

Solstice X.500 Directory Management

 **SunSoft**
A Sun Microsystems, Inc. Business
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.
Part No.: 802-5304-10
Revision A, January 1996

© 1996 Sun Microsystems, Inc. 2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A.

All rights reserved. This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Portions of this product may be derived from the UNIX[®] system, licensed from UNIX System Laboratories, Inc., a wholly owned subsidiary of Novell, Inc., and from the Berkeley 4.3 BSD system, licensed from the University of California. Third-party software, including font technology in this product, is protected by copyright and licensed from Sun's suppliers.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19. The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

TRADEMARKS

Sun, Sun Microsystems, Solstice, SunLink, the Sun logo, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd. OPEN LOOK is a registered trademark of Novell, Inc. PostScript and Display PostScript are trademarks of Adobe Systems, Inc.

All SPARC trademarks are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. SPARCcenter, SPARCcluster, SPARCcompiler, SPARCdesign, SPARC811, SPARCengine, SPARCprinter, SPARCserver, SPARCstation, SPARCstorage, SPARCworks, microSPARC, microSPARC-II, and UltraSPARC are licensed exclusively to Sun Microsystems, Inc. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK[®] and Sun[™] Graphical User Interfaces were developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

X Window System is a trademark of X Consortium, Inc.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN. THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAMS(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.



Contents

Preface.....	xiii
1. Overview	17
Directory Components and Protocols	17
Directory Information	19
Directory Information Base	19
Infrastructure Information	20
Schema.....	20
Knowledge Information	21
Managing Your Directory Service	22
Management Tools.....	22
The RFC1006 Driver	24
2. Initial Configuration.....	27
Default Configuration.....	27
Initial Configuration	28
▼ To Start the DSA	28

▼ Using x500servertool	28
▼ From the command line	29
▼ To Start the LDAP Server	30
▼ To Configure the DSA	31
▼ To Create a Database	32
▼ To Create a Name Context	33
▼ To Add a Remote DSA	33
▼ To Create Knowledge Information	35
▼ Checking References	36
▼ To Stop the DSA	37
3. Advanced Configuration	39
Composing Distinguished Names	39
Configuration Management	40
Directory Schema	41
▼ To Add a Schema Source File	41
▼ To Delete a Schema Source File	43
▼ To Modify a Current Schema	43
Known DSAs	44
▼ To Allow Communication with Known DSAs	44
▼ To Configure a Known DSA	45
▼ To Add a Known DSA Entry	45
▼ To Modify a Known DSA Entry	45
▼ To Delete a Known DSA Entry	46
▼ Checking DSA Availability	46

Access Controls	46
User Access Rights.....	47
Entry Access Controls.....	47
Attribute Access Controls.....	48
DSA Access Rights.....	48
How Access is Controlled.....	49
▼ To Set DSA Access Rights.....	49
Distributed Operations	51
▼ To Set the Default Distributed Operation.....	51
DSA Name Contexts	52
▼ To Edit the Name Contexts List.....	52
Tuning DSA Resource Usage	53
▼ To Tune DSA Resource Usage	54
Tuning Session Parameters.....	54
▼ To Set the Session Parameters	54
LDAP Configuration	55
▼ To Configure the LDAP Server	55
Accounting	56
▼ To Start Accounting.....	57
Error Reporting	57
▼ To Start Error Reporting	57
x500servertool Options.....	58
▼ To Modify the Map	58
▼ To Modify Date and Time Formats	58

4. Extending the Schema	59
How the Schema is Used.	60
Sending a Request to a DSA	60
Receiving a Request from a DUA	60
Receiving Information from a DSA	60
Schema Definition Example	60
Standard Schema Files.	64
Using a Non-Standard Schema	65
Guidelines for Extending the Schema	65
How to Extend the Schema Definition	66
Example of an Extended Schema	67
▼ To Define the Employee Object Class	68
▼ To Add an Employee Entry to the Directory	68
5. Schema Reference Information	71
Attribute Types.	71
Attribute Syntaxes	78
Attribute Sets	80
Object Classes	81
Access Control Information	87
Adding Access Control Attributes.	89
Security Attributes.	90
6. Data Management.	91
Importing Data.	91
x500import Command Syntax	91

Exporting Data	92
x500export Command Syntax	92
Deleting Entries	93
Import and Export Data File Format	93
Attribute Value Syntaxes	95
Any Syntax	97
Data File Example	98
Using the Data Editing Tool	98
▼ To Start x500datatool	98
Selecting an Entry in x500datatool	100
Updating Directory Information	100
▼ To Display the Attributes of an Entry	101
▼ To Modify an Entry	102
▼ To Add a Subordinate Entry	103
▼ To Add a Subordinate Alias	104
▼ To Delete an Entry	104
▼ To Move an Entry	105
▼ To Delete a Subtree	105
▼ To Move a Subtree	106
Backing Up Data	106
▼ To Backup Your Directory	106
▼ To Specify a Backup Schedule	107
▼ To Restore Your Directory	107
7. Troubleshooting	109

Accounting Information	109
Formatted Accounting Information	110
Unformatted Accounting Information	111
The x500trace Utility	112
Running the Trace Utility	112
Trace Output Example	113
Error Messages from x500servertool	114
Schema Error Messages	119
Error Messages from x500datatool	124
Directories and Files	127
Interworking Problems	127

Figures

Figure 1-1	The Directory Model.....	18
Figure 1-2	Directory Information Structure	20
Figure 1-3	Main Window of <code>x500servertool</code>	23
Figure 1-4	RFC1006 Driver and Higher Layer Entities.....	24
Figure 2-1	The DSA Menu.....	29
Figure 2-2	LDAP Server Menu.....	30
Figure 2-3	DSA Addressing Window	31
Figure 2-4	Add Remote DSA Window	34
Figure 2-5	Create Remote Reference Window.....	35
Figure 3-1	Using the Compose Feature.....	40
Figure 3-2	Schema Files Window.....	42
Figure 3-3	Access Control Window.....	50
Figure 3-4	DSA Tuning Window	52
Figure 3-5	Edit Name Contexts Window	53
Figure 3-6	DSA Session Parameters Window	55
Figure 3-7	LDAP Configuration Window.....	56

Figure 6-1	x500datatool Main Window.....	99
Figure 6-2	Entry Attributes Screen for an Organization Entry	102

Tables

Table 5-1	Preferred Delivery Methods	75
Table 5-2	Delivery Methods	79
Table 5-3	User Levels	80
Table 6-1	OR Address Field Keys	96
Table 7-1	x500trace Protocols	113
Table 7-2	x500trace OSI Processes	113
Table 7-3	x500trace Filters	113

Preface

Solstice X.500 Directory Management explains how to configure and manage a Solstice X.500 Directory Server. There are two Solstice X.500 products:

- **Solstice X.500 Directory Server** provides DSA and LDAP server software, and tools to configure and manage a directory server.
- **Solstice X.500 Client Toolkit** provides a DUA client, a client configuration tool, and an application programming interface. For information about this product, see *Solstice X.500 Client Toolkit Management*.

Who Should Use This Book

This book is intended for the person setting up and running a directory service using the Solstice X.500 software. It assumes that you have a basic knowledge of X.500 and general directory concepts, and that you are familiar with the Motif style of user interface.

How This Book Is Organized

Chapter 1, “Overview,” describes the Solstice X.500 directory software and how to use it to provide a directory service.

Chapter 2, “Initial Configuration,” describes how to configure a basic directory server using the default settings wherever possible.

Chapter 3, “Advanced Configuration,” explains all the configuration options.

Chapter 4, “Extending the Schema,” explains why the schema can be extended, and how to extend it.

Chapter 5, “Schema Reference Information,” contains information about the definitions in the standard schema.

Chapter 6, “Data Management,” explains how to use the utilities provided to manage the data in your directory.

Chapter 7, “Troubleshooting,” contains information about error messages and recovery procedures, including general advice about troubleshooting interworking problems. It also contains a list of the directories used by Solstice X.500.

Related Books

The following books are included in this documentation set:

- *Solstice X.500 Directory Management* (this book)
- *Solstice X.500 Client Toolkit Management*
- *Solstice X.500 XDS Programming Reference*
- *Solstice XOM Programming Reference*

What Typographic Changes Mean

The following table describes the typographic changes used in this book.

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. machine_name% You have mail.
AaBbCc123	What you type, contrasted with on-screen computer output	machine_name% su Password:
<i>AaBbCc123</i>	Command-line placeholder: replace with a real name or value	To delete a file, type <code>rm filename</code> .
<i>AaBbCc123</i>	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in <i>User's Guide</i> . These are called <i>class</i> options. You <i>must</i> be root to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

Table P-2 Shell Prompts

Shell	Prompt
C shell prompt	machine_name%
C shell superuser prompt	machine_name#
Bourne shell and Korn shell prompt	\$
Bourne shell and Korn shell superuser prompt	#

Overview



This chapter introduces the components of the Solstice X.500 Directory Server and the management tasks required. It also contains a summary of the tools you use to manage your directory service.

<i>Directory Components and Protocols</i>	<i>page 17</i>
<i>Directory Information</i>	<i>page 19</i>
<i>Directory Information Base</i>	<i>page 19</i>
<i>Infrastructure Information</i>	<i>page 20</i>
<i>Managing Your Directory Service</i>	<i>page 22</i>
<i>Management Tools</i>	<i>page 22</i>

Directory Components and Protocols

The Solstice X.500 software provides a directory system based on the model illustrated in Figure 1-1 on page 18.

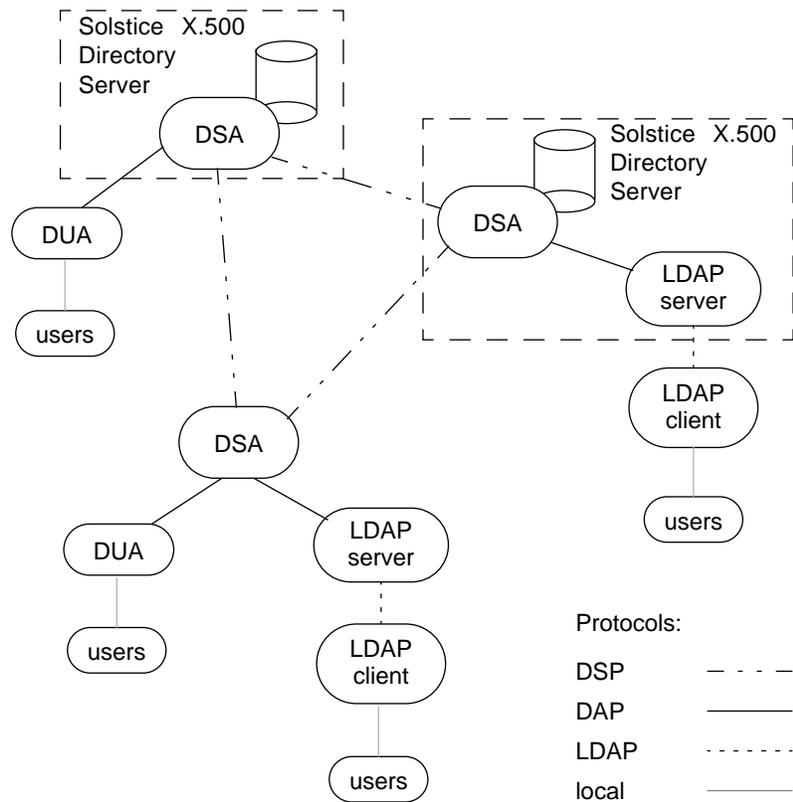


Figure 1-1 The Directory Model

A DSA is a *Directory System Agent*. It is an X.500 server. It accepts a request from a DUA, interacts with the directory to process the request and returns the result to the DUA.

A DUA is a *Directory User Agent*. It is an X.500 client. It interacts with the directory service on behalf of a user. It submits a request to the directory, receives the result from the directory and presents it to the user.

DSAs communicate with each other using the *directory system protocol*, DSP. DUAs and DSAs communicate using the *directory access protocol*, DAP. There is also a *lightweight directory access protocol*, LDAP.

An *LDAP client* is a user agent that accepts requests from a user and uses the LDAP protocol to communicate with an LDAP server. An *LDAP server* behaves as a server to LDAP clients. It uses DAP to communicate with a DSA to satisfy a user's request.

A DSA has an associated database where it stores directory information.

Directory Information

There are two types of directory information: the directory information base (DIB) and infrastructure information.

Directory Information Base

The directory information base (DIB) is the information that is stored by your directory service. Each DSA has a database that holds a part of the DIB.

The DIB consists of entries. An *entry* is a set of *attributes* and their *values*. Every entry has an *object class* attribute, which specifies the kind of object the entry describes, and defines the set of attributes it contains. Some attributes are mandatory and some are optional. The schema defines the attributes that are mandatory and optional for an entry of a given object class.

Directory information is hierarchical, that is, entries are organized in a tree structure. Each entry has a parent entry and may have child entries. The top of the hierarchy is known as the *root entry*. The root entry does not really exist in the DIB, but is the notional parent of any first level entry.

An entry is identified by its *distinguished name* (DN). A distinguished name is a sequence of attributes and values. The final attribute is the *naming attribute* of the entry. This attribute and its value provide the entry's *relative distinguished name* (RDN). The preceding sequence is the distinguished name of the parent entry.

You can also define an *alias entry*. An alias entry is identified by a distinguished name. It contains the name of the directory entry it represents (the aliased object name) and can contain other attributes.

The directory information is divided into name contexts. A name context is subtree of the directory, and is identified by the DN of the entry at the top of the name context. A DSA's database can hold more than one name context.

Figure 1-2 shows an example of how directory information is structured, with the DNs and RDNs of the shaded entries. The entry Country=US is a first level entry.

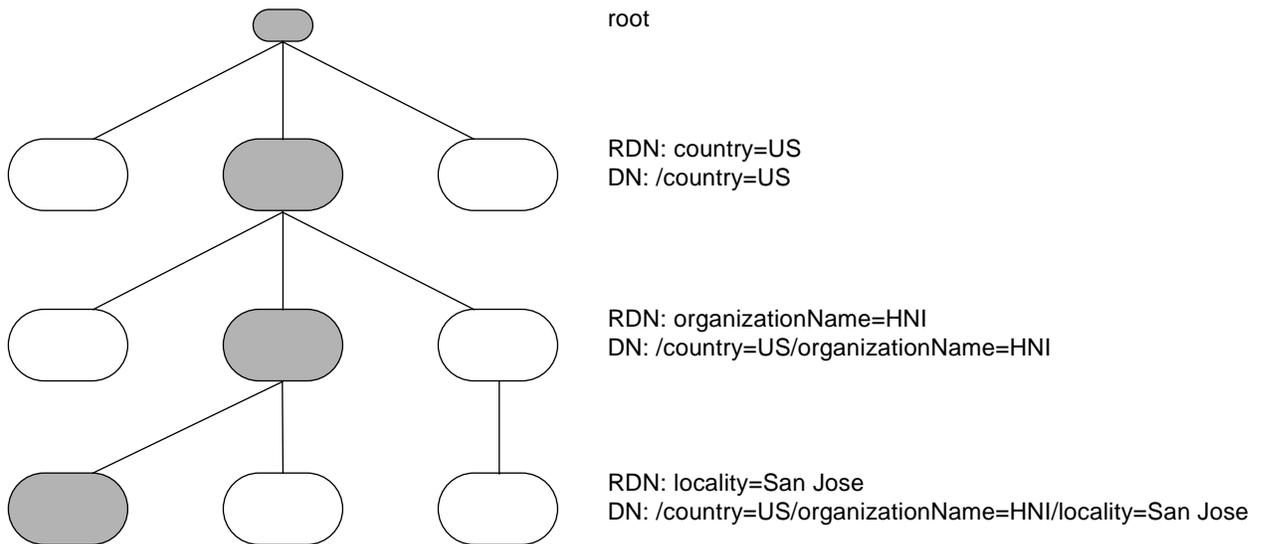


Figure 1-2 Directory Information Structure

Infrastructure Information

Infrastructure information determines how the components of a directory service behave and how DIB information is interpreted. It includes the schema, knowledge information and component configuration information.

Schema

The directory schema is the set of rules that describes the data that can be stored in the directory. It defines the type of entries, the rules for naming them, their structure and their syntax. It also includes information about security for entries held in the directory.

Knowledge Information

A DSA uses knowledge information to pass requests for information to other DSAs. The knowledge information held by a DSA is a list of the name contexts held in the local database and references to DSAs holding other naming contexts. When a DSA receives a request for information, it checks whether it can respond to the request using the information in the local database. If it cannot, it checks the knowledge references to see which is the DSA most likely to be able to process the request. The DSA then does one of the following things:

- Returns the request to the client that submitted it, with information about the DSA that the client should contact with the request. This is known as a *referral*.
- Passes the request to the DSA that can process the request. This is known as *chaining*. The second DSA returns the result to the first DSA, which returns it to the client.
- Broadcasts the request to all the DSAs it has knowledge of. Any DSA that can process the request does so and returns the result to the first DSA. This is known as *multicasting*.

Your DSA can have four types of knowledge references:

- An *upper reference* (sometimes called a *superior reference*) indicates the DSA that holds the naming context above your DSA's naming context in the directory information tree (DIT). A DSA that is not a first-level DSA must have an upper reference. A DSA must not have more than one upper reference defined. If a DSA receives a request it cannot process, and it does not have a reference to the relevant naming context, it sends the request to the DSA defined in the upper reference.
- A *subordinate reference* indicates a naming context that is a child of the naming context held by your DSA.
- A *non-specific subordinate reference* indicates a naming context that is lower in the directory tree but not a child of the naming context held by your DSA.
- A *cross reference* indicates any naming context that can be contacted directly. A cross reference is useful if you receive frequent requests for information from a particular name context, since it provides a direct link to the name context without routing the request through the name context hierarchy.

Managing Your Directory Service

To manage a directory service you must:

- Configure and maintain the Solstice X.500 directory server. See “Directory Components and Protocols” on page 17 for a description of the Solstice X.500 directory server and its components.
- Maintain the infrastructure information. This information includes the schema, knowledge information, and security information controlling access to information in the directory. See Chapter 2, “Initial Configuration” and Chapter 3, “Advanced Configuration” for details of how to set up infrastructure information.
- Maintain the directory information. This is the information you are storing for use by users and applications. See Chapter 6, “Data Management” for information about maintaining the information stored in your directory.

Management Tools

This section contains a summary of the tools available to help you manage your directory service.

Note – When the data entry tools start, there is a short delay while the current schema is loaded from the database. When you change directory information, the result is checked against the schema to ensure that no inconsistencies are introduced in the directory. Loading the schema at the starts speeds up processing of entries.

`x500servertool`

is a graphical tool for configuring and managing the Solstice X.500 directory server. To start the tool, log in as `root` or become `superuser` and type:

```
# /opt/SUNWconn/x500/bin/x500servertool
```

Figure 1-3 on page 23 shows the main window of `x500servertool`.

