.
5. Select the indexes you want to create for the attribute (see Figure 4-1).
6. Repeat Steps 3 through 5 for each attribute listed in Table 4-1 and Table 4-2.
7. Click Save to apply changes.
8. Stop and restart the server.

Figure 4-1 Adding Indexes for Meta-Directory Attributes



Adding Indexes From the Command Line

In iDS 5.0, you load dse.ldif, indexes must be in LDIF, not as below. It's highly recommended that you only do this from the Console.

To add the necessary index attributes from the command line, open and modify the file slapd.ldbm.conf:

1. From a command prompt, stop the Directory Server using the stop-slapd command.
2. Open the file slapd.ldbm.conf (located in the directory NETSITE_ROOT/sloped-ServerInstance/config) and add the following lines in accordance with the guidelines listed in "Determining Which Indexes to Create":

   "index hasSubordinates eq"

   "index mdsLinkToCv pres,eq"

   "index mdsLinkToMv pres,eq"

   "index mdsExcludedCvs pres,eq"

   "index mdsExcludedMvs pres,eq"
3. Save and close the file.

4. Restart the server.

Bulk Loading Data

There are certain adjustments that you can make to speed the time it takes to populate a Meta View with entries from a new data source.

In particular, you should adjust the database cache sizes and the All IDs Threshold before you begin to load your initial sets of data into Meta-Directory.

Adjusting the Database Cache Size

If you are importing a very large database from LDIF, set the Maximum Cache Size parameter as large as possible. The larger you can set this parameter, the faster your database will be imported.

Use the following rule-of-thumb to calculate the maximum cache size:

1. Determine how much free memory you have on your system.
2. Divide the amount of free RAM by two and subtract 1 Mb from the subtotal.

For example, if you have 50 Mb of free memory on your system, divide 50 by 2 (giving a subtotal of 25 Mb) and subtract 1 MB, giving you a final total of 24 Mb. Here, you would set the Maximum Cache Size in Bytes parameter to `24000000`.

At a minimum, you should set the database cache size to 10Mb.

To Set the Maximum Cache Size

To set the Maximum Cache Size in the Sun ONE Directory Server console:

1. Open the Sun ONE Directory Server console.
2. Choose the Configuration tab, expand the Data node, then select Database Settings in the navigation tree.
3. In the right pane, choose the LDBM Plug-in Setting tab.
4. Enter your new setting in the Maximum Cache Size input box.
5. Choose Save to save the new setting.

To set the Maximum Cache Size in the Netscape Directory Server 4.1*x* console:

1. Open the Directory Server console.

2. Choose the Configuration tab and select the Database node in the navigation tree.
3. In the right pane, choose the Performance tab.
4. Enter your new setting in the Maximum Cache Size input box.
5. Choose Save to save the new setting.

Adjusting the All IDs Threshold

The All IDs Threshold is used to limit the size of Directory Server indexes (for a full description of this feature, see "Tuning the All IDs Threshold" on page 104).

Directory Server sets the default value for All IDs Threshold to 4,000. For Meta-Directory, this number is usually too low. For example, if you are flowing 100,000 entries from a Connector View to an empty Meta View, the recommended All IDs Threshold setting is 100,100. If you are flowing 300,000 entries, then set the All IDs Threshold to 300,100.

You should not bulk-load data into a Meta-Directory view until you have adjusted this setting.

NOTE The All IDs Threshold setting alters the way indexes are written, so you must rebuild the indexes after adjusting this setting. Re-importing the data is the fastest and most direct way to create a fresh set of indexes.

For information on tuning the All IDs Threshold, refer to the *Sun ONE Directory Server Administrator's Guide.*

Tuning the Data Servers

Meta-Directory has tuning parameters that pertain to the data servers that Meta-Directory accesses. You can configure these settings through the Data Server > Tuning tab in the Meta-Directory console.

In particular, you should configure the Time of Search to fit your specific system needs.

