Digital X.500 Directory Synchronizer Configuration and Management Guide

Software Version: V1.2

Date: March 1995

Digital Equipment Corporation

Maynard, Massachusetts

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with terms of such license.

No responsibility is assumed for the use or reliability of equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Restricted Rights: Use, duplication, or disclosure by the U.S. 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."

Copyright ©1995 by Digital Equipment Corporation.

All rights reserved.

Printed in U.S.A.

Trademarks

The following trademarks are registered by Digital Equipment Corporation:

digital™
DEC
DECnet
MAILbus
Message Router
VAX
AXP
OpenVMS
ALL-IN-1
MAILbus 400

The following are third-party trademarks:

OSF/1 is a registered trademark of the Open Software Foundation, Inc.
cc:Mail is a registered trademark of cc:Mail, Inc.
Microsoft Mail is a registered trademark of Microsoft Corporation
OfficeVision is a registered trademark of International
PROFS is a registered trademark of International Business Machine Corporation (IBM)
Quickmail is a registered trademark of CE Software
SNADS is a registered trademark of International Business Machine Corporation (IBM)
UNIX is a registered trademark of UNIX System Laboratories

Contents

Preface xi

Chapter 1 Overview

1.1 Introduction 1–1
1.1.1 X.500 and Synchronizer-500 1–2
1.2 Features and Capabilities of Synchronizer-500 1–4
1.3 How Synchronizer-500 Works 1–5
1.3.1 Synchronizer-500's Major Components 1–6
1.3.1.1 Configuration Files 1–6
1.3.1.2 Record Description Files 1–7
1.3.1.3 X.500 Layout File 1–9
1.3.1.4 Metafiles 1–9
1.3.1.5 Foreign Directory Files 1–9
1.3.1.6 Directory Group ID 1–10
1.3.1.7 Synchronizer-500 Identifier Field 1–10
1.3.2 Import Operations 1–11
1.3.2.1 Import Initial Load Operations 1–12
1.3.2.2 Import Update Operations 1–12
1.3.3 Transaction Operations 1–12
1.3.4 Export Operations 1–13
1.3.5 Changes Operations 1–14
1.3.6 Standard Uniqueness Checking 1–16
1.3.7 User-defined Uniqueness Processing 1–18
1.3.8 PROFS and SNADS Alias Generation 1–20
1.3.9 Transferring Foreign Directory Files 1–21

Chapter 2 How Synchronizer-500 Works

2.1 Import Processing 2–2
2.1.1 Converting the Input Foreign Directory File to a New Metafile 2–2
2.1.2 Extracting X.500 Entries into the Old Metafile 2–2
2.1.3 Comparing the Old and New Metafile 2–3
2.2 Transaction Processing 2–4
2.2.1 Converting the Input Foreign Directory Files to Metafiles 2–4
2.2.2 Generating X.500 transactions from Metafiles 2–4
2.3 Export Processing 2–4
2.3.1 Extracting X.500 Entries into the Old Metafile 2–4
2.3.2 Converting the Old Metafile to the Foreign Directory File 2–6
2.4 Changes Processing 2–6
2.4.1 Extracting X.500 Entries into the New Metafile 2–7
2.4.2 Changes Operations Comparison of Old and New Metafiles 2–7
2.4.3 Converting the ADD, MODIFY, and DELETE Metafile Records 2–7

Chapter 3 Synchronizer-500 Set up and Start up

3.1 Customizing the DEC X.500 Directory Service Schema 3–1
3.2 Customizing the Synchronizer-500 X.500 Layout File 3–2
3.3 Setting up and Running a Synchronization Operation 3–2
3.3.1 Creating the Synchronizer-500 Instance Subdirectory 3–3
3.3.2 Setting up Synchronizer-500 Configuration Files 3–4
3.3.2.1 Location of Configuration Files 3–5
3.3.2.2 Configuration File Tables 3–5
3.3.3 Setting up Record Description Files 3–14
3.3.3.1 Import and Transaction Concatenated RDF Statements 3–17
3.3.3.2 RDF Syntax 3–17
3.3.3.3 RDF Conditional Statements 3–18
3.3.3.3.1 Import and Transaction RDF Conditional Statement Syntax 3–18
3.3.3.3.2 Export and Changes RDF Conditional Statement Syntax 3–19
3.3.3.4 Export and Changes Operator Characters 3–20
3.3.3.5 RDF Rules 3–21
3.3.4 Starting Synchronizer-500 3–21

Chapter 4 Suggestions for Running Synchronizer-500

4.1 Limiting the Size of Directory Groups 4–1
4.2 Choose Unambiguous Distinguished Names 4–2
4.3 Choosing an Appropriate X.400 Subscriber-naming Scheme 4–2
4.4 Allocating Disk Space 4–2
4.5 Spreading out Metafiles 4–4
4.6 Monitoring Synchronization Log Files 4–4

Chapter 5 An Example — Synchronizing with Microsoft Mail

5.1 Overview of the Synchronization Process 5–2
5.1.1 Mail System Configuration and Addressing 5–3
5.2 Import Process 5–4
5.2.1 Create the Microsoft Mail Import Subdirectory 5–4
5.2.2 Run the Microsoft Mail Import Utility 5–5
5.2.3 Transfer the Microsoft Mail File 5–5
5.2.4 Determine Format of the Microsoft Mail File 5–5
5.2.5 Writing the Import RDF 5–5
5.2.6 Setting up the Import Configuration File 5–10
5.2.7 Running the Import–The Initial Load 5–13
5.2.8 Running the Import Following the Initial Load 5–18
5.3 Export Process 5–23
5.3.1 Create the Microsoft Mail Export Subdirectory 5–23
5.3.2 Determine Format of the Microsoft Mail Export File 5–24
5.3.3 Writing the Export RDF 5–24
5.3.4 Setting up the Export Configuration File 5–27
5.3.5 Running the Export 5–29
5.3.6 Transfer the Synchronizer-500 Export Output File 5–31
5.4 Changes Process 5–31
5.4.1 Create the Microsoft Mail Changes Subdirectory 5–31
5.4.2 Determine Format of the Microsoft Mail Export File 5–32
5.4.3 Writing the Changes RDFs 5–32
5.4.4 Setting up the Changes Configuration File 5–32
5.4.5 Running the Changes Export 5–35
5.4.6 Transferring Synchronizer-500 Changes Output Files 5–37

Appendix A Sample DDS Synchronization

A.1 Import Process A–1
A.1.1 MAILbus System Activities A–2
A.1.1.1 Choose an Instance A–2
A.1.1.2 Copy the Sample RDF and Configuration File to the MAILbus System A–2
A.1.1.3 Run MAILbus Directory Synchronizer to Export DDS Data A–6
A.1.2 X.500 System Activities A–7
A.1.2.1 Create the DDS Import Subdirectory A–7
A.1.2.2 Transfer the DDS Export File to the X.500 System A–7
A.1.2.3 Setting up the Import RDF A–7
A.1.2.4 Setting up the Import Configuration File A–11
A.1.2.5 Running the Import A–12

Appendix B Sample SMTP Synchronization

B.1 Export Process B–2
B.1.1 Create the SMTP Export Subdirectory B–3
B.1.2 Writing the Export RDF B–3
B.1.3 Setting up the Export Configuration File B–5
B.1.4 Running the Export B–6
B.1.5 Transfer the Synchronizer-500 Export Output File B–8

Appendix C Sample PROFS Synchronization

C.1 Setting up and Running a Synchronization Operation C–1
C.2 Overview C–2
C.3 Import Process C–5
C.3.1 Create the PROFS Import Subdirectory C–5
C.3.2 Determine Format of the PROFS File C–5
C.3.3 Writing the Import RDF C–6
C.3.4 Setting up the Import Configuration File C–11
C.3.5 Running the Import—The Initial Load C–12
C.3.6 Running the Import Following the Initial Load C–16
C.4 Export Process C–19
C.4.1 Create the PROFS Export Subdirectory C–19
C.4.2 Writing the Export RDF C–19
C.4.3 Setting up the Export Configuration File C–20
C.4.4 Running the Export C–21
C.4.5 Transfer the Synchronizer-500 Export Output File C–22

Appendix D X.500 Directory Attributes

Appendix E Synchronizer-500 Error Messages

Appendix F Configuration File Templates

F.1 Import Configuration File Template F–2
F.2 Transaction Configuration File Template F–8
F.3 Export Configuration File Template F–10
F.4 Changes Configuration File Template F–13

Appendix G Synchronizer-500 Transaction Status Codes

Appendix H Synchronizer-500 X.500 Layout File

H.1 Layout File Records H–1
H.1.1 Object Records H–2
H.1.2 Relative Distinguished Name Records H–3
H.1.3 Attribute Records H–4
H.2 Syntaxes H–5
H.3 Uniqueness Check Flags H–7
H.4 Special Purpose Flags H–8

Appendix I Synchronizer-500 Files and their Locations

I.1 Location of Synchronizer-500 Files for an OpenVMS System I–1
I.2 Location of Synchronizer-500 Files for a UNIX System I–4

Index

Examples

5–1 Typical Microsoft Mail File 5–2
5–2 Sample Microsoft Mail X.500 Entry 5–3
5–3 Configuration File for Microsoft Mail Import 5–10
5–4 Initial Load of Microsoft Mail Users Processing 5–13
5–5 Initial Load Import OpenVMS Mail Message 5–15
5–6 Checking the Results of the Initial Load Import—import_microsoft.log 5–16

5–7 X.500 Entry for Microsoft Mail Users 5–18
5–8 Second Import Run Processing 5–19
5–9 Second Import Run Transaction Log and VERBOSE Output 5–20
5–10 Configuration File for Export 5–27
5–11 Microsoft Mail Export Processing 5–29
5–12 Exported X.500 Data in Microsoft Mail Import Format 5–30
5–13 Configuration File for Changes 5–33
5–14 Microsoft Mail Changes Processing 5–35
5–15 changes_microsoft_add.txt Output 5–37
5–16 changes_microsoft_delete.txt Output 5–37
A–1 RDF for DDS Export - dsu_export_dds.rdf A–3
A–2 Configuration File for DDS Export - DSU_CONFIG.DAT_EXPORT_DDS A–6
A–3 RDF for DDS Import - xdsu_import_dds.rdf A–8
A–4 Configuration File for DDS Import - xdsu_config.dat_import_dds A–11
A–5 Initial Load of DDS Users Processing A–13
A–6 Checking the Results of the Import - import_dds.log A–15
A–7 X.500 Entry for a DDS User A–16
B–1 RDF for SMTP Export - xdsu_export_smtp.rdf B–3
B–2 Configuration File for SMTP Export B–5
B–3 SMTP Export Processing B–6
B–4 Exported X.500 Data in SMTP Sendmail Alias Format B–8
C–1 Typical PROFS Nickname Manager File C–3
C–2 Configuration File for PROFS Import C–11
C–3 Initial Load of PROFS Users Processing C–13
C–4 Checking the Results of the Initial Load Import—xdsu_log.log C–15
C–5 X.500 Entry for PROFS User C–16
C–6 Second Import Run Processing C–17
C–7 Second Import Run Transaction Log—xdsu_log.log C–18
C–8 RDF for PROFS Export - xdsu_export_profs.rdf C–19
C–9 Configuration File for PROFS Export C–20
C–10 PROFS Export Processing C–21
C–11 Exported X.500 Data in Nickname Manager Format C–22
F–1 Import Configuration File Template F–2
F–2 Transaction Configuration File Template F–8
F–3 Export Configuration File Template F–10
F–4 Changes Configuration File Template F–13

Figures

1–1 Synchronizer-500 Directory Information Exchange 1–2
1–2 Synchronizer-500 Components 1–6
1–3 Mapping the Record Description File into X.500 1–8
1–4 How Import Operations Work 1–11
1–5 How Export Operations Work 1–13
1–6 How Changes Operations Work 1–15
3–1 Mapping Import and Transaction Foreign Directory File Records to X.500 Attributes 3–15
3–2 Mapping X.500 Attributes to Export and Changes Foreign Directory File Records 3–16
B–1 SMTP Messaging Network Configuration B–2
C–1 PROFS Messaging Network Configuration C–4

Tables

3–1 Import Configuration File 3–6
3–2 Transaction Configuration File 3–8
3–3 Export Configuration File 3–10
3–4 Changes Configuration File 3–12
3–5 RDF Record Formats 3–17
4–1 Import File Size Estimates 4–3
4–2 Transaction File Size Estimates 4–3
4–3 Export File Size Estimates 4–3
4–4 Changes File Size Estimates 4–3
D–1 Standard X.500 Directory Attributes D–2
D–2 emailInfo Auxiliary Class X.500 Directory Attributes D–4
G–1 Synchronizer-500 Transaction Status Codes G–1
H–1 Synchronizer-500 Special Flags H–8
I–1 OpenVMS File Locations I–1
I–2 UNIX File Locations I–4

Preface

Purpose of this Guide

This guide describes the post-installation steps of configuring and managing Digital X.500 Directory Synchronizer (Synchronizer-500).

Synchronizer-500 is a software application that can be used to build a true multivendor directory service, based on the DEC X.500 Directory Service. Synchronizer-500 functions as a ''synchronization engine'', both producing and accepting directory information for and from other vendors' directory services. The exchange and synchronization of properly formatted directory addressing information among multiple vendor platforms can greatly improve the ability of electronic mail users to communicate in a multivendor mail network.

Intended Audience

This guide is primarily intended for the person who will be configuring and managing Synchronizer-500. In this document, this person will be referred to as the Directory Administrator.

Structure of this Guide

This guide has five chapters and nine appendices.

- Chapter 1 is an overview of Synchronizer-500.
- Chapter 2 discusses in depth how Synchronizer-500 works.
- Chapter 3 describes how to set up and start Synchronizer-500 following installation.
- Chapter 4 describes operational considerations when running Synchronizer-500.
- Chapter 5 gives a complete example of synchronizing with a foreign directory (Microsoft Mail).
- Appendix A gives an example of synchronizing with DDS.

- Appendix B gives an example of synchronizing with SMTP.
- Appendix C gives an example of synchronizing with PROFS.
- Appendix D lists the X.500 Directory attributes and their characteristics supported by Directory Synchronizer-500.
- Appendix E lists Synchronizer-500 error messages.
- Appendix F lists the configuration file templates.
- Appendix G lists the Synchronizer-500 transaction status codes.
- Appendix H lists the Synchronizer-500 X.500 Layout file.
- Appendix I lists the Synchronizer-500 files and their locations.

Related Documentation

- *Digital X.500 Directory Synchronizer Installing on an OpenVMS System*
- *Digital X.500 Directory Synchronizer Installing on an OSF/1 System*
- *DEC X.500 Directory Service Installing on an OpenVMS System*
- *DEC X.500 Directory Service Installing on an OSF/1 System*
- *DEC X.500 Directory Service Management* Guide
- *DEC X.500 Directory Service Problem Solving* Guide

Terms, Acronyms and Conventions Used in this Guide

This guide utilizes the following terms, acronyms and conventions:

Terms

- **emailInfo Auxiliary Class** - An X.500 auxiliary class provided by Synchronizer-500 for the purpose of storing electronic mail information about 'Person' objects.
- **Synchronizer-500's root directory** - The directory where Synchronizer-500 is installed. By default this is disk:[DXD$DIRECTORY.XDSU$ROOT] on OpenVMS and /var/dxd/xdsu on UNIX.

Acronyms

DIB - Directory Information Base
DIT - Directory Information Tree
DSA - Directory System Agent
DUA - Directory User Agent
DXIM - Digital X.500 Information Manager
RDN - Relative Distinguished Name
XDSU - Digital X.500 Directory Synchronizer Utility

Chapter 1
Overview

This chapter provides an overview of Digital X.500 Directory Synchronizer (Synchronizer-500 or XDSU). The chapter first briefly discusses what Synchronizer-500 is and then gives an overview of how Synchronizer-500 works.

1.1 Introduction

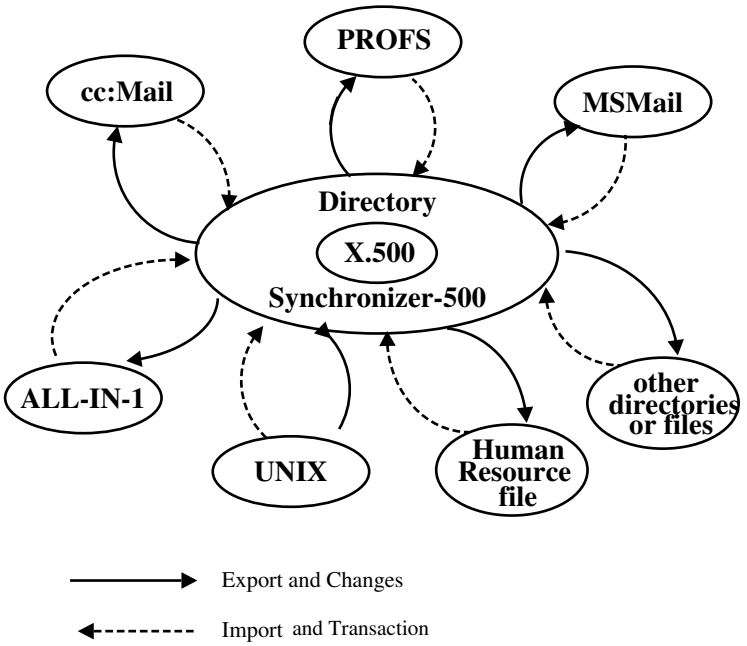
Digital X.500 Directory Synchronizer is a software application that allows the creation of a true enterprise-wide directory service. Synchronizer-500 is a full-featured management Directory User Agent (DUA) which enables the automated and bi-directional exchange of electronic directory information in a multivendor environment. Synchronizer-500 supports the X.500 Directory Information Tree structure suggested in Annex B of the X.521 Recommendations, with customizable extensions which are designed specifically for use in a multivendor electronic mail network.

For purposes of synchronizing electronic mail directories, Synchronizer-500 and Digital's X.500 product form the centerpiece of a powerful multivendor mail network. Using Synchronizer-500, bi-directional synchronization can be achieved between X.500 and electronic mail directories such as: Digital MAILbus/Distributed Directory Service (DDS), MAILbus 400, IBM PROFS or OfficeVision, Microsoft Mail, cc:Mail, UNIX Sendmail, and ALL-IN-1. Support for additional custom, legacy, and newly emerging directory products can be accommodated using Synchronizer-500's customizable configuration files.

As shown in Figure 1–1, Synchronizer-500 uses X.500 as the central repository for all multivendor information. Directory information is put into the X.500 Directory Information Base (DIB) using the import or transaction operations. Directory information is extracted from the X.500 DIB using the export or changes operations.

Synchronizer-500 can also simply be used for the bulk loading and updating of the X.500 directory from a simple flat file in place of DXIM scripts without the need to synchronize with a foreign directory.

Figure 1–1: Synchronizer-500 Directory Information Exchange



Synchronizer-500 has four modes of operation:

- The *import* operation allows the initial loading and updating of the X.500 DIB with master list information from foreign directories.
- The *transaction* operation allows the initial loading and updating of the X.500 DIB with changes information from foreign directories.
- The *export* operation permits information to be extracted from the X.500 DIB for use in an initial load or update of foreign directories.
- The *changes* operation permits the changes-only that have occurred in the X.500 DIB since the last changes operation to be extracted and written to separate output files for use in updating foreign directories.

1.1.1 X.500 and Synchronizer-500

The X.500 Directory Service stores information as **directory entries**, each entry representing an object, such as a person, a place, an organization, or a computer. The directory entries are organized into a **Directory Information Tree (DIT)**. The DIT is a structured database in which some entries are subordinate to others. The hierarchical structure is the basis of each entry's unique, unambiguous name.

Within the Directory Service, there should be one directory entry for each object that you want to represent. Each entry is made up of a set of attributes, each of which has one or more values.

An **attribute** is information of a particular type about an entry. Every attribute in an entry has an **attribute type** and at least one **attribute value**. For example, an entry might have an attribute type of *surname* with an attribute value of *Smith*.

A typical entry can have several attribute values that represent names by which the object is known, such as surnames and common names. However, so that the Directory Service can unambiguously identify an individual entry, each entry also has a **distinguished name**. A distinguished name is made up of a sequence of terms, each of which is called a **relative distinguished name (RDN)**. Each RDN in a distinguished name represents one of the entries that forms the path from the root of the DIT to that entry. Each RDN is an attribute type and value chosen from the attributes of the entry that it names.

For example, a distinguished name might be as follows:

```
/C=US/O=Digital/OU=Sales/CN="Joan Smith"
```

The abbreviation C means *countryName*, O means *organizationName*, OU means *organizationalUnitName*, and CN means *commonName*. The four RDNs listed in the distinguished name are separated by the / character, and are listed in order of hierarchical superiority, with the RDN of the entry closest to the root of the DIT listed first.

Every entry is classified according to the characteristics of the object that it represents. When you create an object you specify what **class** of entry it is. DEC X.500 Directory Service provides several different classes. Each class definition specifies mandatory attributes and optional attributes for entries of that class.

Synchronizer-500 is used to store entries in the DIT based upon the class that is specified by the Directory Administrator. Synchronizer-500 provides primary support for the organizationalPerson class. This includes support for the complete set of mandatory and optional attributes. (See Appendix B of the *DEC X.500 Directory Service Management Guide.*) Extensibility is included in Synchronizer-500 to support other classes such as organizationalRole and residentialPerson.

When Synchronizer-500 is used for purposes of synchronizing electronic mail directories, the entries that will be stored in the X.500 DIT are 'Person' objects. Each 'Person' object entry can be extended to hold additional information about electronic mail. This is accomplished through an an auxiliary class called **emailInfo** provided by Synchronizer-500. This auxiliary class allows for information to be stored in X.500 for the purposes of supplying information for a multivendor electronic mail network.

Note

This document assumes that a 'Person' class, such as organizationalPerson, and the Synchronizer-500 emailInfo auxiliary class is being used.

In addition to specifying the class of objects being synchronized into the X.500 DIT, the Directory Administrator specifies the attribute types and attribute values each entry will have during the configuration of Synchronizer-500. It is up to the Directory Administrator to make sure that all mandatory attributes for the class they are synchronizing with have been specified. If all mandatory attributes have not been specified to Synchronizer-500, the X.500 transaction will fail.

For more information on topics covered in this section, please refer to the *DEC X.500 Directory Service Management Guide*.

1.2 Features and Capabilities of Synchronizer-500

The features and capabilities of Synchronizer-500 are summarized below:

- Record description files (RDFs) permit the import and export of information from/to a foreign directory. RDFs allow the Directory Administrator to define for Synchronizer-500 the format of the foreign directory file being imported, the file being exported to the foreign system, and the add, modify, and delete files for changes operations.
- Synchronizer-500 provides the capability to load foreign directory information into the X.500 DIT as entries in an object class. After the initial load, subsequent import operations generate ADD/MODIFY/DELETE transactions to update the X.500 entries.
- Synchronizer-500 logs all X.500 transactions performed during each import run. The log file contains a record for each transaction. The record lists the transaction type (ADD/MODIFY/DELETE), the entry's distinguished name, and the status (SUCCESS or FAILED) of the transaction. If the transaction has a FAILED status, the reason for the failure is listed.
- On import, Synchronizer-500 can create superior entries in the DSA naming context based on the field CREATE_ROOT in the import configuration file. If, for example, your DSA contains a naming context entry for /c=US/o=Digital and the RDF contains a record for the relative distinguished name (RDN) organizationalUnit=Sales, Synchronizer-500 will create the superior entry /c=US/o=Digital/ou=Sales. Each of the entries added to X.500 during the import run will then have the RDNs for the superior entry as part of its distinguished name.
- On import, Synchronizer-500 can optionally check the uniqueness of selected X.500 attribute values in the emailInfo auxiliary class. By default, these attributes are: X.400 1984 and 1988 standard attributes, PROFS and SNADS user ID and location pairs, X.400 1984 and 1988 addresses, SMTP address, Microsoft Mail address, cc:Mail address, MAILbus address, Quickmail address, and up to 5 additional attributes which can contain other electronic mail user agent address aliases. If the uniqueness check fails, the proposed ADD or MODIFY transaction is rejected and logged.
- On import, Synchronizer-500 provides the Directory Administrator with the capability to specify an X.500 attribute that Synchronizer-500 will guarantee to be unique (up to a selected length) when using import operations to add data to X.500.
- On import, Synchronizer-500 has an optional capability to automatically generate and assign unique PROFS and SNADS user ID and location pairs in the emailInfo auxiliary class.
- On export, Synchronizer-500 uses highly-flexible configuration files that permit the extracted directory information to be tailored into a form ready for use by the receiving foreign directory. Synchronizer-500 allows for an extract of the full directory, a subset of the directory (based on Directory Group ID or selected X.500 attributes), or only the recent changes.
- Synchronizer-500 provides the ability to invoke user-written procedures at a specified point during a run and at the completion of a Synchronizer-500 run. This capability allows the Directory Administrator to perform necessary operations, such as copying the output file to the foreign system.

- A summary report of transaction statistics is generated after all processing has been completed. The report specifies the start- and end-processing time stamps; the Group ID; the total number of transactions processed; the number of successful ADD, MODIFY, and DELETE transactions; and the number of transactions rejected.
- Synchronizer-500 optionally notifies the Directory Administrator, via electronic mail (VMSmail on OpenVMS and sendmail on UNIX), of the start and end of Synchronizer-500 runs, as well as any error conditions causing Synchronizer-500 to end prematurely. This message includes the summary report.
- An unlimited number of separate instances of Synchronizer-500 can be run at any one time. An instance is a single import, transaction, export, or changes run. Any combination of import, transaction, export, or changes runs can be simultaneously executed.

1.3 How Synchronizer-500 Works

Synchronizer-500 uses flat (sequential) text files in a fixed format to facilitate the exchange of information between the X.500 DIT and foreign directories. Synchronizer-500 accepts a formatted directory "feed" of a foreign directory, extracts the information from it into a temporary file (called a metafile) that is compatible with Synchronizer-500, and uses it to load (or update) the appropriate records in Digital's implementation of the 1988 CCITT X.500 Directory Information Base (DIB) for import or transaction operations.

In a similar fashion, Synchronizer-500 extracts information from X.500 into metafiles, and then converts the metafiles to either a complete flat file (for export operations) or flat files containing the changes-only (for changes operations), either of which can be used to update the foreign directory.

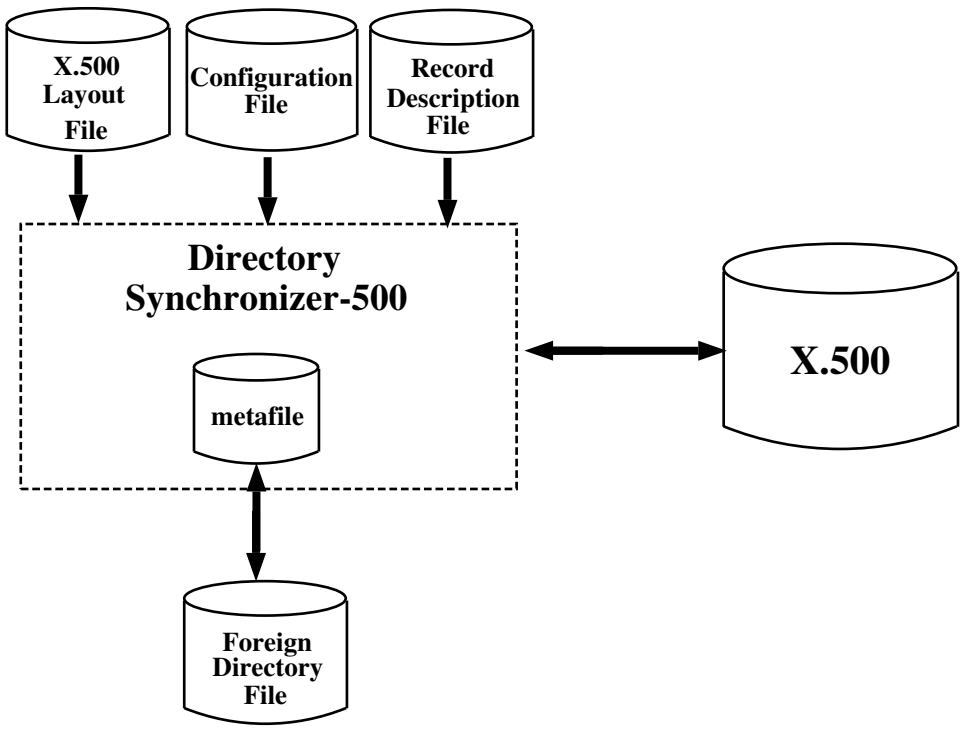
This section defines in more detail the components and some of the features of Synchronizer-500 that support its synchronization capabilities:

- Synchronizer-500's major files and fields:
  - — configuration files
  - — record description files (RDFs)
  - — foreign directory files
  - — metafiles
  - — X.500 layout file
  - — directory Group ID (GID) field
  - — Synchronizer-500 identifier field
- import operation
- transaction operation
- export operation
- changes operation
- uniqueness checking

1.3.1 Synchronizer-500's Major Components

Synchronizer-500's major components comprise a specific set of files and fields. The specific files are: configuration files, RDFs, foreign directory files, metafiles, and the X.500 layout file. The relationship between these files and Synchronizer-500 is shown in Figure 1–2 and explained in the following sections.

Figure 1–2: Synchronizer-500 Components



1.3.1.1 Configuration Files

Configuration files contain the control parameters for a Synchronizer-500 run. The four types of configuration files are *import, transaction, export*, and *changes*. When Synchronizer-500 starts, it reads the configuration file to determine the type of operation (import, transaction, export, or changes), the location of the foreign directory file, the location of the RDF, and other pertinent information.

Chapter 3 explains in more detail how to set up and use configuration files.

1.3.1.2 Record Description Files

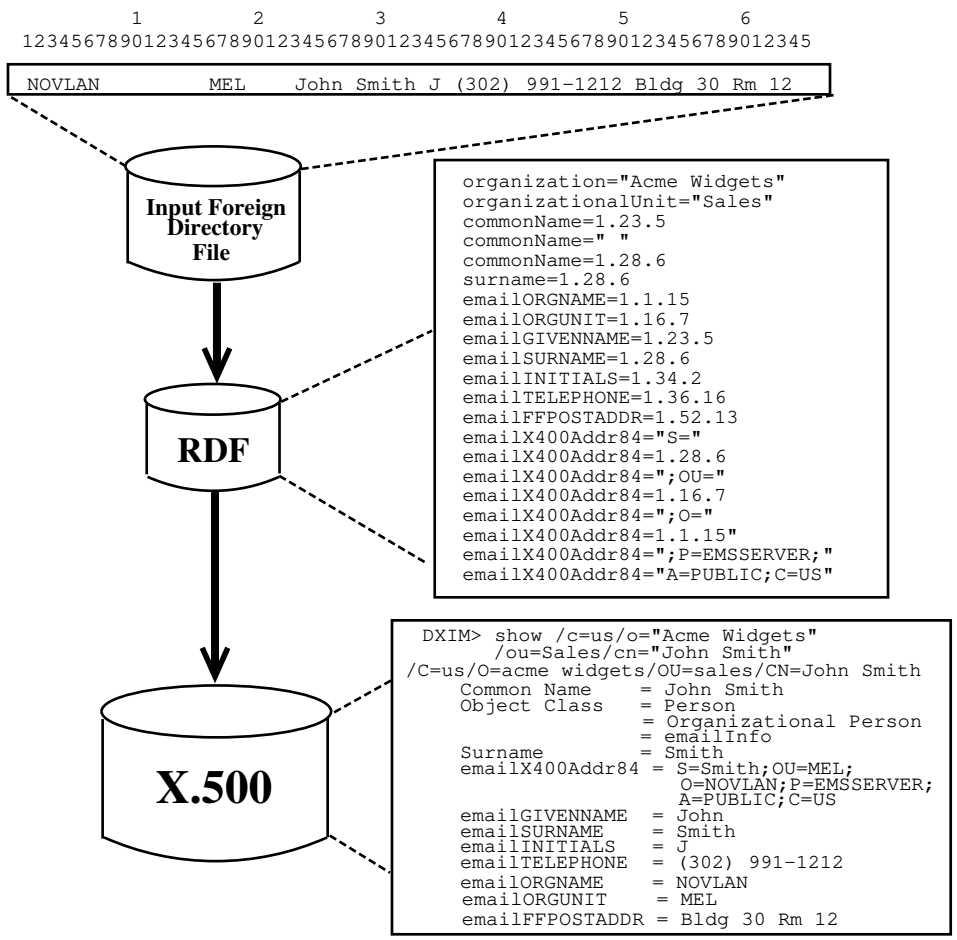
Record description files (RDFs) contain a series of individual statements that describe the format of the flat sequential data files to Synchronizer-500.

For import and transaction operations, Synchronizer-500 uses the information in the RDF(s) to create a mapping between the contents of a foreign directory file and corresponding X.500 attributes that will hold the directory information.

For export and changes operations, Synchronizer-500 uses RDFs to convert the metafile containing X.500 data to the appropriate format for the foreign directory. (See Section 1.3.1.4 for more information on metafiles.)

Figure 1–3 shows an example RDF and how it maps information from the foreign directory file into the appropriate attributes in X.500 for an import operation.

Figure 1–3: Mapping the Record Description File into X.500



Chapter 3 explains how to set up and use RDFs.

The Synchronizer-500 kit includes example RDFs for processing many popular directory formats. These files are placed in the Synchronizer-500 root directory during installation.

1.3.1.3 X.500 Layout File

Synchronizer-500 utilizes an X.500 layout file that provides information about the portions of the X.500 DIB on which Synchronizer-500 will be operating. This information includes rules about the objects and attributes in the DIB. See Appendix H for more information about the X.500 layout file.

Note

If an object or attribute is not described in this file Synchronizer-500 will not be able to operate on it.

1.3.1.4 Metafiles

Metafiles are internal Synchronizer-500 work files. Before Synchronizer-500 processes any directory file, it converts the file to metafile format. A metafile is a compressed record data file that contains all X.500 attributes identified in Synchronizer-500's X.500 layout file. (See Section 1.3.1.3 for more information on the X.500 layout file.) Each attribute in the metafile has sufficient space allocated to it to accommodate the maximum length of that attribute. By converting all data it uses to metafile format, Synchronizer-500 can allow for consistent and accurate comparison of data regardless of the original source or final destination of the data.

By default metafile records are 22550 bytes long. (Note: This number will change if Synchronizer-500's default X.500 layout file is modified.) Because metafile records have large amounts of blanks, they are compressed (blanks reduced to a sentinel byte and a count byte) before they are written to disk to preserve disk space. When a record is brought into memory, it is expanded to its original format.

By default metafiles are deleted at the end of a Synchronizer-500 run. In export and changes mode, metafiles can be saved for future usage by naming them in the appropriate Configuration file records.

See Appendix D for the default supported X.500 attributes and their characteristics.

1.3.1.5 Foreign Directory Files

Foreign directory files are used in different ways, depending on whether Synchronizer-500 is performing an import, transaction, export, or changes operation.

For an *import* operation, the foreign directory file is used as the input data source to a Synchronizer-500 run. The file must be a COMPLETE COPY or MASTER LIST of all the entries in the foreign directory to ensure the completeness and integrity of the X.500 directory. In this mode, Synchronizer-500 does not operate on modification information (such as ADD/MODIFY/DELETE) but instead works with full copies of the foreign directory. (See Section 2.1 for more information on the import operation.)

For a *transaction* operation, three foreign directory files are used as the input data source to a Synchronizer-500 run. The files contain foreign directory modification information (ADD/MODIFY/DELETE) and are used to update the X.500 directory. (See Section 2.2 for more information on the transaction operation.)

For an *export* operation, the foreign directory file is output from a Synchronizer-500 run. It is created by Synchronizer-500 by extracting information from X.500 in the format specified in the RDF. The extracted information can be a complete ''snapshot'' of X.500 or selected subsets of the information. On completion of the export operation, the foreign directory file can be transferred to the foreign system so that it can be merged with or used to update the foreign directory. (See Section 2.3 for more information on the export operation.)