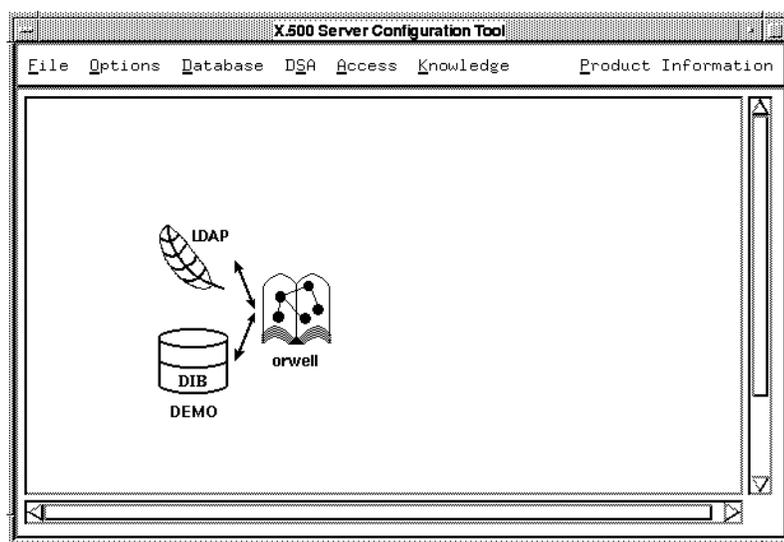


Figure 1-3 Main Window of x500servertool

See Chapter 2, “Initial Configuration” and Chapter 3, “Advanced Configuration” for information about using x500servertool.

x500clienttool

is a graphical tool for configuring and managing clients that use the Solstice X.500 Client Toolkit. See *Solstice X.500 Client Toolkit Management* for information about how to use x500clienttool.

x500import

is a utility for loading information into the directory. See “Importing Data” on page 91 for information about using x500import.

x500export

is a utility for exporting information from the directory into a text file. See “Exporting Data” on page 92 for information about using x500export.

x500datatool

is a graphical tool for managing the entries in your directory. See “Using the Data Editing Tool” on page 98 for information about using x500datatool.

The RFC1006 Driver

The Solstice X.500 Directory Server and the Solstice X.500 Client Toolkit include an RFC1006 driver. The driver implements the RFC1006 protocol, which enables these OSI applications to run over TCP/IP. The RFC1006 driver can be used instead of the OSI Communication Platform (Stack) below the transport layer interface (TLI), as shown in Figure 1-4.

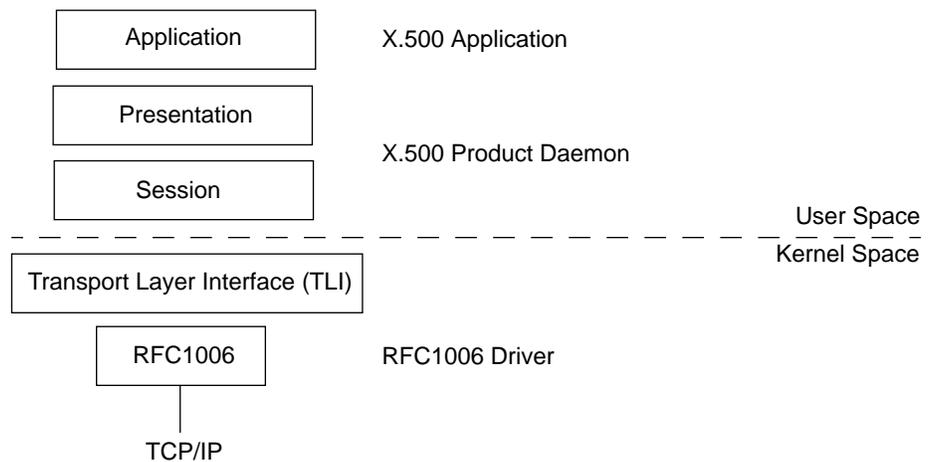


Figure 1-4 RFC1006 Driver and Higher Layer Entities

If you want to run the Solstice X.500 software over TCP/IP only, replace the OSI Stack with the RFC1006 driver. This will free up memory and disk resources.

The RFC1006 package, SUNWrk6, includes three software components:

- The `rk6d(1M)` daemon controls the RFC1006 driver. It is started automatically whenever you reboot your machine. You can fine-tune the configuration of the `rk6d` daemon by stopping it then restarting it manually with some command-line options.

- The `rk6stat(1M)` utility displays statistical information.
- The `rk6trace(1M)` utility traces incoming and outgoing Protocol Data Units (PDUs).

For more information on the RFC1006 driver utilities, see their manpages.

Initial Configuration



This chapter explains how to get a directory server up and running using a minimum configuration using the defaults wherever possible. It contains the following sections:

<i>Default Configuration</i>	<i>page 27</i>
<i>Initial Configuration</i>	<i>page 28</i>
<i>To Start the DSA</i>	<i>page 28</i>
<i>To Start the LDAP Server</i>	<i>page 30</i>
<i>To Configure the DSA</i>	<i>page 31</i>
<i>To Create a Database</i>	<i>page 32</i>
<i>To Add a Remote DSA</i>	<i>page 33</i>
<i>To Create Knowledge Information</i>	<i>page 35</i>
<i>To Stop the DSA</i>	<i>page 37</i>

Use `x500servertool` to configure your DSA. See “Management Tools” on page 22 for information about how to start `x500servertool`. Once your server is working you can modify the configuration using the information in Chapter 3, “Advanced Configuration.”

Default Configuration

A DSA with the default configuration described in this chapter can work as part of a directory service. It has the following characteristics:

- **Limited references to other DSAs.** This means that all queries that cannot be satisfied locally are sent to a small number of DSAs, and may take longer for replies to be returned to users. By default, the DSA chains requests.

See “Distributed Operations” on page 51 for information about setting the DSAs default behavior for queries that it cannot satisfy. See “To Create Knowledge Information” on page 35 for information about adding references to other DSAs.

- **The default schema is described in Chapter 5, “Schema Reference Information”.** You can modify or extend the schema to make it more suitable to your needs. See Chapter 4, “Extending the Schema” for details.
- **Default access control.** By default, the following access controls apply:
 - Only DSAs with *allowed* status can access the directory.
 - Users from all allowed DSAs have read access to the directory.
 - Authenticated users have read access.
 - Unauthenticated users have no access.
 - Users may not modify their own entries.

See “Access Controls” on page 46 for information about specifying access controls for your directory.

Initial Configuration

Complete the tasks in this section to configure a basic directory server. When you have completed them, you will have a working DSA that uses the default configuration, but the directory will not hold any information. See “Default Configuration” on page 27 for a description of the basic configuration, and Chapter 6, “Data Management,” for information about how to put information into the directory.

▼ To Start the DSA

You can start the DSA using `x500servertool` or from the command line.

▼ Using `x500servertool`

1. **Click on the DSA icon.**

A menu is displayed, as shown in Figure 2-1 on page 29.

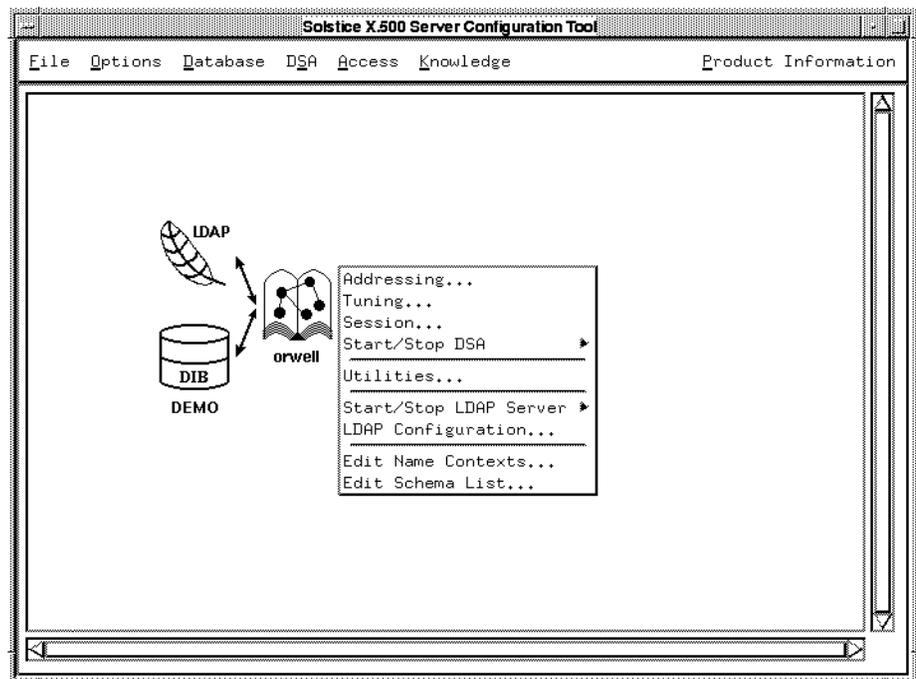


Figure 2-1 The DSA Menu

2. Choose Start DSA from the menu.

The DSA icon changes to indicate that the DSA is running.

Note – When the DSA starts the schema and other information is read from files. This may take a few seconds.

▼ **From the command line**

1. Log in as root or become superuser.

2. Type the following command:

```
# /etc/rc2.d/s93x500server start
```

▼ To Start the LDAP Server

The LDAP server is started using `x500servertool`. The LDAP server cannot be started unless the DSA is running. Complete the following steps:

1. **Click on the LDAP server icon.**
A menu is displayed, as shown in Figure 2-2.

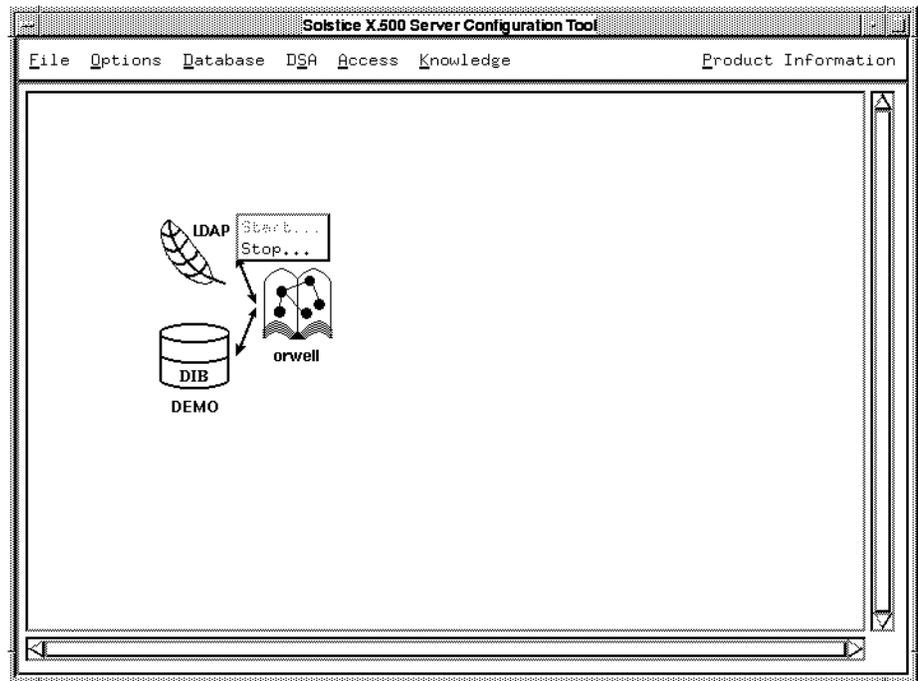


Figure 2-2 LDAP Server Menu

2. **Choose Start from the menu.**
The icon changes to indicate that the LDAP server is running.

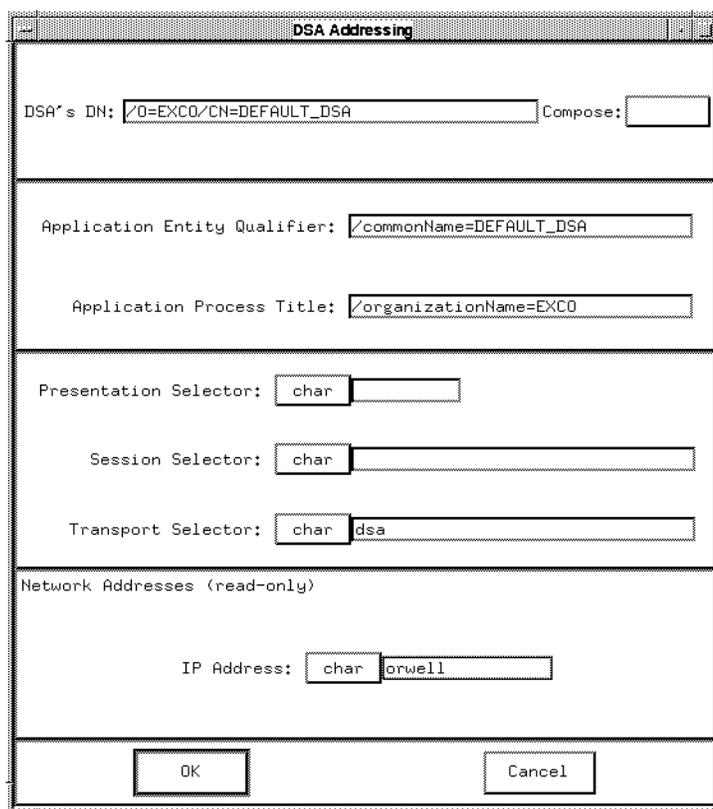
Note – When the LDAP server starts the schema and other information is read from files. This may take a few seconds.

▼ To Configure the DSA

To configure a DSA you must provide addressing information so that it can communicate with other components of the directory service. Complete the following steps:

1. Click **SELECT** on the DSA icon and choose **Addressing** from the pop-up menu.

The DSA Addressing window is displayed, as shown in Figure 2-3.



DSA's DN: Compose:

Application Entity Qualifier:

Application Process Title:

Presentation Selector:

Session Selector:

Transport Selector:

Network Addresses (read-only)

IP Address:

Figure 2-3 DSA Addressing Window

2. Enter the distinguished name (DN) of the DSA.

You can use the Compose button to help you construct the DN. (See “Composing Distinguished Names” on page 39 for information about using the Compose button.)

3. Specify the Application Entity Qualifier and Application Process Title of the DSA.

Use RDN format for these values.

4. Specify the Presentation, Session, and Transport Selectors of the DSA.**5. You can specify the selectors as either character or hexadecimal strings. Use the format button to specify which format you will use, and then type the selector value in the field. If you change the transport selector for a DSA, you must restart the DSA.****6. Press OK to save the information you have specified.**

No other configuration is necessary for the DSA, as the default values are sufficient for a DSA to work. See “Default Configuration” on page 27 for a description of the default DSA configuration settings.

▼ To Create a Database

Note – The first time you start the DSA the database DEMO is loaded.

1. Click SELECT on the DIB icon and choose Create from the pop-up menu.

The Create Database window is displayed. A database will be created with the schema files used in the previously selected database. If you want to use a different schema, you must deselect the database and updated the schema files list before you create the new database. If you are creating the first database for a DSA, the default schema files are used. See “Directory Schema” on page 41 for details of how to specify schema files.

2. Enter the name of the database you want to create.

This name is used to identify the set of files that hold the database.

3. Press OK.

A new database is created. The schema for the database is defined by the list of files in the schema list. The new database is automatically selected when it has been created.

▼ To Create a Name Context

- 1. Choose Edit Name Contexts from the DSA menu.**
The Edit Name Contexts window is displayed.
- 2. Enter the distinguished name (DN) of the name context you want to create.**
You can use the Compose button to help you construct the DN. (See “Composing Distinguished Names” on page 39 for information about using the Compose button.)
- 3. Press Add.**
The name context is added to the list.
- 4. Press OK.**

▼ To Add a Remote DSA

You need to create entries for the remote DSAs that your DSA will communicate with. To create a remote DSA entry, complete the following steps:

- 1. Choose Manage Remote DSAs from the Access menu.**
The Remote DSA Management window is displayed, showing a list of the existing entries for remote DSAs.
- 2. Press Add.**
The Add Remote DSA window is displayed, as shown in Figure 2-4 on page 34.

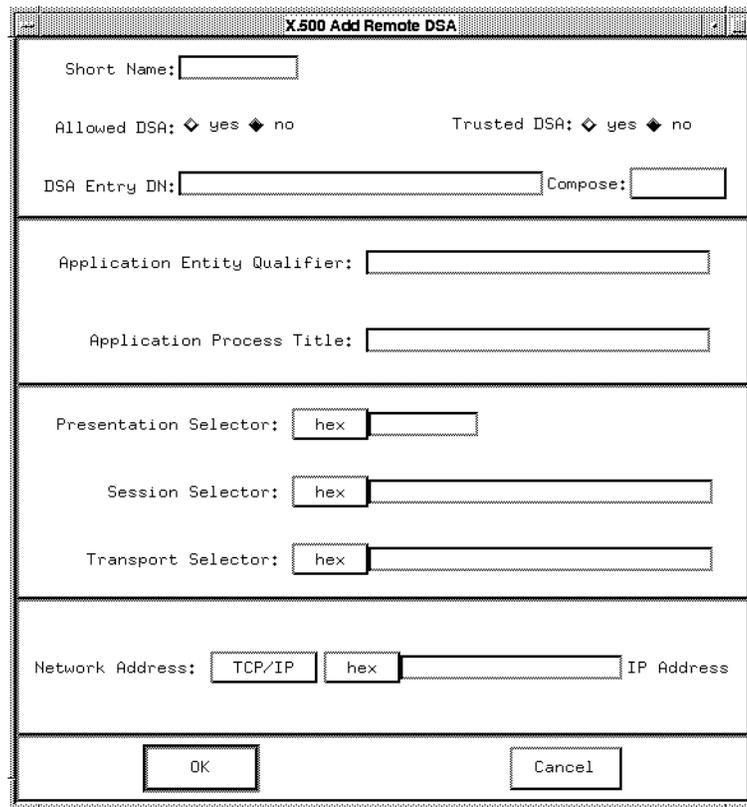


Figure 2-4 Add Remote DSA Window

3. **Enter the short name you want to assign to the remote DSA.**
A DSA short name is the name used to identify the DSA within `x500servertool`, in place of the DSA's distinguished name.
4. **Use the buttons to indicate whether the DSA is an allowed DSA or a trusted DSA (see “Known DSAs” on page 44 for information about allowed and trusted DSAs).**
5. **Enter the DN of the DSA.**
You can use the Compose feature to help you construct the DN.
6. **Enter the application entity qualifier and application process title of the DSA.**

7. Enter the presentation, session, and transport selectors for the DSA.
8. Indicate the network address type and the network address for the DSA.
You can specify the address in hexadecimal or character format.
9. Press OK.

▼ To Create Knowledge Information

You need to specify at least one reference to connect your directory server to other DSAs. To create a reference:

1. **Choose Create Reference from the Knowledge menu.**
A New Reference icon appears on the map, and the Create Remote Reference window is displayed, as shown in Figure 2-5.