You can configure the following data server settings:

DCNS Scheduler

To configure and tune the Data Change Notification System (DCNS), use the DCNS Schedule tab in the Data Server window of the Sun ONE Console. The parameters in this window specify the change log polling schedule and the maximum number of entries to be obtained for each poll of the change log.

The default settings specify that a change log poll will be issued every 15 seconds with a maximum number of 1024 entries to be obtained per poll. To optimize these settings for Meta-Directory, you should adjust the change log settings so it is polled after the join engine has processed at least 50 to 60 percent of the entries that it obtained from the previous poll. This setting will reduce memory consumption because it reduces the number of unprocessed entries added to the queue.

Although you should calculate your system performance, you can form a rough estimate by assuming the average time it takes to move an entry from the change log to the destination Connector View is one second.

For example, if you pull 1024 entries from the change log with each poll, the poll interval should be 512 seconds or more. Another way to reduce the amount of entries queued (and thus reduce memory consumption) would be to increase the polling interval, giving the join engine more time to process entries in the queue before the next poll.

When tuning the polling interval, consider the number of entries being obtained per poll rather than something such as the change log size. Results from queries are maintained internally. Results can overflow the heap when relatively large numbers of entries are obtained per poll.

Configuring the Time of Search

From the Meta-Directory Console, you can specify several tuning parameters for the different Directory Server instances that you have running in your Meta-Directory setup. Of particular interest is the amount of time that a data server should spend before timing-out on a specific search.

By default, the search time is set to 3,600 seconds (one hour). While this should be enough for most situations, you might need to adjust this setting if your system is deprived of resources.

If a system time-out occurs, the Directory Server in question will stop processing. If the log level is set to `dataaccess=3`, the following message will be written to the log file: `DA_USER_TIMELIMIT_EXCEEDED`.

To adjust the time of search:

1. Open the Meta-Directory console.
2. In the left pane, select Meta-Directory in the navigation tree.
3. In the right pane, choose the Data Servers tab to view the list of Directory Server instances used in your Meta-Directory setup.
4. Select the Directory Server instance which you need to adjust. A list of tabs will appear in the pane below.
5. Select the Tuning tab, then specify the new time-out setting in the Maximum Operation Result Time input box.
6. Click Save.

Tuning the I/O Block Time-Out Setting

In Directory Server 4.1*x*, you should adjust the IO Block Time Out setting to prevent the Directory Server from timing out during the time it takes the join engine to synchronize Meta-Directory views. This setting is located in the Directory Server configuration file, found at the following location:

`slapd_`*`SERVER_ROOT`*`/config/slapd.ldbm.conf`

The *IO Block Time Out* setting controls the time-out period for stalled clients. The default setting for the Directory Server's `ioblocktimeout` parameter (180,000,000 ticks on the computer's system clock) may not be great enough to allow the synchronization of a large number of entries. As a result, the Directory Server may time out the synchronization process before your data is fully synchronized. You are more likely to see this on slower clients.

The default value of the `ioblocktimeout` parameter (stored in `slapd.conf`) of 180,000,000 ticks is usually enough. However, keep in mind that the number of ticks per second is computer and platform-dependent. If you see time-out errors, you may need to increase this value.

For more information about the `ioblocktimeout` parameter, refer to the *Sun ONE Directory Server Administrator's Guide.*

Chapter 5

Tuning Directory Server

Sun ONE Meta-Directory uses Sun ONE Directory Server to store its configuration settings. Meta-Directory also uses the Directory Server to store the entries contained in the Meta View. In addition to these required uses of the Directory Server, it's likely you will also use Meta-Directory to synchronize one or more Directory Server user databases.

This chapter describes how to optimize the Directory Server performance when you use it in conjunction with Meta-Directory.