For a *changes* operation, the foreign directory files are output from a Synchronizer-500 run. They are created by Synchronizer-500 by extracting only the changes that have occurred to X.500 since the last changes operation. The foreign directory files are written out in the format specified in the RDF. Separate files are created for ADD transactions, MODIFY transactions, and DELETE transactions. Each output file has its own RDF. On completion of the changes operation, the output foreign directory files can be transferred to the foreign system, where they can be used to update the foreign directory. (See Section 2.4 for more information on the changes operation.)

1.3.1.6 Directory Group ID

Each import and transaction record processed is tagged with one unique Directory Group ID (GID), which is assigned by the Directory Administrator in the configuration file. The Group ID identifies the foreign directory from which the record came. The Group ID is used for identification, maintenance, and reporting functions.

The Group ID is used during import and transaction operations to specify the group of directory entries in X.500 with which to synchronize. It is also used during export or changes operations to allow the Directory Administrator to specify which Directory Group entries to extract.

When Synchronizer-500 creates an X.500 entry, the Group ID is stored in one of the X.500 attributes. The Directory Administrator selects which attribute to use for this purpose in the configuration file.

It is recommended to choose a Group ID that clearly identifies the foreign directory to which the entries belong. For example, CC:MAIL NEW YORK or /c=us/o=Digital/ou=Sales.

1.3.1.7 Synchronizer-500 Identifier Field

Each X.500 entry is tagged with an identifier so that Synchronizer-500 knows that it manages that record. This tag is called the Synchronizer-500 identifier (SYNCH_ID), and is stored in one of the X.500 attributes. The X.500 attribute and the value are specified by the Directory Administrator in the configuration file.

Note

The SYNCH_ID attribute and value should be the same for all Synchronizer-500 instances and should not be changed once its been defined.

Synchronizer-500 will not access a record if the SYNCH_ID value for this run does not not match the SYNCH_ID value found in the specified attribute in the X.500 directory.

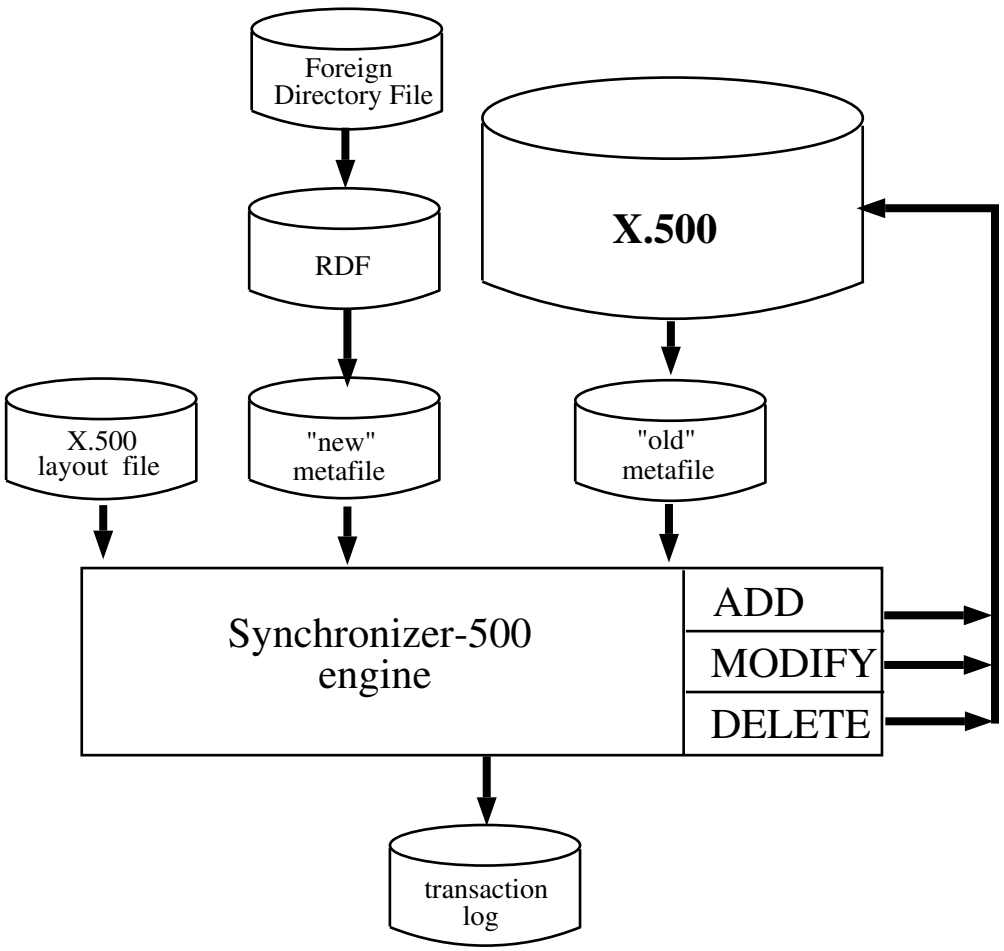
1.3.2 Import Operations

The first time an import operation from a foreign directory is performed is called an initial load. In this operation, all entries in the foreign directory file are ADD transactions to the X.500 directory.

All subsequent imports are called updates. In an update operation, the information in the foreign directory file is compared with the existing entries in X.500 with the specified Group ID and ADD/MODIFY/DELETE transactions to update the X.500 DIT are generated.

Figure 1–4 depicts how the import operation works. The figure is explained in more detail in the subsequent text, Section 1.3.2.1 and Section 1.3.2.2.

Figure 1–4: How Import Operations Work



1.3.2.1 Import Initial Load Operations

Synchronizer-500 first reads all records from the foreign directory file. It then interprets them according to the RDF, generates a distinguished name for each entry according to the configuration file, and writes them to the *new* metafile based upon the X.500 layout file. After all records have been processed, the *new* metafile is sorted on the X.500 DISTINGUISHED NAME.

Synchronizer-500 then extracts from X.500 all directory entries with the specified Synch ID and Group ID in the specified search context, and writes them to a metafile, called the *old* metafile (based upon the X.500 layout file). In an initial load, there are no directory entries with the specified Group ID so the *old* metafile is empty.

Synchronizer-500 compares the two metafiles to see what the differences are between them. As a result of this comparison, Synchronizer-500 recognizes that the old metafile is empty, and performs ADD transactions to the specified class to transfer all records contained in the *new* metafile to X.500. All these initial load ADDs are tagged with the unique Group ID and with the Synch ID as specified by the Directory Administrator in the configuration file. Each ADD transaction is logged in the Synchronizer-500 transaction log file and checked for uniqueness, as described in Section 1.3.6.

1.3.2.2 Import Update Operations

As in the initial load import operation, Synchronizer-500 processes and sorts all records from the foreign directory file into the *new* metafile during import update operations.

Synchronizer-500 extracts from X.500 all directory entries with the specified Synch ID and Group ID and writes them into the *old* metafile. When the extraction process is complete, the *old* metafile is sorted on the X.500 DISTINGUISHED NAME.

Synchronizer-500 compares the two metafiles to see what the differences are between them. By doing this, Synchronizer-500 can generate the necessary ADD/MODIFY/DELETE transactions to ensure that X.500 is synchronized with the foreign directory. All ADD transactions are tagged with the unique Group ID and with the SYNCH_ID, just as in the initial load operation. Each transaction is logged in the Synchronizer-500 transaction log file and checked for uniqueness, as described in Section 1.3.6.

During an import update operation, Synchronizer-500 makes three passes over the information. DELETE transactions are performed in the first pass; MODIFY transactions are performed in the second; and ADD transactions are performed in the third. This processing sequence is done to ensure that proper results occur if an entry's X.500 DISTINGUISHED NAME changes.

1.3.3 Transaction Operations

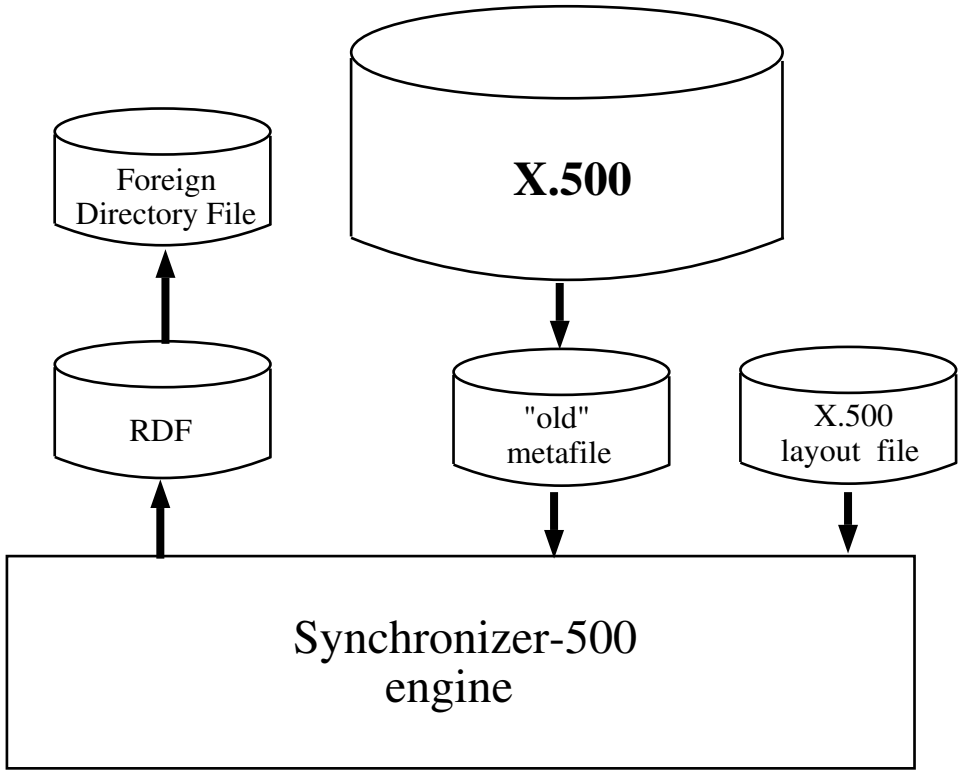
Transaction mode of Synchronizer-500 can be used when a foreign directory contains the "master list" information AND the foreign directory can also generate changes information (in separate ADD, MODIFY, and DELETE files) for input to Synchronizer-500. The changes information is contained in three foreign directory files: ADD, MODIFY, and DELETE transactions. All three are optional (i.e. they don't need to be specified during every transaction mode run.).

Synchronizer-500 processes all records from the DELETE foreign directory file first, then the MODIFY file, and finally the ADD file. Most characteristics of import mode (RDF, initial load, Synch ID, Group ID, logging) are identical to transaction mode. The lone exception is that Synchronizer-500 does not extract current X.500 directory information first for comparison.

1.3.4 Export Operations

Synchronizer-500's export feature allows for all entries in X.500, or a designated subset of them, to be extracted into a sequential, fixed-format file for use by a foreign directory. The export operation process is shown in Figure 1–5.

Figure 1–5: How Export Operations Work



The Directory Administrator can select for export all entries in X.500 with a distinguished value (commonName) or up to 1024 separate subgroups. Selection criteria for the subgroups can be the Group ID within a specific tree, and any of the following X.400 attributes in the emailInfo auxiliary class:

- emailCOUNTRY
- emailADMD
- emailPRMD

- emailORGNAME
- emailORGUNIT
- emailORGUNIT2
- emailORGUNIT3
- emailORGUNIT4

Note

A wild card, the asterisk * character, can be used as the value for the Group ID.

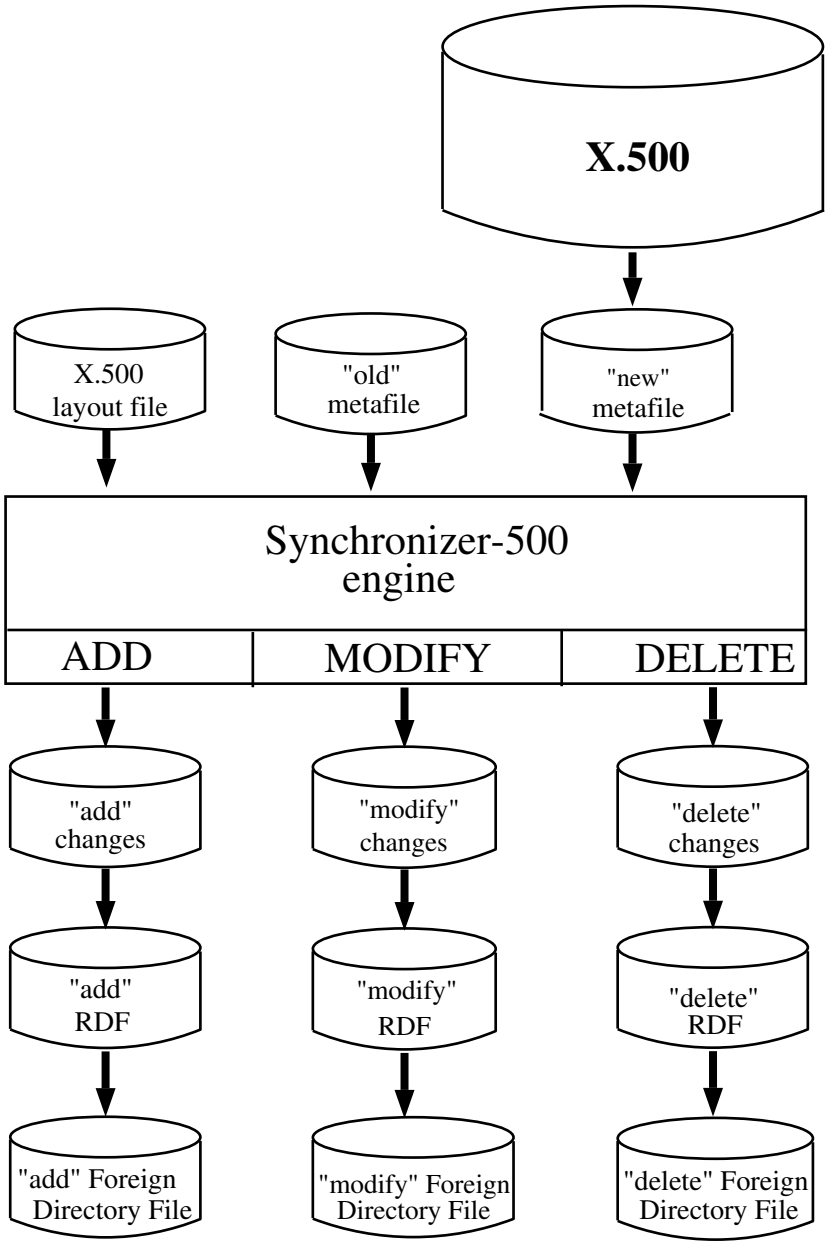
The format of the information written to the output file is specified through the use of RDFs. Selection criteria is specified in the export configuration file. See Section F.3 for detailed information.

The mechanics of the export operation are simple. Synchronizer-500 extracts the specified X.500 directory entries to the *old* metafile. All metafile records are read and written to the output foreign directory file, according to the format specified in the RDF.

1.3.5 Changes Operations

Synchronizer-500's changes operation can detect the updates (ADD/MODIFY/DELETE) that have occurred to X.500 since the last changes run and write those changes-only records to separate output files. The changes operation process is shown in Figure 1–6.

Figure 1–6: How Changes Operations Work



In a changes operation, Synchronizer-500 extracts information from X.500 into a metafile using the same selection criteria as described for an export operation.

It compares this *new* metafile to the *old* metafile and determines the changes that have occurred between the two files. (The old metafile was created by a previous export or changes operation.) It then writes these changes to separate add, modify, and delete output foreign directory files. Each of the add, modify, and delete output files has its own RDF that describes the file's format.

1.3.6 Standard Uniqueness Checking

During the import and transaction process, Synchronizer-500 performs a standard set of uniqueness checks before the actual X.500 ADD and MODIFY transactions occur. A uniqueness check means that the emailInfo auxiliary class is searched for the existence of an entry with the same X.400, PROFS, SNADS, or electronic mail address alias attributes. If such an entry is found, the uniqueness check is deemed a failure, the proposed transaction is rejected, and the failure and reason are recorded in the Synchronizer-500 transaction log file.

This uniqueness checking is a critical element in a multivendor electronic mail directory service. Because no two electronic mail addresses can be the same, Synchronizer-500 will not allow entries to be added into X.500 if an entry already exists with the same electronic mail alias address.

Definitions for standard uniqueness checking are defined in the X.500 layout file (xdsu_layout.dat) and can be individually tailored by the Directory Administrator. See Section H.3 for more information. By default, Synchronizer-500 performs the following uniqueness checks for all ADD and MODIFY transactions:

- X.400 standard attributes - 1984 and 1988
- PROFS username and location
- SNADS username and location
- 1984 X.400 address
- 1988 X.400 address
- SMTP address
- Microsoft Mail address
- cc:Mail address
- MAILbus address
- Quickmail address
- up to 5 additional electronic mail address aliases attributes
- user-specified X.500 attribute (see Section 1.3.7 for more information on user-specified X.500 attribute).

Each uniqueness check using the configuration file parameter UNIQUE_SEARCH_CONTEXT as a base entry for the X.500 search. By default, this parameter is null (e.g. the entire DIT is searched), but can be specified to limit the search for uniqueness checking.

Each uniqueness check is described in more detail below.

- A 1984 X.400 uniqueness check is implemented by doing an X.500 search using the following attributes in the emailInfo auxiliary class (the search only uses attributes that have been defined in the RDF):

  emailPCOUNTRY
  emailADMDNAME
  emailPRMDNAME
  emailORGNAME
  emailORGUNIT
  emailORGUNIT2
  emailORGUNIT3
  emailORGUNIT4
  emailSURNAME
  emailGIVENNAME
  emailINITIALS
  emailGENERATION
  emailDDA
  emailDDA2
  emailDDA3
  emailDDA4

  If this search finds an existing directory entry with the same 1984 X.400 attribute values, the X.500 transaction fails with an X484NOTUNIQ status.

- A 1988 X.400 uniqueness check is implemented by doing an X.500 search using the following attributes in the emailInfo auxiliary class (the search only uses attributes that have been defined in the RDF):

  emailPCOUNTRY
  emailADMDNAME
  emailPRMDNAME
  emailORGNAME
  emailORGUNIT
  emailORGUNIT2
  emailORGUNIT3
  emailORGUNIT4
  emailNAME
  emailDDA
  emailDDA2
  emailDDA3
  emailDDA4

  If this search finds an existing directory entry with the same 1988 X.400 attribute values, the X.500 transaction fails with an X488NOTUNIQ status.

- X.500 electronic mail address alias uniqueness checks are performed. By default, up to 12 electronic mail address aliases can be stored in the emailInfo auxiliary class. The uniqueness checks are implemented by performing an X.500 search for each mail alias. Hence, up to 12 searches can be performed (the search only uses the attributes that have been defined in the RDF):

  emailX400Addr84

emailX400Addr88
emailRFC822Addr
emailCCmailAddr
emailMicrosoftAddr
emailMAILbusAddr
emailQuickmailAddr
emailAddr08
emailAddr09
emailAddr10
emailAddr11
emailAddr12

If any of these searches finds an existing entry with the same attribute values, the X.500 transaction fails with the status codes listed in Appendix G.

- If emailPRUSERNAME is defined in the RDF, the PROFS username and location pair (emailPRUSERNAME and emailPRLOCATION) are checked for uniqueness. If this search finds an existing directory entry with the same emailPRUSERNAME attribute value, the X.500 transaction fails with a PRUNOTUNIQ status.
- If emailSNUSERNAME is defined in the RDF, the SNADS username and location pair (emailSNUSERNAME and emailSNLOCATION) are checked for uniqueness. If this search finds an existing directory entry with the same emailSNUSERNAME attribute value, the X.500 transaction fails with a SNUNOTUNIQ status.

Note

In MODIFY transactions, a uniqueness check is only performed if one of the above attributes has changed. For example, if none of an entry's X.400 standard attribute values have changed between the *old* and *new* metafile, then an X.400 uniqueness check is not performed.

1.3.7 User-defined Uniqueness Processing

Synchronizer-500 provides the Directory Administrator the ability to specify one X.500 attribute that Synchronizer-500 will check, modify (if requested) and guarantee to be unique (up to a selected length). This occurs during the import or transaction operation to add a directory entry to X.500.

Several fields in the configuration file (xdsu_config.dat) are used to allow the directory administrator to define how the uniqueness processing works. Those fields are:

- UNIQUE_FIELD
- UNIQUE_FIELD_LEN
- MAKE_UNIQUE
- TRUNCATE
- UNIQUE_FIELD_SEPARATOR
- UNIQUE_FIELD_CONTEXT

See Appendix F for definitions of each of these fields.

Note

This user-defined uniqueness processing is performed only on ADD transactions. Once Synchronizer-500 has established a unique attribute value for an entry in X.500, Synchronizer-500 will not perform any MODIFY transactions on that attribute.

User-defined uniqueness processing is performed according to the following algorithm:

1. Get the input data, specified by the configuration file record UNIQUE_FIELD; truncate it, if necessary, to the length specified by the configuration record UNIQUE_FIELD_LEN (minimum length of eight characters); and perform a X.500 uniqueness check on that attribute. If the attribute's value is unique (using UNIQUE_SEARCH_CONTEXT if specified), it can be used. Return a status of NORMAL or KEYFLDTRUNC, if the value was truncated. (All status messages are written to the log file, which is located as specified by the Directory Administrator in the configuration file.)

   If the field is not unique, check the value of the configuration record MAKE_UNIQUE. If set to "YES" proceed to step 2. If set to "NO", the transaction is rejected with a status of USERDEFNOTUNIQ.

2. If the process defined in step 1 does not produce a unique value:

   - If a UNIQUE_FIELD_SEPARATOR string is defined, append it to the existing string from step 1
   - Generate a random five-digit ASCII string,
   - Append the ASCII string to the existing data string from step 1,
   - Check the newly generated string to see if the width of the attribute (UNIQUE_FIELD_LEN) is sufficient to include the 5 digits. If not, truncate the data to accommodate the entire 5 digit number. If, however, TRUNCATE=NUMBER is specified in the configuration file, the 5 digit number will be truncated instead of the data.
   - Check the newly generated string for uniqueness.

   If the value is unique, use this value and return KEYFLDMOD.

   If the value is not unique, this step can be repeated up to 1000 times, until a unique value is generated. If a unique value is still not generated, proceed to step 3.

3. If after 1000 attempts a unique value cannot be generated, something is seriously wrong with the way this value has been defined. Return USERDEFNOTUNIQ status and reject the ADD.

If user-defined uniqueness checking is selected, the returned status codes are logged in the Synchronizer-500 transaction log file. (See Appendix G for a complete list of Synchronizer-500 Transaction Status Codes.) A status of NORMAL means the key was unique as is. A status of KEYFLDTRUNC means that the data was truncated. A status of KEYFLDMOD means that the value had digits added in order to make it unique.

1.3.8 PROFS and SNADS Alias Generation

To communicate with IBM PROFS and SNADS systems via the Message Router/PROFS and Message Router/SNADS gateways, a user must have a PROFS/SNADS alias. This alias consists of the IBM USERID and NODE (e.g., USERID = SMITH, NODE = MRPGATE).

Synchronizer-500 gives the Directory Administrator the option to have the PROFS and/or SNADS aliases automatically generated when X.500 entries are created during the import process (see configuration records GEN_PROFS_ALIAS and GEN_SNADS_ALIAS in Appendix F). The aliases are written to the emailPRUSERNAME or emailSNUSERNAME attributes in the emailInfo auxiliary class. If the import RDF contains values for the emailPRUSERNAME and emailSNUSERNAME, then Synchronizer-500 uses these values and does not generate any aliases.

It is recommended that the node names, emailPRLOCATION and emailSNLOCATION, be supplied by the Directory Administrator in the RDF. These names must agree with the values defined in the Message Router/PROFS or Message Router/SNADS gateway database to ensure a complete address (if these gateways are used in the messaging network).

Note

If PROFS or SNADS alias generation is turned on and the corresponding PROFS or SNADS location is not specified in the RDF, the value of SYNCH_ID (see configuration record SYNCH_ID in Appendix F) is substituted and truncated to eight characters, if longer than that length.

The following algorithm is used to generate PROFS and SNADS aliases:

1. Start with the emailSURNAME attribute. If the value is longer than seven characters, truncate it to seven; truncate the emailGIVENNAME value to one character; and append the character to the emailSURNAME value.

   For example:

   ```
   "JOHN SMITHSON" ===> "SMITHSOJ"
   ```

   Search X.500 (using UNIQUE_SEARCH_CONTEXT if specified) to see if this results in a unique attribute value. If it does, add the alias to X.500. If it does not, proceed to step 2.

2. Truncate the value of emailSURNAME to six characters, and emailGIVENNAME to two characters, and append.

   For example:

   ```
   "JOHN SMITHSON" ===> "SMITHSJO"
   ```

   Search X.500 again to see if the result is a unique attribute value. If it does, add the alias to X.500. If it is not unique, proceed to step 3.

3. Take the first character of emailGIVENNAME and the first character of emailSURNAME and append to it the last six random ASCII digits. If neither emailSURNAME nor emailGIVENNAME values exist, use the string "ID".

For example:

```
"JOHN SMITHSON" ===> "JS132112"
```

An example where neither the emailSURNAME or emailGIVENNAME exist would be:

```
"ID132112"
```

Search X.500 to see if the result is a unique attribute value. If it is, add the alias to X.500. If it is not unique, this step can be repeated until either a unique value is found or 100 attempts have been made. If 100 attempts have been made and a unique emailPRUSERNAME or emailSNUSERNAME cannot be generated, return a status of FAILURE.

1.3.9 Transferring Foreign Directory Files

Synchronizer-500 works solely with files located on the system where it is being run. Synchronizer-500 does not provide for extraction of data from a foreign directory nor for transport of the extracted data from the foreign system to the system running Synchronizer-500.

Transport of the data can be accomplished in a number of ways. File transfer utilities such as the OSI File Transfer and Access Method (FTAM), TCP/IP File Transfer Protocol (FTP), or SNA Data Transfer Facility (DTF) can be used. Other methods, such as sending the data in a mail message or via magnetic tape, can be used. Synchronizer-500 lends itself well to being mail enabled and therefore electronic mail is often the best method of transferring the data files.

The method of extracting data from and inserting data into the foreign directory is determined by the nature of that directory and the availability of utilities and tools to do so. Synchronizer-500 provides the files that contain the initial loads and updates but does not perform the updates on the foreign directory directly.

Chapter 2
How Synchronizer-500 Works

This chapter discusses how Synchronizer-500 works by explaining the pertinent internal Synchronizer-500 concepts and data structures.

The main program of Synchronizer-500 executes the following algorithm:

```
Read and interpret the X.500 Layout and Configuration Files

If this is an IMPORT then ...

   Read and interpret the record description file (RDF)
   Read the input foreign directory file, generate distinguished
     names for each entry, convert the data to
     metafile format and write it to the new metafile.
   Extract existing X.500 entries according to configuration parameters
     and write the data to a flat file called the old metafile.
   Compare the old and new metafiles and generate
     DELETE, MODIFY, and ADD transactions to update X.500.

If this is a TRANSACTION then ...

   Read and interpret the record description file (RDF)
   Read the delete, modify, and add input foreign directory files,
     generate distinguished names for each entry,
     convert the data to metafile format and
     write it to the delete, modify, and add metafiles.
   Read the delete, modify, and add metafiles and generate
     DELETE, MODIFY, and ADD transactions to update X.500.

If this is an EXPORT then...

   Read and interpret the record description file (RDF)
   Extract existing X.500 entries according to configuration parameters
     and write the data to a flat file called the old metafile.
   Read the just-extracted old metafile, convert it to
     the format defined in the RDF, and write it to the
     output foreign directory file.

If this is a CHANGES then...
```

```
Read and interpret the ADD, MODIFY, and DELETE
  record description files (RDF)
Extract existing X.500 data according to configuration parameters
  and write the data to a flat file called the new metafile.
Compare the old and new metafiles and write and convert any changes,
  using the formats defined by the ADD, MODIFY, and DELETE RDF's,
  to the ADD, MODIFY, and DELETE output foreign directory files.
```

The subsequent sections of this chapter further describe how this algorithm works for each operation type. Chapter 3 provides more information on the configuration and RDF files.

2.1 Import Processing

This section describes Synchronizer-500 import processing and the use of metafiles. The operations described are:

- converting the input foreign directory file to a *new* metafile
- extracting X.500 entries into the *old* metafile
- comparing the *old* and *new* metafiles

2.1.1 Converting the Input Foreign Directory File to a New Metafile

A *new* metafile is a metafile into which all data that is *new* to X.500 (i.e., coming from a foreign directory file) is placed. All input foreign directory file records are read, converted to metafile format, and written to the *new* metafile. A foreign directory file record is processed and converted to metafile format as specified in the record description file (RDF).

After processing each RDF record for an input foreign directory file record, a DISTINGUISHED NAME string is generated. This string is made up of the RDNs specified in the configuration file by the field DN_SEQUENCE.

The *new* metafile record is compressed (blanks reduced to a sentinel byte and a count byte) and then written to disk.

After processing the entire foreign directory file and converting it to metafile format, the *new* metafile is sorted on the DISTINGUISHED NAME into ascending order to facilitate comparison with the *old* metafile.

2.1.2 Extracting X.500 Entries into the Old Metafile

An *old* metafile is a metafile into which *existing* X.500 entries have been extracted. The entries to be extracted from X.500 are identified in the import configuration file by the following keywords:

- GID_FIELD and GID_MATCH - required keywords which identify the directory Group ID being imported. Synchronizer-500 will only extract entries where GID_FIELD=GID_MATCH.
- SYNCH_ID_FIELD and SYNCH_ID - required keywords which identify which entries Synchronizer-500 can extract, add, modify, and delete. These values should be the same for all Synchronizer-500 instances.

- SEARCH_CONTEXT - optional keyword which can limit the search to an entry base in the DIT (e.g. /C=US/O=DIGITAL). A value for this keyword should be supplied if Synchronizer-500 is only working with a subset of entries in the DIT. (Especially if not specifying a value will cause X.500 to chain to another DSA to satisfy the search request).

In addition, Synchronizer-500 identifies a Distinguished Value (DVAL) field in the X.500 Layout field in which all entries to be extracted must have a value. By default, this field is the *commonName* attribute.

The file name used for the metafile can be internally generated by Synchronizer-500 or can be specified in the RDF record OLD_METAFILE. The *old* metafile is deleted at the end of the Synchronizer-500 run. Metafile records are defined according to the X.500 layout file. Because metafile records have large amounts of blanks, they are compressed before they are written to disk to preserve disk space.

After extracting all specified X.500 entries and converting the data to metafile format, the *old* metafile is sorted on the DISTINGUISHED NAME into ascending order to facilitate comparison with the *new* metafile.

2.1.3 Comparing the Old and New Metafile

At this point, Synchronizer-500 has an *old* metafile, which contains the extracted existing X.500 entries, and a *new* metafile, which is the conversion of the foreign directory file data. Both have been sorted on their DISTINGUISHED NAME strings. Synchronizer-500 compares each old and new record and uses a sort/merge algorithm to generate X.500 transactions.

The metafile comparison is done solely on DISTINGUISHED NAME, according to the following rules:

1. If a record exists in the *old* metafile but not in the *new* metafile, a DELETE transaction to X.500 is performed on the *old* metafile record.
2. If a record exists in both the *new* metafile and the *old* metafile but some data has changed, a MODIFY transaction to X.500 with the new metafile data is performed.
3. If a record exists in the *new* metafile but not in the *old* metafile, an ADD transaction to X.500 is performed on the *new* metafile record.

For example, given the following metafiles:

```
 New Metafile                Old Metafile
--------------------      ---------------------
ADAMS (different data)     ADAMS (different data)
BAKER (same data)          BAKER (same data)
CALHOUN                    COOLIDGE
DAVIS                      FORD (same data)
EDWARDS                    GEORGE
FORD (same data)
```

The results of the Synchronizer-500 metafile comparison would generate:

```
DELETE COOLIDGE
DELETE GEORGE
MODIFY ADAMS
ADD CALHOUN
ADD DAVIS
ADD EDWARDS
```

No transactions would be generated for BAKER or FORD because the data is the same in both files.

2.2 Transaction Processing

This section describes Synchronizer-500 transaction processing and the use of metafiles. The operations described are:

- converting the input foreign directory files into *delete*, *modify*, and *add* metafiles
- generating X.500 transactions from the delete, modify, and add metafiles

2.2.1 Converting the Input Foreign Directory Files to Metafiles

In transaction processing, three input files (delete, modify, and add), containing changes information from a foreign directory, are converted to three metafiles (delete, modify, and add). All input foreign directory file records are read, converted to metafile format, and written to the appropriate metafile as described in import processing (see Section 2.1.1).

2.2.2 Generating X.500 transactions from Metafiles

At this point, Synchronizer-500 generates X.500 DELETE, MODIFY, and ADD transactions (in that order) from the *delete*, *modify*, and *add* metafiles.

2.3 Export Processing

This section describes Synchronizer-500 export processing and the use of metafiles. The operations described are:

- extracting X.500 entries into the *old* metafile
- converting the *old* metafile to the output foreign directory file

2.3.1 Extracting X.500 Entries into the Old Metafile

An *old* metafile is a metafile into which *existing* X.500 entries have been extracted. During Export processing, the entries to be extracted from X.500 are identified in the export configuration file by the following keywords:

- GID_FIELD and GID_MATCH - required keywords which identify the directory Group ID(s) being exported. Synchronizer-500 will only extract entries where GID_FIELD=GID_MATCH. To extract more than one group, set GID_MATCH to a list of Group IDs separated by commas (e.g. GID_MATCH=GROUP1,GROUP2). To extract all Group IDs, set GID_MATCH=*.

- GID_EXCLUDE - optional keyword which identifies the directory Group ID(s) to exclude from the extract. GID_EXCLUDE can only be used when GID_MATCH=*. To exclude more than one group, set GID_EXCLUDE to a list of Group IDs separated by commas (e.g. GID_EXCLUDE=GROUP1,GROUP2).
- SYNCH_ID_FIELD and SYNCH_ID - optional keywords which identify which entries Synchronizer-500 can extract. If specified, Synchronizer-500 will only extract entries where SYNCH_ID_FIELD=SYNCH_ID.
- SEARCH_CONTEXT - optional keyword which can limit the search to an entry base in the DIT (e.g. /C=US/O=DIGITAL). A value for this keyword should be supplied if only a subset of entries in the DIT needs to extracted.

The following configuration file keywords, which represent X.400 standard attributes, can be used to further qualify the subset of entries to be exported from X.500:

- COUNTRY_MATCH
- ADMD_MATCH
- PRMD_MATCH
- ORGNAME_MATCH
- ORGUNIT_MATCH
- ORGUNIT2_MATCH
- ORGUNIT3_MATCH
- ORGUNIT4_MATCH

For example, ADMD_MATCH=ABC will extract all entries in X.500 (that already match the Group ID, Synch ID, and Search Context) with an attribute of emailADMDNAME that has a value of ABC.

Another method of further qualifying the subset of entries to be exported from X.500, is to select a group of entries based on a value in a specific attribute using the configuration keywords:

- GET_FIELD
- GET_VALUE

GET_FIELD specifies any valid attribute listed in the X.500 layout file and GET_VALUE specifies a value to match on in X.500 (the wild card, asterisk * character, is acceptable). For example, GET_FIELD=emailSNLOCATION and GET_VALUE=NACGWY1 will extract all entries in X.500 (that already match the Group ID, Synch ID, Search Context, and any other *_MATCH keywords) with an attribute of emailSNLOCATION that has a value of NACGWY1. If GET_VALUE=* is specified, all entries which have a value defined for emailSNLOCATION are extracted.

In addition, Synchronizer-500 identifies a Distinguished Value (DVAL) field in the X.500 Layout field in which all entries to be extracted must have a value. By default, this field is the *commonName* attribute.

Once all entries in X.500 in the specified SEARCH_CONTEXT with the various match filters are extracted, they are written to the *old* metafile in metafile format.