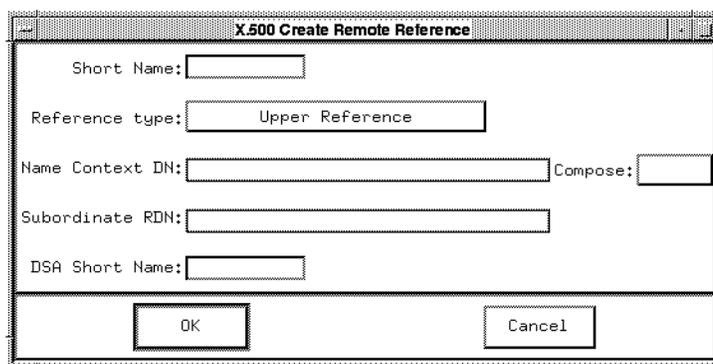


Figure 2-5 Create Remote Reference Window

2. **Enter a Short Name for the reference.**
This name is used to identify the reference so that you can modify it. It is useful to choose a short name that reflects the part of the DIB that the reference points to, or the DSA at which it is stored.
3. **Select the Reference Type.**
See “Knowledge Information” on page 21 for information about types of knowledge reference.

4. Enter the Distinguished Name (DN) of the Name Context that the reference points to.

You can use the Compose button to help construct the DN.

5. If you are creating a Subordinate or Non-specific Subordinate reference, specify the RDN of the subordinate name context.

6. Enter the short name of the DSA that holds the name context that the reference points to.

The DSA short name must already have been defined and an entry for the DSA must already exist in the database. See “To Add a Remote DSA” on page 33 for information about creating entries for remote DSAs.

7. Press OK.

The icon moves to the appropriate part of the screen and its color changes to reflect the reference type.

▼ **Checking References**

To check that a reference is defined correctly and that the DSA holding the information is available, use Ping Reference:

1. Click SELECT on the icon corresponding to the reference you want to test.

2. Choose Ping Reference from the Knowledge menu.

The Ping window is displayed.

3. Press Ping.

The status of the reference is displayed.

You can also check references automatically on a schedule, using the Autoping Reference option on the Knowledge menu. Ping and Autoping provide an application-level test that a referenced DSA is available, and do not provide information about the status of the network. This is useful when you need to monitor the status of a particular reference.

If the status of a reference is Down, the icon flashes and turns red. When the problem is resolved, the icon changes back to its original color, but continues to flash until the status is reset. To reset the status of a reference, click on the icon and select Reset state from the Knowledge menu.

▼ To Stop the DSA

1. Click on the DSA icon.

A menu is displayed, as shown in Figure 2-1 on page 29.

2. Choose Stop DSA from the menu.

The DSA icon changes to indicate that the DSA is stopped.

Note – Stopping the DSA automatically stops the LDAP server.

Advanced Configuration



This chapter explains how to tune the configuration of your DSA. It contains the following sections:

<i>Composing Distinguished Names</i>	<i>page 39</i>
<i>Configuration Management</i>	<i>page 40</i>
<i>Directory Schema</i>	<i>page 41</i>
<i>Known DSAs</i>	<i>page 44</i>
<i>Access Controls</i>	<i>page 46</i>
<i>Distributed Operations</i>	<i>page 51</i>
<i>DSA Name Contexts</i>	<i>page 52</i>
<i>Tuning DSA Resource Usage</i>	<i>page 53</i>
<i>Tuning Session Parameters</i>	<i>page 54</i>
<i>LDAP Configuration</i>	<i>page 55</i>
<i>Accounting</i>	<i>page 56</i>
<i>Error Reporting</i>	<i>page 57</i>
<i>x500servertool Options</i>	<i>page 58</i>

Composing Distinguished Names

There are many places in `x500servertool` where you have to supply a distinguished name. `x500servertool` provides a Compose button that you can use to help you construct a distinguished name (DN). When you press the Compose button, `x500servertool` displays a list of the attribute types that

are permitted as subordinate RDNs of the partly-composed distinguished name that is shown in the DN field. Select the attribute type that you want, and then type the value in the DN field. Figure 3-1 shows an example of using the Compose button to construct the DN of a remote DSA.

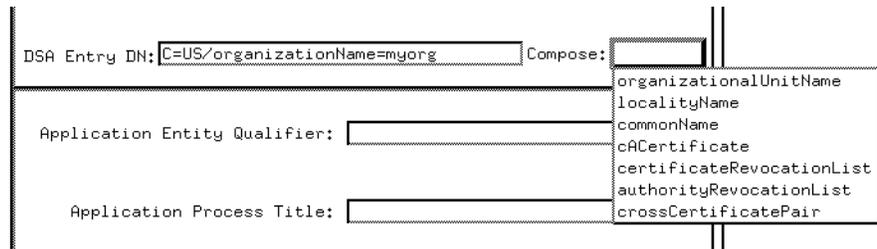


Figure 3-1 Using the Compose Feature

Configuration Management

The current configuration is stored in the `/var/opt/SUNWconn/OSIROOT/conf` directory, in the file `x500servertool.conf`. When you start the tool, this file is copied to `x500servertool.conf.BAK`.

The following features help you manage your configuration:

- **Backup.** This makes a The current safety copy of your configuration file. You should make a backup copy of your file whenever you make major changes. The backup copy stored in the same location and is called `x500servertool.conf.BAK.<timestamp>`, where `<timestamp>` is the time and date when the copy is made.
- **Save as.** This makes a copy of your configuration file that you can then move to a different host. You can specify any name for this file.
- **Load.** This loads the content of `x500servertool.conf` into the configuration file. This is useful if you want to abandon any configuration changes you have made since you started `x500servertool`. If you want to load a different configuration, rename the files so that the configuration you want is the current copy of `x500servertool.conf`.

Note – Take care when specifying the backup file to be loaded, as the file `x500servertool.conf.BAK` does not necessarily contain the most recent backup. Check the file creation times.

- *Print.* This prints a readable version of the current configuration to the file that you specify.

To use the configuration file management features, choose the relevant option from the File menu.

Directory Schema

The directory schema defines the information that can be stored in the directory. Chapter 4, “Extending the Schema,” explains what the schema is, and how you can extend the standard definition. The schema definition is stored in several files. You can modify the schema by specifying other definition files.

Each database contains a compiled version of the schema definition used to create the entries contained in the database. You can make limited modifications to the existing schema for a database. If you know before you create a database that the standard schema is not sufficient for your needs, you should modify the schema definition before you create any entries.

When you create a new database, the schema for the existing database that is selected is automatically used in your new database. If you want to create a new database using a different schema, deselect the current database, specify the list of schema files that you want to use, and then create the database.

▼ To Add a Schema Source File

1. **Start** `x500servertool`.
See “Management Tools” on page 22 for how to start `x500servertool`.
2. **Choose Edit Schema List from the DSA menu.**
The Schema Files window is displayed, listing the files in the current schema, as shown in Figure 3-2 on page 42.

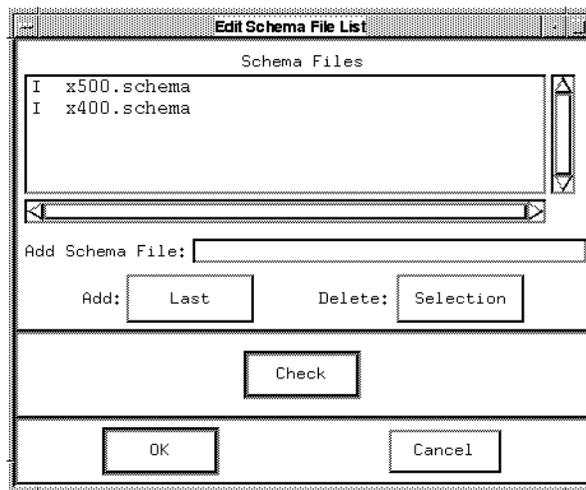


Figure 3-2 Schema Files Window

3. In the Add Schema File field, type the name of the schema file you want to add.

Use the Add pull-down menu to specify the position of the new file in the list. The default schema files are stored under `/opt/SUNWconn/x500/schema/` and have the file extension `.schema`. You can store schema files in any directory. Specify the full pathname if you store a file in a directory other than the default.

4. Press Check.

The definitions files are compiled to check that the resulting schema will work. “Schema Error Messages” on page 119 contains information about errors returned by the schema compiler.

5. Press OK.

▼ To Delete a Schema Source File

Note – You cannot remove a schema file from a database. You can only delete a file from the schema list when there is no database selected.

1. **Start** `x500servertool`.
See “Management Tools” on page 22 for how to start `x500servertool`.
2. **Choose Edit Schema List from the DSA menu.**
The Schema Files menu is displayed, listing the files in the current schema.
3. **Select the file that you want to delete. If you want to delete all the files, you do not need to select any.**
4. **Choose Selection or All from the Delete pull-down menu.**
5. **If you are only removing files and not deleting the whole schema, press Check to check that the remaining definitions file provide a working schema.**
The definitions files are compiled to check that the resulting schema will work. “Schema Error Messages” on page 119 contains information about errors returned by the schema compiler.
6. **Press OK.**

▼ To Modify a Current Schema

Note – Modifying the current schema can result in incompatibilities between databases, and should only be done when necessary to solve problems.

1. **Start** `x500servertool`.
See “Management Tools” on page 22 for how to start `x500servertool`.
2. **Choose Edit Schema List from the DSA menu.**
The Schema Files menu is displayed, listing the files in the current schema.
3. **In the Add Schema File field, type the name of the schema file you want to add.**
Use the Add pull-down menu to specify the position of the new file in the list. Schema extension files should be added after the existing files in the list. The default schema files are stored under `/opt/SUNWconn/x500/schema/`

and have the file extension `.schema`. You can store schema files in any directory. Specify the full pathname if you store a file in a directory other than the default.

4. Press Check.

The definitions files are compiled to check that the resulting schema will work. “Schema Error Messages” on page 119 contains information about errors returned by the schema compiler.

5. Press OK.

Known DSAs

You can configure your DSA so that it can communicate with any remote DSA, or you can define a list of the *allowed* DSAs that it is allowed to communicate with. Add entries to the database for all the DSAs that you want to communicate with and give them all *allowed* status, or set the *AllDSAs* flag.

A *trusted* DSA is granted access rights in your database. For each DSA you enter in the database, indicate whether or not it has *trusted* status. Giving a DSA *trusted* status does not automatically give it *allowed* status. If you want to deny access temporarily to a DSA, you can remove its *allowed* status but it's not necessary to remove it from the database. If you have the *AllDSAs* flag set, this overrides the list of *allowed* DSAs.

▼ To Allow Communication with Known DSAs

1. Start `x500servertool`.

See “Management Tools” on page 22 for how to start `x500servertool`.

2. Choose Access Rights from the Access menu.

The Access Control window is displayed.

3. Use the buttons to indicate whether you want to allow access from all DSAs or only from DSAs with *allowed* status.

4. Press OK.

▼ To Configure a Known DSA

1. **Start x500servertool.**
See “Management Tools” on page 22 for how to start x500servertool.
2. **Choose Manage Remote DSAs from the Access menu.**
The Remote DSA Management window is displayed, listing the DSAs that have entries in the database.

▼ To Add a Known DSA Entry

1. **Press Add.**
The Add Remote DSA window is displayed.
2. **Enter the short name you want to assign to the remote DSA.**
3. **Use the buttons to indicate whether the DSA is an allowed DSA**
4. **Use the buttons to indicate whether the DSA is a trusted DSA.**
5. **Enter the DN of the DSA.**
You can use the Compose button to help you construct the DN.
6. **Enter the application entity qualifier and application process title of the DSA.**
7. **Enter the presentation, session, and transport selectors for the DSA.**
If you change the transport selector for a DSA you must restart the DSA.
8. **Indicate the network address type and the network address for the DSA.**
You can specify the address in hexadecimal or character format.
9. **Press OK.**

▼ To Modify a Known DSA Entry

1. **Select a DSA from the list of Remote DSAs.**
2. **Press Modify.**
The Modify Remote DSA window is displayed.
3. **Make the modifications you require.**
4. **Press OK.**

▼ To Delete a Known DSA Entry

1. Select a DSA from the list of Remote DSAs.

2. Press Delete.

`x500servertool` prompts you for confirmation that you want to delete the DSA. If you want to deny the DSA access temporarily, do not delete its entry but instead modify its entry to remove its allowed status.

3. If you want to delete the DSA entry, press Yes.

4. Press OK.

▼ Checking DSA Availability

1. Start `x500servertool`.

See “Management Tools” on page 22 for how to start `x500servertool`.

2. Choose Ping DSA from the Access menu.

The Ping window is displayed.

3. In the Remote field, enter the shortname of the DSA you want to check.

4. Press Ping.

The state of the DSA is displayed in the State field.

Access Controls

Access to information is controlled by access controls set for entries and access rights granted to users or groups of users. There are eight levels of access:

- none
- compare
- read
- search
- add
- modify
- delete
- administer

A user is granted access to an entry, to the level permitted by the entry and attributes, if the request initiated by the user has rights equal to or higher than the entry's access controls. A user with Administer right has access to all entries and attributes.

User Access Rights

For access control purposes, a user's entry is the entry with the DN supplied in the bind request. A user entry must contain the userPassword attribute. The level of access users have on the DSA where their entries are stored is defined as follows:

- If the xSunUserLevel attribute is not defined in the schema, the user is granted the access rights of an anonymous user.
- If the xSunUserLevel attribute is defined in the schema but the user's entry does not contain this attribute, the user is granted the access rights defined by the default value of the xSunUserLevel attribute.
- If the xSunUserLevel attribute is defined in the schema and the user's entry contains this attribute, the user is granted the access rights defined by the value of the xSunUserLevel attribute in the user's entry.

See "Access Control Information" on page 87 for information about defining the xSunUserLevel attribute and setting a default value.

Entry Access Controls

The level of access required to access an entry in the database is defined as follows:

- If the xSunEntryProtection attribute is not defined in the schema, the entry has the no access control. Individual attributes within the entry can have access controls.
- If the xSunEntryProtection attribute is defined in the schema but the entry does not contain this attribute, the entry has the access control defined by the default value of the xSunEntryProtection attribute.
- If the xSunEntryProtection attribute is defined in the schema and the entry contains this attribute, the entry has the access control defined by the value of the xSunEntryProtection attribute in the entry.

- Regardless of the entry access controls, you can permit users can modify their own entries. See “To Set DSA Access Rights” on page 49 for information about how to use `x500servertool` to permit users to modify their own entries.

See “Access Control Information” on page 87 for information about defining the `xSunEntryProtection` attribute and setting a default value.

Attribute Access Controls

You can also set the protection level for a given attribute within an entry. The level of access required to access an attribute is defined as follows:

- If the attribute definition in the schema contains a protection level, this value is the access control for the attribute.
- If the attribute definition in the schema contains a protecting attribute, the access control for the protected attribute is the value of the protecting attribute if it is present in the entry, or is the default value of the protecting attribute.
- Regardless of the attribute access controls, you can permit users can modify attributes of their own entries. See “To Set DSA Access Rights” on page 49 for information about how to use `x500servertool` to permit users to modify their own entries.

See “Access Control Information” on page 87 for information about defining attribute access controls in the schema.

DSA Access Rights

You can also set access levels for a user from another DSA, based on the status of the remote DSA and whether the user is authenticated by the remote DSA. By default, only users from DSAs with *allowed* or *trusted* status are permitted access, and they are given List and Search access. A DSA that has trusted status does not automatically have allowed status. A user who has been authenticated by a remote DSA is given List and Search access. A user who is not authenticated and is using a DSA that does not have trusted status is only given Blind-compare access. (See your directory client documentation for information about authenticating users.) Use `x500servertool` to set the access rights granted by your DSA.

How Access is Controlled

The access right of a request is calculated as follows:

- If the user's entry is stored by your DSA, the request right is determined by the user's right's, as described in "User Access Rights" on page 47.
- If the user's entry is not in your local database, the request right is determined by the access right of the DSA that submitted the request or that authenticated the user, as described in "DSA Access Rights" on page 48.

If the access rights of the request are equal to or greater than the access controls on the data, the user is granted access to the level permitted by the entry and attribute access controls. The access level granted is the minimum of the user access right, the entry access control, and the attribute access control.

If the DSA is configured to allow users to modify their own entries, this overrides the entry and attribute access control.

A user with Administer right has full access to all entries and attributes.

▼ To Set DSA Access Rights

1. **Start x500servertool.**
See "Management Tools" on page 22 for how to start x500servertool.
2. **Choose Access Rights from the Access menu.**
The Access Control window is displayed, as shown in Figure 3-3 on page 50.

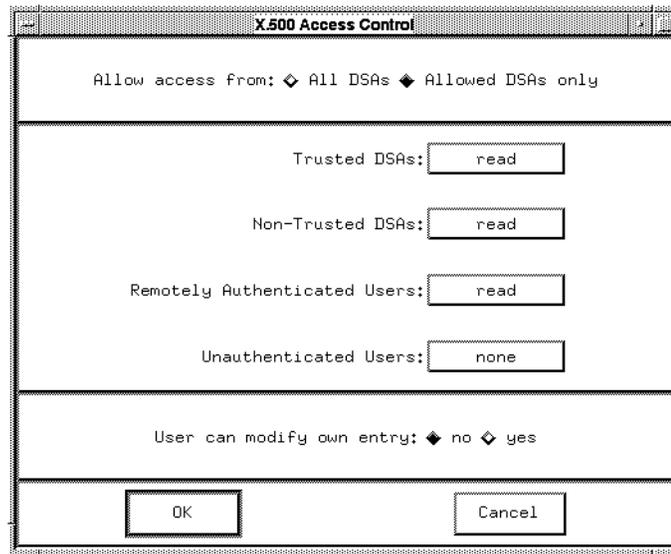


Figure 3-3 Access Control Window

3. **Select the scope of access for remote to DSAs.**
You can permit access from all DSAs, or only from DSAs with *allowed* status.
4. **Set the access rights granted to requests from trusted and non-trusted DSAs.**
5. **Set the access rights granted to users authenticated by a remote DSA and to unauthenticated users.**
This defines the maximum access level granted by your DSA to a user who was authenticated by another DSA.
6. **Specify whether users are permitted to modify their own entries.**
7. **Press OK.**

Distributed Operations

When a DSA receives a request it cannot satisfy from its own local databases, it uses distributed operations to pass the request to other DSAs that may be able to satisfy the request. There are three types of distributed operation:

- *Referral*, when the DSA returns the query to the DUA that submitted it, suggesting other DSAs that may be able to satisfy the request.
- *Chaining*, when the DSA forwards the request to another DSA that may be able to satisfy the request. The DSA uses its knowledge information to select a suitable DSA.
- *Multicasting*, when the DSA forwards the request to all the DSAs it has knowledge of.

A DUA can specify that a DSA use a particular type of distributed operation to handle a request it submits, but the DSA is not obliged to follow the DUA's preference. The Solstice X.500 DSA uses the type of distributed operation that the DUA specifies when possible. If it is not possible, it uses its own default behavior. If a request does not specify the type of distributed operation to be used, the DSA uses its default.