While there are several ways to configure and tune the performance of Sun ONE Directory Server, performance tuning can be grouped into two broad categories:

- Write performance tuning
- Read performance tuning

The main goal of optimizing Directory Server performance with regard to Meta-Directory is to maximize the number of Directory Server operations per second. In light of this, consider the following:

- A single search (read) request takes about 50 to 250 Directory Server operations.
- A single modify (write) request takes about 200 to 1,000 Directory Server operations.

In addition to the number of operations it takes per request, take into consideration that disk access is thousands of times slower than physical memory access. The nature of write requests dictates that they access the disk for each request. This makes them many times slower than read operations, which can often be completed quickly by accessing physical memory caches.

Compared to other Directory Server clients, Meta-Directory is particularly write intensive. Because of this, it is important to optimize the write performance of the Directory Server that's being used in conjunction with Meta-Directory. This chapter begins with a section on how to optimize the performance of Directory Server write operations, then it discusses optimizing Directory Server read performance.

Directory Server Write Performance Tuning

Simply put, write performance is the amount of time it takes to make an update to the Directory Server data stored on disk. When optimizing write performance, you must consider not only the database entry containing you're modifying, but also the many different files that reference the entry. When you add or modify a Directory Server entry, the following write operations take place:

- The new data is written to the entry.
- Any indexes containing attributes in the entry are updated.
- The update is noted in the transaction log.
- The access and audit logs are updated.
- The replication change log is updated if replication is to be performed.
- The Retro Changelog is updated.

Since write performance is directly related to the amount of information that must be written to disk, anything that reduces or eliminates disk traffic will speed updates. This being the case, write-tuning efforts are centered around optimizing the traffic into and out of the disk subsystem. This optimization comes in the following two forms:

- Minimize the amount of information that must be written to disk - such as reducing logging levels.
- Increase the throughput of disk write operations by using faster disks or by spreading the load across multiple disks.

Managing the Information Written to Disk

Although there is not much you can do about optimizing the actual database file (it must be modified whenever there is a change), there are ways that you can minimize the amount of disk writes by carefully selecting the files that are maintained by the Directory Server. These files can often be spread over multiple disks and controllers to improve performance.

Optimizing Indexes

While indexing makes searches much faster, they must be updated by the directory for each write operation. When creating indexes, you must balance the needs of fast data access to directory write performance.

You can make the biggest gain by removing any indexes that are unneeded. If your applications never search on a particular attribute or do so infrequently, then putting an index on that attribute will only slow directory writes without providing any search performance gains. When you first install Directory Server, it is configured with a certain set of default indexes. You should always examine the list of default indexes and remove any that are unnecessary.

It is also important to consider the types of indexes the directory is maintaining. Each type of index, while useful for optimizing particular types of queries, is also associated with a particular cost whenever that index must be maintained. Table 5-1 illustrates the relative cost of maintaining certain types of indexes. The cost is expressed in terms of the number of logical database writes associated with maintaining the index for a given value.

Table 5-1 Relative Cost to Index a Value

Index Type	Cost of Indexing a Value	Example Value “first, middle, last”
Presence	1 (binary)	1
Equality	1 (binary)	1
Approximate	Number of words in value	3
Sub-string	Number of characters in value	17

From the table above you can see that substring indexes are far more expensive to maintain than equality or presence indexes.

To optimize indexes, always use the least expensive type of index based on the types of queries being made to an attribute. For example, if you rarely expect to execute a wildcard search on a "uid" attribute, then you should not maintain a substring index for that attribute.

For more information on indexes and their types, refer to the chapter “Introduction to Directory Server” in the *Sun ONE Directory Server Deployment Guide*. In addition, the “Managing Indexes” chapter in the *Sun ONE Directory Server Administrator’s Guide* contains details on how to configure Directory Server indexes.

The Transaction Log

Sun ONE Directory Server keeps a transaction log that tracks the operations of all the databases it manages. Whenever a database operation (such as a write) is performed, the server logs the operation to the transaction log. If the Directory Server experiences a failure, it uses the transaction log to recover any affected databases.