The file name used for the metafile can be internally generated by Synchronizer-500 or can be specified in the RDF record, OUTPUT_OLD_METAFILE. *Old* metafiles with internally generated names are deleted at the end of the export operation. *Old* metafiles named in the RDF, are retained for future use. (e.g. for the input metafile in Changes mode.)

After extracting all specified X.500 entries and converting them to metafile format, the *old* metafile is sorted on DISTINGUISHED NAME into ascending order.

2.3.2 Converting the Old Metafile to the Foreign Directory File

All *old* metafile records are read, converted to foreign directory file format as specified in the RDF, and written to the output foreign directory file. Output record fields are built in the order in which they are specified in the RDF.

For additional flexibility, the RDF syntax of Synchronizer-500 for export files allows the use of special characters to specify formatting and conditional operations.

If the asterisk * character is specified in the RDF, it denotes: ''find the current end of the output buffer and start writing data there.'' The current end of the output buffer is defined as the last non-null character in the output foreign directory file buffer (the output foreign directory file buffer is initialized as all nulls). (See Section 3.3.3.4 for more information regarding the asterisk * character.)

If the pound sign # character is specified in the RDF, it denotes: ''trim white space (blanks and tabs) from the input data value (X.500_ATTRIBUTE or ''string'') and use the length of that string to store the input data in the output foreign directory file record buffer.'' (See Section 3.3.3.4 for more information regarding the pound sign # character.)

If a conditional statement is found in the RDF the existence of data in the specified X.500 attribute is checked. The question mark ? character denotes taking action based on the existence of data. The tilde ~ character denotes taking action based on the non-existence of data. Data is placed in the output record buffer if:

1. the RDF statement is "if data exists..." (denoted by the ? character) AND data exists in the X.500 attribute
2. the RDF statement is "if data does not exist..." (denoted by the ~ character) AND data does not exist in the X.500 attribute

(See Section 3.3.3.3 for more information regarding RDF conditional statements.)

After the above conversion procedures are completed, the built record is written to disk.

2.4 Changes Processing

This section describes Synchronizer-500 changes processing and the use of metafiles.

Changes operations use a mix of the processing procedures performed by both import and export operations:

- The export operation ''Extracting X.500 entries into the Old Metafile'' (see Section 2.3.1) is used to extract data to the New Metafile.
- The import operation ''Comparing the Old and New Metafile'' (see Section 2.1.3) is used to determine the changes that have occurred to X.500.

- The export operation ‘‘Converting the Old Metafile to the Foreign Directory File’’ (see Section 2.3.2) is used to convert the changed metafile records to output foreign directory file format definitions.

2.4.1 Extracting X.500 Entries into the New Metafile

In Changes operations, the meanings of the *old* and the *new* metafile from those used in import and export operations are reversed. In changes operations, the *old* metafile is the user-specified, previously-existing file and the X.500 extract file is the *new* metafile.

The new metafile in changes operations is extracted identically to the way the old metafile is extracted in export operations.

2.4.2 Changes Operations Comparison of Old and New Metafiles

In Changes operations, the old metafile (user-specified, previously-existing) and the new metafile (just extracted) records are compared, as in import mode, and Synchronizer-500 uses a sort/merge algorithm to generate transactions. The comparison is done solely on the DISTINGUISHED NAME string.

Instead of being X.500 transactions, these transactions are converted and written to three separate output files (ADD, MODIFY, and DELETE).

The Directory Administrator has the option to implement MODIFY output files as DELETE and ADD output files by omitting the value for the configuration record MOD_RDF_FILE in the changes configuration file.

2.4.3 Converting the ADD, MODIFY, and DELETE Metafile Records

ADD, MODIFY, and DELETE Records from the old and new metafile comparison are converted to foreign directory file format as specified in the RDF(s), and written to their respective output foreign directory files. Each of these output files has its own RDF that allows for a different format for each file. (See Section 2.3.2 for more information how the conversion occurs.)

Chapter 3
Synchronizer-500 Set up and Start up

A prerequisite to the use of Synchronizer-500 is the planning and configuration of the X.500 Directory Information Tree (DIT) and the Directory System Agent (DSA). If a DSA has not been configured, Synchronizer-500 will be unable to create entries in the X.500 DIT. The DSA configuration tasks in Chapter 7 of the *DEC X.500 Directory Service Management Guide* must be completed before using Synchronizer-500. Please consult the *DEC X.500 Directory Service Management Guide* for more information.

After Synchronizer-500 is installed, there are three steps that need to be completed before Synchronizer-500 can be used.

- Customizing the DEC X.500 Directory Service Schema
- Customizing the Synchronizer-500 X.500 Layout File
- Setting up Synchronizer-500 to synchronize with a foreign directory

3.1 Customizing the DEC X.500 Directory Service Schema

After Synchronizer-500 is installed, the DEC X.500 Directory Service schema must be customized and recompiled if it is planned to include the Synchronizer-500 email schema file (email.sc). The email.sc file defines the emailInfo auxiliary class. This file is placed in the Synchronizer-500 root directory during installation.

If any attributes from the emailInfo class are to be used include the email schema file into the DEC X.500 schema file, following the instructions below:

1. Copy the email.sc file from the Synchronizer-500 root directory to the X.500 root directory (DXD$DIRECTORY on OpenVMS or /var/dxd on UNIX).
2. Set your current directory to the X.500 root directory (DXD$DIRECTORY on OpenVMS or /var/dxd on UNIX).
3. Edit the DEC X.500 schema file (DXD$SCHEMA.SC on OpenVMS or dxd_schema.sc on UNIX) and insert the following line at the end of the file: INCLUDE "email.sc"
4. Disable and Delete the DSA if already created.

5. Compile the schema:

```
OpenVMS:
$ RUN SYS$SYSTEM:DXD$SCHEMA_COMPILER.EXE
UNIX:
/usr/etc/dxd_sc
```

6. Create and Enable the DSA.

For more information, please see Chapter 5 of the *DEC X.500 Directory Service Management Guide* for instructions on how to customize the schema.

The X.500 Directory Service schema file only needs to be customized and compiled once. If this does not happen, none of the attributes prefaced by 'email' in Appendix D are available for use by Synchronizer-500.

3.2 Customizing the Synchronizer-500 X.500 Layout File

If any other changes were made to the DEC X.500 schema (other than adding the email.sc file), the Synchronizer-500 X.500 Layout File must be updated as well. As detailed in Appendix H, the layout file (xdsu_layout.dat) describes all the X.500 object classes and attributes which Synchronizer-500 can have access to. If any attributes which are not included in the default layout file (see Appendix D) are to be used then the layout file must be updated.

The layout file must also be updated if a standard attribute needs to have a different definition than the default values. This would include the cases where a multi-valued attribute needs more values than defined in column 7 or if the syntax for the attribute in column 5 needs to be changed from printableString to teletexString to allow T.61 characters to be stored in the attribute. See Appendix H for more detail.

3.3 Setting up and Running a Synchronization Operation

There are a number of steps involved in setting up and running the synchronization of a foreign directory with X.500. These steps are:

1. Create the Synchronizer-500 subdirectories for each instance. (Typically every different Synch Type—Import, Transaction, Export, Changes—for each different Group ID represents a separate instance.)
2. Determine how to transport data files between the foreign system and the Synchronizer-500 system.
3. Decide what information should be exchanged between the two directories.
4. Set up the configuration files.
5. Set up the record description files (RDFs).

The following list details the activities in the synchronization operations after the preparation has been completed:

For *import* operations:

- extract a copy of the foreign directory to a file on the foreign system

- move the foreign directory file onto the Synchronizer-500 system
- run the import process to update the foreign directory file information into X.500

For *transaction* operations:

- extract a copy of the foreign directory changes files on the foreign system
- move the foreign directory files onto the Synchronizer-500 system
- run the transaction process to update the foreign directory file information into X.500

For *export* operations:

- run the export process to convert the X.500 entries to the foreign directory format and place the converted entries in an export file
- move the X.500 export file over to the foreign system
- merge the X.500 export file into the foreign directory

For *changes* operations:

- run the changes process to extract the X.500 entries that have changed since the last changes run and place them in export files in the foreign directory format
- move the X.500 changes export files over to the foreign system
- merge the X.500 changes export files into the foreign directory

Note

Synchronizer-500 does not extract data from a foreign directory, transport it to the system where Synchronizer-500 is installed, or merge the exported X.500 data back into the foreign directory. These steps are pre- and post-Synchronizer-500 activities that must be accomplished to complete the directory update process.

The remainder of this chapter describes how to set up the Synchronizer-500 instance subdirectory, configuration files and RDFs. Determining the transport method and what information to exchange is discussed in the example in Chapter 5.

3.3.1 Creating the Synchronizer-500 Instance Subdirectory

In order to keep files for a single Synchronizer-500 run organized, it is recommended that a subdirectory be created under the Synchronizer-500 root directory. When Synchronizer-500 executes, it looks for its configuration file in the directory from which it is being executed and the X.500 layout file and valid group id file one level up from the current directory.

Note

If a subdirectory to the Synchronizer-500 root directory is not created which contains the configuration file, the location of the X.500 layout file and valid_id file must be specified as parameters to Synchronizer-500 on the command line. See Section 3.3.4 for more information on specifying parameters when starting Synchronizer-500.

Synchronizer-500 comes with a procedure which creates the subdirectory and copies the appropriate template configuration file into the subdirectory. To execute this procedure:

1. Set your current directory to the Synchronizer-500 root directory.

2. Execute the procedure xdsu_config_instance:

```
OpenVMS:
$ @XDSU_CONFIG_INSTANCE

UNIX:
#xdsu_config_instance
```

3. The procedure prompts:

```
Enter XDSU instance directory name:
```

   Enter the name of the directory you wish to create. For example, to create a subdirectory to be used for importing Microsoft Mail users into X.500, you might enter MSM_IMPORT.

4. The procedure then prompts:

```
Enter I for Import, E for Export, C for Changes, T for Transaction:
```

   Enter **I** if this is an import run, **E** if this is an export run, **C** if this is a changes run, or **T** if this is a transaction run.

5. The procedure then creates the directory and copies the appropriate template file to the newly created directory.

6. Set your current directory to the newly created directory and proceed with setting up the configuration file and RDFs as described below.

3.3.2 Setting up Synchronizer-500 Configuration Files

The operation of Synchronizer-500 is controlled by entries in the configuration file. These entries determine the behavior and characteristics of Synchronizer-500. As indicated in the opening section to this chapter, it is a prerequisite to using Synchronizer-500 that the DIT has been planned and implemented. This is because the structure of the DIT is indicated when setting up the configuration files. For example, the import configuration file specifies the RDNs used in generating the distinguished name for an entry, and what classes the entries being imported belong to, i.e., Person, organizationalPerson, emailInfo.

There are four types of configuration files–*import*, *transaction*, *export*, and *changes*. Each file type has its own list of keywords.

Entries in the configuration files consist of keyword-value pairs (for example, SYNCH_TYPE=IMPORT). Some keywords must have values, whereas values are optional in others (see Section 3.3.2.2). Optional keywords may be removed, commented out, or left as specified with no value present. Keywords may be in any order, with the lone exception that SYNCH_TYPE must be the first keyword.

Comments can be inserted into a configuration file. Comments on a line begin with the comment character ("!"). Any characters after the comment character are ignored. Synchronizer-500 also ignores blank lines.

Templates for each configuration file type are included in the Synchronizer-500 root directory. When the xdsu_config_instance procedure is executed (see Section 3.3.1) the appropriate configuration template file is copied to the subdirectory for the instance being configured.

Use a text editor to tailor the template file, as needed. (Copies of each type's configuration file template are provided in Appendix F along with detailed descriptions of each keyword.)

3.3.2.1 Location of Configuration Files

Synchronizer-500 looks for the configuration file, xdsu_config.dat, in the current directory where Synchronizer-500 is executed *unless* it is specified as the first parameter on the command line (please refer to Section 3.3.4).

For ease of use and management it is recommended that the configuration files be located in an instance specified subdirectory under the Synchronizer-500 root directory.

3.3.2.2 Configuration File Tables

The following section contains four tables for each configuration type.

The conventions used in these tables are:

- Values allowed for a keyword are stated in parentheses in the column under Meaning.
- All file names are stated as valid file specifications.

See Appendix F for more information on each configuration file record.

Table 3–1: Import Configuration File

Keyword	Meaning	Value Mandatory?
synch_type	Type of synch (IMPORT)	YES
synch_label	Label for Synchronizer-500 output - defaults to 'XDSU'	NO
bind_dsa	Presentation address of DSA to bind to - *Reserved for Future Use*	NO
rdf_file	File name of RDF to use	YES
input_for_dir_file	File name of input foreign directory file	YES
nrec_subscr	Number of records in the input file per subscriber entry	YES
log_file	File name of Synchronizer-500 transaction log file - defaults to 'xdsu_log.log'	NO
end_run_file	File name of procedure to call when import is done	NO
old_metafile	File name of old metafile	NO
new_metafile	File name of new metafile	NO
initial_load	Initial load (YES/NO)	YES
notify	Electronic mail USERID to receive status messages	NO
verbose	Formatted listing of import ADDs/MODIFYs (YES/NO)	NO
encode_delim	Character used to delimit the encoding of characters not available in the T.61 character set	NO
max_transactions	Number of transactions at which to call max_ trans_file	NO
max_fail_transactions	Number of failures at which Synchronizer-500 should terminate	NO
max_trans_file	File name of procedure to call when max_ transactions has been reached	NO
dn_sequence	Order of and RDNs used to define the X.500 Distinguished Name	YES
create_root	How far up the DIT Synchronizer-500 should create superior entries	NO

Table 3–1 (Cont.): Import Configuration File

Keyword	Meaning	Value Mandatory?
search_context	Distinguished Name to base search on during extract	NO
synch_object	X.500 Classes for the entries	YES
synch_id	Name to identify this particular Synchronizer-500 - defaults to 'XDSU'	NO
synch_id_field	X.500 attribute to put SYNCH_ID in	YES
gid_field	X.500 Attribute in which to put the Group ID	YES
gid_match	Group ID character string to use	YES
gen_profs_alias	Generate PROFS alias (YES/NO)	NO
gen_snads_alias	Generate SNADS alias (YES/NO)	NO
admd_space	Generate a "BLANK" as the emailADMDNAME value (YES/NO)	NO
unique_field	X.500 attribute to check/make unique	NO
unique_field_len	Maximum length of unique_field	NO
make_unique	Modify X.500 attribute to make unique (YES/NO) NO	
truncate	Truncate number on unique_field generation (NUMBER)	NO
unique_field_separator	string to separate unique_field value and random suffix if needed to be made unique	NO
unique_search_context	Distinguished Name to base search on during unique field processing	NO

See Section F.1 for more information on import configuration file keywords. See Example 5–3 for a sample import configuration file.

Table 3–2: Transaction Configuration File

Keyword	Meaning	Value Mandatory?
synch_type	Type of synch (TRANSACTION)	YES
synch_label	Label for Synchronizer-500 output - defaults to 'XDSU'	NO
bind_dsa	Presentation address of DSA to bind to - *Reserved for Future Use*	NO
rdf_file	File name of RDF to use	YES
input_add_file	File name of input add foreign directory file	NO
input_mod_file	File name of input modify foreign directory file	NO
input_del_file	File name of input delete foreign directory file	NO
nrec_subscr	Number of records in the input file per subscriber entry	YES
log_file	File name of Synchronizer-500 transaction log file - defaults to 'xdsu_log.log'	NO
end_run_file	File name of procedure to call when transaction is done	NO
add_metafile	File name of add metafile	NO
mod_metafile	File name of modify metafile	NO
del_metafile	File name of delete metafile	NO
initial_load	Initial load (YES/NO)	YES
notify	Electronic mail USERID to receive status messages	NO
verbose	Formatted listing of transaction ADDs/MODIFYs (YES/NO)	NO
encode_delim	Character used to delimit the encoding of characters not available in the T.61 character set	NO
max_transactions	Number of transactions at which to call max_ trans_file	NO
max_fail_transactions	Number of failures at which Synchronizer-500 should terminate	NO
max_trans_file	File name of procedure to call when max_ transactions has been reached	NO
dn_sequence	Order of and RDNs used to define the X.500 Distinguished Name	YES

Table 3–2 (Cont.): Transaction Configuration File

Keyword	Meaning	Value Mandatory?
create_root	How far up the DIT Synchronizer-500 should create superior entries	NO
synch_object	X.500 Classes for the entries	YES
synch_id	Name to identify this particular Synchronizer-500 - defaults to 'XDSU'	NO
synch_id_field	X.500 attribute to put SYNCH_ID in	YES
gid_field	X.500 Attribute in which to put the Group ID	YES
gid_match	Group ID character string to use	YES
gen_profs_alias	Generate PROFS alias (YES/NO)	NO
gen_snads_alias	Generate SNADS alias (YES/NO)	NO
admd_space	Generate a "BLANK" as the emailADMDNAME value (YES/NO)	NO
unique_field	X.500 attribute to check/make unique	NO
unique_field_len	Maximum length of unique_field	NO
make_unique	Modify X.500 attribute to make unique (YES/NO) NO	
truncate	Truncate number on unique_field generation (NUMBER)	NO
unique_field_separator	string to separate unique_field value and random suffix if needed to be made unique	NO
unique_search_context	Distinguished Name to base search on during unique field processing	NO

See Section F.2 for more information on transaction configuration file keywords.

All Transaction keywords are the same as Import keywords except:

- synch_type = TRANSACTION
- replace INPUT_FOR_DIR_FILE with INPUT_ADD_FILE, INPUT_DEL_FILE, INPUT_ MOD_FILE (All three are optional - i.e. If you're only doing adds then you only need to specify the INPUT_ADD_FILE)
- replace OLD_METAFILE and NEW_METAFILE with ADD_METAFILE, DEL_METAFILE, MOD_METAFILE (All three are optional and are only used if you want the temporary metafiles to be created somewhere other than the default directory)
- remove SEARCH_CONTEXT - not used in TRANSACTION mode

Table 3–3: Export Configuration File

Keyword	Meaning	Value Mandatory?
synch_type	Type of synch (EXPORT)	YES
synch_label	Label for Synchronizer-500 output - defaults to 'XDSU'	NO
bind_dsa	Presentation address of DSA to bind to - *Reserved for Future Use*	NO
rdf_file	File name of RDF to use	YES
output_for_dir_file	File name of output foreign directory file	YES
nrec_subscr	Number of records in the input file per subscriber entry	YES
end_run_file	File name of procedure to call when the export is done	NO
output_old_metafile	File name to save old metafile	NO
input_old_metafile	File name of old metafile to use	NO
notify	Electronic mail USERID to receive status messages	NO
verbose	Include more detailed output of Synchronizer-500 activity (YES/NO)	NO
encode_delim	Character used to delimit the encoding of characters not available in the T.61 character set	NO
search_context	Distinguished Name to base search on	NO
synch_id	Name to identify this particular Synchronizer-500 - defaults to 'XDSU'	NO
synch_id_field	X.500 attribute which contains the SYNCH_ID	YES
gid_field	X.500 attribute which contains the Group ID	YES
gid_match	Name of Group ID to extract or *	YES
gid_exclude	Name of Group ID to exclude from the extract	NO
country_match	X.400 country to extract	NO
admd_match	X.400 Admin Domain to extract	NO
prmd_match	X.400 Private Domain to extract	NO
orgname_match	X.400 Organization to extract	NO
orgunit_match	X.400 Orgunit #1 to extract	NO

Table 3–3 (Cont.): Export Configuration File

Keyword	Meaning	Value Mandatory?
orgunit2_match	X.400 Orgunit #2 to extract	NO
orgunit3_match	X.400 Orgunit #3 to extract	NO
orgunit4_match	X.400 Orgunit #4 to extract	NO
get_field	X.500 attribute to extract	NO
get_value	Particular value to extract	NO

See Section F.3 for more information on export configuration file keywords. See Example 5–10 for a sample export configuration file.

Table 3–4: Changes Configuration File

Keyword	Meaning	Value Mandatory?
synch_type	Type of synch (CHANGES)	YES
synch_label	Label for Synchronizer-500 output - defaults to 'XDSU'	NO
bind_dsa	Presentation address of DSA to bind to - *Reserved for Future Use*	NO
add_rdf_file	File name of add RDF file	YES
mod_rdf_file	File name of modify RDF file	NO
del_rdf_file	File name of delete RDF file	YES
output_add_file	File name to which to write the ADD data	YES
output_mod_file	File name to which to write the MODIFY data	NO
output_del_file	File name to which to write the DELETE data	YES
nrec_subscr_add	Number of records per subscriber entry in the output_add_file	YES
nrec_subscr_mod	Number of records per subscriber entry in the output_mod_file	NO
nrec_subscr_del	Number of records per subscriber entry in the output_del_file	YES
end_run_file	File name of procedure to call when the changes run is done	NO
input_old_metafile	File name of old metafile to use	YES
output_new_metafile	File name of metafile to put extracted X.500 data	NO
input_new_metafile	File name of input new metafile	NO
notify	Electronic mail USERID to receive status messages	NO
verbose	Include more detailed output of Synchronizer-500 activity (YES/NO)	NO
encode_delim	Character used to delimit the encoding of characters not available in the T.61 character set	NO
search_context	Distinguished Name to base search on	NO
synch_id	Name to identify this particular Synchronizer-500 - defaults to 'XDSU'	NO
synch_id_field	X.500 attribute which contains the SYNCH_ID	YES

Table 3–4 (Cont.): Changes Configuration File

Keyword	Meaning	Value Mandatory?
gid_field	X.500 attribute which contains the Group ID	YES
gid_match	Name of Group ID to extract or *	YES
gid_exclude	Name of Group ID to exclude from the extract	NO
country_match	X.400 country to extract	NO
admd_match	X.400 Admin Domain to extract	NO
prmd_match	X.400 Private Domain to extract	NO
orgname_match	X.400 Organization to extract	NO
orgunit_match	X.400 Orgunit #1 to extract	NO
orgunit2_match	X.400 Orgunit #2 to extract	NO
orgunit3_match	X.400 Orgunit #3 to extract	NO
orgunit4_match	X.400 Orgunit #4 to extract	NO
get_field	X.500 attribute to extract	NO
get_value	Particular value to extract	NO

See Section F.4 for more information on changes configuration file keywords. See Example 5–13 for a sample changes configuration file.

All Changes keywords are the same as Export keywords except:

- synch_type = CHANGES
- replace RDF_FILE with ADD_RDF_FILE, DEL_RDF_FILE, and (optionally) MOD_RDF_FILE
- replace OUTPUT_FOR_DIR_FILE with OUTPUT_ADD_FILE, OUTPUT_DEL_FILE, and (optionally) OUTPUT_MOD_FILE
- replace NREC_SUBSCR with NREC_SUBSCR_ADD, NREC_SUBSCR_DEL, and (optionally) NREC_SUBSCR_MOD
- add INPUT_OLD_METAFILE
- replace OUTPUT_OLD_METAFILE with OUTPUT_NEW_METAFILE

3.3.3 Setting up Record Description Files

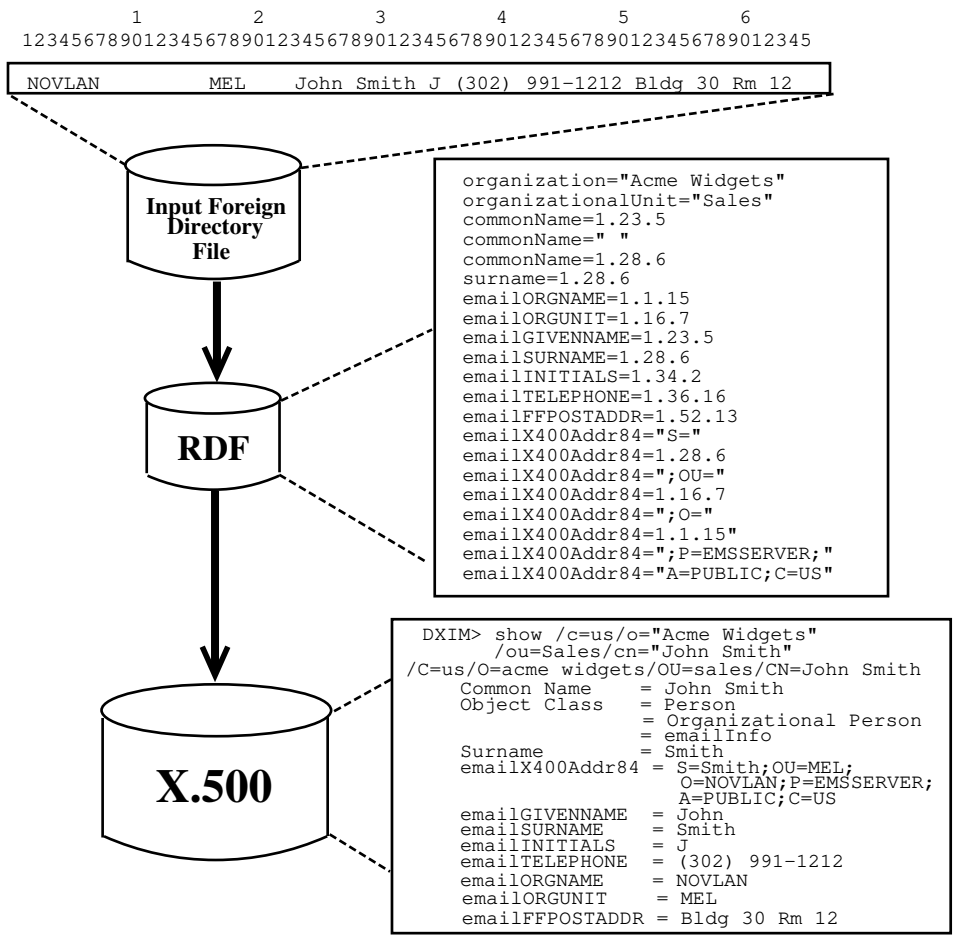
A record description file (RDF) is a sequential file that describes the mapping between the X.500 attributes of the classes for the entries you are synchronizing against and the foreign directory file data. An RDF contains a series of records that describe the information in the foreign directory file and the corresponding X.500 attributes. See Appendix D for the default X.500 attributes and their definitions allowed by Synchronizer-500.

Note

It is the responsibility of the Directory Administrator to include the mandatory attributes in the RDF of the class being used for synchronizing the entries. Please see Appendix B of the *DEC X.500 Directory Service Management Guide* for more information. (For example, *surname* is a mandatory attribute of the *Person* object class.)

During the import and transaction input processes, each line in the RDF maps a piece of information in the foreign directory file or a constant string to an attribute in the X.500 directory, as illustrated in Figure 3–1.

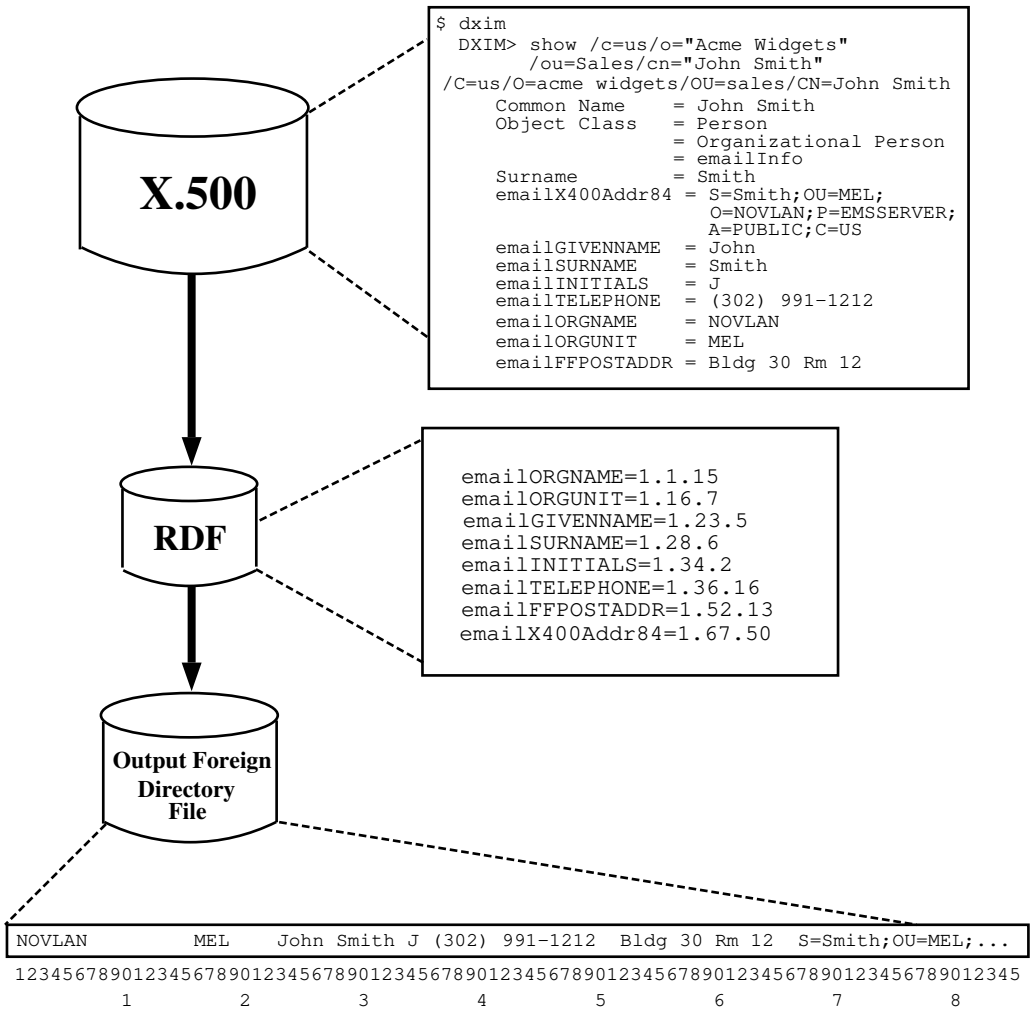
Figure 3–1: Mapping Import and Transaction Foreign Directory File Records to X.500 Attributes



In the example shown in Figure 3–1, each RDF record describes how a portion of the input foreign directory file or a constant string is mapped to an X.500 RDN or attribute. For example, the RDF record "emailORGUNIT=1.16.7" instructs Synchronizer-500 to go out to the input record, get 7 characters of data starting at column 16 ("MEL"), and store this character-string in the X.500 attribute, emailORGUNIT.

During the export and changes output processes, each line in the RDF maps a X.500 attribute to the output file in the format expected by the foreign directory, as illustrated in Figure 3–2.

Figure 3–2: Mapping X.500 Attributes to Export and Changes Foreign Directory File Records



In the example shown in Figure 3–2, each RDF record describes how an X.500 attribute is mapped to a location in the output foreign directory file. For example, the RDF record "emailORGUNIT=1.16.7" will go out to the X.500 attribute, emailORGUNIT, get the first 7 characters of data ("MEL"), and put this character-string in the output foreign directory file record, starting at column 16.

3.3.3.1 Import and Transaction Concatenated RDF Statements

RDF syntax permits concatenation of attributes for import processing by repeating keywords. For example:

```
emailLOCATION=1.40.15
emailLOCATION=1.60.3
```

If the first extract (starting at column 40, width = 15) results in "Pittsburgh#####" and the second extract (starting at column 60, width = 3) results in "PA.", then "Pittsburgh#####PA." is written to the X.500 attribute, emailLOCATION. The # character represents blanks. Note, however, that the in the X.500 schema, this attribute is "Normalized" so that extra blanks are reduced to a single space: "Pittsburgh#PA." To keep extra blanks, the X.500 schema must be updated and the "NONORM" flag must be added to the attribute in the X.500 Layout File (see Section H.4).

3.3.3.2 RDF Syntax

The basic formats of records in the RDF file are shown in Table 3–5.

Table 3–5: RDF Record Formats

Record Format	Use	Example
keyword=record_number.starting_column.width	Import/Export	emailSURNAME=1.1.25
keyword="import-string"	Import only	emailORGNAME="SALES"
"export-string"=record_number.starting_col.width	Export only	"SALES"=1.20.8
?record_number.starting_column.width:action	Import	?1.10.15:emailSURNAME=1.10.15
~record_number.starting_column.width:action	Import	~1.10.15:emailSURNAME="JDoe"
?keyword:action	Export	?emailNAME:emailNAME=3.20.#
~keyword:action	Export	~emailNAME:"*none*"=3.20.#

The meaning of the terms used in the Record Format column are:

- **KEYWORD** is the X.500 attribute to be loaded with data from the foreign directory during import or to be written to the foreign file during export. (See Appendix D for a list of default keywords.) For multi-value keywords, specify name_2 for the second value, name_3 for the third value, etc.
- **RECORD_NUMBER** is a number between 1 and 512. Some directory files contain more than one line of information for each subscriber entry (i.e., the data for an individual may be placed on successive lines rather than on the same line). RECORD_NUMBER allows specification of the record number which contains the data to be imported into the X.500 attribute (for import/transaction operations) or the record number that data is to be written to in the output Foreign Directory File (for export/changes operations). A maximum of 512 records per subscriber is supported.

- **STARTING_COLUMN** is the column number in the input foreign directory file where the attribute starts (for import/transaction operations) or the starting column number in the output foreign directory file record where data is to be written (for export/changes operations).
- **WIDTH** is the length of the attribute to be read (import/transaction operations) or written (export/changes operations). For example, if a WIDTH of 20 is specified for an import process, then Synchronizer-500 begins reading the location specified by the STARTING_ COLUMN parameter and writes the next 20 characters to the indicated attribute in X.500. In an export operation, the first 20 characters in the specified X.500 attribute are written to the export file, beginning at the spot indicated by STARTING_COLUMN.
- **"IMPORT_STRING"** allows import of a constant string of information into a X.500 attribute. For example, emailPCOUNTRY="US" adds the string US as the emailPCOUNTRY attribute value in X.500 during the import process.
- **"EXPORT_STRING"** allows a constant string of information that does not exist in X.500 to be written to the export file. "US"=1.10.2 adds the string US to the output foreign directory file in record 1, columns 10-11.
- The question mark character **?** denotes taking action based on the existence of data. (See Section 3.3.3.3.)
- The tilde character **~** denotes taking action based on the non-existence of data. (See Section 3.3.3.3.)
- **ACTION** is any valid RDF statement to be executed as a result of an RDF conditional statement being true. (See Section 3.3.3.3.)

3.3.3.3 RDF Conditional Statements

For additional flexibility in defining the action that RDF statements can take, the RDF syntax supports conditional RDF statements. These statements have an **IF condition THEN action** structure that allows Synchronizer-500 to take action based on the existence or non-existence of data. (See Table 3–5 for the format of RDF conditional statements.)

The question mark ? character denotes taking action based on the existence of data. The tilde ~ character denotes taking action based on the non-existence of data. The colon *:* character delimits the condition from the action.

3.3.3.3.1 Import and Transaction RDF Conditional Statement Syntax

As seen in Table 3–5, the syntax for import conditional statements is:

```
?record_number.starting_column.width:action
~record_number.starting_column.width:action
```

In an import or transaction RDF, the question mark character ? is used to denote the conditional: "if data *exists* in the input foreign directory file at input record number, column number, and length, then perform action."

For example:

```
?1.10.15:emailSURNAME=1.10.15
```

This example translates to: "if data exists (some non-white space (blanks and/or tabs)) in record number 1 starting at column 10 for 15 characters, then take that data and put it in the X.500 attribute, emailSURNAME."

In an import or transaction RDF, the tilde ~ character is used to denote the conditional: "if data does *not exist* in the input foreign directory file at input record number, column number, and length, then perform action."

For example:

```
~1.10.15:emailSURNAME="John Doe"
```

This example translates to: "if data does not exist in record number 1 starting at column 10 for 15 characters, then put the string "John Doe" in the X.500 attribute, emailSURNAME."

3.3.3.3.2 Export and Changes RDF Conditional Statement Syntax

As seen in Table 3–5, the syntax for the export and changes operations conditional statements are:

```
?keyword:action
~keyword:action
```

In an export or changes RDF, the question mark character ? is used to denote the conditional: "if data exists in the X.500 attribute then perform action."