▼ To Set the Default Distributed Operation

1. **Start `x500servertool`.**
See "Management Tools" on page 22 for how to start `x500servertool`.
2. **Choose Tuning from the DSA menu.**
The DSA Tuning window is displayed, as shown in Figure 3-4 on page 52.

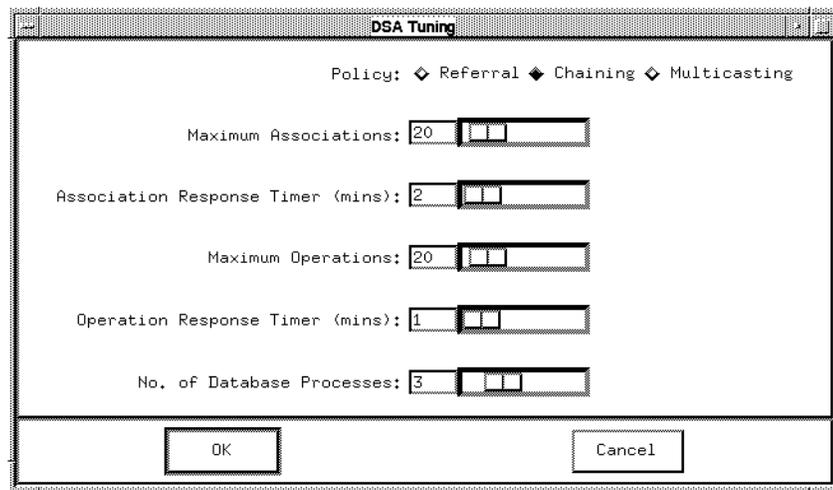


Figure 3-4 DSA Tuning Window

3. Set the distributed operations policy by choosing the relevant Policy button.
4. Press OK.

DSA Name Contexts

Specify the name contexts that a DSA stores in its database. You also need to supply this information to DSAs that need references to these name contexts.

▼ To Edit the Name Contexts List

1. **Start x500servertool.**
See “Management Tools” on page 22 for how to start x500servertool.
2. **Choose Edit Name Contexts from the DSA menu.**
The Edit Name Contexts window is displayed, showing the current list of name contexts held by the DSA, as shown in Figure 3-5 on page 53.

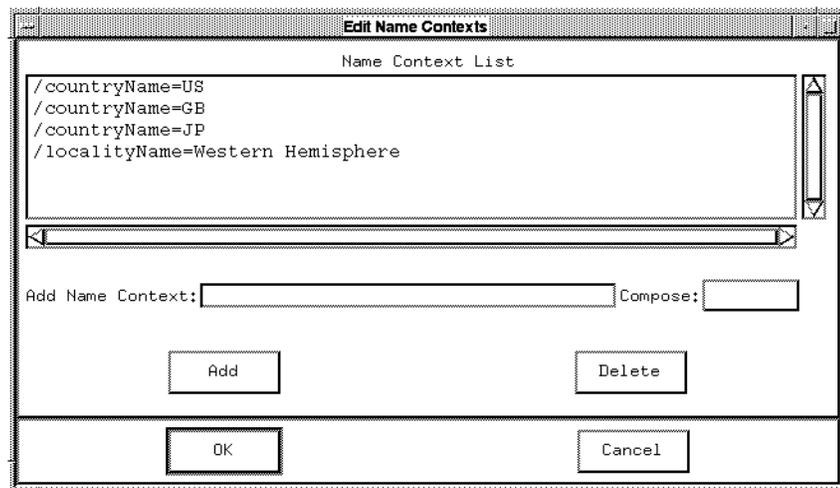


Figure 3-5 Edit Name Contexts Window

3. To add a name context, enter its distinguished name in the Add Name Context field and press Add.
4. To remove a name context, select it from the Name Context List and press Delete.
5. Press OK.

Tuning DSA Resource Usage

You can limit the resources a DSA uses, to ensure that a DSA does not use more resources than is reasonable given the other applications running. You can limit the following:

- The maximum concurrent associations
- The maximum time for which the DSA will attempt to open an association
- The maximum number of concurrent operations a DSA can perform
- The maximum time the DSA will wait for a response for an operation before assuming the operation has failed
- The number of database processes running

▼ To Tune DSA Resource Usage

1. **Start** `x500servertool`.
See “Management Tools” on page 22 for how to start `x500servertool`.
2. **Choose Tuning from the DSA menu.**
The DSA Tuning window is displayed, as shown in Figure 3-4 on page 52.
3. **Use the sliders to set the values you require.**
4. **Press OK.**

Tuning Session Parameters

The session parameters control how directory service components use the underlying network services.

▼ To Set the Session Parameters

1. **Start** `x500servertool`.
See “Management Tools” on page 22 for how to start `x500servertool`.
2. **Choose Session from the DSA menu.**
The DSA Session Parameters window is displayed, as shown in Figure 3-6 on page 55.

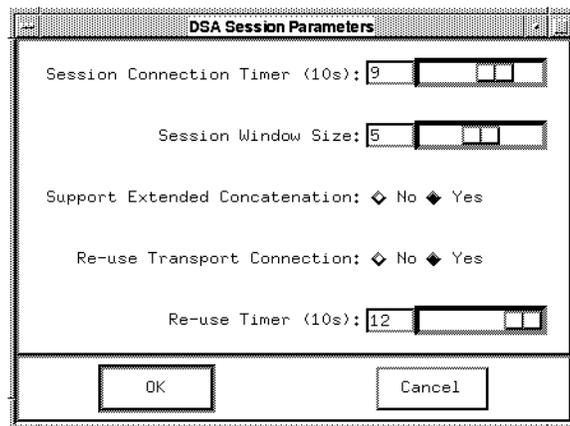


Figure 3-6 DSA Session Parameters Window

3. **Modify the Session Parameters as required.**
4. **Press OK.**

LDAP Configuration

You can specify the port used by LDAP, and the mode it uses, which controls operation timeouts. The maximum number of LDAP clients that can be bound to the LDAP server simultaneously is 255. If you change the LDAP port you must restart the LDAP server.

▼ To Configure the LDAP Server

1. **Start `x500servertool`.**
See “Management Tools” on page 22 for how to start `x500servertool`.
2. **Choose LDAP Configuration from the DSA menu.**
The LDAP Configuration window is displayed, as shown in Figure 3-7 on page 56.

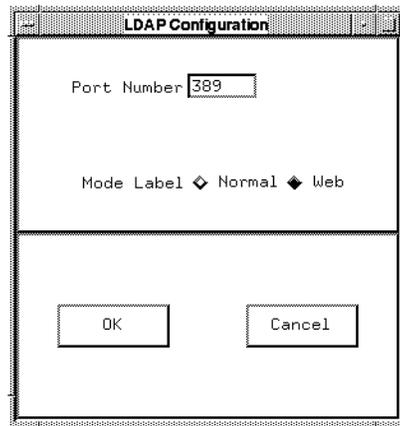


Figure 3-7 LDAP Configuration Window

3. **Enter the number of the port you want to use.**
You must not specify a port that is already being used by another application. By default, LDAP uses port 389.
4. **Specify whether the LDAP server is to work in Normal or Web mode.**
Web mode is intended for use with certain LDAP-Worldwide Web gateways. It sets operation maximum timers to higher values, so that links are maintained longer. Normal mode is the default and is sufficient for most applications, but if you have problems with operations timing out before a response is returned, use Web mode.
5. **Press OK.**

Accounting

The DSA can store accounting information in a file which you can process using a script. By default, accounting information is not stored. See “Accounting Information” on page 109 for details of the accounting information logged.

▼ To Start Accounting

1. **Start** `x500servertool`.
See “Management Tools” on page 22 for how to start `x500servertool`.
2. **Choose Utilities from the DSA menu.**
The DSA Utilities window is displayed.
3. **Select On to start the collection of accounting information.**
4. **Specify the name of a file where the data is to be stored.**
5. **Press OK.**

Error Reporting

You can log error messages in a file, have them sent by electronic mail to an address you specify, or both. By default errors are neither logged nor mailed.

▼ To Start Error Reporting

1. **Start** `x500servertool`.
See “Management Tools” on page 22 for how to start `x500servertool`.
2. **Choose Utilities from the DSA menu.**
The DSA Utilities window is displayed.
3. **If you want errors to be mailed, select Mail and supply an electronic mail address.**
4. **If you want errors to be logged, select File and specify a file name.**
5. **Press OK.**

x500servertool *Options*

The main window of x500servertool is a map of your directory service. You can change the appearance of this map, changing the positioning of the icons and the colors used to represent the components. You can also modify the date and time format used.

Note – On a monochrome monitor the color settings are ignored. Icons representing processes flash when those processes are not started, and in error conditions.

▼ To Modify the Map

- 1. Choose Map from the Options menu.**
The Map Options window is displayed.
- 2. To choose the icon colors, select the colored box next to each icon type. A window is displayed showing the colors available for that icon. Select the color you want for the icon.**
The color changes take effect when you press OK.
- 3. To change the position of the reference icons, select the number of columns (for subordinate references) or rows (for cross references) that you want. You can also change the order in which the icons are displayed, using the Order pull-down menus.**
The positions of the icons are changed when you press OK.
- 4. Press OK.**

▼ To Modify Date and Time Formats

- 1. Choose Date/Time from the Options menu.**
- 2. Specify the date and time formats that you want to use.**
- 3. Press OK.**

Extending the Schema

The Solstice X.500 includes a standard schema definition. The schema defines:

- The object classes known to the directory and the attributes that are mandatory and optional in those object classes.
- The syntax and matching rules for an attribute, whether the attribute may have more than one value, and where appropriate, the set of permitted values or the bounds on the attribute values.
- The object identifiers assigned to the objects and attributes.

You can extend or modify the schema definition that is provided with the Solstice X.500. This chapter explains why you might modify the schema, how to make modifications, and the potential problems you may encounter. It gives you an example of a typical modification. It contains the following sections:

<i>How the Schema is Used</i>	<i>page 60</i>
<i>Schema Definition Example</i>	<i>page 60</i>
<i>Standard Schema Files</i>	<i>page 64</i>
<i>Using a Non-Standard Schema</i>	<i>page 65</i>
<i>Guidelines for Extending the Schema</i>	<i>page 65</i>
<i>How to Extend the Schema Definition</i>	<i>page 66</i>
<i>Example of an Extended Schema</i>	<i>page 67</i>

How the Schema is Used

Each component of a directory service has a copy of the schema. The schemas held by the components are not necessarily identical, but the schemas must be consistent or directory requests cannot be satisfied. For example, a DSA that receives requests from several DUAs must have a schema that includes all the definitions contained in the schemas of all the DUAs.

Sending a Request to a DSA

A DUA verifies that the naming attributes it sends to the directory conform to the schema. If a query or modification request contains information that does not conform, the DUA reports an error or an XDS error is returned.

Receiving a Request from a DUA

When a DSA receives a request it verifies that the attributes in the request conform to the schema. If the request modifies an entry in the directory, the DSA verifies that the resulting entry would conform to the schema before it makes the modification. If the resulting entry would not conform, the DSA reports an error and makes no changes.

Receiving Information from a DSA

When a DUA receives a response from the directory, it passes the information to the application. The application parses the entries using the DUA's schema to decode the information.

Schema Definition Example

This section uses the schema definition for a `person` object to illustrate how objects and attributes are defined. The `Person` object class is used to define entries representing people. Such entries must always contain the common

name and surname of the person being described, and can also contain a telephone number, a password, a description and cross-references to other entries.

```
person OBJECT-CLASS
  SUBCLASS OF top
  MUST CONTAIN {
    commonName,
    surname
  }
  MAY CONTAIN {
    description,
    seeAlso,
    telephoneNumber,
    userPassword
  }
  NAMED BY { commonName }
  PRECEDED BY { organization }
  ::= { objectClass 6 }
```

This definition indicates:

- The name of the object class, `person`
- Which object class it is a subclass of
- The attributes it must contain
- The attributes it may contain
- The naming attribute of an entry of this class
- The position in the directory information tree
- The object identifier assigned to the object class

Each of the attributes must previously have been defined. For example, the Common Name attribute is defined as follows:

```
commonName ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX
  caseIgnoreStringSyntax(SIZE(1..ub-common-name))
  ABBREVIATED TO CN
  ::= { attributeType 3 }
```

This definition indicates:

- The attribute name, `commonName`, and a permitted abbreviation
- The attribute value syntax, including the maximum length of a value (defined by `ub-common-name`)
- The object identifier assigned to the attribute type

The maximum value length must also be defined in the file, though it can be included after a definition that uses it. The maximum length of a `commonName` string is defined as follows:

```
ub-common-name    INTEGER ::= 64
```

The attribute syntax must also be defined in the schema file, for example:

```
caseIgnoreStringSyntaxATTRIBUTE-SYNTAX
CHOICE {
    T61String,
    PrintableString
}
MATCHES FOR EQUALITY SUBSTRINGS
::= { attributeSyntax 4 }
```

This definition indicates:

- The name of the attribute syntax
- The permitted syntax types (in this case T.61 String or Printable String)
- The matching rule for the attribute syntax
- The object identifier assigned to the attribute syntax

The final example in this section is an extract from the file `/opt/SUNWconn/x500/schema/x500.schema` and shows the definition of the person object class. The extract does not contain the root definition for the object classes. See the file `x500.schema` for the complete definition of this object class including the object identifier roots.

```
-- attribute types --

commonName ATTRIBUTE
WITH ATTRIBUTE-SYNTAX
caseIgnoreStringSyntax(SIZE(1..ub-common-name))
```

```
ABBREVIATED TO CN
 ::= { attributeType 3 }

surname ATTRIBUTE
 WITH ATTRIBUTE-SYNTAX
 caseIgnoreStringSyntax(SIZE(1..ub-surname))
 ABBREVIATED TO SN
 ::= { attributeType 4 }

description ATTRIBUTE
 WITH ATTRIBUTE-SYNTAX
 caseIgnoreStringSyntax(SIZE(1..ub-description))
 ::= { attributeType 13 }

telephoneNumber ATTRIBUTE
 WITH ATTRIBUTE-SYNTAX
 telephoneNumberSyntax
 ::= { attributeType 20 }

seeAlso ATTRIBUTE
 WITH ATTRIBUTE-SYNTAX
 distinguishedNameSyntax
 ::= { attributeType 34 }

userPassword ATTRIBUTE
 WITH ATTRIBUTE-SYNTAX
 OCTET STRING(SIZE(1..ub-user-password))
 MATCHES FOR EQUALITY
 ABBREVIATED TO pwd
 PROTECTION LEVEL IS compare
 ::= { attributeType 35 }

-- attribute syntaxes --

distinguishedNameSyntax ATTRIBUTE-SYNTAX
 DistinguishedNameSyntax
 ::= { attributeSyntax 1 }

caseIgnoreStringSyntax ATTRIBUTE-SYNTAX
 CHOICE {
 T61String,
 PrintableString
 }
 MATCHES FOR EQUALITY SUBSTRINGS
```

```

 ::= { attributeSyntax 4 }

telephoneNumberSyntax ATTRIBUTE-SYNTAX
  PrintableString(SIZE(1..ub-telephone-number))
  MATCHES FOR EQUALITY SUBSTRINGS
 ::= { attributeSyntax 12 }

-- upper bounds --

ub-common-name      INTEGER ::= 64
ub-surname          INTEGER ::= 64
ub-description      INTEGER ::= 1024
ub-telephone-number INTEGER ::= 32
ub-user-password    INTEGER ::= 128

-- object classes --

person OBJECT-CLASS
  SUBCLASS OF top
  MUST CONTAIN {
    commonName,
    surname
  }
  MAY CONTAIN {
    description,
    seeAlso,
    telephoneNumber,
    userPassword
  }
  NAMED BY { commonName }
  PRECEDED BY { organization }
 ::= { objectClass 6 }

```

Standard Schema Files

Chapter 5, “Schema Reference Information”, lists the items defined in the standard schema. For details of how these items are defined, see the files containing the definitions. Two schema files, `x500.schema` and `x400.schema`, are shipped with Solstice X.500, and stored in the directory `/opt/SUNWconn/x500/schema`.

The configuration file contains a list of the source files for the currently used schema, and holds a “compiled” version of the schema. By default, the source files are `x500.schema` and `x400.schema`. The schema is compiled from the

source files when you save the configuration. See “Directory Schema” on page 41 for information about specifying schema source files in the configuration file.

Using a Non-Standard Schema

The standard schema definition supplied with Solstice X.500 is sufficient for most applications that use a directory service. However, you may want to extend the definition to include object classes or attributes specific to your organization or to a particular application. For example, if all employees of an organization have an employee number, you might want to add it to the `organizationalPerson` object class, or to create a new object class `employee`.

If you modify the schema, you need to be sure that entries that make use of the modifications can be handled by all the directory entities and applications that might see the information in those entries.

Guidelines for Extending the Schema

- Make sure object identifiers are unique for the whole directory service. The schema compiler checks that the object identifiers used in the schema definition files you supply are used for only one purpose, that is, are unique within the schema source files specified. However, there is no mechanism to check automatically that an object identifier is globally unique for all the DSAs and DUAs in the directory service.

To ensure global uniqueness, you must assign object identifiers that are unique within your organization, and use an object identifier root that uniquely identifies your organization. An object identifier root is an object identifier for your organization, and is assigned to your organization by your national standards body. Define it in the schema file as follows:

```
myoidroot          OBJECT IDENTIFIER ::= { <oid> }
```

where `<oid>` is the object identifier assigned to your organization. You can then define other object identifiers using `myoidroot` as the root:

```
myoid1             OBJECT IDENTIFIER ::= { myoidroot 1 }
```

If you require object identifiers for several different types of objects (new attribute types and new object classes, for example) use your object identifier root to define roots for each class of identifier and then use numbers to distinguish between different objects of the same type.

- Make sure that if you change an object class used by another application, the other application continues to work. You should not remove attributes from an existing object class, or modify the attribute definitions, unless you are sure that the attributes are not used.
- It is less likely to cause problems if you define new object classes than if you modify the standard ones. However, this may result in information being duplicated. You can reduce the duplication between object classes by choosing the standard class that most closely resembles the object class you want, and making your new object class a subclass of that object class. It will then inherit all the attributes of that class. For example, in the standard schema, most object classes are subclasses of `top`. The object classes `organizationalPerson` and `residentialPerson` are subclasses of the object class `person`.
- The following limits apply to the size of the schema definition:

Limit	Maximum
Total number of object classes	150
Total number of attributes	1000
Number of mandatory attributes in an object class	30
Number of optional attributes in an object class	60
Number of naming attributes in an object class	4
Depth of tree (number of RDNs in a DN)	20

How to Extend the Schema Definition

1. Using the information in this chapter and the next, write the definitions of the new or changed schema elements and store them in a file. See the standard schema file `x500.schema` for an example of how to write the definitions.
2. Using `x500servertool`, add this file to the list of schema source files (see “To Add a Schema Source File” on page 41 for details of how to do this).