In a system that performs many writes (such as a with a Meta-Directory system), activity to the transaction log can become significant. To optimize system performance, you want to specify a location for the transaction log that is different from the user database accessed by the Directory Server. This will minimize conflicting disk seeks and writes between the two files.

You can also increase Directory Server write performance by specifying the log file’s checkpoint tag. For details on configuring the transaction log, see the “Tuning Directory Server Performance” chapter of the *Sun ONE Directory Server Administrator’s Guide*.

The Access, Error, and Audit Log Files

The Directory Server also maintains several user log files. Unlike the transaction log, which is stored in DB2 format, the user logs are stored in ASCII text. Specifically, the Directory Server has the following user logs:

- The *access log* tracks front-end Directory Server events such as authentication and binding.
- The *error log* tracks back-end Directory Server events.
- The *audit log* is similar to the transaction log, except it is kept in user-readable ASCII text.

While you can configure each of these logs in various ways, the most import optimization concern is the placement of the logs. Similar to the transaction log, you should locate the user logs in a disk partition that is different from the partition containing the user database.

In addition to controlling the location of the logs, you can also control the size of the logs (based on number of entries per log file or length of time before a new log file is opened), level of verbosity in the logs, and sometimes the deletion of the log files.

For details on configuring the Directory Server user logs, refer to the chapter titled "Monitoring Server and Database Activity" in the *Sun ONE Directory Server Administrator's Guide*.

The Replication Change Log

The Directory Server can be configured for database replication, however this is not required for Sun ONE Meta-Directory to function. In replication, every supplier server maintains a change log. A change log is a record that describes the modifications that have occurred on a replica. When an entry is modified, a change record describing the LDAP operation is recorded in the change log for the respective replica. The supplier server uses the change log to pass modifications made on one replica to the other consumer servers (or in the case of multi-master replication, on other master servers).

Like the user log files, you can specify the location of the change log and how large it should grow before a new change log file is opened. For details on replication and the change log, refer to the chapter "Managing Replication" in the *Sun ONE Directory Server Administrator's Guide*.

Summary of Write Tuning

While this section deals almost entirely with how to configure Sun ONE Directory Server, there are some important considerations to keep in mind when tuning Directory Server with regard to Sun ONE Meta-Directory. To summarize:

- As much as possible, keep the user directory information tree (DIT) on a disk partition that is separate from the partition storing configuration, index, and log files.
- Remove any unnecessary indexes. Many of the default indexes supplied in the directory might not be necessary for your specific database needs. Pay special attention to substring indexes.

- Minimize or eliminate logging. Turn off the access or audit logs if you don't use them on a regular basis or require them for audit purposes.

Tuning Directory Server Read Performance

By design, Sun ONE Directory Server is tuned to perform fast read (search) operations. However, by carefully adjusting some of the Directory Server settings, you can optimize the read performance. Read performance settings can be grouped into the following categories:

- Back-end (database) settings
- Front-end settings

Back-end settings affect the physical make up of the caches and indexes while front-end settings can be used to tune the ways in which clients access the Directory Server indexes and database.

Directory Server Back-End Settings

The Sun ONE Directory Server read and search performance is based on two key back-end technologies: database caching and indexing.

Virtually all of your read and search performance efforts should be directed at optimizing the performance of the Directory Server caches and indexes. Tuning other aspects of the Directory Server will provide minimal benefits if the indexes and caches are not properly tuned.

- **Caching** speeds the processing of searches by storing indexes and entries in physical memory as opposed to on a disk. You tune the performance of the Directory Server by reducing the number of database reads the server has to perform in order to retrieve the desired entries. You can increase Directory Server performance by optimizing the index-to-entry cache ratio.
- **Indexing** speeds the processing of searches by storing the values of specified attributes. If you perform a search on an attribute that has been indexed, the Directory Server can quickly search the index for the entries that meet the search criteria. You can configure the indexes created, tuning the Directory Server based on the specific types of queries the directory has historically been asked to perform.