For example:

```
?emailNAME:emailNAME=3.20.#
```

This syntax example translates to: "if the X.500 attribute emailNAME exists, then put emailNAME data in record number 3 at column 20 for the actual number of characters in the string."

In an export or changes RDF, the tilde ~ character is used to denote the conditional: "if data does not exist in the X.500 attribute, then perform action."

For example:

```
~emailNAME:"*none*"=3.20.#
```

The above syntax example translates to: "if the X.500 attribute emailNAME does not exist or consists of white space (blanks and/or tabs) data, then put the string "*none*" in record number 3 at column 20 for the actual number of characters in the string."

For more information on how RDF conditional statements are used, see the example RDFs in the Synchronizer-500 root directory, which liberally use conditional syntax.

3.3.3.4 Export and Changes Operator Characters

For additional flexibility in creating export data files, the RDF syntax supports two special formatting operators–the asterisk * and the pound sign #. The example below illustrates how to use these characters.

In this example, an export file is needed that concatenates the emailSURNAME and emailGIVENNAME attributes from X.500 and places a period . character between them. The following RDF statements accomplish this concatenation:

```
emailSURNAME=1.1.15
"."=1.16.1
emailGIVENNAME=1.17.10
```

Because the length of the emailSURNAME attribute varies depending on the data in it, the result of the above statements might resemble the following:

```
SMITH           .JOHN
GREEN           .GEORGE
DIAL            .THOMAS
BROWN           .HARRY
```

Excess space exists between the emailSURNAME and emailGIVENNAME. The excess embedded spaces in the export file can be eliminated by using the asterisk * and pound sign # operators.

In an export RDF statement, the field length can be expressed by the pound sign # character. The pound sign # character instructs Synchronizer-500 to use the length of the data found in the specified attribute as the field's width. Trailing spaces are truncated. Thus, "SMITH_ __", shown above, becomes ''SMITH''.

When used in an export RDF statement, the starting column can be specified by the asterisk * character, which means ''write the next output file data field immediately after the last field written.''

Using the asterisk * and pound sign # characters, the RDF shown above can be rewritten to ensure that the embedded spaces are removed:

```
emailSURNAME=1.1.#
"."=1.*.1
emailGIVENNAME=1.*.10
```

The resulting export data then becomes:

```
SMITH.JOHN
GREEN.GEORGE
DIAL.THOMAS
BROWN.HARRY
```

Note

The asterisk * and pound sign # symbols are valid for export and changes operations only. If these symbols are used in import or transaction operations, they are flagged as an error condition and Synchronizer-500 terminates.

3.3.3.5 RDF Rules

RDFs are, by nature, extremely flexible tools. However, a few rules must be observed in writing RDFs:

1. For import and transaction operations, a valid distinguished name must be generated for the structural class you specified for the entries. Values for each of the relative distinguished names in the dn_sequence must be defined.
2. For import and transaction operations, X.500 attribute values are truncated to the lengths specified in the Layout File and an INPTRUNC warning is generated (see Appendix E).
3. For import and transaction operations, trailing blanks/tabs are truncated.
4. For import and transaction operations, the X.500 attribute specified in the configuration file to hold the Group ID (GID_FIELD) will not be available to hold other data.
5. For all operations, ensure that the NREC_SUBSCR value in the configuration file is equal to or greater than the largest RDF RECORD_NUMBER value. At RDF processing time, each RECORD_NUMBER value is checked against the appropriate configuration file NREC_SUBSCR value. If the RDF RECORD_NUMBER is greater than the NREC_SUBSCR value, Synchronizer-500 aborts with an INVRDFNUM status.
6. For all operations, when specifying an attribute that allows multiple values, the attribute name should be specified by appending the keyword with "_n". For example, 4 values are allowed for the attribute "title". The RDF keywords should be specified as "title", "title_2", "title_3", and "title_4".

Note

When attributes are specified with multiple values the order of the values cannot be guaranteed per X.500 recommendations. If the order of the values is important, specify the values in separate attributes rather than storing them in the same attribute.

3.3.4 Starting Synchronizer-500

Synchronizer-500 is started by the following command:

```
OpenVMS:

$ XDSU :==$SYS$SYSTEM:XDSU$MAIN.EXE
$ XDSU [P1 P2 P3]

UNIX:

# /usr/etc/xdsu [p1 p2 p3]
```

Note

On OpenVMS, a symbol must be assigned as a foreign command in order to pass optional parameters to the Synchronizer-500 executable image (XDSU$MAIN.EXE).

The XDSU command can be executed with three optional parameters ([p1 p2 p3]):

- p1 - the first optional parameter is for configuration file. By default, Synchronizer-500 looks for a configuration file with the name xdsu_config.dat in the current directory. If it is necessary to locate a configuration file in a directory other than the current directory where Synchronizer-500 is executed, the location can be specified as the first parameter

on the Synchronizer-500 command line. This parameter can be any valid, fully qualified file specification.

- p2 - the second optional parameter is for X.500 layout file. By default, Synchronizer-500 looks for its X.500 layout file, xdsu_layout.dat, one directory level up ([-] on OpenVMS or ../ on UNIX) from the current directory. If it is necessary to specify an X.500 layout file for Synchronizer-500 to use (other than xdsu_layout.dat), the location and file name can be specified as the second parameter on the Synchronizer-500 command line. This parameter can be any valid, fully qualified file specification.
- p3 - the third optional parameter is for valid id file. By default, Synchronizer-500 looks for its valid id file, xdsu_valid_id.dat, one directory level up ([-] on OpenVMS or ../ on UNIX) from the current directory. If it is necessary to locate the valid id file in a directory other than one directory level up from the current directory, the location can be specified as the third parameter on the Synchronizer-500 command line. This parameter can be any valid fully qualified file specification.

For example, to specify a different configuration file:

```
OpenVMS:

$ XDSU :== $SYS$SYSTEM:XDSU$MAIN.EXE
$ XDSU SYS$LOGIN:MICROSOFT_CONFIG.DAT

UNIX:

/usr/etc/xdsu /usr/user/smith/microsoft_config.dat
```

where SYS$LOGIN:MICROSOFT_CONFIG.DAT is the configuration file name on OpenVMS and /usr/user/smith/microsoft_config.dat on UNIX. This parameter can be any valid, fully qualified file specification.

If the first parameter is not specified, Synchronizer-500 looks for a configuration file in the standard location.

The Synchronizer-500 command can be executed from the operating system level (either from the system prompt or embedded in a command procedure) or from a batch job (regularly scheduled or one-time). Either method starts the Synchronizer-500 run.

If Synchronizer-500 is executed at the system prompt, its output is directed to SYS$OUTPUT for OpenVMS or stdout for UNIX. During a run, Synchronizer-500 outputs "timestamp" messages to show what activity is currently being performed. Many examples of these messages are shown in Chapter 5 and Appendixes A-C. For an example, see Example 5-4.

Chapter 4
Suggestions for Running Synchronizer-500

This chapter offers some suggestions to use when setting up and running Synchronizer-500. The suggestions include:

- Limit directory sizes for better performance and management or if limited resources are available
- Use a DIT structure that minimizes name clashes
- Choose an appropriate X.400 naming scheme that will allow each subscriber to be uniquely identified
- Allocate sufficient disk space to improve performance and lower or eliminate the risk of failure as a result of not enough space
- Spread metafiles across disks to improve performance
- Monitor the log files and enable verbose to find problems

4.1 Limiting the Size of Directory Groups

Best performance can be obtained by breaking up large directory files into several smaller files, each with a separate group ID (GID). Large groups of entries take longer to extract the information from X.500 to make the comparison with the new directory. By breaking up a large directory into several smaller files, much of the extraction and comparison work can be done in parallel, which reduces the overall time required to run a synchronization.

In addition, breaking the directory file into smaller files makes management and troubleshooting easier. When an error occurs in the processing of a relatively small directory file, it is easier to find the problem and restart the job than it is with a very large file. Breaking a foreign directory into smaller, more manageable files should be done as logically as possible.

Finally, breaking the directory file into smaller files uses less resources (memory and disk) per Synchronizer-500 run which may be critical if memory or disk space is limited.

4.2 Choose Unambiguous Distinguished Names

No two entries in X.500 can have the same distinguished name. X.500 will not allow an entry to be added unless it's distinguished name is unique. Try to design a DIT that divides your organization into groupings of entries that cause as few name clashes as possible.

A name clash occurs when two or more entries require the same distinguished name. Every distinguished name must be unique, so, for example, if you have several persons named John Smith working for the same organizational unit, you have a name clash to resolve. Try to divide your organization so that organizational units are small enough for such clashes to be rare. (You can never completely avoid the possibility of name clashes). Please refer to Chapter 3 of the *DEC X.500 Directory Service Management Guide* for information on planning your Directory Information Tree and how to resolve name clashes when they occur.

4.3 Choosing an Appropriate X.400 Subscriber-naming Scheme

When building a multivendor electronic mail directory service, it is important to select a X.400 naming scheme that allows each subscriber to be uniquely identified. (In X.400, no two users can have the same address.) Synchronizer-500 checks to ensure that X.400 1984 and 1988 standard attributes and up to 12 electronic mail address aliases are unique. (See Section 1.3.6 for more information on the uniqueness checking performed by Synchronizer-500.

In a directory with thousands of users, it is likely that some will have the same first or last names. To avoid this type of conflict, choose a naming scheme with enough attributes to distinguish between each user. An example of such a naming scheme is:

SURNAME
GIVENNAME
INITIALS
ORGANIZATION
ORGANIZATIONAL UNIT

For more information on X.400 naming, please refer to *Guidelines for Naming People in an Organization for the Purposes of X.400 Electronic Mail.* A copy can be obtained from your local Digital sales or sales support representative.

4.4 Allocating Disk Space

When synchronizing large directories, Synchronizer-500 may require large amounts of disk space. During a Synchronizer-500 run, several metafiles and output files may be created.

The size of a metafile record may vary. Using the default X.500 Layout File, a full metafile record is 22550 bytes, but metafile records are written to disk in compressed format. Because of this, the size of each record depends on how much information is contained in each entry. A typical input file (e.g. the example used in Chapter 5) uses about 600 bytes per metafile record.

The following tables show how much space to allow for Synchronizer-500 runs for each operation type. The equations for metafile operations usually show that two-times the metafile size is needed. This is because these metafiles have to sorted, which creates a new version of the file. After the sort successfully ends, the old, unsorted version is purged.

Table 4–1: Import File Size Estimates

File	File Size Equation(in bytes)
Old metafile	2 * (Number of existing entries in X.500 for this Group ID * 600 bytes)
New metafile	2 * (Number of entries in input foreign directory file * 600 bytes)

Table 4–2: Transaction File Size Estimates

File	File Size Equation(in bytes)
Add metafile	Number of entries being added to X.500 * 600 bytes
Mod metafile	Number of entries being modified to X.500 * 600 bytes
Del metafile	Number of entries being deleted from X.500 * 600 bytes

Table 4–3: Export File Size Estimates

File	File Size Equation(in bytes)
Output Old metafile	2 * (Number of existing entries in X.500 for this Group ID * 600 bytes)
Output foreign directory file	Number of entries extracted from X.500 * size of foreign record

Table 4–4: Changes File Size Estimates

File	File Size Equation(in bytes)
Output New metafile	2 * (Number of existing entries in X.500 for this Group ID * 600 bytes)
Output add foreign directory file	Number of entries added to X.500 * size of foreign record
Output modify foreign directory file	Number of entries modified in X.500 * size of foreign record
Output delete foreign directory file	Number of entries deleted from X.500 * size of foreign record

For OpenVMS, to calculate the file size in blocks (512 bytes per block), divide the resultant number by 512. For example, an import run is going to extract 2497 records from X.500, and the input foreign directory file has 2521 records. The calculations are:

```
old metafile = 2 * (2497 * 600) = 2,996,400 bytes = 5852 blocks
new metafile = 2 * (2521 * 600) = 3,025,200 bytes = 5909 blocks
```

4.5 Spreading out Metafiles

Often, because of the sheer size of the metafiles, it may be best to spread the metafiles out over several disks. By default, metafiles are created in the default directory. By giving alternate filenames for metafile keywords in the configuration file, metafiles can be spread out.

For import processsing, specify values for OLD_METAFILE and NEW_METAFILE. For transaction processing, specify values for ADD_METAFILE, MOD_METAFILE, and DEL_METAFILE. For export processing, specify a value for OUTPUT_OLD_METAFILE. For changes processing, specify a value for OUTPUT_NEW_METAFILE.

4.6 Monitoring Synchronization Log Files

At the completion of each run, Synchronizer-500 sends a summary message that contains transaction statistics. This message specifies the start- and end-processing time stamps; the Group ID; the total number of transactions processed; the number of successful ADD, MODIFY, and DELETE transactions; and the number of transactions rejected. If you have an idea of what the ADD/MODIFY/DELETE statistics normally are, it will be easier to spot a problem based on the summary message.

The log file (xdsu_log.log or other file name specified in the configuration file) should then be checked after each run so that problems can be detected and corrected quickly. Searching the log file for the string "FAILED" returns any transactions which were not "NORMAL".

When an ADD or MODIFY operation fails, try to ascertain why. A frequent cause of failure is that the operation would have resulted in a duplicate address, most likely the X.400 address.

Enabling verbose mode (VERBOSE=YES) in the configuration file causes Synchronizer-500 to write complete transaction information to the log file and to write diagnostic information to the standard output device.

Chapter 5

An Example — Synchronizing with Microsoft Mail

This chapter presents an example of how to use Synchronizer-500 to synchronize directory information between X.500 and Microsoft Mail for DOS. All of the steps required to set up and configure Synchronizer-500 are discussed in detail.

Several synchronization runs are shown, along with the configuration files, record description files (RDFs), log files, and synchronization results.

The example Microsoft Mail import directory file, the RDFs, and the configuration files used in these examples are distributed with the Synchronizer-500 software. These files are placed in the Synchronizer-500 root directory when Synchronizer-500 is installed. Please refer to these files while reading this chapter.

The following sections describe the import, export, and changes operations in detail. These examples describe all of the aspects of directory synchronization with Microsoft Mail, including the steps necessary for:

- *Import*–Receive and process a Microsoft Mail import file and integrate the Microsoft Mail user records into X.500.
- *Export*–Generate a complete Synchronizer-500 export file that contains all of the information about users in the X.500 DIT, except Microsoft Mail users, and send the file to the Microsoft Mail Directory Administrator to be used as an initial load for the Microsoft Mail directory.
- *Changes*–Generate incremental Synchronizer-500 changes files that contain the transactions that have occurred to the X.500 DIT since the last changes operation (except Microsoft Mail users) and send the files to the Microsoft Mail Directory Administrator to be used to update the Microsoft Mail directory.

5.1 Overview of the Synchronization Process

Microsoft Mail provides a directory service that contains a directory of all local and remote users. Microsoft Mail users can query the directory to find the address of intended recipients and use that information to address a mail message.

The Microsoft Mail addressing scheme is fairly simple compared to other electronic mail systems. It consists of an alias that is used to identify the recipient and the physical address for that alias. The alias consists of an ASCII-character string, with a minimum length of 1 character and a maximum length of 30 characters. The physical address represents the actual location of the recipient, such as a FAX, MCI, SMTP, X.400, or Microsoft Mail address.

The Microsoft Mail Import Utility, available from Microsoft, supports the importing and exporting of address information to and from the Microsoft Mail directory. This integration of directory information is performed by running the utility and specifying one of the following:

- The name of an import file that contains directory information to integrate into Microsoft Mail's directory. (Import, as used here, refers to importing data into Microsoft Mail. Synchronizer-500 is run in export or changes mode to provide this import file.)
- The name of an export file where Microsoft Mail directory information will be written. (Export, as used here, refers to exporting data from Microsoft Mail. Synchronizer-500 is run in import mode to process this export data.)

In either of the above cases, the file format is the same. Each subscriber is represented by a single record, with a maximum length of 512 characters. Each record consists of three mandatory fields:

1. A single-character transaction type that indicates if the record is an add (A) or a delete (D)
2. An alias field by which this address will be known
3. The physical address of the alias

Example 5–1 shows entries from a typical Microsoft Mail file.

Example 5–1: Typical Microsoft Mail File

```
A Microsoft Mail Administrator   NOVLAN/MEL/ADMIN
A Our Guest                      NOVLAN/MEL/Guest
A Jane Black                     NOVLAN/MEL/Black
A Susan Blue                     NOVLAN/MEL/Blue
A Byron Smith                    NOVLAN/MEL/Smith
A Sharon Rotary                  NOVLAN/MEL/Rotary
```

Note

The information shown in Example 5–1 resides in fixed locations in the file, as required by Synchronizer-500. If it did not, it would be necessary to pre-process the file into a new format in which the fields were column-aligned. Synchronizer-500 does not provide the pre-processor to accomplish this.

Example 5–2 shows an X.500 entry created during the Synchronizer-500 import operation.

Example 5–2: Sample Microsoft Mail X.500 Entry

```
dxim> show /c=us/o=digital/ou=mel/cn="susan blue"

/C=us/O=digital/OU=mel/CN=susan blue
    Common Name    = Susan Blue
    Object Class   = Person
                   = Organizational Person
                   = emailInfo
    Surname        = Blue
    Description    = MICROSOFT
    emailX400Addr84 = S=Blue; OU1=MEL; O=NOVLAN; P=EMSERVER; A=PUBLIC; C=US
    emailX400Addr88 = CN=Blue; OU1=MEL; O=NOVLAN; P=EMSERVER; A=PUBLIC; C=US
    emailMicrosoftAddr = NOVLAN/MEL/Blue
    emailMAILbusAddr = 6=Blue@4=MEL@5=NOVLAN@3=EMSERVER@2=PUBLIC@1=US@NCX400
                       @NODE1
    emailPCOUNTRY  = US
    emailPRLOCATION = NJUSDCOA
    emailPRUSERNAME = Blue
    emailSNLOCATION = NACGWYR1
    emailSNUSERNAME = Blue
    emailADMDNAME  = PUBLIC
    emailMHSMAILBOX = NCX400
    emailADDRPART  = NODE1
    emailORGUNIT   = MEL
    emailPRMDNAME  = EMSERVER
    emailNAME      = Susan Blue
    emailORGNAME   = NOVLAN
    emailSURNAME   = Blue
    emailUSE1      = X5SYNCH
    emailUSE5      = Blue-NOVLAN-MEL
    emailMHSUSERID = 6=Blue@4=MEL@5=NOVLAN@3=EMSERVER@2=PUBLIC@1=US
```

5.1.1 Mail System Configuration and Addressing

For the example in this chapter, it is assumed that Microsoft Mail is connected to the messaging backbone via X.400. This will enable users on Microsoft Mail to exchange mail with any other users connected to the backbone.

Microsoft Mail users will address mail to X.400 users as though the recipients were on a remote Microsoft Mail system by selecting an alias from the Microsoft Mail directory (e.g., TO: Smith-John-A). To give Microsoft Mail users the ability to look up X.400 recipients, the Microsoft Mail directory needs to be loaded with these aliases and their addresses.

X.400 users will address mail to Microsoft Mail recipients by composing an address in X.400 syntax (TO: s=Byron; ou1=MEL; o=NOVLAN; p=EMSERVER; a=PUBLIC; c=US). NOTE: the specific manner in which the user enters the X.400 address depends on the gateway they are using. This address can be entered by hand or taken from X.500 via a directory user agent.

5.2 Import Process

The following is a broad outline of the steps that must be performed to import Microsoft Mail directory information into X.500:

1. Create a Synchronizer-500 subdirectory for this instance.
2. Run the Microsoft Mail Import utility to generate an export file.
3. Transfer the Microsoft Mail file to the X.500 system via mail message or file transfer.
4. Determine the format of the Microsoft Mail file and decide how to map the data to X.500 attributes.
5. Write the RDF to reflect this data correspondence.
6. Set up the configuration file with the proper parameters.
7. Run Synchronizer-500 to import the Microsoft Mail information into X.500.
8. Check the log file for any errors.

5.2.1 Create the Microsoft Mail Import Subdirectory

As described in Section 3.3.1, it is recommended that a subdirectory to the Synchronizer-500 root directory be created for this instance. This subdirectory would contain the configuration file, RDF, and input data file for the Microsoft Mail import.

When Synchronizer-500 was installed, a command procedure was placed in the Synchronizer-500 root directory which will create the subdirectory and copy the appropriate configuration file template to that newly created directory. This procedure is executed by the following steps:

1. Set your current directory to the Synchronizer-500 root directory.
2. Execute the xdsu_config_instance procedure:

```
OpenVMS:
$ @XDSU_CONFIG_INSTANCE

UNIX:
#xdsu_config_instance
```

3. The procedure prompts:

```
* Enter XDSU instance directory name:
```

   Enter the name of the directory you wish to create. For example, to create a subdirectory to be used for importing Microsoft Mail users into X.500, you might enter MSM_IMPORT.

4. The procedure then prompts:

```
* Enter Synch Type (Import, Export, Changes, or Transaction):
```

   Enter **I** since this is an import instance.

5. The procedure then creates the directory and copies the import configuration template file to the newly created directory.

6. Set your current directory to the newly created directory and proceed with setting up the configuration file and RDF files as described below.

5.2.2 Run the Microsoft Mail Import Utility

Execute the Microsoft Mail Import Utility in export mode to export the Microsoft Mail directory to a flat file that Synchronizer-500 can use. Please see Section 5.1 for a brief overview on the Microsoft Mail Import Utility. Refer to the Microsoft Mail documentation for more complete information.

The file that is generated becomes the import file for Synchronizer-500.

5.2.3 Transfer the Microsoft Mail File

The Microsoft Mail file that was generated in Section 5.2.2 will be sent to the X.500 Directory Administrator on the X.500 system via mail message. The Directory Administrator receives the message, reads it, and detaches the Microsoft Mail file for further processing.

5.2.4 Determine Format of the Microsoft Mail File

The Microsoft Mail file should be analyzed to determine the format of the data. If the data is not in a fixed format pre-processing is required.

When the analysis of the file is done, the structure of the DIT is used to determine what RDNs and attributes need to be included in the RDF for each entry. This is based upon what classes are being used for the entries being imported.

An analysis of the Microsoft Mail file shown in Example 5–1 yields the following mappings:

```
  X.500
Attribute                    Start      Length
-----------------------------------------------

organizationalUnit             41           3
surname                        45          10
commonName                      3          30
emailORGNAME                   34           6
emailORGUNIT                   41           3
emailSURNAME                   45          10
emailNAME                       3          30
```

In this example, the Person, organizationalPerson, and emailInfo classes are being used. Data is therefore mapped into attributes from those classes. Note that the data file does not contain values for all attributes that need to be specified for an entry and therefore constants will be supplied in the RDF.

5.2.5 Writing the Import RDF

Using the mapping above, the RDF, xdsu_import_microsoft.rdf, is created to tell Synchronizer-500 which RDNs and attributes to create in X.500 and where to get the values for those RDNs and attributes. (The xdsu_import_microsoft.rdf is contained in the Synchronizer-500 root directory and should be referred to while reading the following text.)

The first part of the RDF defines RDNs and their values.

The first two RDF records in xdsu_import_microsoft.rdf establish the default values for the COUNTRY and ORGANIZATION RDNs. Because this information is not contained in the input file, constants must be provided for all entries added during this run. Constants are inserted into the RDF by putting in the values required as quoted strings:

```
country="US"
organization="Digital"
```

Next, the organizationalUnit and commonName RDNs and the surname attribute will be mapped from the input file:

```
organizationalUnit=1.41.3
surname=1.45.10
commonName=1.3.30
```

Note

These field lengths and start/stop positions will vary depending on the Microsoft Mail Postoffice and Network names. This RDF assumes that only local Microsoft users are registered in this Postoffice; and, hence, all users have constant Postoffice and Network names. If this is not the case, some data cleanup or pre-processing may be required on the input file. Synchronizer-500 does not provide the pre-processor to accomplish this.

The remaining RDF records define the attributes that will be used in the emailInfo auxiliary class.

The next records establish the values for the PROFS and SNADS gateway locations since it is assumed that there is a Message Router/PROFS and a Message Router/SNADS gateway in the messaging backbone:

```
emailPRLOCATION="NJUSDCOA"
emailSNLOCATION="NACGWYR1"
```

Again, because this information is not contained in the input file, constants are provided for these attributes. Since there are PROFS and SNADS gateways, PROFS and SNADS alias generation should be turned on. An explanation of how Synchronizer-500 is instructed to generate both PROFS and SNADS aliases via configuration files is given in Section 5.2.6.

The addresses for the Microsoft Mail users must be augmented to include all necessary X.400 global domain identifiers (GDIs). The following three RDF records supply constants for X.400 information that will be used as defaults for each Microsoft Mail user:

```
emailPCOUNTRY="US"
emailADMDNAME="PUBLIC"
emailPRMDNAME="EMSERVER"
```

The ORGNAME, ORGUNIT, and SURNAME fields must be mapped from the input file:

```
emailORGNAME=1.34.6
emailORGUNIT=1.41.3
emailSURNAME=1.45.10
```

Note

These field lengths and start/stop positions will vary depending on the Microsoft Mail

Postoffice and Network names. This RDF assumes that only local Microsoft users are registered in this Postoffice; and, hence, all users have constant Postoffice and Network names. If this is not the case, some data cleanup or pre-processing may be required on the input file. Synchronizer-500 does not provide the pre-processor to accomplish this.

Using the emailSURNAME, emailORGNAME, and emailORGUNIT attributes as a basis, the RDF creates a key field that will be unique in X.500 and puts it in the emailUSE5 attribute. By concatenating these three values, a descriptive name can be derived for use as an alias. This alias value can then be referenced when looking up users in other mail system directories at the To: field. This example uses the format of SURNAME-ORGNAME-ORGUNIT in attribute emailUSE5.

The uniqueness of attribute emailUSE5 is guaranteed by Synchronizer-500 if emailUSE5 is named as a unique attribute (i.e., configuration record, UNIQUE_FIELD=emailUSE5). In addition, because the length of this field is mandated by the Microsoft Mail Import Utility as 30 characters, Synchronizer-500 is instructed to make the emailUSE5 attribute unique up to 30 characters (i.e., configuration record, UNIQUE_FIELD_LEN=30).

```
emailUSE5=1.45.10
emailUSE5="-"
emailUSE5=1.34.6
emailUSE5="-"
emailUSE5=1.41.3
```

Next, the Microsoft Mail alias should be placed in the emailNAME attribute:

```
emailNAME=1.3.30
```

Because our example environment includes both 1984 and 1988 X.400 implementations, both the 1984 and 1988 versions of the X.400 address need to be created. This requires information that resides in three separate places in the Microsoft Mail record as well as constant information.

Build the 1984 X.400 address, and place it in the emailX400Addr84 attribute. Synchronizer-500 builds X.500 attribute values from left to right so the surname needs to be provided first. Supply the constant to identify SURNAME, map the SURNAME value from the input data file, and use the semi-colon ; as a delimiter between attributes:

```
emailX400Addr84="S="
emailX400Addr84=1.45.10
emailX400Addr84=";"
```

The next attribute is ORGUNIT. Supply the constant to identify ORGUNIT and map in the ORGUNIT value from the input data file:

```
emailX400Addr84=" OU1="
emailX400Addr84=1.41.3
emailX400Addr84=";"
```

The ORGNAME attribute comes next. Again, supply the constant to identify ORGNAME and map in the ORGNAME value from the input data file:

```
emailX400Addr84=" O="
emailX400Addr84=1.34.6
emailX400Addr84=";"
```

Last the GDI (Global Domain Identifier) terms: PRMD, ADMD and Country must be supplied. Constants will be provided for these values:

```
emailX00Addr84=" P=EMSERVER; A=PUBLIC; C=US"
```

When Synchronizer-500 sees an X.500 attribute referenced in several RDF lines, it will concatenate the information contained in those RDF records (input file data and string constants). Thus the concatenation of all the RDF records referencing emailX400Addr84 would result in:

```
S=xxxxxxxxxx; OU1=xxx; O=xxxxxx; P=EMSERVER; A=PUBLIC C=US
```

Next, the 1988 X.400 address is built and put in the emailX400Addr88 attribute. This is very similar to the 1984 X.400 address so refer to the text above for explanation.

```
First the commonname:

emailX400Addr88="CN="
emailX400Addr88=1.45.10
emailX400Addr88=";"

Next the orgunit:

emailX400Addr88=" OU1="
emailX400Addr88=1.41.3
emailX400Addr88=";"

Now the organization name:

emailX400Addr88=" O="
emailX400Addr88=1.34.6
emailX400Addr88=";"

Last the GDI (Global Domain Identifier) terms: country, ADMD, and PRMD

emailX400Addr88=" P=EMSERVER; A=PUBLIC; C=US"
```

As before, because emailX400Addr88 is referenced several times, Synchronizer-500 will concatenate the information contained in the RDF records (input file data and string constants). Thus the concatenation of all the RDF records referencing emailX400Addr88 would result in:

```
CN=xxxxxxxxxx; OU1=xxx; O=xxxxxx; P=EMSERVER; A=PUBLIC C=US
```

Since Microsoft Mail users are being imported, their physical address will be put in the attribute emailMicrosoftAddr:

```
emailMicrosoftAddr=1.34.18
```

In this example, it is assumed that MAILbus is part of the X.400 messaging backbone connected via MRX. Because of this, the X.500 emailMHSUSERID attribute needs to present the users' X.400 address in MAILbus format.

Creation of the emailMHSUSERID requires information that resides in three separate places in the Microsoft Mail record. Additional constant information is also needed. A numeric constant to identify each attribute must also be provided:

```
emailMHSUSERID="6="
emailMHSUSERID=1.45.10
emailMHSUSERID="@4="
emailMHSUSERID=1.41.3
emailMHSUSERID="@5="
emailMHSUSERID=1.34.6
emailMHSUSERID="@3=EMSERVER@2=PUBLIC@1=US"
```

The concatenation of all the RDF records using emailMHSUSERID would result in:

```
6=xxxxxxxxxx@4=zzz@5=yyyyyy@3=EMSERVER@2=PUBLIC@1=US
```

In this emailMHSUSERID, *xxxxxxxxxx*, *yyy*, and *zzzzzz* are the Microsoft Mail SURNAME, ORGNAME, and ORGUNIT fields extracted from the indicated location in the Microsoft Mail export file.

Finally, the remainder of the MAILbus address is added, which is of the form:

```
emailMHSUSERID @emailMHSMAILBOX @emailADDRPART
```

Because the name of the emailMHSMAILBOX and VAX name do not appear in the Microsoft Mail file, constants are needed. The Message Router mailbox that MRX uses for servicing the X.400 mail network is named NCX400, and the node that MRX is installed on is NODE1:

```
emailMHSMAILBOX="NCX400"
emailADDRPART="NODE1"
```

Lastly, we need to put the full MAILbus address (emailMHSUSERID @emailMHSMAILBOX @emailADDRPART) information into the emailMAILbusAddr attribute:

```
emailMAILbusAddr="6="
emailMAILbusAddr=1.45.10
emailMAILbusAddr="@4="
emailMAILbusAddr=1.41.3
emailMAILbusAddr="@5="
emailMAILbusAddr=1.34.6
emailMAILbusAddr="@3=EMSERVER@2=PUBLIC@1=US@NCX400@NODE1"
```

This completes the set up of the RDF file. The next step is to set up the configuration file.

5.2.6 Setting up the Import Configuration File

The configuration file (xdsu_config.dat) for this example is set up as shown in Example 5–3. This sample file (xdsu_config.dat_import_microsoft) is included in the Synchronizer-500 root directory and should be referred to while reading the following text.

Example 5–3: Configuration File for Microsoft Mail Import

```
synch_type=IMPORT
!
dn_sequence=country/organization/organizationalUnit/commonName
create_root=organization
synch_object=Person,organizationalPerson,emailInfo
bind_dsa=
!
rdf_file=xdsu_import_microsoft.rdf
initial_load=Y
input_for_dir_file=xdsu_import_microsoft.txt
nrec_subscr=1
gid_field=description
gid_match=MICROSOFT
max_transactions=
max_fail_transactions=
max_trans_file=
end_run_file=xdsu_import_microsoft.com
notify=SYSTEM
synch_id=X5Synch
synch_id_field=emailUSE1
unique_field=emailUSE5
unique_field_len=30
gen_profs_alias=Y
gen_snads_alias=Y
old_metafile=
new_metafile=
log_file=import_microsoft.log
verbose=Y
encode_delim=|
synch_label=Microsoft Mail Import
make_unique=
truncate=
admd_space=
```

This file is an example of a configuration file for an import synchronization and an initial load. The example RDF xdsu_import_microsoft.rdf, example data file xdsu_import_microsoft.txt, and example end_run_file xdsu_import_microsoft.com are used. (These files are contained in the Synchronizer-500 root directory.)

The import configuration file consists of keyword and value pairs. In most cases, not all keywords will be used. Unused keywords can remain blank or can be removed from the configuration file. Descriptions of each keyword and example values are given below. Please see Appendix F for more detailed information.

- *synch_type* must specify the type of synchronization being performed. In this example it is an **IMPORT**.

- *dn_sequence* must specify the order of and the relative distinguished names that make up an entry's distinguished name. In this example, the distinguished name that is generated for each entry upon import is made up of the RDNs: **country, organization, organizationalUnit**, and **commonName**. (As required, the RDF specifies values for each of these RDNs.)
- *create_root* specifies the highest level of superior entries that can be created for this import if necessary. When the entries are added in this example, the superior entries for **organization** and **organizationalUnit** are created if needed.
- *synch_object* specifies the X.500 structural classes for this import. When the entries are imported in this example, they are specified to be of the **Person, organizationalPerson,** and **emailInfo** classes. This value must be entered as a contiguous string (no spaces) with commas separating the classes.
- *bind_dsa* specifies a DSA presentation address. This feature is not enabled at this time. *Reserved for future use.*
- *rdf_file* must specify the file name of the RDF used for this import. In this example, the **xdsu_import_microsoft.rdf** file is used.
- *initial_load* specifies if this is the first time the foreign directory is being imported into X.500. In this example, it is an initial load.
- *import_foreign_dir_file* must specify the file name for the import data file. In this example, the **xdsu_import_microsoft.txt** file is used.
- *nrec_subscr* specifies how many records per entry are in the import data file. In this example, there is only **1** record per entry.
- *gid_field* must specify the X.500 attribute in which the Group ID value is stored. In this example, **DESCRIPTION** is used to store the Synchronizer-500 Group ID.
- *gid_match* specifies the value for the Group ID for this import. In this example, a value of **MICROSOFT** is used.
- *max_transactions* and *max_trans_file* are used together to specify a user-written procedure that should be executed when the number of transactions have been reached.
- *max_fail_transactions* specifies that Synchronizer-500 should stop executing after the specified number of failures have occurred.
- *end_run_file* specifies the file name of a user-written procedure that is executed upon termination of this Synchronizer-500 run. In this example, the command procedure **xdsu_import_microsoft.com** is executed.
- *notify* specifies the electronic mail userid to receive status messages. In this example, **SYSTEM** receives all status messages.
- *synch_id* specifies the tag added to each entry to indicate that Synchronizer-500 "owns" the entry. In this example, the value of **X5Synch** is used.
- *synch_id_field* specifies the X.500 attribute in which to place the synch_id value. In this example, the attribute **emailUSE1** is used.