-
3. Ensure that the DUAs and other DSAs that are also going to handle entries that use the schema extensions have updated schemas.
 - For Solstice X.500 DSAs, copy the schema source file to the system where the DSA runs and use `x500servertool` to add it to the list of schema source files. See “To Add a Schema Source File” on page 41 for details.
 - For Solstice X.500 DUAs, copy the schema source file to the system where the DUA runs and use `x500clienttool` to add it to the list of schema source files. See *Solstice X.500 Client Toolkit Management* for details.

Example of an Extended Schema

This section shows you how to define a new object class, `employee`, containing some standard attributes and some new attributes, and how to use XOM and XDS to add an entry of the `employee` object class to the directory. You can also use the data tools described in Chapter 6, “Data Management” to add the entry to the directory.

▼ To Define the Employee Object Class

The employee object class is to contain the standard attributes surname, telephoneNumber, description, and seeAlso, and two new attributes, loginName and eMail. The ASN.1 definition of the new attributes and the object class is:

```

employee OBJECT-CLASS
  SUBCLASS OF top
  MUST CONTAIN { commonName }
  MAY CONTAIN {
    surname,
    loginName,
    telephoneNumber,
    eMail,
    description,
    seeAlso
  }
  NAMED BY { commonName }
  PRECEDED BY { organization, locality }
  ::= { myNewObjClass 2 }

eMail ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX
    caseExactStringSyntax(SIZE(1..ub-name))
  ::= { myNewAttType 10 }

loginName ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX
    caseExactStringSyntax(SIZE(1..ub-name))
  ::= { myNewAttType 5 }

```

See “Object Classes” on page 81 for details of the NAMED BY and PRECEDED BY clauses in this definition.

▼ To Add an Employee Entry to the Directory

To add an entry for employee John Smith to the directory using XDS:

1. Build a Name object of class DS-DN, containing the distinguished name of the person described in the employee object.
2. Build an Entry object of class Attribute-List, containing:

-
- a. An object of class AVA with attribute type oid and attribute value {myNewObjClass 2}, the object identifier of the employee object class.
 - b. An object of class AVA with attribute type commonName and attribute value "John Smith".
 - c. An object of class AVA with attribute type {myNewAttType 10}, the object identifier for the loginName attribute type, and attribute value johnsmith.
3. Call the Add-Entry() function specifying the Name and Entry objects as arguments. John Smith's entry is added to the directory.

This chapter contains information about the standard schema definitions. it contains the following sections:

<i>Attribute Types</i>	<i>page 71</i>
<i>Attribute Syntaxes</i>	<i>page 78</i>
<i>Attribute Sets</i>	<i>page 80</i>
<i>Object Classes</i>	<i>page 81</i>
<i>Access Control Information</i>	<i>page 87</i>

Attribute Types

This section contains an alphabetical list of the attribute types. Attribute types are defined as follows:

```
<attribute-type> ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX
    <attribute-syntax>
  [MATCHES FOR <matching-rule>]
  [SINGLE VALUE]
  [ABBREVIATED TO <abbreviation>]
  [PROTECTED BY <attribute> | SHIELD ENTRY | CONTROL ACCESS]
  [LDAP STRING IS <string>]
  ::= <object-identifier>
```

where:

- `<attribute-type>` is the name of the attribute type. The length of the name must not exceed 40 characters. The maximum number of attributes the schema can contain is 1000.
- `<attribute-syntax>` is the syntax of the attribute values.
- `<matching-rule>` is the matching rule that applies to attribute values. Specifying the matching rule is optional. If you do not specify a matching rule, the exact-match rule for value syntax is used.
- `SINGLE VALUE` indicates that this attribute cannot have multiple values. If this clause is not present, the attribute can have multiple values.
- `ABBREVIATED TO <abbreviation>` indicates an alternative name for the attribute, such as C for country.
- `SHIELD ENTRY`, `CONTROL ACCESS` and `PROTECTED BY` are used to provide security for information in the directory. See “Access Control Information” on page 87 for information about using these elements.
- `LDAP STRING IS <string>` specifies the string used within the LDAP protocol to identify this attribute type. The string must be unique for each attribute type. If no string is specified, the attribute name is used.
- `<object-identifier>` is the object identifier of the attribute type.

The standard schema definition contains the following attribute types:

aliasedObjectName

In an alias entry, holds the distinguished name of the object which the alias describes.

businessCategory

Information concerning the occupation of some common objects such as objects describing people. The length of the values of this attribute must not exceed 128 characters.

commonName

The name by which an object is commonly known. This name is usually chosen by the person or organization it describes, or by the organization that owns the object it describes, and conforms to the naming conventions of the country or

culture with which it is associated. The length of the values of this attribute must not exceed 64 characters. You can use the abbreviation CN for commonName.

countryName

The country where the named object resides, or with which the object is connected. The attribute value is one of the codes defined in ISO 3166. The length of the values of this attribute must not exceed 2 characters. You can use the abbreviation C for countryName.

description

A description of an object. The length of the values of this attribute must not exceed 1024 characters.

destinationIndicator

The country and city associated with the object (the addressee). Used by the Public Telegram Service, according to Recommendations F.1 and F.3. The length of the values of this attribute must not exceed 128 characters.

facsimileTelephoneNumber

The telephone number and, optionally, parameters of a facsimile machine associated with an object.

internationalISDNNumber

The international ISDN number associated with an object. The length of the values of this attribute must not exceed 16 characters.

knowledgeInformation

A human-readable description of the knowledge information held by a DSA.

localityName

The geographical area where an object is located or with which the object is associated. The length of the values of this attribute must not exceed 128 characters. You can use the abbreviation L for localityName.

member

A group of names associated with an object.

mhs-or-addresses

The O/R address of an entity that sends and receives messages using a Message Handling System (MHS).

mhs-message-store

The name of a Message Store.

objectClass

The object class to which the object belongs.

organizationName

The name of the organization with which an object is affiliated. The length of the values of this attribute must not exceed 64 characters. You can use the abbreviation O for organizationName.

organizationalUnitName

A subdivision of an organization. The length of the values of this attribute must not exceed 64 characters. You can use the abbreviation OU for organizationalUnitName.

owner

The name of an object that has responsibility for the object the entry describes.

postalAddress

The address information required for the physical delivery of postal messages by the postal authority. The address consists of not more than 6 lines of not more than 30 characters.

postalCode

The postal code of an object. The length of the values of this attribute must not exceed 40 characters.

postOfficeBox

The Post Office Box by which the object will receive physical postal delivery. The length of the values of this attribute must not exceed 40 characters.

preferredDeliveryMethod

The object's preferred methods of receiving information, in decreasing order of preference. The delivery methods are indicated by integers and the standard methods defined are listed in Table 5-1.

Table 5-1 Preferred Delivery Methods

Delivery Method	Integer	Delivery Method	Integer
any-delivery-method	0	g3-facsimile-delivery	5
mhs-delivery	1	g4-facsimile-delivery	6
physical-delivery	2	ia5-terminal-delivery	7
telex-delivery	3	videotex-delivery	8
teletex-delivery	4	telephone-delivery	9

presentationAddress

The presentation address associated with an object that represents an OSI application entity.

physicalDeliveryOfficeName

The name of the city, town or village where the physical delivery office is situated. The length of a value of this attribute must not exceed 128 characters.

registeredAddress

A mnemonic for an address associated with an object at a particular city location. The mnemonic is registered in the country in which the address is located, and is used in the provision of a public telegram service.

roleOccupant

The name of an object which fulfills an organizational role. Note that this is not the name of the role itself.

searchGuide

Suggested search criteria optionally included in an entry expected to be used as the base-object for a search operation. The attribute consists of an optional identifier for the class of object sought and combinations of attribute types and logical operators to be used to construct a filter. You can also specify the matching rule to be applied.

seeAlso

The names of other directory objects which describe some other aspect of the same real-world object. For example, an `organizationalPerson` object could contain a `seeAlso` attribute containing the distinguished name of the `residentialPerson` object that describes the same person.

serialNumber

The serial number of a device. The length of the values of this attribute must not exceed 64 characters.

stateOrProvince

The geographical subdivision in which the named object is located or with which it is associated. The length of the values of this attribute must not exceed 128 characters. You can use the abbreviation `ST` for `stateOrProvince`.

streetAddress

The street address at which the named object is located or with which it is associated. The address that is used for local distribution and physical delivery in a postal address. The length of each value of this attribute must not exceed 128 characters.

surname

The family name of a person, normally inherited from a parent or assumed on marriage, by which the person is commonly known. The length of a value of this attribute must not exceed 64 characters. You can use the abbreviation `SN` for `surname`.

supportedApplicationContext

The object identifiers of the application contexts supported by the OSI application entity represented by an object.

telephoneNumber

The telephone number associated with an object. The length of a value of this attribute must not exceed 32 characters.

teletexTerminalIdentifier

The Teletex terminal identifier and, optionally, parameters for a teletex terminal associated with an object. The length of a value of this attribute must not exceed 24 characters.

telexNumber

The telex number, country code, and answerback code of a telex terminal associated with an object. The length of the telex number must not exceed 14 characters. The length of the answerback code must not exceed 8 characters.

title

The position or function of an object within an organization. The length of the values of this attribute must not exceed 64 characters.

userPassword

The length of the values of this attribute must not exceed 128 characters.

x121Address

An address associated with an object. The address is in the format defined by CCITT Recommendation X.121. The length of the values of this attribute must not exceed 15 characters.

xSunUserLevel

The level of access granted to a user.

xSunEntryProtection

The level of protection assigned to an entry.

Attribute Syntaxes

This section contains an alphabetical list of the attribute syntaxes. Attribute syntaxes are defined as follows:

```
<attribute-syntax> ATTRIBUTE-SYNTAX
  <syntax>
  [MATCHES FOR <matching-rule>]
  ::= <object-identifier>
```

where:

- `<attribute-syntax>` is the name of the syntax of the attribute values. The length of the name must not exceed 40 characters.
- `<matching-rule>` is the matching rule that applies attribute values. Specifying the matching rule is optional. If you do not specify a matching rule, the exact-match rule for value syntax is used.
- `<syntax>` is the attribute syntax, and is one of the following:
 - ANY
Either a hexadecimal string, or a binary file.
 - BitString
A hexadecimal string.
 - Boolean
True or False, case insensitive.
 - CaseExactString
 - Case Ignore IA5String
 - CaseIgnoreString
 - CaseIgnoreStringList
A series of values in CaseIgnoreString syntax.
 - DSubmitPermission
Permission type, OR Address, User DN, Group DN
 - DistinguishedNameSyntax
A sequence of attribute value assertions.
 - FacsimileTelephoneNumber
A numeric string.
 - GeneralString

- GeneralizedTime
- GraphicString
- Guide
- Ia5String
- Integer
- NumericString
- Integer
- ObjectDescriptor
- ObjectIdentifierSyntax
Object identifier in hexadecimal format.
- OctetString
A hexadecimal string, can be read from a file.
- OrAddress
A sequence of attribute value assertions.
- OrName
OR Address, user's distinguished name
- PostalAddress
Six lines of addressing information.
- PreferredDeliveryMethod
Numeric values in decreasing order of preference. The delivery methods are indicated by integers, and the standard methods defined are listed in Table 5-2.

Table 5-2 Delivery Methods

Delivery Method	Integer	Delivery Method	Integer
any-delivery-method	0	g3-facsimile-delivery	5
mhs-delivery	1	g4-facsimile-delivery	6
physical-delivery	2	ia5-terminal-delivery	7
telex-delivery	3	videotex-delivery	8
teletex-delivery	4	telephone-delivery	9

- PresentationAddress
P-selector, S-Selector, T-Selector, NSAP
- PrintableString

- **T61String**
A string of T.61 characters.
- **TelephoneNumber**
A numeric string.
- **TeletexTerminalIdentifier**
Ttx number, Graphic set, Control set, Page format, Term. cap, Private
- **TelexNumber**
Telex number, country code, answerback
- **UTCTime**
String in UTC format.
- **VideotexString**
- **VisibleString**
- **XSunUserLevelSyntax**
The user levels are listed in Table 5-3.

Table 5-3 User Levels

User Level	Integer	User Level	Integer
none	0	add	4
compare	1	modify	5
read	2	delete	6
search	3	administer	7

Attribute Sets

Several attribute sets are defined, as a short way of specifying a set of attributes that frequently occur together in an object class.

The syntax for defining an attribute set is:

```

<attributesetname> ATTRIBUTE-SET
  CONTAINS {
    <attribute>
  }
  ::= {<object-identifier> }
    
```

where:

- `<attribute-set>` is the name of the attribute set. The length of the name must not exceed 40 characters.
- `<attribute>` is a comma-separated list of attributes and attribute sets to be included in this attribute set
- `<object-identifier>` is the object identifier assigned to the attribute set.

The sets defined in the standard schema are:

telecommunicationAttributeSet

containing the attributes `facsimileTelephoneNumber`, `internationalISDNNumber`, `telephoneNumber`, `teletexTerminalIdentifier`, `telexNumber`, `x121Address`, `preferredDeliveryMethod`, `destinationIndicator`, and `registeredAddress`.

postalAttributeSet

containing the attributes `physicalDeliveryOfficeName`, `postalAddress`, `postalCode`, `postOfficeBox`, and `streetAddress`.

localeAttributeSet

containing the attributes `localityName`, `stateOrProvinceName`, and `streetAddress`.

organizationalAttributeSet

containing the attributes `description`, `localeAttributeSet`, `postalAttributeSet`, `telecommunicationAttributeSet`, `businessCategory`, `seeAlso`, `searchGuide`, and `userPassword`.

Object Classes

This section contains an alphabetical list of the object classes that are defined in the standard schema. Attribute syntaxes are defined as follows:

```
<object-class> OBJECT-CLASS
  SUBCLASS OF <superclass>
  MUST CONTAIN {
    <attributes>
```

```

    }
    MAY CONTAIN {
    <attributes>
    }
    [MAXIMUM OF <number>]
    [NAMED BY {<attribute>}]
    PRECEDED BY {<attribute-list>}
    LDAP STRING IS <string>
    ::= { <object-identifier> }

```

where:

- <object-class> is the name of the object class. The length of the name must not exceed 40 characters. The maximum number of object classes the schema can contain is 150.
- <superclass> is the object class of which this object class is a subclass. An object inherits the mandatory and optional attributes of its parent. All objects are subclasses of object class top and therefore inherit the mandatory attribute objectIdentifier. An object class can be a subclass of up to 20 superclasses.
- <attributes> is a comma-separated list of the attributes and attribute sets that an entry of this object class must or may contain. The number of attributes an entry must contain cannot exceed 30. The number of optional attributes in an entry is 60.
- MAXIMUM OF <number> is an optional clause which indicates the maximum number of relative distinguished names (RDNs) of this class allowed in a distinguished name. This cannot exceed ten. When this clause is not included, the limit is one.
- NAMED BY <attribute> is an optional clause indicating the naming attributes of an entry of this class, that is, the attributes used in the relative distinguished name. The number of naming attributes of an entry must not exceed 4.
- PRECEDED BY <object class> indicates the object class that can precede an entry of this object class in an RDN.
- LDAP STRING IS <string> specifies the string used within the LDAP protocol to identify this object class. The string must be unique for each object class. If no string is specified, the name of the object class is used.
- <object-identifier> is the object identifier assigned to the object class.

The following object classes are defined in the standard schema:

alias

An alternative name for an object. Objects of class *alias* must contain the attribute *aliasedObjectName*.

applicationProcess

Used to define entries representing application processes. An application process is an element of an open system that performs information processing for a particular application. Entries of object class *applicationProcess* must contain the attribute *commonName* and may contain the attributes *description*, *localityName*, *organizationalUnitName* and *seeAlso*.

applicationEntity

Used to define entries representing application entities. Entries of object class *applicationEntity* must contain the attributes *commonName* and *presentationAddress*, and may contain the attributes *description*, *localityName*, *organizationName*, *organizationalUnitName*, *seeAlso* and *supportedApplicationContext*.

certificationAuthority

Used to define entries representing objects that act as certification authorities. Entries of object class *certificationAuthority* must contain the attributes *cACertificate*, *certificateRevocationList* and *authorityRevocationList*, and may contain the attribute *crossCertificatePair*.

country

Identifies country entries in the directory. Entries of object class *country* must contain the attribute *countryName* and may contain the attributes *description* and *searchGuide*.

device

Used to define objects representing devices (for example modems and CD-ROM drives). Entries of object class *device* must contain the attribute *commonName* and may contain the attributes *description*, *localityName*, *organizationName*, *organizationalUnitName*, *owner*, *seeAlso* and *serialNumber*.

dSA

Used to define entries representing DSAs. Object class *dSA* is a subclass of object class *applicationEntity*. Entries of object class *dSA* inherit the attributes of object class *applicationEntity*, and may also contain the attribute *knowledgeInformation*.

groupOfNames

Used to define entries representing an unordered set of names of objects or other groups. Entries of object class *groupOfNames* must contain the attributes *commonName* and *member*, and may contain the attributes *description*, *organizationName*, *organizationalUnitName*, *owner*, *seeAlso* and *businessCategory*.

locality

Used to define entries in the directory that describe locality. Entries of object class *locality* must contain at least one of the attributes *localityName* or *stateOrProvinceName*, and can also contain the attributes *description*, *searchGuide*, *seeAlso* and *streetAddress*.

mhs-distribution-list

Used to define entries in the directory that hold distribution lists. Entries of object class *mhs-distribution-list* must contain the attributes *commonName*, *mhs-dl-submit-permissions*, and *mhs-or-addresses*, and may contain the attributes *description*, *organizationName*, *organizationalUnitName*, *owner*, *seeAlso*, *mhs-deliverable-content-types*, *mhs-deliverable-eits*, *mhs-dl-members*, and *mhs-preferred-delivery-methods*.

mhs-message-store

Used to define Message Store entries in the directory. Object class `mhs-message-store` is a subclass of object class `applicationEntity`. Entries of object class `mhs-message-store` inherit the attributes of object class `applicationEntity`, and may also contain the attributes `description`, `owner`, `mhs-supported-optional-attributes`, `mhs-supported-automatic-actions`, and `mhs-supported-content-types`.

mhs-message-transfer-agent

Used to define Message Transfer Agent (MTA) entries in the directory. Object class `mhs-message-transfer-agent` is a subclass of object class `applicationEntity`. Entries of object class `mhs-message-transfer-agent` inherit the attributes of object class `applicationEntity`, and may also contain the attributes `description`, `owner` and `mhs-deliverable-content-length`.

mhs-user

Used to define directory entries representing Message Handling System (MHS) users. Entries of object class `mhs-user` must contain the attribute `mhs-or-addresses` and may contain the attributes `mhs-deliverable-content-length`, `mhs-deliverable-content-types`, `mhs-deliverable-eits`, `mhs-message-store` and `mhs-preferred-delivery-methods`.

mhs-user-agent

Used to define user agent entries in the directory. Object class `mhs-user-agent` is a subclass of object class `applicationEntity`. Entries of object class `mhs-user-agent` inherit the attributes of object class `applicationEntity`, and may also contain the attributes `owner`, `mhs-deliverable-content-length`, `mhs-deliverable-content-types`, `mhs-deliverable-eits` and `mhs-or-addresses`.

organization

Used to define organization entries in the directory. Entries of object class `organization` must contain the attribute `organizationName` and may also contain the entries in the attribute set `organizationalAttributeSet`.

organizationalPerson

Used to define entries representing people employed by, or in some way associated with, an organization. Object class *organizationalPerson* is a subclass of object class *person*. Entries of object class *organizationalPerson* inherit the attributes of object class *person*, and may also contain the attributes *organizationalUnitName* and *title* and the attribute sets *localeAttributeSet*, *postalAttributeSet* and *telecommunicationAttributeSet*.

organizationalRole

Used to define entries representing a role or position within an organization. An organizational role is usually filled by an organizational person, but it may also be filled by a non-human entity. Entries of object class *organizationRole* must contain the attribute *commonName* and may contain the attributes *description*, *preferredDeliveryMethod*, *roleOccupant*, *seeAlso* and *organizationalUnitName*, and the attribute sets *localeAttributeSet*, *postalAttributeSet* and *telecommunicationAttributeSet*.

organizationalUnit

Used to define entries representing subdivisions of organizations. Entries of object class *organizationalUnit* must contain the attribute *organizationalUnitName* and may also contain the entries in the attribute set *organizationalAttributeSet*.

person

Used to define entries representing people. Entries of object class *person* must contain the attributes *commonName* and *surname*, and may also contain the attributes *description*, *seeAlso*, *telephoneNumber* and *userPassword*.

residentialPerson

Used to define entries representing a person in the residential environment. Object class *residentialPerson* is a subclass of object class *person*. Entries of object class *residentialPerson* inherit the attributes of object class *person*, must also contain the attribute *localityName*, and may also contain the attributes *preferredDeliveryMethod* and *businessCategory* and the attribute sets *localeAttributeSet*, *postalAttributeSet* and *telecommunicationAttributeSet*.

strongAuthenticationUser

Used to define entries for objects that participate in Strong Authentication. Entries of object class `strongAuthenticationUser` must contain the attribute `userCertificate`.

top

This object class has no real-world equivalent object, but is defined to ensure that all other object classes contain the attribute `objectClass`. All object classes are subclasses of object class `top`.