When optimizing the performance of the Directory Server back-end, you must balancing the configuration of the following cache and index settings:

- Cache settings
  - Database cache
  - Entry cache
- Index settings
  - All IDs threshold

Tuning the Directory Server Caches

The Directory Server utilizes caches to speed search performance by storing database information in physical memory. In addition to faster reads, caches increase the speed of query returns because entries stored in the entry cache have already been converted from DB2 database format to LDAP format.

Sun ONE Directory Servers has two distinct types of caches:

- The *database cache* stores pages from the database, caching both indexes and data.
- The *entry cache* stores the most recently access entries form the directory. It uses a least-recently-used algorithm to ensure that the most frequently accessed directory entries are available in physical memory.

While the database cache is set to the size of the maximum amount of physical memory it should consume, the entry cache setting is based on the maximum number of entries it should hold.

To maximize directory read performance, you must cache as much directory data in physical memory as possible. By preventing the directory from having to read information from disk, you can increase read performance by eliminating the disk subsystem. When sizing the cache settings, there are three rules you must follow:

- **Your database cache must always be large enough to hold the databases indexes.**

  While the database cache should hold more than just the indexes, you must always ensure that it is at least big enough to hold the directory indexes. If it isn't, the directory will be forced to read indexes from disk for every search request, which will quickly bring directory throughput to a virtual halt.

- **Your database and entry caches must always fit into available, physical memory.**

  If the size of the two caches combined is bigger than the amount of available physical memory on the machine, the operating system will begin to use virtual memory, swapping the cache to and from disk. This can cause a significant amount of disk "thrashing" that will quickly bring not just the directory, but the entire system to a virtual halt.

- **The database cache is more important than the entry cache.**

  When given a choice between allocating memory to the database cache or to the entry cache, you should generally favor the database over the entry cache.

Sizing the Database and Entry Cache Settings

To maximize Directory Server read performance, you will want to allocate as much available memory as possible to the caches. As very general rule, efficient operation can be achieved by allocating memory between the caches in the ratio of 75 percent for the database cache and 25 percent for the entry cache.

Sizing the Database Cache

The size of the database cache can be set through the Sun ONE Console. It is important to note, however, that the actual amount of memory used by the database cache can exceed the size you specify by up to 25 percent. This is due to additional memory required to manage the cache itself. Also, it is important that the database cache (with overhead) should not be set to consume over two gigabytes (2 Gb) of memory. It is not capable of using more memory than that.

Sizing the Entry Cache

Unlike the database cache, the entry cache size is set not by the amount of memory you would like it to consume but by the maximum number of entries you would like it to hold. The actual amount of memory it will consume is a function of the average entry size.

For example, if your average entry size is 1Kb, and you specify that the entry cache should hold a maximum of 10,000 entries, then the amount of memory the entry cache will consume will be 12.5 Mb, calculated as follows:

(1 Kb / entry * 10,000 entries) = 10 Mb + 25 percent for cache management overhead.

To determine the average entry size, you will need to use your best judgment of the data the directory will hold. You can determine the size of an entry by adding the number of characters needed to list the entry and adding a carriage-return and line-feed for each line in the entry. For example, the entry below for Lenny Riceman is 398 bytes.

```
dn: uid=lriceman, ou=People, o=siroe.com
cn: Lenny Riceman
sn: Riceman
givenname: Lenny
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
ou: Accounting
ou: People
l: Sunnyvale
uid: lriceman
mail: lriceman@siroe.com
telephonenumber: +1 408 555 4798
facsimiletelephonenumber: +1 408 555 9751
roomnumber: 4612
userpassword: cocoloco
```

Computing the Database

While bigger caches are usually better, performance can degrade if `dbcachesize` is too large. The database cache size exceeds the available free memory, the process will begin to thrash virtual memory as it writes clean and dirty pages to disk, resulting in severe performance degradation. (You can detect this behavior by using one of the system monitoring tools such as `vmstat` on Solaris that provides statistics for virtual memory usage.)