- *unique_field* specifies an X.500 attribute Synchronizer-500 will check for uniqueness. In this example, the attribute **emailUSE5** will be checked by Synchronizer-500. (See MAKE_UNIQUE, below.)
- *unique_field_len* specifies the length to guarantee uniqueness for unique_field. In this example, emailUSE5 is guaranteed to be unique up to **30** characters.
- *gen_profs_alias* specifies whether or not the PROFS alias should be generated. In this example, PROFS aliases are generated.
- *gen_snads_alias* specifies whether or not the SNADS alias should be generated. In this example, SNADS aliases are generated.
- *old_metafile* specifies the file name and location of the old metafile.
- *new_metafile* specifies the file name and location of the new metafile.
- *log_file* specifies the file name for the Synchronizer-500 transaction log file. In this example, the transactions will be logged to the file **import_microsoft.log** in the current directory where Synchronizer-500 is executed.
- *verbose* specifies that a formatted listing of each entry Synchronizer-500 attempts to import into X.500 is written to the log file. In this example, verbose mode is enabled.
- *encode_delim* specifies a character to use as a delimiter when Synchronizer-500 needs to encode characters not available in the T.61 character set. In this example, the | character is used as the delimiter.
- *synch_label* specifies the label for Synchronizer-500 mail messages and end-of-run synopses. In this example, the label **Microsoft Mail Import** is used.
- *make_unique* specifies if Synchronizer-500 should modify the UNIQUE_FIELD to make it unique. The default value of "YES" is used in this example since no value was specified.
- *truncate* is used in combination with UNIQUE_FIELD and MAKE_UNIQUE. This field specifies whether the data or the number should be truncated if necessary when modifying UNIQUE_FIELD to be unique.
- *admd_space* specifies that the attribute emailADMDNAME should have a blank as its value.

5.2.7 Running the Import–The Initial Load

Synchronizer-500 is then started by issuing the following command from the directory where the configuration, RDF, and foreign data files reside:

```
OpenVMS:

$ XDSU :== $SYS$SYSTEM:XDSU$MAIN.EXE
$ XDSU

UNIX:

/usr/etc/xdsu
```

As Synchronizer-500 executes, its processing is logged to your terminal device as shown in Example 5–4 (unless you are operating in batch in which case the process is logged to the batch log file). The processing should be examined as it occurs to review the results of the import run.

Example 5–4: Initial Load of Microsoft Mail Users Processing

```
Feb 28 14:39:55 processing layout file [-]XDSU_LAYOUT.DAT
Feb 28 14:39:56 processing configuration file XDSU_CONFIG.DAT
Feb 28 14:39:56 sending mail message to SYSTEM
Feb 28 14:39:56 initializing and binding to X.500
Feb 28 14:39:57 processing RDF file xdsu_import_microsoft.rdf
Feb 28 14:39:57 converting input file xdsu_import_microsoft.txt to new metafile
Feb 28 14:39:57 6 input records read
Feb 28 14:39:57 6 new metafile records written
Feb 28 14:39:57 sorting new metafile
Feb 28 14:39:58 searching X.500 directory data
    SEARCH  -
     where   description = MICROSOFT -
       and     EMAILUSE1 = X5Synch -
       and    COMMONNAME = *
Feb 28 14:39:58 found 0 X.500 directory entries
Feb 28 14:39:58 extracting X.500 directory data to old metafile
Feb 28 14:39:58 extracted 0 directory entries into old metafile
Feb 28 14:39:58 comparing old and new metafiles
Feb 28 14:39:58 processing ADD records
    ADD /C=US/O=DIGITAL/OU=MEL/CN=BYRON SMITH
    ADD /C=US/O=DIGITAL/OU=MEL/CN=JANE BLACK
    ADD /C=US/O=DIGITAL/OU=MEL/CN=MICROSOFT MAIL ADMINISTRATOR
    ADD /C=US/O=DIGITAL/OU=MEL/CN=OUR GUEST
    ADD /C=US/O=DIGITAL/OU=MEL/CN=SHARON ROTARY
    ADD /C=US/O=DIGITAL/OU=MEL/CN=SUSAN BLUE
Feb 28 14:40:00 6 records added
Feb 28 14:40:00 creating summary file
Feb 28 14:40:00 executing end run file xdsu_import_microsoft.com

XDSU V1.2-1
Microsoft Mail Import run started at: Tue Feb 28 14:39:55 1995

IMPORT SYNCH SUMMARY:

RECORDS READ/WRITTEN SUMMARY:
```

Example 5–4 Cont'd on next page

Example 5–4 (Cont.): Initial Load of Microsoft Mail Users Processing

```
Foreign directory file records read: 6
New metafile records written: 6
Old metafile records written: 0

TRANSACTION SUMMARY:

Number of successful transactions: 6
Number of failed transactions:     0
Number of successful add transactions:    6
Number of successful modify transactions: 0
Number of successful delete transactions: 0
Number of failed add transactions:    0
Number of failed modify transactions: 0
Number of failed delete transactions: 0

CONFIGURATION FILE SUMMARY:

synch_type: IMPORT
synch_label: Microsoft Mail Import
bind_dsa:
rdf_file: xdsu_import_microsoft.rdf
input_for_dir_file: xdsu_import_microsoft.txt
nrec_subscr: 1
log_file: import_microsoft.log
end_run_file: xdsu_import_microsoft.com
old_metafile:
new_metafile:
initial_load: Y
notify: SYSTEM
verbose: Y
encode_delim: |
max_transactions:
max_fail_transactions:
max_trans_file:
dn_sequence: country/organization/organizationalUnit/commonName
create_root: organization
search_context:
synch_object: Person,organizationalPerson,emailInfo
synch_id: X5Synch
synch_id_field: emailUSE1
gid_field: description
gid_match: MICROSOFT
gen_profs_alias: Y
gen_snads_alias: Y
admd_space:
unique_field: emailUSE5
unique_field_len: 30
make_unique:
truncate:
unique_field_separator:
unique_search_context:

Microsoft Mail Import run ended at: Tue Feb 28 14:40:00 1995
Feb 28 14:40:00 sending mail message to SYSTEM
```

Since a value for the notify parameter has been specified, a mail message is generated at the beginning and end of the Synchronizer-500 run (see Example 5–5).

Example 5–5: Initial Load Import OpenVMS Mail Message

```
From:   CLMNTE::SYSTEM
To:     SYSTEM
CC:
Subj:   Microsoft Mail Import run started at: Tue Feb 28 14:39:55 1995

Microsoft Mail Import run started at: Tue Feb 28 14:39:55 1995


From:   CLMNTE::SYSTEM
To:     SYSTEM
CC:
Subj:   Microsoft Mail Import run successfully ended at: Tue Feb 28 14:40:00 1995

XDSU V1.2-1
Microsoft Mail Import run started at: Tue Feb 28 14:39:55 1995

IMPORT SYNCH SUMMARY:

RECORDS READ/WRITTEN SUMMARY:

Foreign directory file records read: 6
New metafile records written: 6
Old metafile records written: 0

TRANSACTION SUMMARY:

Number of successful transactions: 6
Number of failed transactions:     0
Number of successful add transactions:    6
Number of successful modify transactions: 0
Number of successful delete transactions: 0
Number of failed add transactions:    0
Number of failed modify transactions: 0
Number of failed delete transactions: 0

CONFIGURATION FILE SUMMARY:
```

Example 5–5 Cont'd on next page

Example 5–5 (Cont.): Initial Load Import OpenVMS Mail Message

```
synch_type: IMPORT
synch_label: Microsoft Mail Import
bind_dsa:
rdf_file: xdsu_import_microsoft.rdf
input_for_dir_file: xdsu_import_microsoft.txt
nrec_subscr: 1
log_file: import_microsoft.log
end_run_file: xdsu_import_microsoft.com
old_metafile:
new_metafile:
initial_load: Y
notify: SYSTEM
verbose: Y
encode_delim: |
max_transactions:
max_fail_transactions:
max_trans_file:
dn_sequence: country/organization/organizationalUnit/commonName
create_root: organization
search_context:
synch_object: Person,organizationalPerson,emailInfo
synch_id: X5Synch
synch_id_field: emailUSE1
gid_field: description
gid_match: MICROSOFT
gen_profs_alias: Y
gen_snads_alias: Y
admd_space:
unique_field: emailUSE5
unique_field_len: 30
make_unique:
truncate:
unique_field_separator:
unique_search_context:

Microsoft Mail Import run ended at: Tue Feb 28 14:40:00 1995
```

The results of the initial load should be examined in the transaction log file specified by the field LOG_FILE in the configuration file (see Example 5–6). If no file was specified, the transaction log file name is xdsu_log.log. The configuration file shown in Example 5–3 specifies the log file name and enables verbose mode.

Example 5–6: Checking the Results of the Initial Load Import—import_microsoft.log

```
Microsoft Mail Import run started at: Tue Feb 28 14:39:55 1995

A INFORMATIONAL Parent Entry created: /C=US/O=Digital/OU=MEL

A NORMAL /C=US/O=DIGITAL/OU=MEL/CN=BYRON SMITH  - status=SUCCESS
```

Example 5–6 Cont'd on next page

Example 5–6 (Cont.): Checking the Results of the Initial Load Import—import_microsoft.log

```
  SURNAME = Smith
  DESCRIPTION = MICROSOFT
  EMAILX400ADDR84 = S=Smith; OU1=MEL; O=NOVLAN; P=EMSERVER; A=PUBLIC; C=US
  EMAILX400ADDR88 = CN=Smith; OU1=MEL; O=NOVLAN; P=EMSERVER; A=PUBLIC; C=US
  EMAILMICROSOFTADDR = NOVLAN/MEL/Smith
  EMAILMAILBUSADDR = 6=Smith@4=MEL@5=NOVLAN@3=EMSERVER@2=PUBLIC@1=US@NCX400@NODE1
  EMAILPCOUNTRY = US
  EMAILPRLOCATION = NJUSDCOA
  EMAILPRUSERNAME = Sm78168
  EMAILSNLOCATION = NACGWYR1
  EMAILSNUSERNAME = Sm46865
  EMAILADMDNAME = PUBLIC
  EMAILMHSMAILBOX = NCX400
  EMAILADDRPART = NODE1
  EMAILORGUNIT = MEL
  EMAILPRMDNAME = EMSERVER
  EMAILNAME = Byron Smith
  EMAILORGNAME = NOVLAN
  EMAILSURNAME = Smith
  EMAILUSE1 = X5Synch
  EMAILUSE5 = Smith-NOVLAN-MEL
  EMAILMHSUSERID = 6=Smith@4=MEL@5=NOVLAN@3=EMSERVER@2=PUBLIC@1=US
                        .
                        .
                        .
Microsoft Mail Import run ended at: Tue Feb 28 14:40:00 1995
```

Examining the log file shows that entries for the Microsoft Mail users were correctly added.

In Example 5–6, the second line of the log file indicates that a superior entry of /C=US/O=Digital/OU=MEL was created. This occurred because of the distinguished name sequence (DN_SEQUENCE) and CREATE_ROOT records specified in the configuration file. In this example, the DN_SEQUENCE specified that each entry should have a distinguished name of *country/organization/organizationalUnit/commonName* and, if need be, superior entries should be created as high as the organization level (CREATE_ROOT). Because the DIT did not contain a superior entry for the organizationalUnit, one was created by Synchronizer-500. This is necessary before any entries can be added to X.500 with the specified distinguished name sequence.

In Example 5–6, the third line represents the normal log output. The first term, A, indicates an ADD operation; the second term, NORMAL, shows that overall success of the indicated operation (NORMAL or FAILED). The third term, /C=US/O=DIGITAL..., is the distinguished name for the entry that was added. The last term, status, shows the status for a transaction. See Appendix G for a list of transaction status codes.

The remainder of the log file in Example 5–6 is a dump of the attributes for the entry and is only shown when VERBOSE=YES. VERBOSE mode performs a formatted listing of all data being entered for a particular directory entry. Each attribute has its own line, and each line displays a label that describes the X.500 attribute being added and the data value itself. VERBOSE mode is set in the configuration file. To enable VERBOSE mode, enter YES. Leave the field blank or enter NO to disable VERBOSE mode.

Example 5–7 shows an X.500 entry created as a result of the import.

Example 5–7: X.500 Entry for Microsoft Mail Users

```
dxim> show /C=US/O=digital/OU=MEL/CN="Byron Smith"

    /C=US/O=digital/OU=MEL/CN="Byron Smith"
      Common Name    = Byron Smith
      Object Class   = Person
                     = Organizational Person
                     = emailInfo
      Surname        = Smith
      Description    = MICROSOFT
      emailX400Addr84 = S=Smith; OU1=MEL; O=NOVLAN; P=EMSERVER; A=PUBLIC; C=US
      emailX400Addr88 = CN=Smith; OU1=MEL; O=NOVLAN; P=EMSERVER; A=PUBLIC; C=US
      emailMicrosoftAddr = NOVLAN/MEL/Smith
      emailMAILbusAddr = 6=Smith@4=MEL@5=NOVLAN@3=EMSERVER@2=PUBLIC
                         @1=US@NCX400@NODE1
      emailPCOUNTRY  = US
      emailPRLOCATION = NJUSDCOA
      emailPRUSERNAME = Sm63560
      emailSNLOCATION = NACGWYR1
      emailSNUSERNAME = Sm26049
      emailADMDNAME  = PUBLIC
      emailMHSMAILBOX = NCX400
      emailADDRPART  = NODE1
      emailORGUNIT   = MEL
      emailPRMDNAME  = EMSERVER
      emailNAME      = Byron Smith
      emailORGNAME   = NOVLAN
      emailSURNAME   = Smith
      emailUSE1      = X5Synch
      emailUSE5      = Smith-NOVLAN-MEL
      emailMHSUSERID = 6=Smith@4=MEL@5=NOVLAN@3=EMSERVER@2=PUBLIC@1=US
```

5.2.8 Running the Import Following the Initial Load

Once the initial load of Microsoft Mail users has been performed, each new run generates only the ADD/MODIFY/DELETE transactions needed to update X.500.

To set up Synchronizer-500 for subsequent import operations, the configuration file flag, INITIAL_LOAD, must be changed from YES to NO. Nothing else in the configuration file changes; the RDF remains the same.

When the import operation occurs, the entries in the import file are compared with the Microsoft Mail entries in X.500 identified by the Group ID (gid_match=MICROSOFT). From that comparison, ADD, MODIFY, and DELETE transactions, as appropriate, are generated to update X.500.

Example 5–8 shows the processing that results from the import update run. In this example, the entry for Black is deleted; the entry for Blue is modified; and a new entry, Reed, is added. Following the import run, the log file should be checked, as was done with the initial load run.

Example 5–8: Second Import Run Processing

```
Feb 28 15:14:23 processing layout file [-]XDSU_LAYOUT.DAT
Feb 28 15:14:24 processing configuration file XDSU_CONFIG.DAT
Feb 28 15:14:24 sending mail message to SYSTEM
Feb 28 15:14:25 initializing and binding to X.500
Feb 28 15:14:26 processing RDF file xdsu_import_microsoft.rdf
Feb 28 15:14:26 converting input file xdsu_import_microsoft.txt to new metafile
Feb 28 15:14:26 6 input records read
Feb 28 15:14:26 6 new metafile records written
Feb 28 15:14:26 sorting new metafile
Feb 28 15:14:27 searching X.500 directory data
    SEARCH  -
     where   description = MICROSOFT -
       and     EMAILUSE1 = X5Synch -
       and    COMMONNAME = *
Feb 28 15:14:27 found 6 X.500 directory entries
Feb 28 15:14:27 extracting X.500 directory data to old metafile
Feb 28 15:14:27 extracted 6 directory entries into old metafile
Feb 28 15:14:27 sorting old metafile
Feb 28 15:14:28 comparing old and new metafiles
Feb 28 15:14:28 processing DELETE records
    DELETE /C=US/O=DIGITAL/OU=MEL/CN=JANE BLACK
    DELETE /C=US/O=DIGITAL/OU=MEL/CN=SUSAN BLUE
Feb 28 15:14:28 2 records deleted
Feb 28 15:14:28 processing MODIFY records
Feb 28 15:14:28 0 records modified
Feb 28 15:14:28 processing ADD records
    ADD /C=US/O=DIGITAL/OU=MEL/CN=MARY REED
    ADD /C=US/O=DIGITAL/OU=MEL/CN=SUZANNE BLUE
Feb 28 15:14:29 2 records added
Feb 28 15:14:29 creating summary file
Feb 28 15:14:29 executing end run file xdsu_import_microsoft.com

XDSU V1.2-1
Microsoft Mail Import run started at: Tue Feb 28 15:14:23 1995

IMPORT SYNCH SUMMARY:

RECORDS READ/WRITTEN SUMMARY:

Foreign directory file records read: 6
New metafile records written: 6
Old metafile records written: 6

TRANSACTION SUMMARY:

Number of successful transactions: 4
Number of failed transactions:     0
Number of successful add transactions:    2
Number of successful modify transactions: 0
Number of successful delete transactions: 2
Number of failed add transactions:    0
Number of failed modify transactions: 0
Number of failed delete transactions: 0

CONFIGURATION FILE SUMMARY:
```

Example 5–8 Cont'd on next page

Example 5–8 (Cont.): Second Import Run Processing

```
synch_type: IMPORT
synch_label: Microsoft Mail Import
bind_dsa:
rdf_file: xdsu_import_microsoft.rdf
input_for_dir_file: xdsu_import_microsoft.txt
nrec_subscr: 1
log_file: import_microsoft.log
end_run_file: xdsu_import_microsoft.com
old_metafile:
new_metafile:
initial_load: N
notify: SYSTEM
verbose: Y
encode_delim: |
max_transactions:
max_fail_transactions:
max_trans_file:
dn_sequence: country/organization/organizationalUnit/commonName
create_root: organization
search_context:
synch_object: Person,organizationalPerson,emailInfo
synch_id: X5Synch
synch_id_field: emailUSE1
gid_field: description
gid_match: MICROSOFT
gen_profs_alias: Y
gen_snads_alias: Y
admd_space:
unique_field: emailUSE5
unique_field_len: 30
make_unique:
truncate:
unique_field_separator:
unique_search_context:

Microsoft Mail Import run ended at: Tue Feb 28 15:14:29 1995
Feb 28 15:14:30 sending mail message to SYSTEM
```

The transaction log and VERBOSE output for this run are shown in Example 5–9.

Note

In Example 5–9, the old common name (NAME) was Susan Blue. The modification is to make the new common name to be Suzanne Blue. The rest of the information is the same. However, since commonName is part of the distinguished name, the old entry has to be deleted and then the new entry is added.

Example 5–9: Second Import Run Transaction Log and VERBOSE Output

Example 5–9 Cont'd on next page

Example 5–9 (Cont.): Second Import Run Transaction Log and VERBOSE Output

```
Microsoft Mail Import run started at: Tue Feb 28 15:14:25 1995

D NORMAL /C=US/O=DIGITAL/OU=MEL/CN=JANE BLACK  - status=SUCCESS

  SURNAME = Black
  DESCRIPTION = MICROSOFT
  EMAILX400ADDR84 = S=Black; OU1=MEL; O=NOVLAN; P=EMSERVER; A=PUBLIC; C=US
  EMAILX400ADDR88 = CN=Black; OU1=MEL; O=NOVLAN; P=EMSERVER; A=PUBLIC; C=US
  EMAILMICROSOFTADDR = NOVLAN/MEL/Black
  EMAILMAILBUSADDR = 6=Black@4=MEL@5=NOVLAN@3=EMSERVER@2=PUBLIC@1=US@NCX400@NODE1
  EMAILPCOUNTRY = US
  EMAILPRLOCATION = NJUSDCOA
  EMAILPRUSERNAME = Bl46342
  EMAILSNLOCATION = NACGWYR1
  EMAILSNUSERNAME = Bl63815
  EMAILADMDNAME = PUBLIC
  EMAILMHSMAILBOX = NCX400
  EMAILADDRPART = NODE1
  EMAILORGUNIT = MEL
  EMAILPRMDNAME = EMSERVER
  EMAILNAME = Jane Black
  EMAILORGNAME = NOVLAN
  EMAILSURNAME = Black
  EMAILUSE1 = X5Synch
  EMAILUSE5 = Black-NOVLAN-MEL
  EMAILMHSUSERID = 6=Black@4=MEL@5=NOVLAN@3=EMSERVER@2=PUBLIC@1=US

D NORMAL /C=US/O=DIGITAL/OU=MEL/CN=SUSAN BLUE  - status=SUCCESS

  SURNAME = Blue
  DESCRIPTION = MICROSOFT
  EMAILX400ADDR84 = S=Blue; OU1=MEL; O=NOVLAN; P=EMSERVER; A=PUBLIC; C=US
  EMAILX400ADDR88 = CN=Blue; OU1=MEL; O=NOVLAN; P=EMSERVER; A=PUBLIC; C=US
  EMAILMICROSOFTADDR = NOVLAN/MEL/Blue
  EMAILMAILBUSADDR = 6=Blue@4=MEL@5=NOVLAN@3=EMSERVER@2=PUBLIC@1=US@NCX400@NODE1
  EMAILPCOUNTRY = US
  EMAILPRLOCATION = NJUSDCOA
  EMAILPRUSERNAME = Bl45214
  EMAILSNLOCATION = NACGWYR1
  EMAILSNUSERNAME = Bl04831
  EMAILADMDNAME = PUBLIC
  EMAILMHSMAILBOX = NCX400
  EMAILADDRPART = NODE1
  EMAILORGUNIT = MEL
  EMAILPRMDNAME = EMSERVER
  EMAILNAME = Susan Blue
  EMAILORGNAME = NOVLAN
  EMAILSURNAME = Blue
  EMAILUSE1 = X5Synch
  EMAILUSE5 = Blue-NOVLAN-MEL
  EMAILMHSUSERID = 6=Blue@4=MEL@5=NOVLAN@3=EMSERVER@2=PUBLIC@1=US

A NORMAL /C=US/O=DIGITAL/OU=MEL/CN=MARY REED  - status=SUCCESS
```

Example 5–9 Cont'd on next page

Example 5–9 (Cont.): Second Import Run Transaction Log and VERBOSE Output

```
  SURNAME = Reed
  DESCRIPTION = MICROSOFT
  EMAILX400ADDR84 = S=Reed; OU1=MEL; O=NOVLAN; P=EMSERVER; A=PUBLIC; C=US
  EMAILX400ADDR88 = CN=Reed; OU1=MEL; O=NOVLAN; P=EMSERVER; A=PUBLIC; C=US
  EMAILMICROSOFTADDR = NOVLAN/MEL/Reed
  EMAILMAILBUSADDR = 6=Reed@4=MEL@5=NOVLAN@3=EMSERVER@2=PUBLIC@1=US@NCX400@NODE1
  EMAILPCOUNTRY = US
  EMAILPRLOCATION = NJUSDCOA
  EMAILPRUSERNAME = Re70652
  EMAILSNLOCATION = NACGWYR1
  EMAILSNUSERNAME = Re53573
  EMAILADMDNAME = PUBLIC
  EMAILMHSMAILBOX = NCX400
  EMAILADDRPART = NODE1
  EMAILORGUNIT = MEL
  EMAILPRMDNAME = EMSERVER
  EMAILNAME = Mary Reed
  EMAILORGNAME = NOVLAN
  EMAILSURNAME = Reed
  EMAILUSE1 = X5Synch
  EMAILUSE5 = Reed-NOVLAN-MEL
  EMAILMHSUSERID = 6=Reed@4=MEL@5=NOVLAN@3=EMSERVER@2=PUBLIC@1=US

A NORMAL /C=US/O=DIGITAL/OU=MEL/CN=SUZANNE BLUE  - status=SUCCESS

  SURNAME = Blue
  DESCRIPTION = MICROSOFT
  EMAILX400ADDR84 = S=Blue; OU1=MEL; O=NOVLAN; P=EMSERVER; A=PUBLIC; C=US
  EMAILX400ADDR88 = CN=Blue; OU1=MEL; O=NOVLAN; P=EMSERVER; A=PUBLIC; C=US
  EMAILMICROSOFTADDR = NOVLAN/MEL/Blue
  EMAILMAILBUSADDR = 6=Blue@4=MEL@5=NOVLAN@3=EMSERVER@2=PUBLIC@1=US@NCX400@NODE1
  EMAILPCOUNTRY = US
  EMAILPRLOCATION = NJUSDCOA
  EMAILPRUSERNAME = Bl49530
  EMAILSNLOCATION = NACGWYR1
  EMAILSNUSERNAME = Bl76619
  EMAILADMDNAME = PUBLIC
  EMAILMHSMAILBOX = NCX400
  EMAILADDRPART = NODE1
  EMAILORGUNIT = MEL
  EMAILPRMDNAME = EMSERVER
  EMAILNAME = Suzanne Blue
  EMAILORGNAME = NOVLAN
  EMAILSURNAME = Blue
  EMAILUSE1 = X5Synch
  EMAILUSE5 = Blue-NOVLAN-MEL
  EMAILMHSUSERID = 6=Blue@4=MEL@5=NOVLAN@3=EMSERVER@2=PUBLIC@1=US

Microsoft Mail Import run ended at: Tue Feb 28 15:14:31 1995
```

In this example, because modification of the data caused the value of the distinguished name to change, a MODIFY transaction did not occur. If an attribute value changes that is not part of the distinguished name than a MODIFY transaction occurs. When Synchronizer-500 performs a MODIFY transaction, both the old and new information for the modified entry is written to the log file.

5.3 Export Process

The next step in the example is to perform an export operation. The purpose is to export a copy of selected portions of X.500 in Microsoft Mail import format so that the copy can serve as an import initial load into the Microsoft Mail directory.

The following is a broad outline of the steps needed to export X.500 information to be imported into Microsoft Mail directory:

1. Create a Synchronizer-500 subdirectory for this instance.
2. Determine the format of the file to be exported to Microsoft Mail and decide how to map the X.500 attributes into this format.
3. Write the RDF to reflect this data correspondence.
4. Set up the configuration file with the proper parameters.
5. Run Synchronizer-500 to export X.500 information to files that are compatible with the Microsoft Mail Import Utility.
6. Transfer these export files to the Microsoft Mail Directory Administrator via mail message.

5.3.1 Create the Microsoft Mail Export Subdirectory

As described in Section 3.3.1, it is recommended that a subdirectory to the Synchronizer-500 root directory be created. This subdirectory would contain the configuration file, RDF, and output data file for the Microsoft Mail export.

When Synchronizer-500 was installed, a command procedure was placed in the Synchronizer-500 root directory which will create the subdirectory and copy the appropriate configuration file template to that newly created directory. This procedure is executed by the following steps:

1. Set your current directory to the Synchronizer-500 root directory.
2. Execute the xdsu_config_instance procedure:

```
OpenVMS:
$ @XDSU_CONFIG_INSTANCE

UNIX:
#xdsu_config_instance
```

3. The procedure prompts:

```
* Enter XDSU instance directory name:
```

   Enter the name of the directory you wish to create. For example, to create a subdirectory to be used for exporting entries from X.500 in Microsoft Mail format, you might enter MSM_EXPORT.

4. The procedure then prompts:

```
* Enter Synch Type (Import, Export, Changes, or Transaction):
```

   Enter **E** since this is an export instance.

5. The procedure then creates the directory and copies the export configuration template file to the newly created directory.

6. Set your current directory to the newly created directory and proceed with setting up the configuration file and RDF files as described below.

5.3.2 Determine Format of the Microsoft Mail Export File

The format of the output of the Synchronizer-500 export run must be Microsoft Mail Import format (i.e., operation, alias, physical address). In addition:

- Because this run is the initial load of data from X.500 into Microsoft Mail, all transactions will be ADDs (A).
- In this example environment, an attempt has been made during all Synchronizer-500 import runs (from Microsoft Mail and other sources) to use the X.500 attribute, emailUSE5, as a unique alias field. Therefore, if the attribute emailUSE5 contains a value, Synchronizer-500 will map this value to the Microsoft Mail alias field. For this example, if emailUSE5 does not exist for a X.500 entry, an alias using the emailSURNAME, emailGIVENNAME, and emailORGUNIT attributes will be created.
- Since all backbone users have their Microsoft Mail physical addresses expressed as X.400 addresses, the X.400 GDI information from X.500 must be provided. The X.500 fields–emailGIVENNAME, emailINITIALS, emailSURNAME, emailPCOUNTRY, emailADMDNAME, emailPRMDNAME, emailORGNAME, and emailORGUNIT–will be mapped to the Microsoft Mail physical address field.

The mapping of the X.500-to-Microsoft Mail import file will be:

```
MSMail import
   Field      Start-End   Length     X.500 Attributes
-------------------------------------------------------------------------

"A"             1 -   2       2
Alias           3 -  32      30      emailUSE5 or
                                     emailSURNAME-emailGIVENNAME-emailORGUNIT
Physical       33 - 536     504      emailGIVENNAME
address                              emailINITIALS
                                     emailSURNAME
                                     emailPCOUNTRY
                                     emailADMDNAME
                                     emailPRMDNAME
                                     emailORGNAME
                                     emailORGUNIT1-4
```

5.3.3 Writing the Export RDF

Using the mapping above, the RDF (xdsu_export_microsoft.rdf) is created to tell Synchronizer-500 which X.500 attributes to map to the output file and where to put that information. (xdsu_export_microsoft.rdf is included in Synchronizer-500's root directory and should be referred to while reading the following text.)

The use of emailUSE5 as a unique X.500 attribute was chosen for the examples written for Synchronizer-500.

As explained in Chapter 3, an RDF record may contain a conditional clause. These conditional clauses test for the existence or non-existence of data. Because the existence of the X.500 attributes needed cannot be guaranteed, the example uses several conditional statements.

For the export process to begin, the Microsoft Mail Import Utility must be told that this is an ADD operation. This instruction should be anchored in column 1:

```
"A"=1.1.#
```

The next step is to specify an alias string that is displayed when the Microsoft Mail user pulls up the X.400 recipient list. In this example environment, if Synchronizer-500 was used to load the directory, then an attempt was made to have the attribute emailUSE5 contain a unique alias for the X.500 entry.

The following statement in the example RDF tests for the existence of data in the X.500 emailUSE5 attribute. Synchronizer-500 will execute the RDF statement (put the data from the X.500 attribute to the output file) only if there is data in the emailUSE5 attribute.

```
?emailUSE5:emailUSE5=1.*.#
```

If, for whatever reason, emailUSE5 is empty, an alias, which should be unique, must be algorithmically generated by concatenating the emailSURNAME, emailGIVENNAME, and emailORGUNIT as a best approximation for a unique alias value. The dash - character should be used to delimit these values:

```
~emailUSE5:emailSURNAME=1.*.#
~emailUSE5:"-"=1.*.#
~emailUSE5:emailGIVENNAME=1.*.#
~emailUSE5:"-"=1.*.#
~emailUSE5:emailORGUNIT=1.*.#
```

Note

Uniqueness is not guaranteed by using this concatenation method.

All the above RDF statements are prefaced with the ~emailUSE5 conditional statement. Conditional statements prefaced by the tilde ~ character test for the non-existence of data, in this case, non-existence of data in the X.500 emailUSE5 attribute. When it finds an attribute prefaced by the tilde ~ character, Synchronizer-500 executes the RDF statement (put the data from another X.500 attribute or string constant to the output file) only if no data exists in the X.500 attribute being tested.

The physical address is created next. It should be anchored in column 33, which is the starting column for the physical address field.

The first element of the physical address is the personal name. For the personal name, GIVENNAME and INITIALS are optional, but SURNAME is mandatory. The personal name can be constructed as GIVENNAME.INITIALS.SURNAME, GIVENNAME.SURNAME, or just SURNAME.

To begin construction of the personal name, the string, /pn=, is placed into the output stream:

```
" /pn="=1.33.#
```

Conditional statements prefaced by the question mark ? character test for the existence of data. When it finds an attribute prefaced by the question mark ? character, Synchronizer-500 executes the RDF statement (put the data from an X.500 attribute or string constant to the output file) only if data exists in the attribute being tested.

The question mark ? character should be used to search for the existence of emailGIVEN-NAME before putting it in the output stream. If it does exist, the period . delimiter should be placed in the output stream:

```
?emailGIVENNAME:emailGIVENNAME=1.*.#
?emailGIVENNAME:"."=1.*.#
```

The same procedure as noted for emailGIVENNAME should be performed for emaillINITIALS:

```
?emailINITIALS:emailINITIALS=1.*.#
?emailINITIALS:"."=1.*.#
```

Because emailSURNAME is mandatory, no conditional syntax is used in this RDF:

```
emailSURNAME=1.*.#
```

Synchronizer-500 allows the option for up to four emailORGUNITs to an entry. For this reason, the RDF for ORGUNIT must include conditional syntax. Use the question mark ? character to test for the existence of a emailORGUNITn value before putting it in the output stream. If an emailORGUNIT(s) value exists, it should be prefaced with the string, /ou=:

```
?emailORGUNIT:"/ou="=1.*.#
?emailORGUNIT:emailORGUNIT=1.*.#
?emailORGUNIT2:"/ou="=1.*.#
?emailORGUNIT2:emailORGUNIT2=1.*.#
?emailORGUNIT3:"/ou="=1.*.#
?emailORGUNIT3:emailORGUNIT3=1.*.#
?emailORGUNIT4:"/ou="=1.*.#
?emailORGUNIT4:emailORGUNIT4=1.*.#
```

emailORGNAME is also optional. The question mark ? character should be used to test for its existence before putting it in the output stream. If the emailORGNAME attribute exists, the string, /o=, should be placed in the output stream:

```
?emailORGNAME:"/o="=1.*.#
?emailORGNAME:emailORGNAME=1.*.#
```

The X.400 GDIs–COUNTRY, ADMD, and PRMD–are mandatory in an X.400 address. However, these GDIs may not have been entered in X.500 for local users. The RDF for these terms is constructed to use these values, if they exist. If values do not exist, defaults will be supplied via string-constant RDF records. The RDF assumes that if COUNTRY is specified then the GDIs are fully specified. If COUNTRY is not specified, then conditional RDFs will supply the defaults for all three terms.

The first step is to use the question mark ? to search for the existence of emailPRMDNAME, emailADMDNAME, and emailPCOUNTRY:

```
?emailPRMDNAME:"/PRMD="=1.*.#
?emailPRMDNAME:emailPRMDNAME=1.*.#
?emailADMDNAME:"/ADMD="=1.*.#
?emailADMDNAME:emailADMDNAME=1.*.#
?emailPCOUNTRY:"/c="=1.*.#
?emailPCOUNTRY:emailPCOUNTRY=1.*.#
```

The next step is to use the tilde ~ character to test for a value for COUNTRY. This RDF is constructed to assume that if there is no value for the X.500 attribute COUNTRY then there are no other GDIs and defaults are used for ADMD and PRMD:

```
~emailPCOUNTRY:"/PRMD=EMSERVER"=1.*.#
~emailPCOUNTRY:"/ADMD=PUBLIC"=1.*.#
~emailPCOUNTRY:"/c=US"=1.*.#
```

5.3.4 Setting up the Export Configuration File

The configuration file (xdsu_config.dat) for this example is set up as shown in Example 5–10. This sample file (xdsu_config.dat_export_microsoft) is included in the Synchronizer-500 root directory and should be referred to while reading the following text.

Example 5–10: Configuration File for Export

```
synch_type=EXPORT
!
bind_dsa=
search_context=
!
rdf_file=xdsu_export_microsoft.rdf
input_old_metafile=
output_old_metafile=export_microsoft.old
output_for_dir_file=export_microsoft.dat
nrec_subscr=1
gid_field=description
gid_match=*
gid_exclude=MICROSOFT
country_match=
admd_match=
prmd_match=
orgname_match=
orgunit_match=
orgunit2_match=
orgunit3_match=
orgunit4_match=
end_run_file=xdsu_export_microsoft.com
notify=SYSTEM
synch_id=X5Synch
synch_label=Microsoft_Mail_export
get_field=
```

This file is an example of a configuration file for an export synchronization. The example RDF xdsu_export_microsoft.rdf and example end_run_file xdsu_export_microsoft.com are used. (These files are contained in the Synchronizer-500 root directory.)

The export configuration file consists of keyword and value pairs. In most cases, not all keywords will be used. Unused keywords can remain blank or be removed from the configuration file. Descriptions of each keyword and example values are given below. Please see Appendix F for more detailed information.