Access Control Information

Access to information in the directory is controlled by a combination of entry and attribute protection levels and user access rights. See “Access Controls” on page 46 for more information about how access controls are defined and used.

A user’s access rights are defined by the attribute `user-level`, which is defined as follows:

```
xSunUserLevel ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX XSunUserLevelSyntax
  CONTROL ACCESS
  DEFAULT <value>
  ::= { sun 15 }
```

This attribute can be included in the entry that describes the user. The `CONTROL ACCESS` clause indicates that this is the attribute defining the user’s access rights. See “Attribute Syntaxes” on page 78 for information about the attribute syntax. The default value supplied is used if no specific access right is given for a user.

Protection levels for directory information can be set for the whole entry or for a given attribute. The protection level for an entry is defined by the attribute `entry-protection`, which is defined as follows:

```
xSunEntryProtection ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX XSunActionLevelSyntax
  SHIELD ENTRY
  DEFAULT <value>
  ::= { sun 16 }
```

The SHIELD ENTRY clause indicates that this is the attribute defining the level of protection of an entry. XSunActionLevel Syntax is an integer syntax defined as follows:

```
XSunActionLevelSyntax ::= INTEGER {
    none      (0),
    compare   (1),
    read      (2),
    search    (3),
    add       (4),
    modify    (5),
    delete    (6)
}
```

The default value supplied is used if no specific access right is given for an entry.

You can also define the protection level for a given attribute, by including a reference to a protection attribute in the attribute definition, as follows:

```
<attribute> ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX <syntax>
    PROTECTED BY <prot-attrib> | PROTECTION LEVEL IS <prot-level>
    DEFAULT <default-value>
    ::= { oid }
```

where:

- <attribute> is the name of the attribute you are defining
- <syntax> is the attribute syntax
- <prot-attrib> is the name of the attribute whose value determines the protection level of the attribute being defined
- <prot-level> is the protection level assigned if this attribute is self-protecting. You must not specify both PROTECTED BY and PROTECTION LEVEL clauses.
- <default-value> specifies the protection level for the attribute if a protecting attribute is defined but is not present in a given entry. You must not specify both PROTECTION LEVEL and DEFAULT clauses.
- <oid> is the object identifier of the attribute being defined

The protecting attribute is defined as follows:

```
<prot-attribute> ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX XSunActionLevelSyntax
    ::= { oid }
```

A protecting attribute automatically has Administer protection level.

You can use a combination of entry protection and attribute protection. The attribute protection overrides the entry protection.

Adding Access Control Attributes

The default schema definition supplied with Solstice X.500 does not contain access control attribute definitions. If you want to use access control, you need to add the `XSunUserLevel` and `XSunEntryProtection` attributes to the schema. If you make these attributes optional attributes of object class `top`, they will be inherited by all other object classes. The file `secure_extension.schema` in the `/opt/SUNWconn/x500/schema` directory shows how to extend the schema to support access control.

Security Attributes

Solstice X.500 supports simple authentication based on a user's distinguished name and password. Note that the passwords are transmitted in clear as part of the bind protocol.

The standard schema includes a number of attributes used to provide strong authentication. Solstice X.500 can handle these attributes but provides no further support for strong authentication. The attributes are:

- AlgorithmIdentifier
- AuthorityRevocationList
- cACertificate
- certificate
- CertificateList
- CertificatePair
- Certificates
- certificaterevocationList
- CertificationPath
- crossCertificatePair
- CrossCertificates
- ForwardCertificationPath
- serialNumber
- SubjectPublicKeyInfo
- userCertificate
- validity
- version

Data Management



This chapter explains how to load information into your directory and how to manage the data after it has been loaded. It contains the following sections:

<i>Importing Data</i>	<i>page 91</i>
<i>Exporting Data</i>	<i>page 92</i>
<i>Using the Data Editing Tool</i>	<i>page 98</i>
<i>Backing Up Data</i>	<i>page 106</i>

Importing Data

Use the `x500import` utility to add entries to the directory. The utility reads a file containing the entries and attributes you want to add, with the object class, naming attributes, other attributes and attribute values for the entries. “Import and Export Data File Format” on page 93 describes the format of the data file.

`x500import` *Command Syntax*

To run `x500import`, log in as root or become superuser and type:

```
# /opt/SUNWconn/x500/bin/x500import -f datafile [-e errorfile] [-l logfile] [-p] [-v]
```

datafile is the name of the file containing the entries to be added to the directory. The format of this file is described in “Import and Export Data File Format”.

errorfile is the name of a file that contains any entries that could not be added.

logfile is the name of a file that contains a list of errors. If you do not specify a *logfile*, the errors are directed to standard output.

-p instructs `x500import` to parse the data file and report any syntax errors, but not to add any entries to the directory. The *datafile* is left unchanged, and syntax errors are reported to *logfile*.

-v instructs `x500import` to report detailed information.

Exporting Data

Use the `x500export` utility to list the entries contained in the directory. The exported data is written to a file, using the format described in “Import and Export Data File Format” on page 93.

`x500export` *Command Syntax*

To run `x500export`, log in as `root` or become `superuser` and type

```
# /opt/SUNWconn/x500/bin/x500export -s subtree [-f datafile] [-d bindir] [-m maxsize]
```

subtree indicates the root entry of the subtree that you want to export. This entry and the whole subtree must be within your database.

datafile is the name of the file where the entries are listed. If you do not specify a file, standard output is used.

bindir is the name of the directory where files containing binary attribute values are stored.

maxsize is the maximum size (in bytes) of binary attribute value that is written to the *datafile*. Values that exceed this limit are written to separate files in the *bindir* directory instead, and the name of the file is recorded in *datafile*. Values not exceeding this limit are written in the *datafile* as hexadecimal.

Note – `x500export` does not change any entries in your database.

Deleting Entries

You can also use `x500import` to delete entries in the directory. In this case you need only specify the distinguished name of the entry. For example:

```
# a delete example
delete /C=US/O=AnyCo
```

You cannot use `x500import` to delete an entry that has child entries.

Import and Export Data File Format

The data file used by `x500import` and `x500export` is formatted according to the following rules:

- An entry you want to add has the following format:

```
add DN
ocl: objectclass = o-value
att: attribute = att-value
.
.
.
att: attribute = att-value
```

where:

- *DN* is the distinguished name of the entry you are adding
- *objectclass = o-value* indicates the object class of the entry you are adding to the directory. You can specify more than one value for the object class.

- *attribute* is an ordinary attribute of the entry and *att-value* is an attribute value. You can specify multiple values for an attribute, and you can specify multiple attributes for an entry.

See “Data File Example” on page 98 for an example of a datafile.

The object class must be specified before any other attributes. You can specify several attributes and naming attributes for an entry. Each equals sign (=) indicates a new value for an attribute, so you can specify multiple values to an attribute.

Entries written to the data file by `x500export` are written in this `add` format.

- An entry you want to delete has the following format:

```
delete DN
```

where:

- *DN* is the distinguished name of the entry you want to delete
- Comment lines start with a hash sign (#).
- Blank lines are ignored.
- Leading spaces are ignored after the equals sign are ignored, but trailing spaces after the attribute value are not.
- Fields in a complex attribute value are terminated by a semicolon (including the last field, so that there are no trailing spaces).
- To enter a long attribute value, use the continuation character + as the first character on a line. Where the + character follows other characters it is treated as a text character.
- The / and ; characters are significant unless preceded by the accept flag, \.
- Values of syntax Any that do not exceed *maxsize* are written as hexadecimal strings in the *datafile*. Values exceeding *maxsize* are stored as binary in a file in the *bindir* directory and the file name is stored in the *datafile*. See “Data File Example” on page 98 for an example of how to specify the file.

Attribute Value Syntaxes

The following attribute value syntaxes are supported:

- ANY
Either a hexadecimal string, or a binary file.
- BitString
A hexadecimal string.
- Boolean
True or False, case insensitive.
- CaseExactString
- Case Ignore IA5String
- CaseIgnoreString
- CaseIgnoreStringList
A series of values in CaseIgnoreString syntax.
- DSubmitPermission
Permission type; OR Address; User DN; Group DN;
- DistinguishedNameSyntax
A sequence of attribute value assertions separated by / characters. If there are multiple distinguished attributes, these are separated by & characters. For example /C=US/O=AnyCo&L=NewYork.
- FacsimileTelephoneNumber
A numeric string.
- GeneralString
- GeneralizedTime
- GraphicString
- Guide
- Ia5String
- Integer
- NumericString
- Integer

- **ObjectDescriptor**
- **ObjectIdentifierSyntax**
Object identifier in hexadecimal format.
- **OctetString**
A hexadecimal string, can be read from a file.
- **OrAddress**
Attribute value assertions separated by / characters, for example /C=US/O=AnyCo/OU1=Headquarters/OU1=Finance/S=Jones. The permitted keys are shown in Table 6-1.

Table 6-1 OR Address Field Keys

Field	Key	Field	Key
Country name	C	Administrative management domain	ADMD
Organization	O	Private management domain	PRMD
Surname	S	Organizational units 1 to 4	OU1 to OU4
Given name	G	X.121 address	X121
Generation qualifier	GQ	Terminal Id	T-TY
user agent Id	UA-ID	Domain defined attributes	any other key

- **OrName**
OR Address; user's distinguished name;
- **PostalAddress**
Six lines of addressing information, each (including the last) terminated with a semi-colon (;). All components must be present even if no value is given.
- **PreferredDeliveryMethod**
Numeric values in decreasing order of preference, separated by semi-colons (;).
- **PresentationAddress**
P-selector; S-Selector; T-Selector; NSAP;
- **PrintableString**

- **T61String**
A string of T.61 characters.
- **TelephoneNumber**
A numeric string.
- **TeletexTerminalIdentifier**
Ttx number; Graphic set; Control set; Page format; Term. cap; Private;
- **TelexNumber**
Telex number; country code; answerback;
- **UTCTime**
String in UTC format.
- **VideotexString**
- **VisibleString**
- **XSunActionLevelSyntax**
- **XSunUserLevelSyntax**

“Attribute Syntaxes” on page 78 contains more information about these supported attribute syntaxes.

Any Syntax

A value of syntax Any is supplied to the directory either as a hexadecimal string or in a binary file. Use `x500datatool` to set a length limit for values supplied as strings. Any value that exceeds this limit is held in a file and is not manipulated directly by `x500datatool`, `x500import`, or `x500export`.

Values of syntax Any must be encoded by an application before they are passed to the directory. You cannot use `x500datatool` or `x500import` to access multiple values of an attribute with this syntax. The values are concatenated. You can use these tools to replace an existing attribute. XDS and LDAP fully support correctly-encoded multivalued attributes of syntax Any.

Data File Example

The following data file adds two entries to the directory:

```
# entry of type "organization"
add /C=US/organizationName=WM
  ocl: objectClass=550404
  att: organizationName=WM
  att: postalAddress = 1 Long Street; Large Town; Far Country;;;
                    = 2 Short Street; Small Town;;;Zip123;
  att: binaryPicture
      file: /tmp/picture.octetstring
  att: longAttribute = a very long attribute that does not fit on
                    + one line so includes the continuation character

# entry of type "als.organization"
add /C=US/organizationName=BABMT
  ocl: objectClass=550604
      =550601
  att: organizationName=BABMT
  att: aliasedObjectName=/countryName=US/organizationName=MTC
```

Using the Data Editing Tool

The data editing tool, `x500datatool`, enables you to modify the data in your directory. The following sections summarize how to use `x500datatool`. The descriptions assume that you are familiar with the standard features of Motif interfaces.

▼ **To Start `x500datatool`**

1. **Log in as root or become superuser.**
 Anyone using `x500datatool` has the equivalent of Administer access rights to all data in the directory, so access to the tool is restricted.
2. **Start `x500datatool` by typing:**

```
# /opt/SUNWconn/x500/bin/x500datatool
```

The main window of `x500datatool` is displayed.

3. Set the current name context by choosing one of the available name contexts from the Name Contexts menu.

The distinguished name of entry that corresponds to the name context automatically appears in the Current Entry field. The name context must already exist. See “DSA Name Contexts” on page 52 for details of how to create a name context using `x500servertool`.

Figure 6-1 shows an example of the `x500datatool` main window. The File menu contains an Exit option. The Options menu enables you to specify a limit on the size of attribute values of syntax Any that are handled directly by the tool. Any values exceeding this size are held in files stored in the default directory. Use the Options menu to specify this default directory. The files are identified by the attribute name and a number assigned by the tool.

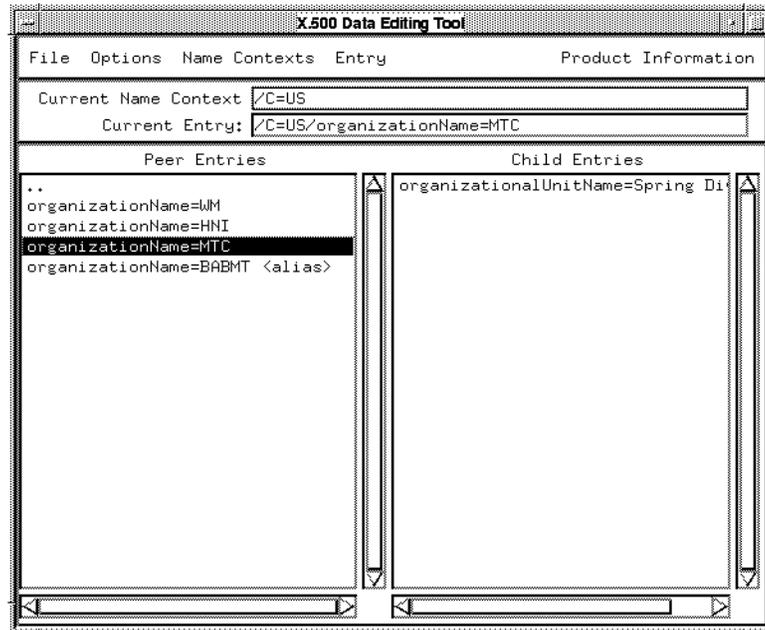


Figure 6-1 `x500datatool` Main Window

The Peer Entries list shows the current entry, which is highlighted, and all other entries which have the same parent entry as the current entry. The Child Entries list shows the child entries of the current entry. Any entries which are aliases are marked in the lists with `<alias>`.

If you have just created a new database and the name context is empty, no current entry is displayed and the Peer and Child Entries lists are empty. You must add an entry corresponding to the name context before you can add any other entries. See “To Add a Subordinate Entry” on page 103 for details of how to add an entry. When you create this first entry in the name context, all the object classes defined in the schema will be displayed.

You can now display an entry, add a subordinate entry, or modify or delete the current entry.

Selecting an Entry in x500datatool

To select an entry that is displayed in the Peer Entries or Child Entries list, double-click on the entry. The distinguished name of the entry you select is displayed in the Current Entry field and is highlighted in the list.

If the entry you want to select is not in the Peer or Child Entries lists displayed, you can move up or down the directory tree until the entry is listed. To move up the tree, double-click on the `..` line at the top of the Peer Entries list. (If there is no `..` line at the top of the Peer Entries list, the current entry is the top entry of the name context.) To move down the tree, select a child entry. The selected entry becomes the current entry and its child entries are displayed.

You can also make an entry the Current Entry by typing its distinguished name in the Current Entry field. `x500datatool` checks that an entry of that name exists.

Updating Directory Information

When you update directory information using `x500datatool`, the tool checks the DSA’s schema to ensure that the resulting information is valid. A remote DUA might have only a subset of the DSA’s schema, so may be unable to handle the entries after they have been updated.

▼ To Display the Attributes of an Entry

1. Start x500datatool.

See “To Start x500datatool” on page 98 for how to start x500datatool.

2. Select the entry you want to display.

The distinguished name of the entry is displayed in the Current Entry field.

3. Choose Modify Current Entry from the Entry menu.

The Entry Attributes window is displayed, showing the current attribute values. The object class of the entry is shown in the top bar of the window. For most entries there is more than one window of attributes. Use the Forward and Back buttons to move between windows.

For each attribute that has a value, the Used box is checked.

For a multivalued attribute, the first value is displayed on the main screen. Press Multi to display a list of all the attribute values. Note that the order in which the values of an attribute are displayed depends on the order in which they are returned from the database and is not fixed.

Binary attribute values (values of syntax Any) are not displayed; instead, the name of a file containing the value is shown.

4. Press Cancel to close the Entry Attributes screen.

Figure 6-2 on page 102 is an example of an entry of class organization.