To size your database and entry caches, use the following technique:

- database cache (assuming 100% of your free memory can be given to Sun ONE Directory Server)

1. Identify the amount of free memory available when your system configuration is operating normally (use system monitoring tools).
2. Take 75 percent of your free memory as the base of your database cache size.

3. Divide the base database cache size (computed above) by 1.25 to account for the cache overhead, and use the result as the database cache size value.

   If the result is larger than 1.6 Gb, reduce it to 1.6 Gb, the maximum dbcache size.

   If the result is smaller than the size of all database indexes, increase the database cache to cover the size of all the indexes without exceeding available memory. Calculate the total index size (add up the sizes of `*.db2` files in the Directory Server database directory and subtract the size of user database file `id2entry.db2`).

- entry cache

1. Take 25 percent of your free memory and use this as the base for your entry cache size.
2. Divide the base entry cache size by 1.25 to account for cache overhead.
3. Divide the result by the average entry size and use the result as the entry cache size (maximum number of cache entries).

   Determine the size of an entry by adding up the size of all its attributes. Examine your data closely to determine average entry size.

A Short Example

As an example, assume your system has the following properties:

- 150 Mb of available physical memory (RAM)
- An estimated entry size of 1Kb / entry

To size the database cache:

150 Mb * .75 = 112 Mb to allocate to the base database cache

112 Mb / 1.25 = 90 Mb for the Database cache

To size the entry cache:

150 Mb - 112 Mb = 38 Mb for base entry cache

38 Mb / 1.25 = 30 Mb avail for entries

30 Mb / 1 Kb = max 30,000 cache entries

To configure Directory Serve cache sizes, do the following:

1. From the Directory Server console, select the Configuration tab.
2. Navigate to the database performance configuration settings:

- In the Sun ONE Directory Server console, expand the Data node in the navigation tree, then select Database Settings. In the right pane, choose the LDBM Plug-In Setting tab.
- In the Netscape Directory Server console, select Database node in the navigation tree, then select the Performance tab in the right pane.

The current database performance settings appear.

3. Enter the amount of memory you want to make available for open index files in the Maximum Cache Size text box.
4. In Directory Server 4.1*x*, enter the number of entries you want the server to keep in memory in the Maximum Entries in Cache text box (note that Sun ONE Directory Server does not support this configuration setting).

Monitoring the Database Cache

Once you have set your initial cache sizes, you should monitor your cache utilization from time to time to ensure your caches are being used efficiently. You can determine the effectiveness of your caches by examining the Database Performance Counters available through the Sun ONE Console. you can view the performance counters by selecting the Status tab in the Directory Serve console.

Under optimal conditions, both the entry and the database cache hit ratios will be above 95 percent. If either hit ratio is less than 95 percent, and you have additional available physical memory, you should consider increasing your cache size to increase the hit ratio.

If you have been running the Directory Server for several days and find that your entry cache is not filled to the maximum level, you might also consider lowering the size of your entry cache to just above the high-water mark and giving any freed memory to the database cache.

For information about changing entry cache settings using the Sun ONE Console, refer to the *Sun ONE Directory Server Administrator's Guide.*

Monitoring Performance From the Command Line

In Netscape Directory server 4.1x, you can also monitor the database cache settings through the entry "`cn=monitor, cn=ldbm`". To read the database monitor entry from the command line, use the following command:

```
ldapsearch -b "cn=monitor,cn=ldbm" -s base "objectclass=*"
```

The cache parameters are listed in Table 5-2:

Table 5-2 Cache Parameters for ldapsearch Command

Console Label	Monitor Entry Attribute	Description
Entry cache hits	`entrycachehits`	Number of requests filled from the entry cache.
Entry cache tries	`entrycachetries`	Number of total requests to the entry cache.
Entry cache hit ratio	`entrycachehitratio`	Percentage of requests filled from the entry cache.
Current number of entries in entry	`currententrycachesize`	Current number of entries in the entry cache.
Hits (under the Database Cache heading)	`dbcachehits`	Number of requests filled from the database cache.
Tries (under the Database Cache heading)	`dbcachetries`	Number of total requests to the database cache.
Hit ratio (under the Database Cache heading)	`dbcachehitratio`	Percentage of requests filled from the database cache.

Sizing the Entry Cache to Prevent Over-Allocation of Memory

In some cases, the Directory Server entry cache settings may be set to a value that results in the over-allocation of memory to the ns-slapd process, resulting in the slow but consistent growth of the `ns-slapd` process until it is so large that it causes every operation of the machine to require an excessive amount of time, and the Directory Server needs to be restarted because of the memory over-allocation.

The entry cache, which is used to store a copy of LDAP entries as they are being read, is most useful when the same entries are being pulled up repeatedly within a reasonable period of time. In the case of an ISP, at a given time there are usually fewer repeats of previously searched LDAP entries than there are original searches. In this case, or in similar situations where there are large numbers of users performing searches, it may be desirable to lower the entry cache, especially if the size of the entry cache is occasionally contributing to the over allocation of memory to the `ns-slapd` process.

Tuning the Directory Server Indexes

Proper indexing is the most important thing you can do to improve the Directory Server read performance. To optimize the indexing of your database, you need to understand the types of searches that are being submitted to your Directory Server database. Knowing this, you can create indexes for the attributes that are used in the searches.

For example, if your only application is a mail server, it will most likely search only for an exact match on the `UID` attribute. You would therefore set an index on the `UID` attribute. Setting indexes on other attributes would be necessary only if your Directory Server is accessed by other types of applications.

You can determine the types of searches your directory is handling by examining the access log. The access log records search requests in `SRCH` records. For example, you might see the following log entry:

```
[05/Mar/2001:09:23:35 -0800] conn=4 op=6
  SRCH base="ou=People,o=siroe.com" scope=2 filter="(uid=jsmith)"
```

By looking through the access log for all search requests, you can determine the type and frequency of the queries being performed. Your goal is to ensure that all commonly performed searches are fully indexed, meaning all attributes used in a search has a corresponding index.

NOTE While indexing makes searches much faster, the server must manage the indexes you create whenever it writes an entry. As a result, maintaining indexes causes adds, modifies, deletes, and imports to be slower than they would be without indexes. Because of this, you must strike a balance between read performance and write performance. Write performance, and how indexing affects it, is discussed in the section above, “Directory Server Write Performance Tuning.”

It’s important to keep in mind that any search on an unindexed attribute will require the server to physically look at each entry in the directory tree (DIT) to determine if it meets the search criteria. Searching the entire DIT is slow and expensive.

With Sun ONE Directory Server, you can use the access log to directly locate searches that referenced unindexed attributes. Do this by searching the access log for `RESULT` records that contain a `notes=U` entry. For example:

```
[02/Mar/1999:11:42:49 -0800] conn=4 op=6 RESULT err=0 tag=101
  nentries=1 etime=0 notes=U
```

By matching the connection (conn) and operation number (op) fields with the corresponding SRCH record, you can determine which searches are being performed without indexes.

When configuring indexes, it is also important to use the correct type of index to match the types of searches being performed. Using incorrect index types will at best impact the write-performance of the directory; at worst, incorrect index types provide the equivalent of using no indexes at all.

Indexing and index types are described in depth in the Sun ONE Directory Server documentation. Specifically, the "Introduction to Directory Server" chapter in the *Sun ONE Directory Server Deployment Guide* contains an overview of indexing and index types, while the "Managing Indexes" chapter in the *Sun ONE Directory Server Administrator's Guide* provides details of creating and maintaining indexes.