- *synch_type* must specify the type of synchronization being performed. In this example it is an **EXPORT**.
- *bind_dsa* specifies a DSA presentation address. This feature is not enabled at this time. *Reserved for future use.*
- *search_context* specifies the part of the naming context Synchronizer-500 should limit its searches to. In this example, no search context is given. Specifying "search_context= /c=US/o=Digital" would limit the search tree and could be more efficient.
- *rdf_file* must specify the file name of the RDF used for this export. In this example, the **xdsu_export_microsoft.rdf** file is used.
- *input_old_metafile* allows a previous extract of X.500 to be used by specifying the file name of the old metafile.
- *output_old_metafile* permits saving the metafile by specifying the file name for it. In this example, the old metafile is saved in the file **export_microsoft.old**. This is necessary for use in the changes operation portion of this example. (See Section 5.4.)
- *output_for_dir_file* must specify the file name to which to write the exported X.500 data. In this example, the **export_microsoft.dat** file will contain the exported X.500 data.
- *nrec_subscr* specifies how many records per entry are written to the output data file. In this example, there is only **1** record written per entry.
- *gid_field* must specify the X.500 attribute in which the Group ID value is stored. In this example, **DESCRIPTION**, is used to store the Synchronizer-500 Group ID.
- *gid_match* specifies the Group ID value to export. In this example, all Group IDs are exported.
- *gid_exclude* specifies a Group ID value to exclude from the export. In this example, the Group ID of **MICROSOFT** is excluded from the export since the Microsoft users should not be included in the output file.
- *country_match, admd_match, prmd_match, orgname_match, orgunit_match, orgunit2_ match, orgunit3_match, orgunit4_match* are used to further qualify the entries to be exported from X.500.
- *end_run_file* specifies the file name of a user-written procedure that is executed upon termination of this Synchronizer-500 run. In this example, the command procedure **xdsu_ export_microsoft.com** is executed.
- *notify* specifies the electronic mail userid to receive status messages. In this example, **SYSTEM** receives all status messages.
- *synch_id* specifies the tag which indicates that Synchronizer-500 "owns" the entry. In this example, the value of **X5Synch** is used.
- *synch_label* specifies the label for Synchronizer-500 mail messages and end-of-run synopses. In this example, the label **Microsoft_Mail_Export** is used.

- *get_field* and *get_value* are used together. These keywords allow a group of entries to be exported from X.500 based upon a value in a specific attribute.

5.3.5 Running the Export

Synchronizer-500 is started by issuing the following command from the directory where the configuration, RDF files reside:

```
OpenVMS:

$ XDSU :== $SYS$SYSTEM:XDSU$MAIN.EXE
$ XDSU

UNIX:

/usr/etc/xdsu
```

As Synchronizer-500 executes, its processing is logged to your terminal device as shown in Example 5–11. The processing should be examined as it occurs to review the results of the export run. When the export run finishes, a mail message containing a summary of the run is sent to the address specified by the configuration file parameter "notify=".

Example 5–11: Microsoft Mail Export Processing

```
Feb 28 15:40:46 processing layout file [-]XDSU_LAYOUT.DAT
Feb 28 15:40:46 processing configuration file XDSU_CONFIG.DAT
Feb 28 15:40:46 sending mail message to SYSTEM
Feb 28 15:40:47 initializing and binding to X.500
Feb 28 15:40:48 processing RDF file xdsu_export_microsoft.rdf
Feb 28 15:40:48 searching X.500 directory data
Feb 28 15:40:48 found 50 X.500 directory entries
Feb 28 15:40:48 extracting X.500 directory data to export metafile
Feb 28 15:40:51 extracted 50 directory entries into export metafile
Feb 28 15:40:51 sorting export metafile
Feb 28 15:40:52 converting metafile to output file export_microsoft.dat
Feb 28 15:40:52 50 metafile records read
Feb 28 15:40:52 50 output file records written
Feb 28 15:40:52 creating summary file
Feb 28 15:40:52 executing end run file xdsu_export_microsoft.com

XDSU V1.2-1
Microsoft_Mail_export run started at: Tue Feb 28 15:40:46 1995

EXPORT SYNCH SUMMARY:

RECORDS READ/WRITTEN SUMMARY:

Old metafile records written: 50
Old metafile records read:    50
Foreign directory file records written: 50

CONFIGURATION FILE SUMMARY:
```

Example 5–11 Cont'd on next page

Example 5–11 (Cont.): Microsoft Mail Export Processing

```
synch_type: EXPORT
synch_label: Microsoft_Mail_export
bind_dsa:
rdf_file: xdsu_export_microsoft.rdf
output_for_dir_file: export_microsoft.dat
nrec_subscr: 1
end_run_file: xdsu_export_microsoft.com
output_old_metafile: export_microsoft.old
input_old_metafile:
notify: SYSTEM
verbose:
encode_delim:
search_context:
synch_id: X5Synch
synch_id_field:
gid_field: description
gid_match: *
gid_exclude: MICROSOFT
country_match:
admd_match:
prmd_match:
orgname_match:
orgunit_match:
orgunit2_match:
orgunit3_match:
orgunit4_match:
get_field:
get_value:

Microsoft_Mail_export run ended at: Tue Feb 28 15:40:52 1995
Feb 28 15:40:53 sending mail message to SYSTEM
```

Example 5–12 shows part of the output file, export_microsoft.dat, that was generated from the export run. This is the file that should be examined to see if information has been extracted from X.500 and formatted correctly.

Example 5–12: Exported X.500 Data in Microsoft Mail Import Format

```
A ADAMS-MARILYN-ADAMSMA          /pn=MARILYN.ADAMS/ou=USROV/o=Digital
                                        /PRMD=EMSERVER/ADMD=PUBLIC/c=US
A ABEAR-MARK-ABEARMA             /pn=MARK.ABEAR/ou=USDCA/o=Digital
                                        /PRMD=EMSERVER/ADMD=PUBLIC/c=US
A ADAMS-DONNA-ADAMSDO            /pn=DONNA.ADAMS/ou=USDCO/o=Digital
                                        /PRMD=EMSERVER/ADMD=PUBLIC/c=US
A ADAMS-TIMOTHY-ADAMSTI          /pn=TIMOTHY.ADAMS/ou=USDCO/o=Digital
                                        /PRMD=EMSERVER/ADMD=PUBLIC/c=US
A ABRAMS-ANN-ABRAMAN             /pn=ANN.ABRAMS/ou=USDCO/o=Digital
                                        /PRMD=EMSERVER/ADMD=PUBLIC/c=US
A ADAMS-TOM-ADAMSTO              /pn=TOM.ADAMS/ou=USFOR/o=Digital
                                        /PRMD=EMSERVER/ADMD=PUBLIC/c=US
A ADAMS-TAMMY-ADAMSTA            /pn=TAMMY.ADAMS/ou=USCOL/o=Digital
                                        /PRMD=EMSERVER/ADMD=PUBLIC/c=US
```

Note

In Example 5–12, the /pn values are wrapped to the second line for ease of reading. The /pn values would normally be on the same line in the output file.

5.3.6 Transfer the Synchronizer-500 Export Output File

The final step in the export process is to transfer the output file to the Microsoft Mail Directory Administrator so that is can be merged into the Microsoft Mail directory. Electronic mail or a file transfer mechanism can be used as the transport for moving the output file. The end_run_file, xdsu_export_microsoft.com, can include the steps for the transfer process so it automatically occurs at the end of the export run.

5.4 Changes Process

The next step in the example is to perform a changes operation. In this operation, the Synchronizer-500 run provides two files (ADD and DELETE) that specify the changes that have occurred to X.500 since the last changes run was made. (In the case of the first changes run, the changes are those that have occurred since the last export run). The ADD and DELETE files are in Microsoft Mail import format and are used to update the Microsoft Mail directory, which was initially loaded with the export file from the previous section.

The following is a broad outline of the steps performed in providing changes X.500 information for the Microsoft Mail directory:

1. Create a Synchronizer-500 subdirectory for this instance.
2. Determine the format of the files to be exported to Microsoft Mail and decide how to map the X.500 attributes into this format.
3. Write the RDF to reflect this correspondence.
4. Set up the configuration file with the proper parameters.
5. Run Synchronizer-500 to export X.500 information to files that are compatible for the Microsoft Mail import file utility.
6. Transfer these files to the Microsoft Mail Directory Administrator via mail message.

Much of the analysis work done during the export process is the same in the changes process. Therefore, rather than repeat this information, references to the applicable section under the export discussion are provided.

5.4.1 Create the Microsoft Mail Changes Subdirectory

See Section 5.3.1 for the instructions on how to run the procedure to create a subdirectory for the Microsoft Mail Changes run.

5.4.2 Determine Format of the Microsoft Mail Export File

See section Section 5.3.2 for the analysis of this file. The only differences from the initial export operation are:

1. ADD transactions are implemented as A
2. DELETE transactions are implemented as D
3. MODIFY transactions are implemented as DELETE and ADD transactions. A record that contains the old, unmodified record is written to the output DELETE file. Another record is written to the output ADD file containing the new, modified record.

Note

It is important that the DELETE file be applied against the Microsoft Mail directory *before* the ADD file. If this sequence is not observed, unpredictable results may occur.

5.4.3 Writing the Changes RDFs

The files xdsu_changes_microsoft_add.rdf and xdsu_changes_microsoft_delete.rdf were created to tell Synchronizer-500 the mapping of X.500 attributes to their respective output files. (These RDFs are included in Synchronizer-500's root directory and should be referred to while reading the following text.)

For more information on xdsu_changes_microsoft_add.rdf, see the description in Section 5.3.3 of the export RDF file. The only difference between the export RDF and the changes RDF is the name.

For more information on xdsu_changes_microsoft_delete.rdf, see the description of the export RDF file in Section 5.3.3. The only difference between the _ADD and the _DELETE RDF is the use of the DELETE (D) identifier:

```
"D"=1.1.#
```

5.4.4 Setting up the Changes Configuration File

The configuration file (xdsu_config.dat) for this example is set up using the format shown in Example 5–13. This file is included in the Synchronizer-500 root directory and should be referred to while reading the following text.

Example 5–13: Configuration File for Changes

```
synch_type=CHANGES
!
bind_dsa=
search_context=/country=us/o=Digital
!
add_rdf_file=xdsu_changes_microsoft_add.rdf
mod_rdf_file=
del_rdf_file=xdsu_changes_microsoft_delete.rdf
input_old_metafile=export_microsoft.old
output_new_metafile=export_microsoft.new
output_add_file=changes_microsoft_add.txt
output_mod_file=
output_del_file=changes_microsoft_delete.txt
nrec_subscr_add=1
nrec_subscr_mod=
nrec_subscr_del=1
gid_field=description
gid_match=*
gid_exclude=MICROSOFT
country_match=
admd_match=
prmd_match=
orgname_match=
orgunit_match=
orgunit2_match=
orgunit3_match=
orgunit4_match=
end_run_file=xdsu_changes_microsoft.com
notify=SYSTEM
synch_id=X5Synch
synch_label=Microsoft_Changes_Updates
input_new_metafile=
get_field=
get_value=
```

This file is an example of a configuration file for a changes export synchronization. The example RDFs xdsu_changes_microsoft_add.rdf and xdsu_changes_microsoft_delete.rdf, and example end_run_file xdsu_changes_microsoft.com are used. (These files are contained in the Synchronizer-500 root directory.)

The changes configuration file is very similar to the export conifiguration file described in Section 5.3.4. The following keywords are different for the changes configuration file.

- *synch_type* must specify the type of synchronization being performed. In this example it is a **CHANGES** operation.
- *add_rdf_file* must specify the file name of the RDF to be used for the ADD output foreign directory file. In this example, the **xdsu_changes_microsoft_add.rdf** file is used.
- *mod_rdf_file* must specify the file name of the RDF to be used for the MODIFY output foreign directory file. In this example, there is no modify RDF. Modify transactions will be written as DELETE and ADD records.

- *del_rdf_file* must specify the file name of the RDF to be used for the DELETE output foreign directory file. In this example, the **xdsu_changes_microsoft_delete.rdf** file is used.
- *input_old_metafile* allows a previous extract of X.500 to be used by specifying the file name of the old metafile. In this example, the file **export_microsoft.old** is used as the old metafile. Synchronizer-500 uses this file to compare extracted X.500 information to determine the changes that have occurred to X.500 since the last changes run. The file used for the first changes run is the output metafile from the export run done previously (export OUTPUT_OLD_METAFILE). (See Section 5.3.) All subsequent changes runs use the OUTPUT_NEW_METAFILE of the previous Synchronizer-500 changes run.
- *output_new_metafile* permits saving the new metafile by specifying the file name for it. In this example, the new metafile is saved in the file **export_microsoft.new**. This new metafile is used as the OUTPUT_OLD_METAFILE in subsequent changes runs by being renamed by the end-of-run command procedure to .old.
- *output_add_file* must specify the file name to which to write the ADD X.500 data. In this example, the **changes_microsoft_add.txt** file will contain the ADD X.500 data.
- *output_mod_file* must specify the file name to which to write the MODIFY X.500 data. In this example, a MODIFY output file is not created. MODIFY transactions are implemented as DELETEs and ADDs.
- *output_del_file* must specify the file name to which to write the DELETE X.500 data. In this example, the **changes_microsoft_delete.txt** file will contain the DELETE X.500 data.
- *nrec_subscr_add* specifies how many records per entry are written to the ADD output data file. In this example, there is only **1** record written per ADD entry.
- *nrec_subscr_mod* specifies how many records per entry are written to the MODIFY output data file. In this example, there is no MODIFY output file created. MODIFY transactions are implemented as DELETEs and ADDs.
- *nrec_subscr_del* specifies how many records per entry are written to the DELETE output data file. In this example, there is only **1** record written per DELETE entry.
- *end_run_file* specifies the file name of a user-written procedure that is executed upon termination of this Synchronizer-500 run. In this example, the command procedure **xdsu_changes_microsoft.com** is executed. The command procedure, xdsu_changes_ microsoft.com, renames the .new metafile to .old for use in the next changes run.
- *synch_label* specifies the label for Synchronizer-500 mail messages and end-of-run synopses. In this example, the label **Microsoft_Changes_Updates** is used.

5.4.5 Running the Changes Export

Synchronizer-500 is started by issuing the following command from the directory where the configuration and RDF files reside:

```
OpenVMS:

$ XDSU :== $SYS$SYSTEM:XDSU$MAIN.EXE
$ XDSU

UNIX:

/usr/etc/xdsu
```

As Synchronizer-500 executes, its processing is logged to your terminal device as shown in Example 5–14. The processing should be examined as it occurs to review the results of the changes run. When the changes run finishes, a mail message containing a summary of the run is sent to the address specified by the configuration file parameter "notify=".

Example 5–14: Microsoft Mail Changes Processing

```
Feb 28 15:59:31 processing layout file [-]XDSU_LAYOUT.DAT
Feb 28 15:59:31 processing configuration file XDSU_CONFIG.DAT
Feb 28 15:59:31 sending mail message to SYSTEM
Feb 28 15:59:32 initializing and binding to X.500
Feb 28 15:59:33 processing RDF file xdsu_changes_microsoft_delete.rdf
Feb 28 15:59:33 processing RDF file xdsu_changes_microsoft_add.rdf
Feb 28 15:59:33 searching X.500 directory data
Feb 28 15:59:33 found 50 X.500 directory entries
Feb 28 15:59:33 extracting X.500 directory data to new metafile
Feb 28 15:59:35 extracted 50 directory entries into new metafile
Feb 28 15:59:35 sorting new metafile
Feb 28 15:59:36 comparing old and new metafiles
Feb 28 15:59:36 processing DELETE records
Feb 28 15:59:36 0 metafile records used
Feb 28 15:59:36 0 output file records written to changes_microsoft_delete.txt
Feb 28 15:59:36 processing MODIFY records
Feb 28 15:59:37 1 metafile records used
Feb 28 15:59:37 1 output file records written to changes_microsoft_delete.txt
Feb 28 15:59:37 1 output file records written to changes_microsoft_add.txt
Feb 28 15:59:37 processing ADD records
Feb 28 15:59:37 0 metafile records used
Feb 28 15:59:37 0 output file records written to changes_microsoft_add.txt
Feb 28 15:59:37 creating summary file
Feb 28 15:59:37 executing end run file xdsu_changes_microsoft.com

XDSU V1.2-1
Microsoft_Changes_Updates run started at: Tue Feb 28 15:59:31 1995

CHANGES SYNCH SUMMARY:

RECORDS READ/WRITTEN SUMMARY:
```

Example 5–14 Cont'd on next page

Example 5–14 (Cont.): Microsoft Mail Changes Processing

```
Old metafile records read:    50
New metafile records written: 50
ADD metafile records used:    0
MOD metafile records used:    1
DEL metafile records used:    0
ADD foreign directory file records written: 1
MOD foreign directory file records written: 0
DEL foreign directory file records written: 1

CONFIGURATION FILE SUMMARY:

synch_type: CHANGES
synch_label: Microsoft_Changes_Updates
bind_dsa:
add_rdf_file: xdsu_changes_microsoft_add.rdf
mod_rdf_file:
del_rdf_file: xdsu_changes_microsoft_delete.rdf
output_add_file: changes_microsoft_add.txt
output_mod_file:
output_del_file: changes_microsoft_delete.txt
nrec_subscr_add: 1
nrec_subscr_mod:
nrec_subscr_del: 1
end_run_file: xdsu_changes_microsoft.com
input_old_metafile: export_microsoft.old
output_new_metafile: export_microsoft.new
input_new_metafile:
notify: SYSTEM
verbose:
encode_delim:
search_context: /country=us/o=Digital
synch_id: X5Synch
synch_id_field:
gid_field: description
gid_match: *
gid_exclude: MICROSOFT
country_match:
admd_match:
prmd_match:
orgname_match:
orgunit_match:
orgunit2_match:
orgunit3_match:
orgunit4_match:
get_field:
get_value:

Microsoft_Changes_Updates run ended at: Tue Feb 28 15:59:37 1995
Feb 28 15:59:38 sending mail message to SYSTEM
```

The two output file, changes_microsoft_add.txt and changes_microsoft_delete.txt, are generated from the changes run as shown in Example 5–15 and Example 5–16, respectively. An examination of these files shows that the information has been extracted from X.500 and formatted correctly.

Example 5–15: changes_microsoft_add.txt Output

```
A Edwards-NOVLAN-CYO              /pn=Edwards/ou=CYO/o=NOVLAN/PRMD=EMSERVER
                                         /ADMD=PUBLIC/c=US
A ADAMS-CHRISTINE-ADAMSCH         /pn=CHRISTINE.ADAMS/ou=USMEL/o=Digital
                                         /PRMD=EMSERVER/ADMD=PUBLIC/c=US
A Jones-NOVLAN-CYO                /pn=Jones/ou=CYO/o=NOVLAN/PRMD=EMSERVER
                                         /ADMD=PUBLIC/c=US
A Peterson-NOVLAN-CYO             /pn=Peterson/ou=CYO/o=NOVLAN/PRMD=EMSERVER
                                         /ADMD=PUBLIC/c=US
A ADKINS-TERRY-ADKINTE            /pn=TERRY.ADKINS/ou=USMEL/o=Digital
                                         /PRMD=EMSERVER/ADMD=PUBLIC/c=US
A ADDISON-VICKY-ADDISVI           /pn=VICKY.ADDISON/ou=USWNP/o=Digital
                                         /PRMD=EMSERVER/ADMD=PUBLIC/c=US
```

Example 5–16: changes_microsoft_delete.txt Output

```
D SMITHSON-USMEL-MEL              /pn=DAVE.SMITHSON/ou=USMEL/o=Digital
                                         /PRMD=EMSERVER/ADMD=PUBLIC/c=US
D MOORE-USMEL-MEL                 /pn=ALLEN.MOORE/ou=USMEL/o=Digital
                                         /PRMD=EMSERVER/ADMD=PUBLIC/c=US
D THOPMSON-USWNP-WNP              /pn=RICHARD.THOMPSON/ou=USWNP/o=Digital
                                         /PRMD=EMSERVER/ADMD=PUBLIC/c=US
```

Note

In Example 5–15 and Example 5–16, the /pn values are wrapped to the second line for ease of reading. The /pn values would normally be on the same line in the output files.

5.4.6 Transferring Synchronizer-500 Changes Output Files

The final step in the changes process is to transfer the output files to the Microsoft Mail Directory Administrator so that it can be merged into the Microsoft Mail directory. Electronic mail or a file transfer mechanism can be used as the transport for moving the output file. The end_run_file, xdsu_changes_microsoft.com, can include the steps for the transfer process so it automatically occurs at the end of the export run.

Appendix A
Sample DDS Synchronization

This appendix gives an example of how to synchronize export data from the MAILbus Distributed Directory Service (DDS) and perform an initial load import into X.500. All of the steps required are discussed in detail. The synchronization runs are shown, along with the configuration files, RDFs, log files, and results.

A.1 Import Process

The import of data from DDS into X.500 requires that data in DDS be written to a flat file for Synchronizer-500 to use. This example uses MAILbus Directory Synchronizer to export DDS data to a flat file. MAILbus Directory Synchronizer was used to synchronize foreign directories with DDS, so each DDS subscriber entry should already have a Group ID (GID) associated with it. This Group ID allows identification of the entries that will be exported from DDS and then imported into X.500.

The import of DDS data into the X.500 DIT requires that activities be performed on both the MAILbus system and the Synchronizer-500 system. These activities must be executed for each Group ID you wish to export from DDS and then subsequently import into X.500 so that entries will be properly identified.

The following is a broad outline of the steps that must be performed to import DDS information into X.500:

On the **MAILbus** system:

1. Choose an existing instance or set up a new instance subdirectory for the export.
2. Copy the sample RDF and configuration file to the MAILbus system and edit as appropriate.
3. Run MAILbus Directory Synchronizer to export entries with the specified Group ID from DDS.

On the **X.500** system:

1. Create a Synchronizer-500 subdirectory for the DDS import.
2. Transfer the DDS export file to the Synchronizer-500 system.

3. Set up the import configuration file.
4. Set up the import RDF file.
5. Run the import process to add the DDS entries to X.500.

The next sections describe these activities in detail.

A.1.1 MAILbus System Activities

The following sections describe the activities on the MAILbus system. Please refer to the MAILbus Directory Synchronizer documentation for more information.

A.1.1.1 Choose an Instance

The first step is to choose a MAILbus Directory Synchronizer instance from which to run the export operation. The location is a subdirectory of the MAILbus Directory Synchronizer directory tree.

For this example INSTANCE 03 was chosen, which uses the subdirectory [.DSU_03].

A.1.1.2 Copy the Sample RDF and Configuration File to the MAILbus System

Synchronizer-500 comes with a sample RDF and configuration file for the MAILbus Directory Synchronizer to export all fields in DDS with a specified Group ID. These sample files can be found in the Synchronizer-500 root directory with the names dsu_export_dds.rdf and dsu_config.dat_export_dds.

Follow these steps to put the sample files into place:

1. Copy the dsu_export_dds.rdf and dsu_config.dat_export_dds files from the Synchronizer-500 system to the [.DB] subdirectory of the instance chosen in Section A.1.1.1 on the MAILbus system.
2. Rename the dsu_config.dat_export_dds file to dsu_config.dat.
3. Edit dsu_config.dat and make site specific changes to the export configuration file. The following record values need to be changed: directory specification for RDF_FILE and OUTPUT_FOR_DIR_FILE, DDS_GID_USED, DDS_GID_MATCH, and DSU_ID. Please refer to Example A–2.

The sample RDF file used to export DDS data to a flat file is shown in Example A–1. The RDF, dsu_export_dds.rdf, will export ALL fields in DDS for each subscriber. One line per field is written to the output file. If data does not exist in a field, a blank is written to the output file.

Example A–1: RDF for DDS Export - dsu_export_dds.rdf

```
! DSU_EXPORT_DDS.RDF
! Export Synch for use by MAILbus Directory Synchronizer
! to generate a flat file of specified DDS users to be used as
! an import file into Digital X.500 Directory Synchronizer.
!
! 11-October-1993
!
!
!   RDF for export of DDS data to be imported into X.500
!   Export one DDS field per line
!
?ADDRPART:ADDRPART=1.1.#
~ADDRPART:" "=1.1.#
?ADDRPART2:ADDRPART2=2.1.#
~ADDRPART2:" "=2.1.#
?ADMDNAME:ADMDNAME=3.1.#
~ADMDNAME:" "=3.1.#
?DDA:DDA=4.1.#
~DDA:" "=4.1.#
?DDA2:DDA2=5.1.#
~DDA2:" "=5.1.#
?DDA3:DDA3=6.1.#
~DDA3:" "=6.1.#
?DDA4:DDA4=7.1.#
~DDA4:" "=7.1.#
?FFPOSTADDR:FFPOSTADDR=8.1.#
~FFPOSTADDR:" "=8.1.#
?FFPOSTADDR2:FFPOSTADDR2=9.1.#
~FFPOSTADDR2:" "=9.1.#
?FFPOSTADDR3:FFPOSTADDR3=10.1.#
~FFPOSTADDR3:" "=10.1.#
?FFPOSTADDR4:FFPOSTADDR4=11.1.#
~FFPOSTADDR4:" "=11.1.#
?GENERATION:GENERATION=12.1.#
~GENERATION:" "=12.1.#
?GIVENNAME:GIVENNAME=13.1.#
~GIVENNAME:" "=13.1.#
?GIVENNAME2:GIVENNAME2=14.1.#
~GIVENNAME2:" "=14.1.#
?INITIALS:INITIALS=15.1.#
~INITIALS:" "=15.1.#
?INTLOCATION:INTLOCATION=16.1.#
~INTLOCATION:" "=16.1.#
?INTPHONE:INTPHONE=17.1.#
~INTPHONE:" "=17.1.#
?LANGUAGE:LANGUAGE=18.1.#
~LANGUAGE:" "=18.1.#
?LOCATION:LOCATION=19.1.#
~LOCATION:" "=19.1.#
?MAILADDRESS:MAILADDRESS=20.1.#
~MAILADDRESS:" "=20.1.#
?MHSMAILBOX:MHSMAILBOX=21.1.#
~MHSMAILBOX:" "=21.1.#
?MHSMAILBOX2:MHSMAILBOX2=22.1.#
```

Example A–1 Cont'd on next page

Example A–1 (Cont.): RDF for DDS Export - dsu_export_dds.rdf

```
~MHSMAILBOX2:" "=22.1.#
?MHSUSERID:MHSUSERID=23.1.#
~MHSUSERID:" "=23.1.#
?MHSUSERID2:MHSUSERID2=24.1.#
~MHSUSERID2:" "=24.1.#
?MRX_FLAGS:MRX_FLAGS=25.1.#
~MRX_FLAGS:" "=25.1.#
?NADMDNAME:NADMDNAME=26.1.#
~NADMDNAME:" "=26.1.#
?NAME:NAME=27.1.#
~NAME:" "=27.1.#
?NCOUNTRY:NCOUNTRY=28.1.#
~NCOUNTRY:" "=28.1.#
?NPRMDNAME:NPRMDNAME=29.1.#
~NPRMDNAME:" "=29.1.#
?ORGNAME:ORGNAME=30.1.#
~ORGNAME:" "=30.1.#
?ORGUNIT:ORGUNIT=31.1.#
~ORGUNIT:" "=31.1.#
?ORGUNIT2:ORGUNIT2=32.1.#
~ORGUNIT2:" "=32.1.#
?ORGUNIT3:ORGUNIT3=33.1.#
~ORGUNIT3:" "=33.1.#
?ORGUNIT4:ORGUNIT4=34.1.#
~ORGUNIT4:" "=34.1.#
?PCOUNTRY:PCOUNTRY=35.1.#
~PCOUNTRY:" "=35.1.#
?POSNROLE:POSNROLE=36.1.#
~POSNROLE:" "=36.1.#
?PRMDNAME:PRMDNAME=37.1.#
~PRMDNAME:" "=37.1.#
?PRLOCATION:PRLOCATION=38.1.#
~PRLOCATION:" "=38.1.#
?PROXY:PROXY=39.1.#
~PROXY:" "=39.1.#
?PRUSERNAME:PRUSERNAME=40.1.#
~PRUSERNAME:" "=40.1.#
?REVERSE:REVERSE=41.1.#
~REVERSE:" "=41.1.#
?REVERSE2:REVERSE2=42.1.#
~REVERSE2:" "=42.1.#
?SNLOCATION:SNLOCATION=43.1.#
~SNLOCATION:" "=43.1.#
?SNUSERNAME:SNUSERNAME=44.1.#
~SNUSERNAME:" "=44.1.#
?SUBID:SUBID=45.1.#
~SUBID:" "=45.1.#
?SURNAME:SURNAME=46.1.#
~SURNAME:" "=46.1.#
?SURNAME2:SURNAME2=47.1.#
~SURNAME2:" "=47.1.#
?TELEPHONE:TELEPHONE=48.1.#
~TELEPHONE:" "=48.1.#
?TERMINALID:TERMINALID=49.1.#
~TERMINALID:" "=49.1.#
```

Example A–1 Cont'd on next page

Example A–1 (Cont.): RDF for DDS Export - dsu_export_dds.rdf

```
?TITLE:TITLE=50.1.#
~TITLE:" "=50.1.#
?UAID:UAID=51.1.#
~UAID:" "=51.1.#
?1UNUSED1:1UNUSED1=52.1.#
~1UNUSED1:" "=52.1.#
?1UNUSED2:1UNUSED2=53.1.#
~1UNUSED2:" "=53.1.#
?1UNUSED3:1UNUSED3=54.1.#
~1UNUSED3:" "=54.1.#
?1UNUSED4:1UNUSED4=55.1.#
~1UNUSED4:" "=55.1.#
?2UNUSED1:2UNUSED1=56.1.#
~2UNUSED1:" "=56.1.#
?2UNUSED2:2UNUSED2=57.1.#
~2UNUSED2:" "=57.1.#
?2UNUSED3:2UNUSED3=58.1.#
~2UNUSED3:" "=58.1.#
?2UNUSED4:2UNUSED4=59.1.#
~2UNUSED4:" "=59.1.#
?3UNUSED1:3UNUSED1=60.1.#
~3UNUSED1:" "=60.1.#
?3UNUSED2:3UNUSED2=61.1.#
~3UNUSED2:" "=61.1.#
?3UNUSED3:3UNUSED3=62.1.#
~3UNUSED3:" "=62.1.#
?3UNUSED4:3UNUSED4=63.1.#
~3UNUSED4:" "=63.1.#
?USE1:USE1=64.1.#
~USE1:" "=64.1.#
?USE2:USE2=65.1.#
~USE2:" "=65.1.#
?USE3:USE3=66.1.#
~USE3:" "=66.1.#
?USE4:USE4=67.1.#
~USE4:" "=67.1.#
?USE5:USE5=68.1.#
~USE5:" "=68.1.#
?USE6:USE6=69.1.#
~USE6:" "=69.1.#
?VMSUSERNAME:VMSUSERNAME=70.1.#
~VMSUSERNAME:" "=70.1.#
?VMSNODENAME:VMSNODENAME=71.1.#
~VMSNODENAME:" "=71.1.#
?VMSADDRPART:VMSADDRPART=72.1.#
~VMSADDRPART:" "=72.1.#
?X121ADDRESS:X121ADDRESS=73.1.#
~X121ADDRESS:" "=73.1.#
```

The sample export configuration file for use in exporting DDS data is shown in Example A–2.

Example A–2: Configuration File for DDS Export - DSU_CONFIG.DAT_EXPORT_DDS

```
synch_type=EXPORT
rdf_file=DSU$03_DB:DSU_EXPORT_DDS.RDF
input_old_metafile=
output_old_metafile=
output_for_dir_file=DSU$03_OUTPUT:EXPORT_DDS.DAT
nrec_subscr=73
dds_gid_used=USE1
dds_gid_match=ALL-IN-1
dds_gid_exclude=
dds_country_match=
dds_admd_match=
dds_prmd_match=
dds_orgname_match=
dds_orgunit_match=
dds_orgunit2_match=
dds_orgunit3_match=
dds_orgunit4_match=
end_run_file=
notify=SYSTEM
dsu_id=
! new V2.0 keywords
dsu_label=DDS EXPORT
```

In this example:

- The RDF file, DSU_EXPORT_DDS.RDF, is used.
- The output file created is EXPORT_DDS.DAT.
- 73 lines per subscriber are written to the output file.
- The DDS field, USE1, is used to extract the subscribers with a Group ID of ALL-IN-1.
- The SYSTEM account is notified of the start, end, and any errors in this Synchronizer run.
- The label "DDS EXPORT" is used in all Synchronizer mail messages and summary reports.

A.1.1.3 Run MAILbus Directory Synchronizer to Export DDS Data

MAILbus Directory Synchronizer is started by issuing the following command:

```
$ @SYS$MANAGER:MB$CONTROL START=DSU_03
```

When the export finishes, a VMSmail message is sent to SYSTEM. The log file, SYS$LOGIN:DSU$03.LOG, should be examined to review the results of the export. An output file is created with the name EXPORT_DDS.DAT.

This flat file will be moved to the Synchronizer-500 system and used as input for importing the user data into X.500.

A.1.2 X.500 System Activities

The following sections describe the activities on the Synchronizer-500 system.

A.1.2.1 Create the DDS Import Subdirectory

Create a Synchronizer-500 subdirectory for this instance. See Section 3.3.1 for the instructions on how to run the xdsu_config_instance procedure.

When the subdirectory has been created, set your current directory to the newly created directory and proceed with the next steps below.

A.1.2.2 Transfer the DDS Export File to the X.500 System

The export file, EXPORT_DDS.DAT, that was generated as described in Section A.1.1.3 will to be used as the import data file to Synchronizer-500. This file must be transferred from the MAILbus system to the X.500 system into the subdirectory for this run.

The configuration file parameter, END_RUN_FILE=, provides a convenient means of initiating the procedure for transferring the file. By specifying the file name of a user-written command procedure, a program can be called to transport the EXPORT_DDS.DAT output file to the X.500 system.

Note

The file name should remain the same regardless of how the file is transferred.

A.1.2.3 Setting up the Import RDF

A sample RDF file, xdsu_import_dds.rdf, has been provided in the Synchronizer-500 root directory which maps the DDS export file, EXPORT_DDS.DAT, into X.500 RDNs and attributes. This file must be moved from the Synchronizer-500 root directory into the subdirectory for this run.

Once the file xdsu_import_dds.rdf has been copied to the subdirectory, edit the file to specify the X.500 RDNs and attributes you are using in your DIT. Sample RDNs and attributes are provided.

As explained in Chapter 3, each line in the RDF consists of an X.500 attribute keyword, a record number, a starting column number, and a length. The record number is used because there is more than one line per user. Example A–3 presents the example RDF.