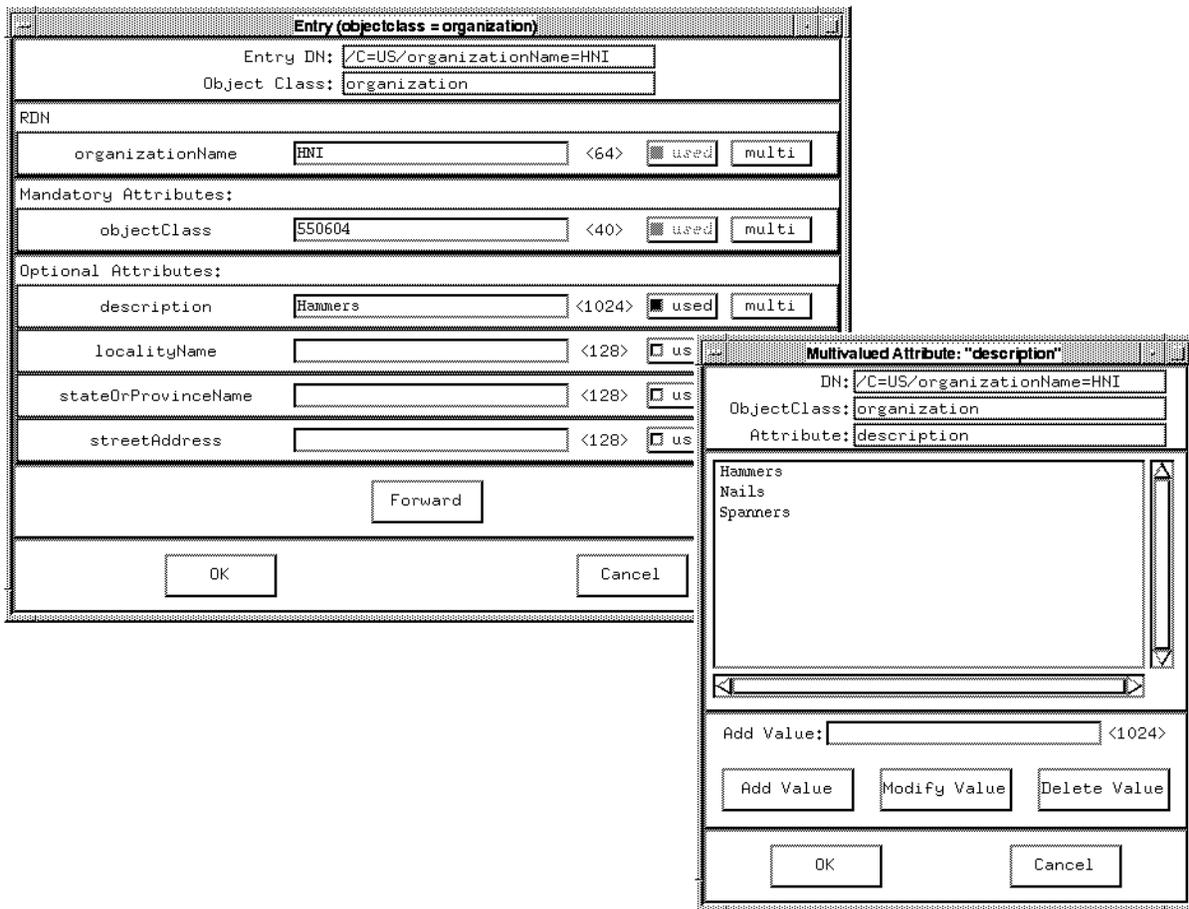


Figure 6-2 Entry Attributes Screen for an Organization Entry

▼ To Modify an Entry

1. **Start x500datatool.**
See "To Start x500datatool" on page 98 for how to start x500datatool.
2. **Select the entry you want to modify.**
The distinguished name of the entry is displayed in the Current Entry field.

3. Choose Modify Current Entry from the Entry menu.

The Entry Attributes window is displayed, showing the current attribute values. The object class of the entry is shown in the top bar of the window. For most entries there is more than one window of attributes. Use the Forward and Back buttons to move between windows.

4. Modify the attribute values as required.

If an attribute has a value, the marker in the Used button is checked.

For a multivalued attribute, press Multi to display the list of current values.

- To modify a value, select the value from the list, modify it in the Value field, and press Modify Value.
- To add a new value, type the value in the Value field and press Add Value.
- Press OK.

5. Press OK**▼ To Add a Subordinate Entry****1. Start x500datatool.**

See “To Start x500datatool” on page 98 for how to start x500datatool.

2. Select the entry for which you want to add a subordinate.

The distinguished name of the entry is displayed in the Current Entry field.

3. Press and hold MB1 on the Add Subordinate Entry option of the Entry menu.

A list of object classes allowed as subordinates of the current entry is displayed.

4. Choose the object class of the entry you want to add.

The Add Entry window for that object class is displayed. For most entries there is more than one window of attributes. Use the Forward and Back buttons to move between windows.

5. Enter the value of the RDN.**6. Enter a value for each mandatory attribute.****7. Enter values for the optional attributes that you want to use.**

To enter more than one value for an attribute, press Multi to display the Multivalued Attribute window and add each value that you require.

8. Press OK.

▼ **To Add a Subordinate Alias**

- 1. Start x500datatool.**
See “To Start x500datatool” on page 98 for how to start x500datatool.
- 2. Select the entry for which you want to add a subordinate.**
The distinguished name of the entry is displayed in the Current Entry field.
- 3. Press and hold MB1 on the Add Subordinate Alias option of the Entry menu.**
A list of alias object classes allowed as subordinates of the current entry is displayed.
- 4. Choose the object class of the alias you want to add.**
The Add Entry window for that object class is displayed.
- 5. Enter the distinguished name of the aliased object.**
- 6. Enter values for the optional attributes that you want to use.**
To enter more than one value for an attribute, press Multi to display the Multivalued Attribute window and add each value that you require.
- 7. If no naming attribute is specified in the schema for this alias object class, indicate which of the attributes you have specified is the naming attribute.**
For most aliased object classes, this step should not be necessary because the naming attribute will be defined. You cannot specify an attribute that has multiple values to be the naming attribute.
- 8. Press OK.**

▼ **To Delete an Entry**

- 1. Start x500datatool.**
See “To Start x500datatool” on page 98 for how to start x500datatool.
- 2. Select the entry that you want to delete.**
The distinguished name of the entry is displayed in the Current Entry field.

3. Choose Delete Current Entry from the Entry menu.

If the entry has subordinates, you are prompted to confirm whether or not you want to delete the entry.

Note – There is no automatic way to reload entries you have deleted.

▼ To Move an Entry**1. Start x500datatool.**

See “To Start x500datatool” on page 98 for how to start x500datatool.

2. Select the entry that you want to move.

The distinguished name of the entry is displayed in the Current Entry field.

3. Choose Modify Current Entry from the Entry menu.

The Entry Attributes window is displayed.

4. Modify the value of the RDN attribute.

Modifying the RDN of an entry changes the distinguished name of that entry and moves it to a different part of the directory.

Note – Entries of certain object classes cannot be moved. If you try to modify the RDN of an entry that cannot be moved an error message is displayed.

▼ To Delete a Subtree**1. Start x500datatool.**

See “To Start x500datatool” on page 98 for how to start x500datatool.

2. Select the entry at the top of the subtree that you want to delete.

The distinguished name of the entry is displayed in the Current Entry field.

3. Choose Delete Current Entry from the Entry menu.

You are prompted to confirm that you want to delete the entry and all subordinate entries.

Note – There is no automatic way to reload entries you have deleted.

▼ To Move a Subtree

1. **Start `x500datatool`.**
See “To Start `x500datatool`” on page 98 for how to start `x500datatool`.
2. **Select the entry at the top of the subtree that you want to move.**
The distinguished name of the entry is displayed in the Current Entry field.
3. **Choose Modify Current Entry from the Entry menu.**
The Entry Attributes window is displayed.
4. **Modify the value of the RDN attribute.**
Modifying the RDN of an entry changes the distinguished name of that entry and moves it to a different part of the directory. Any child entries are also moved.

Note – Entries of certain object classes cannot be moved. If you try to modify the RDN of an entry that cannot be moved an error message is displayed.

Backing Up Data

Use `x500servertool` to make regular backup copies of your directory. This ensures that if there is a problem on your system and the database becomes corrupt you have a recent copy of the data in your directory and can restore it. You can back up your database immediately, as described in “To Backup Your Directory”, or you can specify a schedule for backups, as described in “To Specify a Backup Schedule” on page 107. For information about restoring a backup copy, see “To Restore Your Directory” on page 107.

▼ To Backup Your Directory

1. **Log in as `root` or become superuser and start `x500servertool` by typing:**

```
# /opt/SUNWconn/x500/bin/x500servertool
```

2. **Choose Backup from the Database menu and choose Immediate from the submenu.**
The Backup Database window is displayed.

3. **Enter a file name for the backup copy.**
This name is used to identify the backup files.
4. **Press Backup.**

▼ To Specify a Backup Schedule

1. **Log in as root or become superuser and start x500servertool by typing:**

```
# /opt/SUNWconn/x500/bin/x500servertool
```

2. **Choose Backup from the Database menu and choose Scheduled from the submenu.**
The Scheduled Backup window is displayed.
3. **Enter a file name for the backup copy.**
This name is used to identify the backup files.
4. **Specify the time and date at which you want the database to be first backed up.**
5. **Specify interval at which you want the backup to be repeated.**
6. **Specify whether or not you want timestamps to be appended to the backup file names.**
7. **Press Schedule.**

To cancel a scheduled backup, choose Cancel Scheduled Backup from the Database menu.

▼ To Restore Your Directory

1. **Log in as root or become superuser and start x500servertool by typing:**

```
# /opt/SUNWconn/x500/bin/x500servertool
```

2. Choose Restore from the Database menu.

The Restore Database window is displayed.

3. Enter the file name of the backup you want to restore.**4. Press Restore.**

The database is restored from the backup you specified. The backup files are not deleted.

When you restore a backup copy of a database, you automatically restore the schema that was associated with that database. If you made any changes to the schema or to the directory information since the database was backed up, you need to repeat those changes.

Troubleshooting



This chapter contains general information about solving problems with your directory, hints about interworking problems, lists of error messages and recovery procedures, and information about the files and directories used by Solstice X.500. It contains the following sections:

<i>Accounting Information</i>	<i>page 109</i>
<i>The x500trace Utility</i>	<i>page 112</i>
<i>Error Messages from x500servertool</i>	<i>page 114</i>
<i>Schema Error Messages</i>	<i>page 119</i>
<i>Error Messages from x500datatool</i>	<i>page 124</i>
<i>Directories and Files</i>	<i>page 127</i>
<i>Interworking Problems</i>	<i>page 127</i>

Accounting Information

You can collect accounting information for your DSA using the accounting daemon, `x500account`. You can start accounting using `x500servertool` or from the command line. See “Accounting” on page 56 for information about starting accounting using `x500servertool`. To start the daemon from the command line, log in as `root` or become `superuser` and type:

```
# /opt/SUNWconn/x500/bin/x500account -f logfile -p
```

logfile is the name of the file where accounting information is stored. If you do not specify a file, information is directed to standard output.

-p indicates that you want the output to be formatted (as shown in “Formatted Accounting Information, not raw data (as shown in “Unformatted Accounting Information” on page 111).

Note – If you start accounting using `x500servertool`, the accounting information is not formatted.

Formatted Accounting Information

If you use the -p option, the accounting information is formatted as follows:

```
Bind from type: DN
User: user-DN
PSEL: psel, SSEL: ssel, TSEL: tsel, NSAP: nsap
Bind start: timestamp, Duration:t
No. of Ops: n, Arg length: argl
No. of Results: r, result length: rl
```

where:

- *type* is the type of bind, and is *DUA*, *LDAP to DSA* or *from DSA*
- *entity-DN* is the distinguished name of the component bound to your DSA
- *user-DN* is the distinguished name of the user making the request
- *psel*, *ssel*, *tset*, and *nsap* are the addressing information of the remote entity (these are always 0x for LDAP binds)
- *timestamp* is the time at which the bind was established, in UTC format
- *t* is the duration of the bind, in seconds
- *n* is the number of directory operations that took place while the bind was established
- *argl* is the length in bytes of operational arguments (such as filters)
- *r* is the number of results returned to the remote entity
- *rl* is the length in bytes of the result string returned to the remote entity

The following example shows information about three binds: an outgoing call to a DSA, an incoming call from a DUA, and an incoming call from an LDAP server.

```

Bind to DSA: /organizationName=ANYCORP/commonName=DEFAULT
  DSA is Trusted
  DSA is Allowed
  PSEL: Ox, SSEL: Ox, TSEL: Ox647361, NSAP: Ox12345678
  Bind start: 951122161543Z, Duration: 132
  No. of Ops: 0, Arg length: 0
  No. of Results: 0, Result length: 0
Bind from DUA: /organizationName=ANYCORP/commonName=DEFAULT_DUA
  User:
  PSEL: Ox, SSEL: Ox, TSEL: Ox647561, NSAP: Ox7F000001
  Bind start: 951122163123Z, Duration: 2
  No. of Ops: 0, Arg length: 0
  No. of Results: 0, Result length: 0
Bind from LDAP:
  User:
  PSEL: Ox, SSEL: Ox, TSEL: Ox, NSAP: Ox
  Bind start: 951122163123Z, Duration: 2
  No. of Ops: 0, Arg length: 0
  No. of Results: 0, Result length: 60

```

Unformatted Accounting Information

If you do not use the `-p` option, the accounting information is recorded as follows:

```

%d - bind type (1 = DUA, 2 = from DSA, 3 = to DSA, 4 = LDAP)
%s - DN of DSA (or DUA)
%d - 1 = is Trusted DSA only meaningful for a bind from a DSA
%d - 1 = is Allowed DSAonly meaningful for a bind from a DSA
%s - user DN only meaningful for a bind from a DUA
%s - Presentation selector
%s - Session selector
%s - Transport selector
%s - Network address
%s - start time in UTC Time format.
%d - duration of bind (secs)
%d - number or operations
%d - length of arguments (bytes)
%d - number of results
%d - length of results (bytes)

```

The following example shows information about the same three binds: an outgoing call to a DSA, an incoming call from a DUA, and an incoming call from an LDAP server.

```
3,/organizationName=ANYCORP/commonName=DEFAULT,2,1,,0x,0x,0x647361,
Ox12345678,951122161543Z,132,0,0,0,0
1,/organizationName=ANYCORP/commonName=DEFAULT_DUA,2,1,,0x,0x,
Ox647561,Ox7F000001,951122163123Z,1,0,0,0,0
4,/organizationName=ANYCORP/commonName=LDAP_USER,2,1,,0x,0x,0x,0x,
951122163123Z,1,0,0,0,0
```

The x500trace Utility

Use the X.500 trace utility to analyze information sent to and received by directory components. The trace utility detects protocol errors which in turn help you diagnose problems with construction of protocol elements.

Running the Trace Utility

To run the trace utility, log in as root or become superuser and type:

```
prompt# /opt/SUNWconn/x500/bin/x500trace [-dfr] [-p <protocols>] [-i <processes>] [<filters>]
```

The command-line options are:

- d: Returns a full trace of ROSE and Association Control Service Elements (ACSEs).
- f: Returns a full buffer trace. Without this option, only the first twenty lines of a message bodypart is displayed.
- r: Suppresses the names of PDU service elements from the trace. This option is valid only when you specify the pdu filter.

Valid protocols are shown in Table 7-1. If you do not specify a `-p` option, then all protocols are traced.

Table 7-1 x500trace Protocols

Protocol	Description
dap	Traces DAP
dsp	Traces DSP

Valid processes are shown in Table 7-2.

Table 7-2 x500trace OSI Processes

Process	Description
x500dsad	Traces the DSA server process
x500duad	Traces the DUA process
x500ldad	Traces the DAP part of the LDAP process

Valid filters are shown in Table 7-3.

Table 7-3 x500trace Filters

Filters	Description
debug	Recovers a full trace of the remote operation service element (ROSE) and ACSE presentation. The <code>-d</code> command-line option has the same effect as this filter.
events	Returns status and error messages recovered from the local DSA, LDAP server or DUA.
pdu	Returns ASN.1 decoding of incoming and outgoing messages.

Trace Output Example

The following example shows the trace output from a bind.

```
# /opt/SUNWconn/x500/bin/x500trace -p dap event pdu
x500duad      14:55:07[9973] DAP Bind Confirmation
```

```

b1 39 RO-BIND Result *[17] Length=57 (57)
  31 37 *[DirectoryBindResult] Length=55 (55)
    a0 35 Credentials *[0] Length=53 (53)
      a0 33 SimpleCredentials *[0] Length=51 (51)
        30 31 *[SimpleCredentials] Length=49 (49)
          a0 28 name *[0] Length=40 (47)
            30 26 *[DistinguishedName] Length=38 (38)
              31 0e *[RelativeDistinguishedName] Length=14 (36)
                30 0c *[AttributeValueAssertion] Length=12 (12)
                  06 03 [type] Length=3 (10)
                    55040a
                    OrganizationName
                  13 05 [PrintableString] Length=5 (5)
                    414e59434f
                    A N Y C O
                31 14 *[RelativeDistinguishedName] Length=20 (20)
                  30 12 *[AttributeValueAssertion] Length=18 (18)
                    06 03 [type] Length=3 (16)
                      550403
                      CommonName
                    13 0b [PrintableString] Length=11 (11)
                      44454641554c545f445341
                      D E F A U L T _ D S A
            a2 05 password *[2] Length=5 (5)
              04 03 [OctetString] Length=3 (3)
                61646d
                a d m
  
```

Error Messages from x500servertool

Note - This section does not contain an exhaustive list of all the error messages x500servertool returns, since many of them are self-explanatory. If you receive an error message which is not listed here and is not self explanatory, contact your authorized service provider.

Attempt to restore a database while no current database is selected

You must select a database before you can restore data.

Cannot read <reference-name> reference description

The entry for the DSA that holds the referenced name context has been deleted. Replace the DSA entry or update the reference.

Can't access database file <file>

The named file cannot be accessed. Check that the file is in the correct location and has appropriate permissions.

Can't backup schema file <file> to <backup> (reason)

The named file cannot be backed up. Use the information in the reason message to fix the problem.

Can't compile file <file> (reason)

The schema compiler cannot compile the named file. Use the information in the reason message to fix the problem. See "Schema Error Messages" on page 119 for information about schema errors.

Can't connect to process <process> (reason)

where reason is on of:

- Bind refused: multiple binding attempt
- Bind refused: wrong dib type
- Bind result invalid: unexpected
- Bind result invalid: wrong dib type
- Bind refused: bad sap or no channel
- Bind refused: already in progress
- Bind refused: already exists
- Error: while opening dib file
- Unknown reason

The named process cannot be contacted by x500servertool. Repeat the actions that caused this error. Try stopping and restarting the named process.

Can't create name context <name context> (internal error)

An internal error occurred. Contact your authorized service provider.

Can't delete database file <file>

An internal error occurred. Contact your authorized service provider.

Can't delete DSA <dsa name> (can't contact DSA)

An internal error has occurred and the DSA cannot be contacted. Try stopping and starting the DSA. If you see the same error, contact your authorized service provider.

Can't delete name context <name context> (unknown reason)

An internal error occurred. Contact your authorized service provider.

Can't get \$OSIROOT variable

This variable is used internally to identify the directories used by the software.