Tuning the All IDs Threshold

Each index that the directory server uses is comprised of a table of index keys and matching list of entry IDs. That is, for each index key there is a list of directory entry IDs that match the key. This entry ID list is used by Directory Server to build a list of candidate entries that may match a specified search filter.

Directory Server sets a size limit for each entry ID list. This size limit is globally applied to every index key managed by the server and it is called the *All IDs Threshold*. When the size of an individual entry ID list reaches this boundary, the server replaces that entry ID list with an *All IDs token*.

The All IDs token causes the server to assume that all directory entries match the index key. In effect, the All IDs token causes the server to behave as if no index was available for that particular key and a search on it will cause the server to search the entire directory database. The assumption is that some other aspect of the search filter will allow the server to narrow its candidate list before processing the list.

For more details on the All IDs threshold and how to calculate the best setting, refer to the chapter titled "Managing Indexes" in the *Sun ONE Directory Server Administrator's Guide*.

Directory Server Front-End Settings

The front-end server settings let you manage your server's performance by limiting the amount of resources the Directory Server puts into client search requests. Although LDAP clients can specify that the server use smaller values for the Size Limit and Time Limit settings, you should review the following configuration settings to optimize system performance:

- **Size limit** specifies the maximum number of entries the server will return to the client in response to a search operation. If this limit is reached, the server returns any entries it has located that match the search request, as well as an exceeded size limit error. The default value for this parameter is 2,000. Decreasing this value could reduce your average search time but will also limit the number of results returned on very large searches.
- **Time limit** specifies the maximum amount of real time (in seconds) you want the server to spend performing a search request. If this limit is reached during a search, the server returns any entries it has located that match the search request, as well as an exceeded time limit error. The default value for this parameter is 3,600 (1 hour). Decreasing this value will produce similar results to decreasing Size Limit.
- **Idle Time-out** specifies the time (in seconds) you want the server to maintain an idle connection before terminating.
- **Look-through limit** specifies the maximum number of entries the server will check when seeking candidate entries in response to a search request. If this limit is reached, the server returns any entries it has located that match the search request as well as an exceeded size limit error.

  The default value for this parameter is 5,000. Decreasing this value could reduce the average search time per request but will also produce more exceeded size limit errors and empty search requests.

  As a guideline, set this parameter 10 percent above your value for Size Limit.
- **Maximum file descriptors** sets the maximum number of file descriptors available to the Directory Server. This setting effects the number of simultaneous connection that can be made to Directory Server.

To Modify the Directory Server Settings

1. Open the Directory Server console and select the Configuration tab.

2. From the Configuration tab, select the root entry in the navigation tree in the left pane.

   The server-wide configuration tabs appear in the right pane.

3. Select the Performance tab in the right pane.

   The current front-end server performance settings appear.

4. Specify the maximum number of entries the server will return to the client in response to a search operation by entering a new value in the Size Limit text box. If you do not want to set a limit, type `-1` in this text box.

5. Specify the maximum amount of real time (in seconds) you want the server to spend performing a search request in the Time Limit text box. If you do not want to set a limit, enter a zero (`0`) in this text box.

6. Specify the time (in seconds) you want the server to maintain an idle connection it terminates in the Idle Timeout text box. If you do not want to set a limit, enter a zero (`0`) in this text box.

7. Set the maximum number of file descriptors available to the Directory Server in the Max Number of File Descriptors text box. (Note that this option is not available on Windows NT or IBM AIX systems.)

While the values set on these parameters are hard limits on the resources that the server will apply to each request, they are not enforced on the root DN. For further details on how these parameters impact your server's searching performance, refer to the “Managing Indexes” chapter in the *Sun ONE Directory Server Administrator’s Guide*.

Index

F

I

J

M

N

O

P

S

T