Example A–3: RDF for DDS Import - xdsu_import_dds.rdf

```
!
! XDSU_IMPORT_DDS.RDF
! Import Synch from DDS
!
! 11-October-1993
!
!
!   RDF for import synch which will synch
!   DDS against X.500.
!   (expects input file to have been generated by MAILbus
!     Directory Synchronizer using the RDF, DSU_EXPORT_DDS.RDF).
!
!   One DDS field per line
!
!** First define the X.500 RDNs**
!
! Default the COUNTRY ORGANIZATION, and ORGANIZATIONALUNITs RDN values
!
country="US"
organization="Digital"
organizationalUnit="Sales"
organizationalUnit_2="Mass."
!
! The commonName RDN is built from the input file and consists
! of GIVENNAME INITIALS SURNAME
!
commonName=13.1.16
commonName=" "
commonName=15.1.5
commonName=" "
commonName=46.1.40
!
! The surname attribute comes from the input file
!
surname=46.1.40
!
! Define the emailInfo auxiliary class attributes
!
! X.400 O/R address values - If data does not exist in input file
! supply constants
!
?35.1.2:emailPCOUNTRY=35.1.2
~35.1.2:emailPCOUNTRY="US"
?3.1.50:emailADMDNAME=3.1.50
~3.1.50:emailADMDNAME="PUBLIC"
?37.1.16:emailPRMDNAME=37.1.16
~37.1.16:emailPRMDNAME="EMSERVER"
?30.1.64:emailORGNAME=30.1.64
~30.1.64:emailORGNAME="MARKETING"
!
! Put the reverse lookup data in the emailMAILbusAddr attribute
!
emailMAILbusAddr=41.1.255
!
```

Example A–3 Cont'd on next page

Example A–3 (Cont.): RDF for DDS Import - xdsu_import_dds.rdf

```
! Build the 1984 X.400 address in the emailX400Addr84 attribute
! Consists of GIVENNAME, INITIALS, SURNAME, ORGANIZATIONALUNIT, ORGANIZATION,
! PRMD, ADMD, and COUNTRY
!
!
emailX400Addr84="G="
emailX400Addr84=13.1.16
emailX400Addr84="; I="
emailX400Addr84=15.1.5
emailX400Addr84="; S="
emailX400Addr84=46.1.40
emailX400Addr84="; OU1="
emailX400Addr84=31.1.32
emailX400Addr84="; O="
emailX400Addr84=30.1.64
emailX400Addr84="; P=EMSERVER; A=PUBLIC; C=US"
!
! Build the 1988 X.400 address in the emailX400Addr88 attribute
! Consists of COMMONNAME, ORGANIZATIONALUNIT, ORGANIZATION,
! PRMD, ADMD, and COUNTRY
!
emailX400Addr88="CN="
emailX400Addr88=13.1.16
emailX400Addr88=" "
emailX400Addr88=15.1.5
emailX400Addr88=" "
emailX400Addr88=46.1.40
emailX400Addr88="; OU1="
emailX400Addr88=31.1.32
emailX400Addr88="; O="
emailX400Addr88=30.1.64
emailX400Addr88="; P=EMSERVER; A=PUBLIC; C=US"
!
! Input the rest of the data
!
emailADDRPART=1.1.25
emailADDRPART2=2.1.25
emailDDA=4.1.137
emailDDA2=5.1.137
emailDDA3=6.1.137
emailDDA4=7.1.137
emailFFPOSTADDR=8.1.40
emailFFPOSTADDR2=9.1.40
emailFFPOSTADDR3=10.1.40
emailFFPOSTADDR4=11.1.40
emailGENERATION=12.1.3
emailGIVENNAME=13.1.16
emailGIVENNAME2=14.1.16
emailINITIALS=15.1.5
emailINTLOCATION=16.1.20
emailINTPHONE=17.1.32
emailLANGUAGE=18.1.2
emailLOCATION=19.1.32
emailMAILADDRESS=20.1.20
emailMHSMAILBOX=21.1.24
```

Example A–3 Cont'd on next page

Example A–3 (Cont.): RDF for DDS Import - xdsu_import_dds.rdf

```
emailMHSMAILBOX2=22.1.24
emailMHSUSERID=23.1.255
emailMHSUSERID2=24.1.255
emailMRX_FLAGS=25.1.2
emailNADMDNAME=26.1.16
emailNAME=27.1.64
emailNCOUNTRY=28.1.3
emailNPRMDNAME=29.1.16
emailORGUNIT=31.1.32
emailORGUNIT2=32.1.32
emailORGUNIT3=33.1.32
emailORGUNIT4=34.1.32
emailPOSNROLE=36.1.40
emailPRLOCATION=38.1.8
emailPROXY=39.1.32
emailPRUSERNAME=40.1.8
emailREVERSE=41.1.255
emailREVERSE2=42.1.255
emailSNLOCATION=43.1.8
emailSNUSERNAME=44.1.8
emailSUBID=45.1.20
emailSURNAME=46.1.40
emailSURNAME2=47.1.40
emailTELEPHONE=48.1.32
emailTERMINALID=49.1.24
emailTITLE=50.1.10
emailUAID=51.1.20
email1UNUSED1=52.1.40
email1UNUSED2=53.1.40
email1UNUSED3=54.1.40
email1UNUSED4=55.1.40
email2UNUSED1=56.1.40
email2UNUSED2=57.1.40
email2UNUSED3=58.1.40
email2UNUSED4=59.1.40
email3UNUSED1=60.1.40
email3UNUSED2=61.1.40
email3UNUSED3=62.1.40
email3UNUSED4=63.1.40
emailUSE1=64.1.80
emailUSE2=65.1.80
emailUSE3=66.1.80
emailUSE4=67.1.80
emailUSE5=68.1.80
emailUSE6=69.1.80
emailVMSUSERNAME=70.1.32
emailVMSNODENAME=71.1.6
emailVMSADDRPART=72.1.25
emailX121ADDRESS=73.1.15
```

A.1.2.4 Setting up the Import Configuration File

A sample configuration file for importing DDS subscriber data into X.500 has been provided. This file is called xdsu_config.dat_import_dds and is located in the Synchronizer-500 root directory.

To utilize this sample import configuration file:

1. Copy xdsu_config.dat_import_dds from the Synchronizer-500 root directory into the sub-directory for this run and give it the name xdsu_config.dat.
2. Edit xdsu_config.dat to include site specific information for:

   DN_SEQUENCE
   CREATE_ROOT
   SYNCH_OBJECT
   GID_FIELD
   GID_MATCH
   SYNCH_ID
   SYNCH_ID_FIELD

The sample configuration file is shown in Example A–4.

Example A–4: Configuration File for DDS Import - xdsu_config.dat_import_dds

```
synch_type=IMPORT
!
dn_sequence=country/organization/organizationalUnit/commonName
create_root=organization
synch_object=Person,organizationalPerson,emailInfo
bind_dsa=
!
rdf_file=xdsu_import_dds.rdf
initial_load=YES
input_for_dir_file=export_dds.dat
nrec_subscr=73
gid_field=description
gid_match=ALL-IN-1
max_transactions=
max_fail_transactions=
max_trans_file=
end_run_file=
notify=SYSTEM
synch_id=X5Synch
synch_id_field=emailUSE1
unique_field=
unique_field_len=
gen_profs_alias=NO
gen_snads_alias=NO
old_metafile=
new_metafile=
log_file=import_dds.log
verbose=YES
```

Example A–4 Cont'd on next page

Example A–4 (Cont.): Configuration File for DDS Import - xdsu_config.dat_import_dds

```
encode_delim=|
synch_label=DDS Import
make_unique=
truncate=
admd_space=
```

In this example:

- The distinguished name that is generated for each entry upon import is made up of the RDNs: country, organization, organizationalUnit, and commonName. (As required, the RDF specifies values for each of these RDNs.)
- When the entries are added, create the superior entry for organization, organizationalUnit, and organizationalUnit if needed. (Note that there are two levels of organizationalUnits.)
- When the entries are imported, they are specified to be of the Person, organizationalPerson, and emailInfo classes.
- The RDF file, xdsu_import_dds.rdf, is used.
- The value for initial load is set to YES.
- The input file is export_dds.dat and there are 73 lines per subscriber entry.
- The X.500 attribute *DESCRIPTION* is used to store the Synchronizer-500 Group ID, ALL-IN-1.
- The SYSTEM account is notified of the start, end, and any errors in this run.
- The SYNCH_ID, X5SYNCH, is stored in the X.500 attribute emailUSE1.
- The transactions will be logged to the file import_dds.log in verbose mode.
- The label "DDS IMPORT" is used in all Synchronizer-500 mail messages and summary reports.

A.1.2.5 Running the Import

Synchronizer-500 is started by issuing the following command from the directory where the configuration, RDF, and DDS data files reside:

```
OpenVMS:

$ XDSU :== $SYS$SYSTEM:XDSU$MAIN.EXE
$ XDSU

UNIX:

/usr/etc/xdsu
```

As Synchronizer-500 executes, its processing is logged to your terminal device as shown in Example A–5. The processing should be examined as it occurs to review the results of the initial load. When the initial load finishes, a mail message containing a summary of the run is sent to the address specified by the configuration file parameter "notify=".

The results of the initial load should be examined in the transaction log file import_dds.log (Example A–6).

Example A–5: Initial Load of DDS Users Processing

```
Feb 28 12:36:42 processing layout file [-]XDSU_LAYOUT.DAT
Feb 28 12:36:43 processing configuration file XDSU_CONFIG.DAT
Feb 28 12:36:43 initializing and binding to X.500
Feb 28 12:36:44 processing RDF file xdsu_import_dds.rdf
Feb 28 12:36:44 converting input file export_dds.dat to new metafile
Feb 28 12:36:44 73 input records read
Feb 28 12:36:44 1 new metafile records written
Feb 28 12:36:44 sorting new metafile
Feb 28 12:36:45 searching X.500 directory data
    SEARCH  -
     where   description = ALL-IN-1 -
       and     EMAILUSE1 = X5Synch -
       and    COMMONNAME = *
Feb 28 12:36:45 found 0 X.500 directory entries
Feb 28 12:36:45 extracting X.500 directory data to old metafile
Feb 28 12:36:45 extracted 0 directory entries into old metafile
Feb 28 12:36:45 comparing old and new metafiles
Feb 28 12:36:45 processing ADD records
    ADD /C=US/O=DIGITAL/OU=SALES/OU=MASS./CN=JOHN M SMITH
Feb 28 12:36:45 1 records added
Feb 28 12:36:45 creating summary file

XDSU V1.2-1
DDS Import run started at: Tue Feb 28 12:36:42 1995

IMPORT SYNCH SUMMARY:

RECORDS READ/WRITTEN SUMMARY:

Foreign directory file records read: 73
New metafile records written: 1
Old metafile records written: 0

TRANSACTION SUMMARY:

Number of successful transactions: 1
Number of failed transactions:     0
Number of successful add transactions:    1
Number of successful modify transactions: 0
Number of successful delete transactions: 0
Number of failed add transactions:    0
Number of failed modify transactions: 0
Number of failed delete transactions: 0

CONFIGURATION FILE SUMMARY:
```

Example A–5 Cont'd on next page

Example A–5 (Cont.): Initial Load of DDS Users Processing

```
synch_type: IMPORT
synch_label: DDS Import
bind_dsa:
rdf_file: xdsu_import_dds.rdf
input_for_dir_file: export_dds.dat
nrec_subscr: 73
log_file: import_dds.log
end_run_file:
old_metafile:
new_metafile:
initial_load: YES
notify: SYSTEM
verbose: YES
encode_delim: |
max_transactions:
max_fail_transactions:
max_trans_file:
dn_sequence: country/organization/organizationalUnit/commonName
create_root: organization
search_context:
synch_object: Person,organizationalPerson,emailInfo
synch_id: X5Synch
synch_id_field: emailUSE1
gid_field: description
gid_match: ALL-IN-1
gen_profs_alias: NO
gen_snads_alias: NO
admd_space:
unique_field:
unique_field_len:
make_unique:
truncate:
unique_field_separator:
unique_search_context:

DDS Import run ended at: Tue Feb 28 12:36:46 1995
```

When the import initial load has completed, the log file shown in Example A–6 should be reviewed to determine if the entries have been imported correctly. The configuration file shown in Example A–4 specifies the log file name and enables verbose mode.

Example A–6: Checking the Results of the Import - import_dds.log

```
DDS Import run started at: Wed Mar 15 12:36:43 1995

A INFORMATIONAL Parent Entry created: /C=US/O=Digital/OU=Sales

A INFORMATIONAL Parent Entry created: /C=US/O=Digital/OU=Sales/OU=Mass.

A NORMAL /C=US/O=DIGITAL/OU=SALES/OU=MASS./CN=JOHN M SMITH  - status=SUCCESS

  SURNAME = SMITH
  DESCRIPTION = ALL-IN-1
  EMAILX400ADDR84 = G=JOHN; I=M; S=SMITH; OU1=CCC; O=DIGITAL; P=EMSERVER;
    A=PUBLIC; C=US
  EMAILX400ADDR88 = CN=JOHN M SMITH; OU1=CCC; O=DIGITAL; P=EMSERVER; A=PUBLIC;
    C=US
  EMAILMAILBUSADDR = SMITH@A1@HERB
  EMAILPCOUNTRY = US
  EMAILINITIALS = M
  EMAILPRLOCATION = X5SYNCH
  EMAILPRUSERNAME = SMITHJ
  EMAILSNLOCATION = X5SYNCH
  EMAILSNUSERNAME = SMITHJ
  EMAILADMDNAME = PUBLIC
  EMAILGIVENNAME = JOHN
  EMAILMHSMAILBOX = A1
  EMAILADDRPART = HERB
  EMAILORGUNIT = CCC
  EMAILPRMDNAME = EMSERVER
  EMAILTELEPHONE = 301-918-5555
  EMAILFFPOSTADDR = 6406 Ivy Lane
  EMAILFFPOSTADDR2 = Greenbelt, MD
  EMAILNAME = JOHN SMITH
  EMAILORGNAME = DIGITAL
  EMAILSURNAME = SMITH
  EMAILUSE1 = X5Synch
  EMAILMHSUSERID = SMITH
  EMAILREVERSE = SMITH@A1@HERB

DDS Import run ended at: Wed Mar 15 12:36:46 1995
```

Examining the log file in Example A–6 shows that entries for DDS users were correctly added.

In Example A–6, the second and third lines of the log file indicate that superior entries of /C=US/O=Digital/OU=Sales and /C=US/O=Digital/OU=Sales/OU=Mass. were created. This occurred because of the distinguished name sequence (DN_SEQUENCE) and CREATE_ROOT records specified in the configuration file. In this example, the DN_SEQUENCE specified that each entry should have a distinguished name of *country/organization/organizationalUnit/commonName* and, if need be, superior entries should be created as high as the organization level (CREATE_ROOT). Because the DIT did not contain superior entries for the two organizationalUnits, they were created by Synchronizer-500. This is necessary before any entries can be added to X.500 with the specified distinguished name sequence.

In Example A–6, the fourth line represents the normal log output. The first term, A, indicates an ADD operation; the second term, NORMAL, shows that overall success of the indicated operation (NORMAL or FAILED). The third term, /C=US/O=DIGITAL..., is the distinguished name for the entry that was added. The last term, status, shows the status for a transaction. See Appendix G for a list of transaction status codes.

The remainder of the log file in Example A–6 is a dump of the attributes for the entry and is only shown when VERBOSE=YES.

Example A–7 shows an X.500 entry created as a result of the import run.

Example A–7: X.500 Entry for a DDS User

```
dxim> show /C=us/O=digital/OU=sales/OU=mass./CN="JOHN M SMITH"

    /C=US/O=digital/OU=Sales/OU=Mass./CN="JOHN M SMITH"
      Common Name    = JOHN M SMITH
      Object Class   = Person
                     = Organizational Person
                     = emailInfo
      Surname        = SMITH
      Description    = ALL-IN-1
      emailX400Addr84 = G=JOHN; I=M; S=SMITH; OU1=CCC; O=DIGITAL; P=EMSERVER;
                        A=PUBLIC; C=US
      emailX400Addr88 = CN=JOHN M SMITH; OU1=CCC; O=DIGITAL; P=EMSERVER;
                        A=PUBLIC; C=US
      emailMAILbusAddr = SMITH@A1@HERB
      emailPCOUNTRY  = US
      emailINITIALS  = M
      emailPRLOCATION = X5SYNCH
      emailPRUSERNAME = SMITHJ
      emailSNLOCATION = X5SYNCH
      emailSNUSERNAME = SMITHJ
      emailADMDNAME  = PUBLIC
      emailGIVENNAME = JOHN
      emailMHSMAILBOX = A1
      emailADDRPART  = HERB
      emailORGUNIT   = CCC
      emailPRMDNAME  = EMSERVER
      emailTELEPHONE = 301-918-5555
      emailFFPOSTADDR = 6406 Ivy Lane
      emailFFPOSTADDR2 = Greenbelt, MD
      emailNAME      = JOHN SMITH
      emailORGNAME   = DIGITAL
      emailSURNAME   = SMITH
      emailUSE1      = X5Synch
      emailMHSUSERID = SMITH
      emailREVERSE   = SMITH@A1@HERB
```

In this example, data with a Group ID of ALL-IN-1 has been successfully exported from DDS and then imported into X.500. The process detailed in this chapter would then be repeated for each Group ID value identifying data that should be imported into X.500.

Appendix B
Sample SMTP Synchronization

This appendix gives a brief example of how Synchronizer-500 can be used to export data from X.500 for use in an SMTP mail network. The export process generates a sendmail alias file that SMTP users can use to "look up" X.400 addresses. The example SMTP configuration file, RDF, and end run file can be found in the Synchronizer-500 root directory.

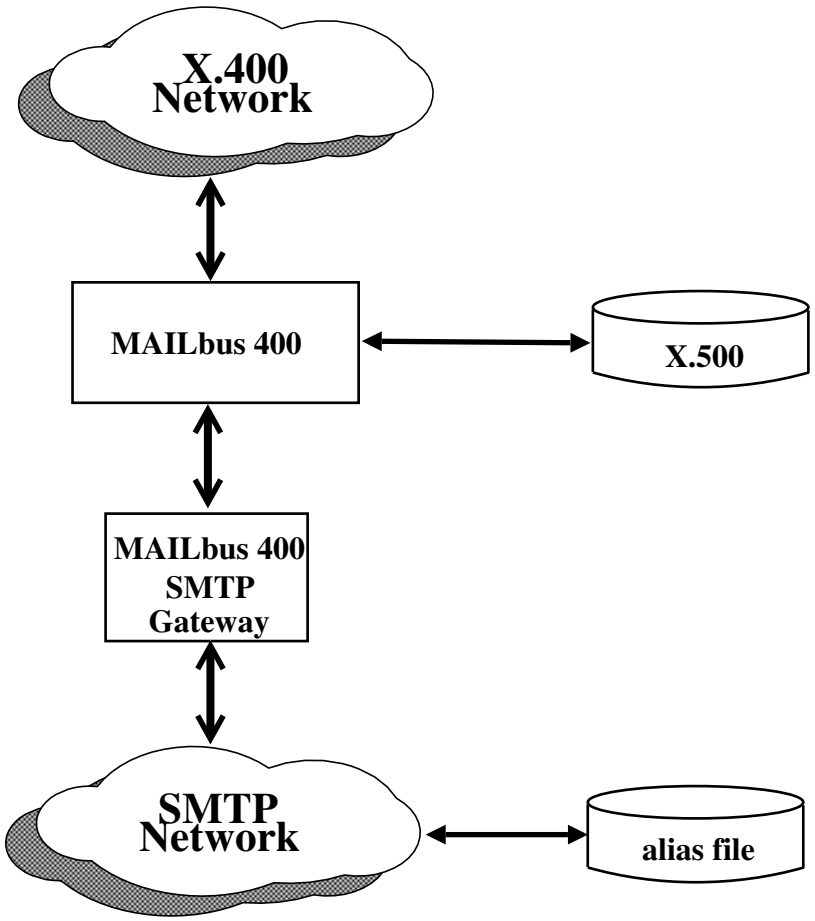
Many SMTP user agents do not have a directory of all users. It is generally not possible, therefore, to import SMTP user data into X.500 unless a site specific directory is maintained. Because there is no standard directory, this example will not show importing SMTP user data into X.500.

The messaging network configuration for this example is shown in Figure B–1.

For this example, it is assumed that SMTP users are connected through the MAILbus 400 SMTP gateway to MAILbus 400 and then out to the X.400 network. SMTP users will address mail to X.400 users by specifying their X.400 address along with a Gateway RFC 822 Domain. For example,

```
TO: /c=us/a=public/p=emserver/o=digital/ou=sales/s=smith/g=jane
    /dd.common=jane_smith/@tamvm2.x400
```

Figure B–1: SMTP Messaging Network Configuration



B.1 Export Process

The purpose of the export process is to extract selected portions of X.500 and format the data in SMTP address format so that the output file can be used as a sendmail alias file.

The following is a broad outline of the steps needed to export X.500 entries:

1. Create a Synchronizer-500 subdirectory for this instance.
2. Determine the format of the sendmail alias file and decide how to map the X.500 attributes into this format.
3. Write the RDF to reflect this data correspondence.
4. Set up the configuration file with the proper parameters.

5. Run Synchronizer-500 to export X.500 entries in the format desired.

6. Transfer the export file to the SMTP system.

B.1.1 Create the SMTP Export Subdirectory

Create a Synchronizer-500 subdirectory for this instance. See Section 3.3.1 for the instructions on how to run the xdsu_config_instance procedure to create a subdirectory for the SMTP export run.

When the subdirectory has been created, set your current directory to the newly created directory and proceed with the next steps below.

B.1.2 Writing the Export RDF

Example B–1 shows a sample RDF that is used in exporting X.500 entries to SMTP address format. This file (xdsu_export_smtp.rdf) is included in the Synchronizer-500 root directory and should be referred to while reading the following text.

Example B–1: RDF for SMTP Export - xdsu_export_smtp.rdf

```
!
! XDSU_EXPORT_SMTP.RDF
! This RDF generates a file which can be moved into /usr/lib/aliases in UNIX
! 11-October-1993
!
! RDF for export of data from X.500 to UNIX sendmail alias file.
!
! If emailUSE5 defined, this is the 'alias' we want to use with sendmail. Always
! proceed the alias with an underscore character so as not to accidentally
! redirect mail (via an alias) from an account that actually is on this
! machine. (assumes no usernames start with _).
!
"_"=1.*.#
?emailUSE5:emailUSE5=1.*.#
!
! If no USE5, make the combination of surname, givenname, and orgunit
! into the alias field.
!
~emailUSE5:emailSURNAME=1.*.#
~emailUSE5:"-"=1.*.#
~emailUSE5:emailGIVENNAME=1.*.#
~emailUSE5:"-"=1.*.#
~emailUSE5:emailORGUNIT=1.*.#
!
! A colon marks the end of the alias name
!
":  "=1.*.#
!
! Next, we need to create the actual addressing string required for
! a sendmail user to route mail through the MAILbus 400 SMTP gateway.
! First, we take the X.400 address and then we append the gateway followed
! by the RFC 822 domain.
! The final format of the address will be:
```

Example B–1 Cont'd on next page

Example B–1 (Cont.): RDF for SMTP Export - xdsu_export_smtp.rdf

```
! /c=country/a=ADMD/p=PRMD/o=org/ou1=orgunit1/ou2=orgunit2.../s=surname
!    /g=givenname/dd.common=common_name/@gateway rfc 822 domain
!
! Define constants for the GDI terms:
!
"/c=us/a=public/p=emserver/o="=1.*.#
!
! Put ORGNAME and then ORGUNIT if it exists
!
emailORGNAME=1.*.#
?emailORGUNIT:"/ou1="=1.*.#
?emailORGUNIT:emailORGUNIT=1.*.#
!
! Surname comes next and then GIVENNAME if it exists
!
"/s="=1.*.#
emailSURNAME=1.*.#
?emailGIVENNAME:"/g="=1.*.#
?emailGIVENNAME:emailGIVENNAME=1.*.#
!
! Next is the common name
!
"/dd.common="=1.*.#
?emailNAME:emailNAME=1.*.#
!
! If the common name is not specified, let's try to build one using the
! givenname and surname.
!
~emailNAME:emailGIVENNAME=1.*.#
~emailNAME:"_"=1.*.#
~emailNAME:emailSURNAME=1.*.#
!
! Last is the Gateway RFC 822 Domain
!
"/@tamvm2.x400"=1.*.#
!
! Now, for naming purposes, let's just put a comment field here for
! people who want to grep the nickname file for a user. Note that the
! alias file has no provisions for inline comments so we will use a
! second line, immediately following the alias name.
!
! First the comment delimiter
!
"# "=2.1.#
!
! If there is a common name, use it. It probably contains descriptive
! info which we can't get elsewhere.
!
?emailNAME:emailNAME=2.*.#
!
! If the common name is not specified, let's try to build one using the
! givenname and surname.
!
~emailNAME:emailGIVENNAME=2.*.#
~emailNAME:" "=2.*.#
```

Example B–1 Cont'd on next page

Example B–1 (Cont.): RDF for SMTP Export - xdsu_export_smtp.rdf

```
~emailNAME:emailSURNAME=2.*.#
!
! Put a separator here and then put the telephone number (or not available)
" - "=2.*.#
emailTELEPHONE=2.*.#
~emailTELEPHONE:" Telephone not available."=2.*.#
```

B.1.3 Setting up the Export Configuration File

Example B–2 shows a sample export configuration file for use in exporting X.500 entries to SMTP address format. This file (xdsu_config.dat_export_smtp) is included in the Synchronizer-500 root directory and should be referred to while reading the following text.

Example B–2: Configuration File for SMTP Export

```
synch_type=EXPORT
!
bind_dsa=
search_context=/country=us/organization=digital
!
rdf_file=xdsu_export_smtp.rdf
input_old_metafile=
output_old_metafile=
output_for_dir_file=export_smtp.dat
nrec_subscr=2
gid_field=description
gid_match=*
gid_exclude=smtp
country_match=
admd_match=
prmd_match=
orgname_match=
orgunit_match=
orgunit2_match=
orgunit3_match=
orgunit4_match=
end_run_file=xdsu_export_smtp.com
notify=system
synch_id=x5synch
synch_label=SMTP_export
get_field=
get_value=
```

This file is an example of a configuration file for an export synchronization. The example RDF xdsu_export_smtp.rdf and example end_run_file xdsu_export_smtp.com are used. (These files are contained in the Synchronizer-500 root directory.)

The example export configuration file consists of keyword and value pairs. In this example not all keywords are used. The unused keywords can remain blank or can be removed from the configuration file. Descriptions of the example values are given below. Please see Appendix F for more detailed information.

In this example:

- The search will start on the X.500 tree /country=us/organization=digital.
- The RDF file, xdsu_export_smtp.rdf, is used.
- The output file, export_smtp.dat, is generated with 2 records per entry.
- The X.500 attribute DESCRIPTION is used to extract every entry (GID_MATCH=*) except those with a DESCRIPTION attribute value of SMTP (GID_EXCLUDE=SMTP).
- Upon termination of this Synchronizer-500 run, the command procedure xdsu_export_smtp.com is executed.
- The system account is notified of the start, end, and any errors in this Synchronizer-500 run.
- The label "SMTP_export" is used in all Synchronizer-500 mail messages and summary reports.

B.1.4 Running the Export

Synchronizer-500 is started by issuing the following command from the directory where the configuration file and RDF reside:

```
OpenVMS:

$ XDSU :== $SYS$SYSTEM:XDSU$MAIN.EXE
$ XDSU

UNIX:

/usr/etc/xdsu
```

As Synchronizer-500 executes, its processing is logged to your terminal device as shown in Example B–3. The processing should be examined as it occurs to review the results of the export run. When the export run finishes, a mail message containing a summary of the run is sent to the address specified by the configuration file parameter "notify=".

Example B–3: SMTP Export Processing

```
Feb 28 16:02:15 processing layout file [-]XDSU_LAYOUT.DAT
Feb 28 16:02:15 processing configuration file XDSU_CONFIG.DAT
Feb 28 16:02:15 initializing and binding to X.500
Feb 28 16:02:16 processing RDF file xdsu_export_smtp.rdf
Feb 28 16:02:16 searching X.500 directory data
Feb 28 16:02:16 found 35 X.500 directory entries
Feb 28 16:02:17 extracting X.500 directory data to export metafile
Feb 28 16:02:18 extracted 35 directory entries into export metafile
Feb 28 16:02:18 sorting export metafile
Feb 28 16:02:19 converting metafile to output file export_smtp.dat
Feb 28 16:02:19 35 metafile records read
Feb 28 16:02:19 70 output file records written
Feb 28 16:02:19 creating summary file
Feb 28 16:02:19 executing end run file xdsu_export_smtp.com
```

Example B–3 Cont'd on next page

Example B–3 (Cont.): SMTP Export Processing

```
XDSU V1.2
SMTP_export run started at: Tue Feb 28 16:02:15 1995

EXPORT SYNCH SUMMARY:

RECORDS READ/WRITTEN SUMMARY:

Old metafile records written: 35
Old metafile records read:    35
Foreign directory file records written: 70

CONFIGURATION FILE SUMMARY:

synch_type: EXPORT
synch_label: SMTP_export
bind_dsa:
rdf_file: xdsu_export_smtp.rdf
output_for_dir_file: export_smtp.dat
nrec_subscr: 2
end_run_file: xdsu_export_smtp.com
output_old_metafile:
input_old_metafile:
notify: system
verbose:
encode_delim: |
search_context: /country=us/organization=digital
synch_id: X5synch
synch_id_field:
gid_field: description
gid_match: *
gid_exclude: smtp
country_match:
admd_match:
prmd_match:
orgname_match:
orgunit_match:
orgunit2_match:
orgunit3_match:
orgunit4_match:
get_field:
get_value:

SMTP_export run ended at: Tue Feb 28 16:02:19 1995
```

Example B–4 shows a portion of the output file, export_smtp.dat, that was generated from the export run. To make it easier to read, the first line of every entry is wrapped to show the entire address. In the actual output file, the address is on one line.

Example B–4: Exported X.500 Data in SMTP Sendmail Alias Format

```
_APPLE-BOB-D:  /c=us/a=public/p=emserver/o=Digital/ou1=SALES/s=APPLE/g=BOB
     /dd.common=BOB D. APPLE/@tamvm2.x400
# BOB D. APPLE -  Telephone not available.
_WILLIAMSON-BILL-B:  /c=us/a=public/p=emserver/o=Digital/ou1=SRV/s=WILLIAMSON
     /g=BILL/dd.common=BILL B. WILLIAMSON/@tamvm2.x400
# BILL B. WILLIAMSON -  Telephone not available.
_SMITH-JOHN-CCC:  /c=us/a=public/p=emserver/o=DIGITAL/ou1=CCC/s=SMITH/g=JOHN
     /dd.common=JOHN SMITH/@tamvm2.x400
# JOHN SMITH - 301-918-5555
_MOOG-MARK-CCC:  /c=us/a=public/p=emserver/o=DIGITAL/ou1=CCC/s=MOOG/g=MARK
     /dd.common=Mark Moog/@tamvm2.x400
# Mark Moog -  Telephone not available.
_Reed-NOVLAN-MEL:  /c=us/a=public/p=emserver/o=NOVLAN/ou1=MEL/s=Reed
     /dd.common=Mary Reed/@tamvm2.x400
# Mary Reed -  Telephone not available.
_Guest-NOVLAN-MEL:  /c=us/a=public/p=emserver/o=NOVLAN/ou1=MEL/s=Guest
     /dd.common=Our Guest/@tamvm2.x400
# Our Guest -  Telephone not available.
_Rotary-NOVLAN-MEL:  /c=us/a=public/p=emserver/o=NOVLAN/ou1=MEL/s=Rotary
     /dd.common=Sharon Rotary/@tamvm2.x400
# Sharon Rotary -  Telephone not available.
_Blue-NOVLAN-MEL:  /c=us/a=public/p=emserver/o=NOVLAN/ou1=MEL/s=Blue
     /dd.common=Suzanne Blue/@tamvm2.x400
# Suzanne Blue -  Telephone not available.
_ADMIN-NOVLAN-MEL:  /c=us/a=public/p=emserver/o=NOVLAN/ou1=MEL/s=ADMIN
     /dd.common=Microsoft Mail Administrator/@tamvm2.x400
# Microsoft Mail Administrator -  Telephone not available.
_Smith-NOVLAN-MEL:  /c=us/a=public/p=emserver/o=NOVLAN/ou1=MEL/s=Smith
     /dd.common=Byron Smith/@tamvm2.x400
# Byron Smith -  Telephone not available.
```

B.1.5 Transfer the Synchronizer-500 Export Output File

The final step in the export process is to transfer the output file over to the SMTP system. The configuration file parameter, END_RUN_FILE=, provides a convenient means of initiating this step. By specifying the file name of a command procedure, a program can be called to transport the output file over to the SMTP system.

Appendix C
Sample PROFS Synchronization

This appendix gives an example of how to synchronize an IBM PROFS directory with X.500. All of the steps required to set up and configure Synchronizer-500 are shown in detail. Several synchronization runs are shown, along with the configuration files, log files, and results.

The PROFS directory file and the record description files (RDFs) used in these examples are distributed with the Synchronizer-500 software. These files are placed in the Synchronizer-500 root directory when Synchronizer-500 is installed.

C.1 Setting up and Running a Synchronization Operation

There are a number of steps involved in preparing for bi-directional synchronization of a foreign directory with the X.500 directory:

1. Create a Synchronizer-500 subdirectory for the import and export operations.
2. Determine how to transport data files between the foreign system and the Synchronizer-500 system.
3. Decide what information should be exchanged between the two directories.
4. Set up the import and export configuration files.
5. Set up the import and export RDFs.

Once the preparation is complete, the following list details the activities that perform both the import and export synchronization operations:

- Run the procedure to create the subdirectory.
- Extract a copy of the foreign directory to a file on the foreign system.
- Move the foreign directory file onto the Synchronizer-500 system.
- Run the import process to put the foreign directory file information into X.500.
- Run the export process to extract specified X.500 entries and write the entries to a file in the format required by the foreign directory.
- Move the X.500 export file over to the foreign system.

- Merge the X.500 export file into the foreign directory.

Note

Synchronizer-500 does not extract data from a foreign directory, transport it to the system where Synchronizer-500 is installed, or merge the exported X.500 data back into the foreign directory. These steps are pre- and post-Synchronizer-500 activities that must be accomplished to complete the directory update process.

The next sections describe PROFS directory synchronization and illustrate both the import and export operations.

C.2 Overview

IBM PROFS uses a directory service called the Nickname Manager. Nickname Manager contains a directory of all local and remote recipients. PROFS users can query the directory to find the USERID and NODE name of intended recipients. PROFS users use this information to address a note or document.

The PROFS addressing scheme consists of a USERID and NODE, each of which can be up to eight characters long. Other information such as department name, first name, last name, and initials can also be included in the Nickname Manager.

Example C–1 shows entries from a typical PROFS Nickname Manager data file.

Note

The format of the Nickname Manager File is customizable. Therefore, the format used on your PROFS system may differ from the format shown in Example C–1.