Can't get current database name

No database is currently select. Try selecting (or reselecting) a database.

Can't get DSA parameters

An internal error occurred. Contact your authorized service provider.

Can't get session layer configuration

An internal error occurred. Contact your authorized service provider.

Can't initialize compiler (reason)

An internal error occurred. Contact your authorized service provider.

Can't initialize mbx library (reason)

An internal error occurred. Contact your authorized service provider.

Can't load colors (invalid file format)

One of the configuration files is corrupted. Try restarting x500servertool. If you still see the same error, contact your authorized service provider.

Can't load configuration file (invalid header)

One of the configuration files is corrupted. Try restarting x500servertool. If you still see the same error, contact your authorized service provider.

Can't load ldap configuration (invalid file format)

One of the configuration files is corrupted. Try restarting x500servertool.
If you still see the same error, contact your authorized service provider.

Can't load references (invalid file format)

An internal error has occurred. Contact you authorized service provider.

Can't load reference coordinates (invalid file format)

One of the configuration files is corrupted. Try restarting x500servertool.
If you still see the same error, contact your authorized service provider.

Can't load remote DSAs (invalid file format)

An internal error has occurred. Contact you authorized service provider.

Can't open file filename (reason)

Use the information in the reason message to fix the problem.

Can't set <AllDsas> flag (can't contact DSA)

An internal error has occurred. Contact you authorized service provider.

Can't set DSA flags

An internal error has occurred. Contact you authorized service provider.

Can't set DSA parameters

An internal error has occurred. Contact you authorized service provider.

Can't unselect database

An internal error has occurred. Contact you authorized service provider.

Deletion of schema file not allowed (database exists)

You cannot remove a schema file that is being used, because it is likely to cause problems with the schema.

Internal error

An internal error has occurred. Contact you authorized service provider.

Internal error (missing bmr)

An internal error has occurred. Contact you authorized service provider.

Internal error (no method)

An internal error has occurred. Contact you authorized service provider.

Invalid suffix <suffix> (reason)

where reason is one of:

- even number of digits expected
- hexadecimal digits expected
- unexpected non printable character

There is an error in an attribute value you have supplied. Correct the value and try again.

- internal conversion error
- calloc failed

An internal error has occurred. Retry the actions that caused the error. If you see the same error, contact your authorized service provider.

Mailbox access failure (reason) while contacting <process>

An internal error has occurred. Stop and restart x500servertool.

Memory allocation failed for <number> bytes

An internal error has occurred. Stop and restart x500servertool.

No connection to <name> process

An internal error has occurred. Contact your authorized service provider.

Schema compilation error (error)

The schema could not be compiled for the reason given. Use the information in the reason message to fix the problem. See “Schema Error Messages” on page 119 for information about schema errors.

X.500 not initialized. Please run `</etc/rc2.d/S93x500server start>`

The X.500 server processes are not initialized. When Solstice X.500 is first installed you must start the server processes from the command line so that they are correctly initialized, using the command given. If this error occurs after Solstice X.500 has been running for some time, contact your authorized service provider.

Schema Error Messages

This section contains an alphabetical list of the error messages that can be returned by the schema compiler.

`<attribute>` is not known as an attribute or abbreviation

The value given in the error message is not a recognized attribute type or abbreviation. Correct the attribute type.

`does not start with a slash`

A distinguished name or relative distinguished name is not specified correctly.

`empty`

No distinguished name value is specified.

`empty OID`

An object identifier is not specified correctly.

`incorrect OBJECT-CLASS`

There is a mismatch between an object class and a value. Check that you have specified all object classes and values correctly.

`information is present at two incompatible places`

An attribute is specified as being both mandatory and optional in an object class definition.

`invalid OID`

The object identifier specified is invalid. Check that you specified the correct object identifier.

invalid schema modification: <error>

There is an error in an object class or attribute definition. For object classes, the error is one of the following:

- "subclass of" clause
You cannot change the SUBCLASS OF list.
- "named by" clause
You cannot change the NAMED BY list.
- "component" clause
You cannot change the object identifier of an object class.
- "must contain" clause
You cannot change the mandatory attributes.
- "may contain" clause
You cannot remove optional attributes.
- "preceded by" clause
You cannot remove the PRECEDED BY object class.
- "maximum of" clause
You cannot decrease the maximum number of attributes.
- "may attribute -> must attribute"
You cannot make a previously optional attribute mandatory.

For attributes, the error is one of the following:

- "attribute syntax" clause
You cannot change the attribute syntax.
- "component" clause
You cannot change the object identifier of an attribute.
- "protected by" clause
You cannot modify the protecting attribute.
- "protection level" clause
You cannot modify the protection level.
- "shield entry" clause
You cannot modify a SHIELD ENTRY definition.
- "control access" clause
You cannot modify a CONTROL ACCESS definition.

- "multi value" clause
You cannot make a previously multivalued attribute single-valued.

loop detected

A loop has been detected in a schema definition. Check that the subclass definitions in your schema are correct and consistent.

name too long

The name of the specified entity exceeds 40 characters.

OID <oid> is used more than once

The OID specified in the error message is used more than once. OIDs must be unique.

OID should start with 0, 1 or 2 instead of <value>

An object identifier is not correctly specified.

OID is too long

An object identifier is not correctly specified. It exceeds the maximum permitted length of 200 characters.

syntax is pre-defined

You are attempting to redefine a standard attribute syntax. This is not permitted.

The alias OBJECT-CLASS must be a SUBCLASS of top

The object class alias is not specified correctly. It must have the OID given in the error message.

The alias OBJECT-CLASS must have OID { joint-iso-ccitt ds objectClass 1 }

The OID for object class alias is not specified correctly. It must have the OID given in the error message.

The aliasedObjectName ATTRIBUTE must have distinguishedNameSyntax for ATTRIBUTE-SYNTAX

The syntax for attributes of object class aliasedObjectName is not specified correctly. It must be the syntax given in the error message.

The `aliasedObjectName ATTRIBUTE` must have OID { joint-iso-ccitt ds attributeType 1 }

The attribute `aliasedObjectName` has an incorrect OID. It must have the OID specified in the error message.

The `<attribute> ATTRIBUTE` has the OID of the `aliasedObjectName ATTRIBUTE`

The specified attribute has an incorrect OID. It must not have the OID for object class `aliasedObjectName`.

The `<attribute> ATTRIBUTE` has the OID of the `objectClass ATTRIBUTE`

The specified attribute has an incorrect OID. It must not have the OID for object class `attribute`.

the equals sign is missing from `<value>`

A value is not specified correctly.

The `objectClass ATTRIBUTE` must have OID { joint-iso-ccitt ds attributeType 0 }

The OID for object class `attribute` is not specified correctly. It must have the OID given in the error message.

The `objectClass ATTRIBUTE` must have `objectIdentifierSyntax` for `ATTRIBUTE-SYNTAX`

The syntax for attributes of object class `attribute` is not specified correctly. It must be the syntax given in the error message.

The top `OBJECT-CLASS` can not be a `SUBCLASS`

The object class is not specified correctly. It cannot be a subclass of any other class.

The top `OBJECT-CLASS` must have OID { joint-iso-ccitt ds objectClass 0 }

The OID for object class `top` is not specified correctly. It must have the OID given in the error message.

The <object-class> OBJECT-CLASS has the OID of the top OBJECT-CLASS

The OID of the specified object class is incorrect. It must not be the same as the OID used for object class top.

The <object-class> OBJECT-CLASS is the SUBCLASS of nothing (only top can be like that)

The specified OID has no superior class specified.

The <object-class> OBJECT-CLASS has the OID of the alias OBJECT-CLASS

The OID of the specified object class is incorrect. It must not be the same as the OID used for object class alias.

two or more clauses cannot be used together

An attribute definition includes more than one of the PROTECTED BY, SHIELD ENTRY and CONTROL ACCESS clauses. Only one of these can be included.

protection attribute should be INTEGER

The value of the protection attribute must be an integer. Specify the correct value.

undefined

In the schema definition, an element is referenced but not defined.

Error Messages from x500datatool

Note - This section does not contain an exhaustive list of all the error messages `x500datatool` returns since many of them are self-explanatory. If you receive an error message which is not listed here and is not self explanatory, contact your authorized service provider.

The error messages returned by `x500datatool` have the format:

```
<general problem>:  
<reason>  
<suggestion>
```

The `reason` component of the message may itself be a composite of messages returned by underlying services.

In the rest of this section, the error messages are listed in alphabetical order of the `general problem` component.

```
Bad selection of attribute to act as RDN:  
Please select an attribute which can act as a valid RDN
```

An attribute you have specified in a distinguished name is not a naming attribute.

```
Cannot get list of name contexts:  
naming context problem - <reason>  
Possible Schema inconsistency.
```

where `reason` is one of the following:

- Error creating naming contexts file
- Error searching naming contexts file
- Error opening naming contexts file
- Error reading from naming contexts file
- Error accessing naming contexts file
- Undefined naming context

An internal error has occurred. Repeat the action that caused the error. If you see the same error, contact your authorized service provider.

```
Cannot open add entry menu:  
Unable to get list of inferior object classes.  
Possible Schema inconsistency.
```

An internal error has occurred. Repeat the action that caused the error. If you see the same error, contact your authorized service provider.

```
Error in getting object class for current entry:  
object class problem - <Reason>  
There may be a schema inconsistency
```

where reason is one of the following:

- Error opening object class file
- Error searching object class file
- Error reading from object class file
- Error accessing object class file
- Undefined object class
- Object class violation

An internal error has occurred. Repeat the action that caused the error. If you see the same error, contact your authorized service provider.

```
Error in getting object class structure information:  
<Reason>  
There may be a schema inconsistency
```

where reason is one of the following:

- attribute problem - Error opening attributes file
- attribute problem - Error searching attributes file
- attribute problem - Error reading from attributes file
- attribute problem - Error accessing attributes file
- attribute problem - Undefined attribute
- attribute problem - No such value or attribute

- attribute problem - Undefined attribute type
- name problem - Naming error - no such object
- No attributes defined for objectclass "<Objectclass>"
- No attribute with OID "<OID>" found
- No objectclass with name "<name>" found

An internal error has occurred. Repeat the action that caused the error. If you see the same error, contact your authorized service provider.

```
Error in getting values for current object:
<Reason>
There may be a database problem
```

where reason is one of the following:

- attribute problem - Wrong attribute syntax ID
- attribute problem - Undefined attribute
- attribute problem - No such value or attribute
- attribute problem - Undefined attribute type
- name problem - Naming error - invalid attribute syntax
- name problem - Naming error - no such object

An internal error has occurred. Repeat the action that caused the error. If you see the same error, contact your authorized service provider.

```
Received signal - terminating.
```

An internal error has occurred, and x500datatool cannot start, or must stop processing. Try restarting x500datatool. If this error occurs frequently, contact your authorized service provider.

```
X.500 Data Editing Tool: Could not load schema information. There
may be a schema inconsistency
```

An internal error has occurred, and x500datatool cannot start. Check that the schema source files are accessible. Try restarting x500datatool. If this error occurs frequently, contact your authorized service provider.

Directories and Files

This following directories are used by Solstice X.500 software:

- `/etc/opt/SUNWconn/OSIROOT`
- `/etc/opt/SUNWconn/fsp`
- `/etc/opt/licenses`
- `/etc/rc2.d`
- `/opt/SUNWconn/x500/examples/programming`
- `/opt/SUNWconn/x500/include`
- `/opt/SUNWconn/x500/lib`
- `/opt/SUNWconn/x500/sbin`
- `/opt/SUNWconn/x500/schema`
- `/var/opt/SUNWconn/OSIROOT/conf`
- `/var/opt/SUNWconn/OSIROOT/x500/conf`
- `/var/opt/SUNWconn/OSIROOT/x500/default`
- `/var/opt/SUNWconn/OSIROOT/x500/database`

If your Solstice X.500 reports errors relating to access to files or directories, check that these directories are accessible.

Interworking Problems

Most interworking problems have one of the following causes:

- **Local errors.**
Check that you have fixed any errors reported by the Solstice X.500 software and in the software you are interworking with.
- **Schema.**
Check that the schemas are aligned. Note that the schemas need not be identical but they must be compatible. OIDs must be unique.

- Security policy.
Check that the access control and authentication policies are compatible. A user may be receiving error messages because access controls prevents information being seen, and not because the information does not exist.
- Bind timeout.
If binds are timing out, check network interconnection, working from the bottom up. If lower layer performance is slow you need to increase timeout values to prevent directory queries timing out before results are returned.

Index

A

- access control
 - sschema definition, 47
- access controls, 44
 - attribute, 46, 48
 - authentication, 50
 - default, 28
 - definitions, 87 to 89
 - syntax, 87
 - DSA, 48, 49
 - entry, 46, 47
 - levels, 46
 - request, 49
 - user, 47
 - user modify, 48, 50
- access rights
 - See access controls, 44
- accounting, 56, 109
 - formatted output, 110
 - starting, 57, 109
 - unformatted output, 111
- alias
 - distinguished name, 104
 - entry, 19
 - naming attribute, 104
 - object class, 104
 - optional attributes, 104
- ASN.1, 113

- association control service element (ACSE), 112

- attribute
 - access controls, 48
 - mandatory, 19
 - multivalued, 103
 - naming, 19
 - optional, 19
- attribute sets
 - definitions, 80 to 81
 - syntax, 80
- attribute syntaxes
 - definitions, 78 to 80
 - syntax, 78
- attribute types
 - definitions, 71 to 77
 - syntax, 71
- autoping
 - reference, 36

B

- backup
 - cancelling, 107
 - database, 106
 - schedule, 107

C

- chaining, 21, 51
- child entry, 19
- client, 18
- compose, 39
- configuration, 27
 - backing up, 40
 - current, 40
 - default, 27
 - loading, 40
 - printing, 41
 - saving, 40
- configuring
 - schema, 64
- cross reference, 21
- current entry
 - deleting, 105
 - x500datatool, 100

D

- DAP, 18
 - tracing, 113
- database, 19
 - backing up, 106
 - backup
 - scheduling, 107
 - creating, 32, 41
 - name, 32
 - name context, 52
 - restoring, 108
 - schema, 32, 41
 - selecting, 32
- DIB, 19
- directories, 127
- directory access protocol
 - See DAP, 18
- directory information, 19
- directory information base
 - See DIB, 19
- directory system, 17
 - model, 17
- Directory System Agent

- See DSA, 18

- directory system protocol
 - See DSP, 18
- Directory User Agent
 - SeeDUA, 18
- distinguished name, 19
 - composing, 40
 - current entry in x500datatool, 100
 - of alias, 104
- distributed operations, 51
 - chaining, 51
 - default, 51
 - multicasting, 51
 - policy, 52
 - referral, 51

DN

- See distinguished name, 19

DSA, 18

- access controls, 44, 48, 49
- addressing, 31
- AllDSAs flag, 44
- allowed, 34, 43, 44
- ping, 46
- resource usage, 53
- session usage, 54
- starting, 28, 29
- stopping, 37
- trusted, 34, 44
- tuning, 54

DSP, 18

- tracing, 113

DUA, 18

- DUA events
 - tracing, 112

E

- entries
 - adding, 91
 - deleting, 93
- entry, 19
 - access controls, 47
 - adding a subordinate, 103
 - adding subordinate alias, 104

- alias, 19
- deleting, 104
- displaying, 101
- modifying, 102
- error messages
 - logging, 57
 - mailing, 57

I

- information
 - loading, 91
 - managing, 91
- interworking problems, 127

K

- knowledge information, 21
 - creating, 35
 - distributed operations, 51

L

- LDAP, 18
 - configuring, 55
 - normal or web mode, 56
 - port, 55
- LDAP server
 - starting, 30
- lightweight directory access protocol
 - See LDAP, 18

M

- mandatory attribute
 - adding, 103
- multicasting, 21, 51
- multivalued attribute, 103

N

- name context, 19
 - adding, 33, 53
 - creating, 33
 - current, 52
 - DSA, 52

- reference, 52
- removing, 53
- naming attribute, 19
 - of alias, 104
- non-specific subordinate reference, 21

O

- object class, 19
- object classes
 - creating, 66
 - definitions, 81 to 87
 - example definition, 68
 - inheritance, 66
 - modifying, 66
 - syntax, 81
- object identifiers
 - defining, 65
 - uniqueness, 65
- optional attribute
 - adding, 103
- optional attributes
 - of alias, 104
- OSI
 - communication platform (stack), 24

P

- parent entry, 19
- ping
 - DSA, 46
 - reference, 36
- protecting attribute, 88
- protection level, 88
- protocol data unit (PDU), 25
- protocol trace, 112

R

- RDN, 19
- reference
 - checking, 36
 - creating, 35
 - name, 35

- ping, 36
- status, 36
- to DSA, 36
- reference type, 35
 - cross, 21
 - non-specific subordinate, 21
 - subordinate, 21
 - superior, 21
 - upper, 21
- referral, 21, 51
- relative distinguished name, 19
- remote DSA
 - adding, 33, 45
 - configuring, 45
 - deleting, 46
 - listing, 45
 - modifying, 45
 - name, 34
- remote operations service (ROSE)
 - tracing, 112
- restore
 - database, 108
- RFC1006, 24
- rk6d, 24
- rk6stat, 25
- rk6trace, 25
- root entry, 19

S

- schema, 19, 20, 41
 - access control, 87
 - access controls, 47
 - adding a file, 42
 - attribute sets, 80
 - attribute types, 71
 - checking, 42, 43
 - compilation, 64
 - configuring, 64
 - definition, 59
 - deleting a file, 43
 - errors, 119
 - example, 67
 - example definition, 60

- files, 41
 - how to extend, 66
 - list, 41
 - modifying, 43, 65
 - guidelines, 65
 - object classes, 66, 81
 - object identifiers, 65
 - purpose, 59
 - restoring, 108
 - security attributes, 90
 - standard definitions, 64, 71
 - storage, 64
- security attributes, 90
- select
 - entry in x500datatool, 100
- server, 18
- session parameters, 54
- subordinate alias
 - adding, 104
- subordinate entry
 - adding, 103
- subordinate reference, 21
- subtree, 19
 - deleting, 105
 - moving, 106
- superior reference, 21
- syntax supported, 95

T

- TCP/IP, 24
- trace utility, 112
- transport layer interface (TLI), 24
- tree structure, 19

U

- unauthenticated user
 - access controls, 50
- upper reference, 21

X

- x500.schema, 64

- x500account, 109
 - starting, 109
- x500clienttool, 23
- x500datatool, 24, 98
 - child entries, 100
 - current entry, 100
 - errors, 124 to 126
 - name context, 98, 100
 - new database, 100
 - options, 99
 - peer entries, 100
 - selecting an entry, 100
 - starting, 98
- x500export, 23, 92
 - file format, 93
 - syntaxes supported, 95
- x500import, 23, 91
 - file format, 93
 - syntaxes supported, 95
- x500servertool, 22
 - accounting, 109
 - backup, 106
 - date and time format, 58
 - errors, 114 to 119
 - icons, 58
 - map options, 58
- x500trace, 112
 - running, 112