Example C–1: Typical PROFS Nickname Manager File

```
APPLE           BOB         D APPLEC   SALES PROFSA
BAKER           DONALD      E BAKERD   SALES PROFSB
CHARLES         THOMAS      F CHARLEST ADMIN PROFSA
DELTA           BOBBY       G DELTAB   ADMIN PROFSB
EPSILON         EDWARD      H EPSILONE SALES PROFSA
FOXTROT         BILL        I FOXTROTB MFG   PROFSB
GOPHER          WILLIAM     J GOPHERW  ENG   PROFSA
HODDING         HAROLD      K HODDINGH ENG   PROFSB
INDIA           ELLEN       E INDIAE   SALES PROFSC
JENDERSON       THOMAS      L JENDERSO SRV   PROFSB
KELLISON        RICHARD     M KELLISON SRV   PROFSA
LOM             GEORGE      N LOMG     SWS   PROFSB
MONDALE         MIKE        K MONDALEM MFG   PROFSA
NILSON          LARRY       P NILSONL  MFG   PROFSB
OPPERHEIMER     LARRY       Q OPEERHEI MFG   PROFSA
PONDERANT       DONALD      P PONDERAN MFG   PROFSB
QUESOME         DONALD      R QUESOMED SALES PROFSB
RONDERSON       RONALD      S RONDERSO MFG   PROFSA
SONDHEIM        SONNY       S SONDHEIM ADMIN PROFSB
TAGGART         SUE         T TAGGARTS ENG   PROFSA
UNISON          NICK        U UNISONN  ENG   PROFSC
VANCE           WILLIAM     V VANCEW   SALES PROFSC
XRAY            MARIE       C XRAYM    MIS   PROFSC
WILLIAMSON      BILL        B WILLIAMS SRV   PROFSC
YELLOWSTONE     HARRY       B YELLOWST MANGT PROFSC
ZEBRA           ROGER       R ZEBRAR   DIS   PROFSC
ZIZI            JIM         Q ZIZIJ    TEST  PROFSD
```

Note

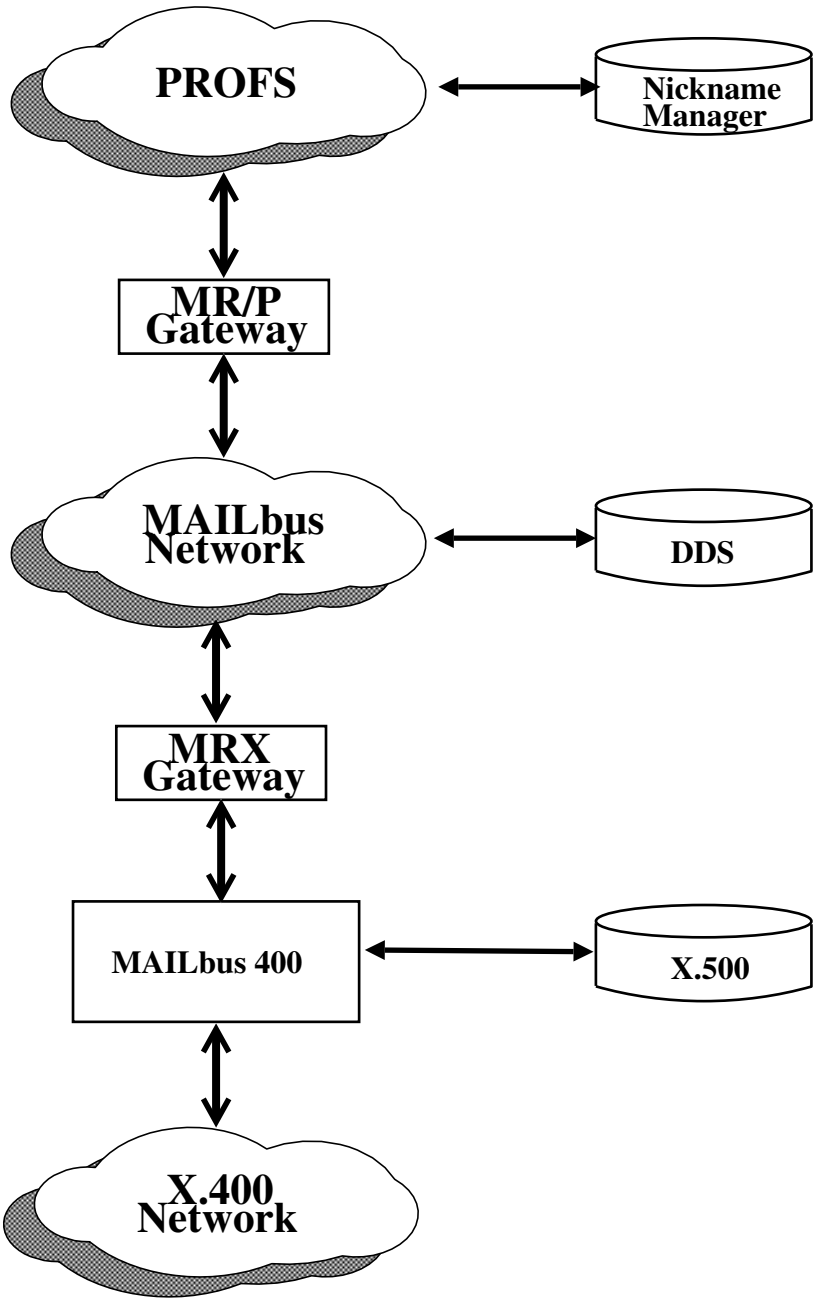
In this example, the data extracted from the Nickname Manager resides in fixed locations in the file, as required by Synchronizer-500. If it did not, then it would be necessary to process the file into a new format in which the fields were column-aligned. Synchronizer-500 does not provide this preprocessor.

For the purposes of this example, the network configuration shown in Figure C–1 is used.

In this example, a VAX node named MRPGWY is connected through a Message Router/PROFS (MR/P) gateway to a PROFS network, which includes PROFS nodes PROFSA, PROFSB, and PROFSC. The MAILbus network includes an MRX gateway which has a connection to a MAILbus 400 system. The MAILbus 400 system has connections to other X.400 systems.

Through this messaging network, PROFS users will be able to exchange messages with MAILbus users as well as recipients in the X.400 environment. PROFS users address mail to MAILbus and X.400 users as though the recipients were on a remote PROFS system (TO: MRPGWY(SMITHJ), for example). MAILbus users address mail to PROFS users by composing an address in Message Router syntax, for example, TO: PR_U=JONESJ @PR_L=PROFSA @MRP @MRPGWY, where JONESJ and PROFSA are the PROFS USERID and NODE, respectively. X.400 users address mail to PROFS users by composing an address in X.400 format, for example, TO: /s=JONES/g=JACOB/ou=MEL/o=DIGITAL/p=EMSSERVER/a=PUBLIC/c=US.

Figure C–1: PROFS Messaging Network Configuration



The MR/P gateway uses information from DDS to translate the addresses from PROFS to MAILbus format (which could be an X.400 address), and vice versa, as messages pass through the gateway. DDS contains a PROFS alias, which is an address in the USERID/NODE format

expected by PROFS, for each MAILbus/X.400 user. To give PROFS users the ability to look up MAILbus/X.400 recipients, the Nickname Manager must be loaded with these aliases. In addition, DDS must be loaded with information from the Nickname Manager so that MAILbus users can look up PROFS recipients, and X.500 must be loaded with information from the Nickname Manager so that X.400 users can look up PROFS recipients.

The following sections explain the process of importing the information from the PROFS Nickname Manager into X.500. For information on how to import the PROFS Nickname Manager directory into DDS, see the *MAILbus Directory Synchronizer Configuration and Management Guide*.

C.3 Import Process

The first step in the import process is to create a subdirectory which will contain the PROFS Nickname Manager data file, the RDF file and the configuration file. These are all the files that Synchronizer-500 needs to execute the import process.

The next step in preparing to import information from PROFS is to transfer the Nickname Manager data file to the system running Synchronizer-500 (it does not matter whether the transfer is accomplished via file transfer, mail message, or tape). Once this has been done:

- The format of the data file should be examined.
- A decision should be made as to how to map the data to X.500 attributes.
- The RDF(s) should be created to reflect this data correspondence.
- The configuration file(s) should be set up with the proper parameters.

The import process can then be run.

C.3.1 Create the PROFS Import Subdirectory

Create a Synchronizer-500 subdirectory for this instance. See Section 3.3.1 for the instructions on how to run the xdsu_config_instance procedure to create a subdirectory for the PROFS import run.

When the subdirectory has been created, set your current directory to the newly created directory and proceed with the next steps below.

C.3.2 Determine Format of the PROFS File

Once the Nickname Manager directory file has been transferred to the system running Synchronizer-500, it should be placed in the subdirectory created as described in Section C.3.1 above.

Analysis of the Nickname Manager data file, shown in Example C–1, yields the following mappings:

```
Nickname                                X.500
 Field        Start      Length     Attribute
-----------------------------------------------

LASTNAME        2          16      SURNAME
LASTNAME        2          16      emailSURNAME
FIRSTNAME      18          12      emailGIVENNAME
INITIAL        30           2      emailINITIALS
USERID         32           8      emailPRUSERNAME
DEPARTMENT     41           6      emailORGUNIT
NODE           47           8      emailPRLOCATION
```

In addition to these mappings, a MAILbus and an X.400 address must be determined for the PROFS users.

MAILbus addresses consist of three parts: MHSUSERID, MHSMAILBOX, and ADDRPART. The ADDRPART field is the VMS NODE executing the Message Router PROFS Gateway (MRPGWY). The MHSMAILBOX field is the name of the Message Router mailbox serviced by the PROFS Gateway (MR/P). The MHSUSERID field is constructed according to the following algorithm:

```
PR_U=xxxxxxxx @PR_L=yyyyyyyy
```

where *xxxxxxxx* is the eight-character PROFS USERID and *yyyyyyyy* is the eight-character PROFS NODE.

X.400 addresses consist of several standard attributes:

SURNAME (S)
GIVENNAME (G)
INITIALS (I)
4 ORGANIZATIONAL UNITS (OU1-OU4)
ORGANIZATION (O)
PRMD (P)
ADMD (A)
COUNTRY (C)

The SURNAME, GIVENNAME, and INITIALS will be mapped from the data file. The remainder of the attributes will contain constant values.

Once all the fields in the Nickname Manager data file have been mapped to attributes in X.500 the import RDF can be set up.

C.3.3 Writing the Import RDF

Next the RDF (xdsu_import_profs.rdf) is created to tell Synchronizer-500 which RDNs and attributes to create in X.500 and where to get the values for those RDNs and attributes. In addition, the RDF will contain constant values for information that is the same for all of the PROFS users, such as X.500 RDNs and X.400 GDI information. (The xdsu_import_profs.rdf is included in the Synchronizer-500 root directory and should be referred to while reading the following text.)

As explained in Chapter 3, each line in the RDF consists of an X.500 attribute keyword, a record number, a starting column number, and a length value. The record number is used in cases where there is more than one line per user in the foreign directory file. Because the Nickname Manager data file does not use more than one line per user, the record number is always *1*.

As further explained in Chapter 3, an RDF record may contain a conditional clause. These conditional clauses test for the existence or non-existence of data. Some of the RDF statements in this example make use of conditional statements, prefaced with the question mark ? or tilde ~ character.

The beginning part of the RDF defines the values for the RDNs. Because this information is not contained in the input file, the first three RDF records tell Synchronizer-500 to define constant values for the X.500 RDNs.

```
country="US"
organization="Digital"
organizationalUnit="NAA"
```

The next record maps the lastname into the surname attribute.

```
surname=1.2.16
```

The next five RDF records build the commonName RDN. The first four records of this set are conditional, that is, they are only executed if data exists in the specified location. Only the SURNAME value is mandatory in this commonName RDN.

The records:

```
?1.18.12:commonName=1.18.12
?1.18.12:commonName=" "
```

test for the existence of the PROFS FIRSTNAME (using the question mark ? character) and puts the FIRSTNAME and a space character in the commonName, if data exists.

The next two records:

```
?1.30.1:commonName=1.30.1
?1.30.1:commonName=". "
```

test for the existence of the PROFS INITIAL and put the INITIAL and the string, ". ", in the commonName, if data exists.

Because the SURNAME value is mandatory for commonName, the PROFS field, LASTNAME, is moved to the commonName attribute:

```
commonName=1.2.16
```

The remaining RDF records map information into the emailInfo auxiliary class.

Map the PROFS user's FIRSTNAME, INITIAL, and LASTNAME data into the X.500 email-GIVENNAME, emailINITIALS, and emailSURNAME attributes. This information becomes part of the user's X.400 address. This mapping is done by the following records:

```
emailGIVENNAME=1.18.12
emailINITIALS=1.30.1
emailSURNAME=1.2.16
```

The next five RDF records build a common name for the emailName attribute. The emailName attribute is built like the commonName RDN above. The first four records of this set are conditional, that is, they are only executed if data exists in the specified location. Only the SURNAME field is mandatory in this common name expression.

```
?1.18.12:emailNAME=1.18.12
?1.18.12:emailNAME=" "
?1.30.1:emailNAME=1.30.1
?1.30.1:emailNAME=". "
emailNAME=1.2.16
```

Conditional RDF statements can be used to build a human-readable alias value primarily for use in PC LAN lookups. The format of this value is SURNAME-GIVENNAME-INITIALS. In this example, the emailUSE5 attribute will be used to hold the alias and Synchronizer-500 will guarantee its uniqueness as shown in the configuration file, Example C–2. This value will be constructed by testing for values in SURNAME, GIVENNAME, and INITIALS and then concatenate the values separated by a dash - character.

```
emailUSE5=1.2.16
?1.18.12:emailUSE5="-"
?1.18.12:emailUSE5=1.18.12
?1.30.1:emailUSE5="-"
?1.30.1:emailUSE5=1.30.1
```

If for some reason the above combination does not yield a unique value, Synchronizer-500 appends numbers to the end of the string to ensure uniqueness.

The X.400 address is completed by providing the ORGUNIT (from the DEPARTMENT field) and constant values for the ORGNAME, PRMD, ADMD, and COUNTRY:

```
emailORGUNIT=1.41.5
emailORGNAME="Digital"
emailPRMDNAME="EMSERVER"
emailADMDNAME="PUBLIC"
emailPCOUNTRY="US"
```

If no data resides in the DEPARTMENT field, an ORGUNIT value should be provided by using a negative conditional RDF statement to ensure that a properly designed name space for X.400 can be created:

```
~1.41.5:emailORGUNIT="Washington DC"
```

The actual PROFS NODE and USERID should be mapped in from the input file to the X.500 attributes emailPRLOCATION and emailPRUSERNAME.

```
emailPRUSERNAME=1.32.8
emailPRLOCATION=1.47.8
```

Because our example configuration also includes a SNADS connection, Synchronizer-500 will be used to generate the SNADS USERID. (See Example C–2). The SNADS location is provided as a constant value:

```
emailSNLOCATION="NACGWYR1"
```

The emailMHSUSERID attribute must be created with information that resides in two separate places in the Nickname Manager File (USERID and NODE). The information in these fields must be combined with other terms ("PR_U=" and "@PR_L=") to complete the value:

```
emailMHSUSERID="PR_U="
 emailMHSUSERID=1.32.8
 emailMHSUSERID="@PR_L="
 emailMHSUSERID=1.47.8
```

When Synchronizer-500 sees an RDF line repeated, it concatenates those fields together. Thus, the lines shown above result in a emailMHSUSERID of:

```
PR_U=xxxxxxxx@PR_L=yyyyyyyy
```

where *xxxxxxx* and *yyyyyyyy* are the PROFS USERID and NODE terms extracted from the indicated location in the Nickname Manager File.

Finally, the MAILbus address is added in the form:

```
MHSUSERID @MHSMAILBOX @ADDRPART
```

Because the name of the MHSMAILBOX and VAX node name needed for ADDRPART do not appear in the Nickname Manager File, they must be provided as constants in the RDF by putting in the values required as quoted strings:

```
emailMHSMAILBOX="MRP"
emailADDRPART="MRPGWY"
```

Now all of the components of the MAILbus address need to be put into the emailMAILbusAddr attribute.

```
emailMAILbusAddr="PR_U="
emailMAILbusAddr=1.32.8
emailMAILbusAddr="@PR_L="
emailMAILbusAddr=1.47.8
emailMAILbusAddr="@MRP@MRPGWY"
```

The last part of the RDF constructs the 1984 and 1988 X.400 addresses. The 1984 address is put in the emailX400Addr84 attribute and the 1988 address is put in the emailX400Addr88 attribute.

The 1984 X.400 address is constructed first. The existence of GIVENNAME is checked first. If a value is there, it is concatenated to the constant identifier and is followed by a semi-colon.

```
?1.18.12:emailX400Addr84="G="
?1.18.12:emailX400Addr84=1.18.12
?1.18.12:emailX400Addr84=";"
```

The next three records check for the existence of an initial in the Nickname Manager file. If it is there, put in the constant identifier for initials and then the value, followed by a semi-colon.

```
?1.30.1:emailX400Addr84=" I="
?1.30.1:emailX400Addr84=1.30.1
?1.30.1:emailX400Addr84=";"
```

Because SURNAME is mandatory in the X.400 address, no conditional RDF syntax is used here. Put the constant identifier in the attribute along with the value.

```
emailX400Addr84=" S="
emailX400Addr84=1.2.16
emailX400Addr84=";"
```

The next part of the X.400 address is the ORGUNIT. First check to see if data is in the department field in the input file. If no data exists, provide a constant for the ORGUNIT value.

```
?1.41.5:emailX400Addr84=" OU1="
 ?1.41.5:emailX400Addr84=1.41.5
 ?1.41.5:emailX400Addr84=";"
 ~1.41.5:emailX400Addr88=" OU1=Washington DC;"
```

The final parts of the 1984 X.400 address, are the ORGANIZATION, PRMD, ADMD, and COUNTRY attributes. Constants will be supplied for each of these values.

```
emailX400Addr84=" O=Digital; P=EMSERVER; A=PUBLIC; C=US"
```

As explained previously in this Appendix, when Synchronizer-500 sees an RDF keyword repeated, it concatenates those attributes together. Thus the resultant emailX400Addr84 attribute would be:

```
emailX400Addr84= G=gggggg; I=i; S=ssssssss; OU=ououou ;O=Digital;
                 P=EMSERVER; A=PUBLIC; C=US
```

where *gggggg, i, sssssss* and *ououou* are the GIVENNAME, INITIALS, SURNAME, and ORGANIZATIONAL UNIT.

Lastly the 1988 X.400 address is constructed. The same logic is used to build both the 1984 and 1988 X.400 addresses. The only difference is that in the 1988 address the term Common Name (CN) is used instead of separate attributes for GIVENNAME, INITIALS, and SURNAME.

```
emailX400Addr88="CN="
?1.18.12:emailX400Addr88=1.18.12
?1.18.12:emailX400Addr88=" "
?1.30.1:emailX400Addr88=1.30.1
?1.30.1:emailX400Addr88=" "
emailX400Addr88=1.2.16
emailX400Addr88=";"
?1.41.5:emailX400Addr88=" OU1="
?1.41.5:emailX400Addr88=1.41.5
?1.41.5:emailX400Addr88=";"
~1.41.5:emailX400Addr88=" OU1=Washington DC;"
emailX400Addr88=" O=Digital; P=EMSERVER; A=PUBLIC; C=US"
```

The resultant emailX400Addr88 attribute value would be:

```
emailX400Addr88= CN=gggggg i. ssssssss; OU=ououou ;O=Digital;
                 P=EMSERVER; A=PUBLIC; C=US
```

where *gggggg, i, sssssssss* and *ououou* are the GIVENNAME, INITIALS, SURNAME, and ORGANIZATIONAL UNIT.

That completes the set up of the RDF file. The next step is to set up the configuration file.

C.3.4 Setting up the Import Configuration File

The configuration file (xdsu_config.dat) for this example is set up as shown in Example C–2. This file (xdsu_config.dat_import_profs) is included in the Synchronizer-500 root directory and should be referred to while reading the following text.

Example C–2: Configuration File for PROFS Import

```
synch_type=IMPORT
!
dn_sequence=country/organization/organizationalUnit/commonName
create_root=organizationalUnit
synch_object=Person,organizationalPerson,emailInfo
bind_dsa=
!
rdf_file=xdsu_import_profs.rdf
initial_load=YES
input_for_dir_file=xdsu_import_profs.txt
nrec_subscr=1
gid_field=description
gid_match=PROFS
max_transactions=
max_fail_transactions=
max_trans_file=
end_run_file=xdsu_import_profs.com
notify=SYSTEM
synch_id=X5Synch
synch_id_field=emailUSE1
unique_field=emailUSE5
unique_field_len=30
gen_profs_alias=
gen_snads_alias=YES
old_metafile=
new_metafile=
log_file=
verbose=
encode_delim=|
synch_label=PROFS (OfficeVision) Import
make_unique=
truncate=
admd_space=
```

This file is an example of a configuration file for an import synchronization and initial load. The example RDF file xdsu_import_profs.rdf, example data file xdsu_import_profs.txt, and example end_run_file xdsu_import_profs.com are used. (These files are contained in the Synchronizer-500 root directory.)

In this example:

- The distinguished name that will be built is made up of the RDNs, country, organization, organizationalUnit, and commonName.
- When the entries are added, create the superior entry for organizationalUnit if needed.
- When entries are imported, they are specified to be of the Person, organizationalPerson and emailInfo classes.
- The example RDF, xdsu_import_profs.rdf and example data file are used.
- The X.500 attribute DESCRIPTION is used to store the Group ID, PROFS.
- Upon termination of the Synchronizer-500 run, the command procedure xdsu_import_profs.com is executed.
- The SYNCH_ID, X5Synch, is stored in the X.500 attribute emailUSE1.
- The X.500 attribute emailUSE5 is guaranteed by Synchronizer-500 to be unique, up to 30 characters.
- A SNADS alias is generated.
- The label "PROFS (OfficeVision) Import" is used in all Synchronizer-500 mail messages and summary reports.

C.3.5 Running the Import—The Initial Load

Synchronizer-500 is started by issuing the following command from the directory where the configuration, RDF, and PROFS data files reside:

```
OpenVMS:

$ XDSU :== $SYS$SYSTEM:XDSU$MAIN.EXE
$ XDSU

UNIX:

/usr/etc/xdsu
```

As Synchronizer-500 executes, its processing is logged to your terminal device as shown in Example C–3. The processing should be examined as it occurs to review the results of the initial load. When the initial load finishes, a mail message containing a summary of the run is sent to the address specified by the configuration file parameter "notify=".

The results of the initial load should be examined in the transaction log file xdsu_log.log (Example C–4).

Example C–3: Initial Load of PROFS Users Processing

```
Feb 28 16:55:56 processing layout file [-]XDSU_LAYOUT.DAT
Feb 28 16:55:56 processing configuration file XDSU_CONFIG.DAT
Feb 28 16:55:57 initializing and binding to X.500
Feb 28 16:55:57 processing RDF file xdsu_import_profs.rdf
Feb 28 16:55:57 converting input file xdsu_import_profs.txt to new metafile
Feb 28 16:55:58 27 input records read
Feb 28 16:55:58 27 new metafile records written
Feb 28 16:55:58 sorting new metafile
Feb 28 16:55:59 searching X.500 directory data
Feb 28 16:55:59 found 0 X.500 directory entries
Feb 28 16:55:59 extracting X.500 directory data to old metafile
Feb 28 16:55:59 extracted 0 directory entries into old metafile
Feb 28 16:55:59 comparing old and new metafiles
Feb 28 16:55:59 processing ADD records
Feb 28 16:56:09 27 records added
Feb 28 16:56:09 creating summary file
Feb 28 16:56:09 executing end run file xdsu_import_profs.com

XDSU V1.2
PROFS (OfficeVision) Import run started at: Tue Feb 28 16:55:56 1995

IMPORT SYNCH SUMMARY:

RECORDS READ/WRITTEN SUMMARY:

Foreign directory file records read: 27
New metafile records written: 27
Old metafile records written: 0

TRANSACTION SUMMARY:

Number of successful transactions: 27
Number of failed transactions:     0
Number of successful add transactions:    27
Number of successful modify transactions: 0
Number of successful delete transactions: 0
Number of failed add transactions:    0
Number of failed modify transactions: 0
Number of failed delete transactions: 0

CONFIGURATION FILE SUMMARY:
```

Example C–3 Cont'd on next page

Example C–3 (Cont.): Initial Load of PROFS Users Processing

```
synch_type: IMPORT
synch_label: PROFS (OfficeVision) Import
bind_dsa:
rdf_file: xdsu_import_profs.rdf
input_for_dir_file: xdsu_import_profs.txt
nrec_subscr: 1
log_file: XDSU_LOG.LOG
end_run_file: xdsu_import_profs.com
old_metafile:
new_metafile:
initial_load: YES
notify:
verbose:
encode_delim: |
max_transactions:
max_fail_transactions:
max_trans_file:
dn_sequence: country/organization/organizationalUnit/commonName
create_root: organizationalUnit
search_context:
synch_object: Person,organizationalPerson,emailInfo
synch_id: X5Synch
synch_id_field: emailUSE1
gid_field: description
gid_match: PROFS
gen_profs_alias:
gen_snads_alias: YES
admd_space:
unique_field: emailUSE5
unique_field_len: 30
make_unique:
truncate:
unique_field_separator:
unique_search_context:

PROFS (OfficeVision) Import run ended at: Tue Feb 28 16:56:09 1995
```

Example C–4: Checking the Results of the Initial Load Import—xdsu_log.log

```
PROFS (OfficeVision) Import run started at: Tue Feb 28 16:55:57 1995

A INFORMATIONAL Parent Entry created: /C=US/O=Digital/OU=NAA

A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=BILL B. WILLIAMSON  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=BILL I. FOXTROT  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=BOB D. APPLE  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=BOBBY G. DELTA  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=DONALD E. BAKER  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=DONALD P. PONDERANT  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=DONALD R. QUESOME  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=EDWARD H. EPSILON  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=ELLEN E. INDIA  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=GEORGE N. LOM  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=HAROLD K. HODDING  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=HARRY B. YELLOWSTONE  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=JIM Q. ZIZI  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=LARRY P. NILSON  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=LARRY Q. OPPERHEIMER  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=MARIE C. XRAY  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=MIKE K. MONDALE  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=NICK U. UNISON  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=RICHARD M. KELLISON  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=ROGER R. ZEBRA  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=RONALD S. RONDERSON  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=SONNY S. SONDHEIM  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=SUE T. TAGGART  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=THOMAS F. CHARLES  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=THOMAS L. JENDERSON  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=WILLIAM J. GOPHER  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=WILLIAM V. VANCE  - status=SUCCESS

PROFS (OfficeVision) Import run ended at: Tue Feb 28 16:56:10 1995
```

Examining the log in Example C–4 shows that a superior entry for /C=US/O=Digital/OU=NAA has been created and the entries for the PROFS users have been correctly entered. The first term, A, indicates an ADD operation; the second term, NORMAL, shows that overall success of the indicated operation (NORMAL or FAILED). The third term, /C=US/O=DIGITAL..., is the distinguished name for the entry that was added. The last term, status, shows the status for a transaction. See Appendix G for a list of transaction status codes.

Example C–5 shows an X.500 entry created as a result of the import run.

Example C–5: X.500 Entry for PROFS User

```
dxim> show /C=US/O=digital/OU=NAA/CN="BOB D. APPLE"

    /C=US/O=digital/OU=NAA/CN="BOB D. APPLE"
      Common Name    = BOB D. APPLE
      Object Class   = Person
                     = Organizational Person
                     = emailInfo
      Surname        = APPLE
      Description    = PROFS
      emailX400Addr84 = G=BOB; I=D; S=APPLE; OU1=SALES; O=Digital; P=EMSERVER;
                        A=PUBLIC; C=US
      emailX400Addr88 = CN=BOB D APPLE; OU1=SALES; O=Digital; P=EMSERVER;
                        A=PUBLIC; C=US
      emailMAILbusAddr = PR_U=APPLEC@PR_L=PROFSA@MRP@MRPGWY
      emailPCOUNTRY  = US
      emailINITIALS  = D
      emailPRLOCATION = PROFSA
      emailPRUSERNAME = APPLEC
      emailSNLOCATION = NACGWYR1
      emailSNUSERNAME = APPLEB
      emailADMDNAME  = PUBLIC
      emailGIVENNAME = BOB
      emailMHSMAILBOX = MRP
      emailADDRPART  = MRPGWY
      emailORGUNIT   = SALES
      emailPRMDNAME  = EMSERVER
      emailNAME      = BOB D. APPLE
      emailORGNAME   = Digital
      emailSURNAME   = APPLE
      emailUSE1      = X5Synch
      emailUSE5      = APPLE-BOB-D
      emailMHSUSERID = PR_U=APPLEC@PR_L=PROFSA
```

C.3.6 Running the Import Following the Initial Load

Once the initial load of PROFS users has been performed, each new import run generates only the ADD/MODIFY/DELETE transactions needed to update X.500. For import runs after the initial load, the configuration file flag, INITIAL_LOAD, must be changed from YES to NO.

When Synchronizer-500 receives a new copy of the Nickname Manager File for import after the initial load, the data is placed into a *new* metafile. The existing entries (from the initial load) reside in an *old* metafile. Synchronizer-500 compares the entries in the two metafiles and, from that comparison, generates ADD/MODIFY/DELETE transactions to update X.500.

Example C–6 shows the processing that results from the import update run. In this example, the entry for BAKER is deleted, the entry for CHARLES is modified, and an entry for a new user named ANDERSON is added. Following the import run, the log file should be checked as was done with the initial load run.

Example C–6: Second Import Run Processing

```
Feb 28 17:18:55 processing layout file [-]XDSU_LAYOUT.DAT
Feb 28 17:18:55 processing configuration file XDSU_CONFIG.DAT
Feb 28 17:18:55 initializing and binding to X.500
Feb 28 17:18:56 processing RDF file xdsu_import_profs.rdf
Feb 28 17:18:56 converting input file xdsu_import_profs.txt to new metafile
Feb 28 17:18:56 27 input records read
Feb 28 17:18:56 27 new metafile records written
Feb 28 17:18:56 sorting new metafile
Feb 28 17:18:57 searching X.500 directory data
Feb 28 17:18:57 found 27 X.500 directory entries
Feb 28 17:18:57 extracting X.500 directory data to old metafile
Feb 28 17:18:59 extracted 27 directory entries into old metafile
Feb 28 17:18:59 sorting old metafile
Feb 28 17:19:00 comparing old and new metafiles
Feb 28 17:19:00 processing DELETE records
Feb 28 17:19:00 1 records deleted
Feb 28 17:19:00 processing MODIFY records
Feb 28 17:19:00 1 records modified
Feb 28 17:19:00 processing ADD records
Feb 28 17:19:01 1 records added
Feb 28 17:19:01 creating summary file
Feb 28 17:19:01 executing end run file xdsu_import_profs.com

XDSU V1.2
PROFS (OfficeVision) Import run started at: Tue Feb 28 17:18:55 1995

IMPORT SYNCH SUMMARY:

RECORDS READ/WRITTEN SUMMARY:

Foreign directory file records read: 27
New metafile records written: 27
Old metafile records written: 27

TRANSACTION SUMMARY:

Number of successful transactions: 3
Number of failed transactions:     0
Number of successful add transactions:    1
Number of successful modify transactions: 1
Number of successful delete transactions: 1
Number of failed add transactions:    0
Number of failed modify transactions: 0
Number of failed delete transactions: 0

CONFIGURATION FILE SUMMARY:
```

Example C–6 Cont'd on next page

Example C–6 (Cont.): Second Import Run Processing

```
synch_type: IMPORT
synch_label: PROFS (OfficeVision) Import
bind_dsa:
rdf_file: xdsu_import_profs.rdf
input_for_dir_file: xdsu_import_profs.txt
nrec_subscr: 1
log_file: XDSU_LOG.LOG
end_run_file: xdsu_import_profs.com
old_metafile:
new_metafile:
initial_load: NO
notify:
verbose:
encode_delim: |
max_transactions:
max_fail_transactions:
max_trans_file:
dn_sequence: country/organization/organizationalUnit/commonName
create_root: organizationalUnit
search_context:
synch_object: Person,organizationalPerson,emailInfo
synch_id: X5Synch
synch_id_field: emailUSE1
gid_field: description
gid_match: PROFS
gen_profs_alias:
gen_snads_alias: YES
admd_space:
unique_field: emailUSE5
unique_field_len: 30
make_unique:
truncate:
unique_field_separator:
unique_search_context:

PROFS (OfficeVision) Import run ended at: Tue Feb 28 17:19:01 1995
```

Example C–7: Second Import Run Transaction Log—xdsu_log.log

```
PROFS (OfficeVision) Import run started at: Tue Feb 28 17:18:55 1995

D NORMAL /C=US/O=DIGITAL/OU=NAA/CN=DONALD E. BAKER  - status=SUCCESS
M NORMAL /C=US/O=DIGITAL/OU=NAA/CN=THOMAS F. CHARLES  - status=SUCCESS
A NORMAL /C=US/O=DIGITAL/OU=NAA/CN=PATRICK T. ANDERSON  - status=SUCCESS

PROFS (OfficeVision) Import run ended at: Tue Feb 28 17:19:02 1995
```

C.4 Export Process

Exporting X.500 information is very similar to importing information into X.500. A Synchronizer-500 subdirectory for this instance is created, the xdsu_config.dat file is set up with the appropriate keywords and the RDF file is set up to define the attributes to be exported and their location in the output file.

C.4.1 Create the PROFS Export Subdirectory

Create a Synchronizer-500 subdirectory for this instance. See Section 3.3.1 for the instructions on how to run the xdsu_config_instance procedure to create a subdirectory for the PROFS export run.

When the subdirectory has been created, set your current directory to the newly created directory and proceed with the next steps below.

C.4.2 Writing the Export RDF

The format of the output of the Synchronizer-500 export run must be in the Nickname Manager File format. Example C–8 shows an example of the type of RDF to use to accomplish this formatting. This file (xdsu_export_profs.rdf) is included in the Synchronizer-500 root directory and should be referred to while reading the following text.

Example C–8: RDF for PROFS Export - xdsu_export_profs.rdf

```
! XDSU_EXPORT_PROFS.RDF
! Export Synch to IBM PROFS or Office Vision V1R1
! 11-October-1993
!
emailSURNAME=1.2.16
emailGIVENNAME=1.18.11
emailINITIALS=1.30.1
emailPRUSERNAME=1.32.8
emailPRLOCATION=1.47.8
emailORGUNIT=1.41.5
```

The RDF in Example C–8 specifies:

- Write the emailSURNAME attribute starting at record 1, column 2, for a total of 16 characters.
- Write the emailGIVENNAME attribute starting at column 18 for a total of 11 characters.
- Write the emailINITIALS attribute starting at column 30 for a total of 1 character.
- Write the emailPRUSERNAME attribute starting at column 32 for a total of 8 characters.
- Write the emailPRLOCATION field starting at column 47 for a total of 8 characters.
- Write the emailORGUNIT field starting at column 41 for a total of 5 characters.

C.4.3 Setting up the Export Configuration File

The next step is to set up the export configuration file (xdsu_config.dat) as shown in Example C–9. This file (xdsu_config.dat_export_profs) is included in the Synchronizer-500 root directory and should be referred to while reading the following text.

Example C–9: Configuration File for PROFS Export

```
synch_type=EXPORT
!
bind_dsa=
search_context=/country=us/organization=digital
!
rdf_file=xdsu_export_profs.rdf
input_old_metafile=
output_old_metafile=
output_for_dir_file=export_profs.dat
nrec_subscr=1
gid_field=description
gid_match=*
gid_exclude=PROFS
country_match=
admd_match=
prmd_match=
orgname_match=
orgunit_match=
orgunit2_match=
orgunit3_match=
orgunit4_match=
end_run_file=copy_to_profs.com
notify=SYSTEM
synch_id=X5Synch
synch_label=PROFS Export
get_field=
get_value=
```

This file is an example of a configuration file for an export synchronization. The example RDF xdsu_export_profs.rdf is used. (This file is contained in the Synchronizer-500 root directory.)

The example export configuration file consists of keyword and value pairs. In this example, not all keywords are used. The unused keywords can remain blank or can be removed from the configuration file. Descriptions of the example values are given below. Please see Appendix F for more detailed information.

In this example:

- The search will start on the X.500 tree /country=us/organization=digital.
- The RDF, xdsu_export_profs.rdf, is used.
- The output file, export_profs.dat, is generated with 1 record per entry.
- The X.500 attribute DESCRIPTION is used to extract every entry (GID_MATCH=*) except those with a DESCRIPTION attribute value of PROFS (GID_EXCLUDE=PROFS).
- Upon termination of this Synchronizer-500 run, the command procedure copy_to_ profs.com is executed.

- The SYSTEM account is notified of the start, end, and any errors in this Synchronizer-500 run.
- The label "PROFS Export" is used in all Synchronizer-500 mail messages and summary reports.

C.4.4 Running the Export

Synchronizer-500 is started by issuing the following command from the directory where the configuration file and RDF reside:

```
OpenVMS:

$ XDSU :== $SYS$SYSTEM:XDSU$MAIN.EXE
$ XDSU

UNIX:

/usr/etc/xdsu
```

As Synchronizer-500 executes, its processing is logged to your terminal device as shown in Example C–10. The processing should be examined as it occurs to review the results of the export run. When the export run finishes, a mail message containing a summary of the run is sent to the address specified by the configuration file parameter "notify=".

Example C–10: PROFS Export Processing

```
Feb 28 17:32:53 processing layout file [-]XDSU_LAYOUT.DAT
Feb 28 17:32:53 processing configuration file XDSU_CONFIG.DAT
Feb 28 17:32:53 initializing and binding to X.500
Feb 28 17:32:54 processing RDF file xdsu_export_profs.rdf
Feb 28 17:32:54 searching X.500 directory data
Feb 28 17:32:54 found 10 X.500 directory entries
Feb 28 17:32:54 extracting X.500 directory data to export metafile
Feb 28 17:32:55 extracted 10 directory entries into export metafile
Feb 28 17:32:55 sorting export metafile
Feb 28 17:32:56 converting metafile to output file export_profs.dat
Feb 28 17:32:56 10 metafile records read
Feb 28 17:32:56 10 output file records written
Feb 28 17:32:56 creating summary file
Feb 28 17:32:56 executing end run file copy_to_profs.com

XDSU V1.2
PROFS Export run started at: Tue Feb 28 17:32:52 1995

EXPORT SYNCH SUMMARY:

RECORDS READ/WRITTEN SUMMARY:

Old metafile records written: 10
Old metafile records read:    10
Foreign directory file records written: 10

CONFIGURATION FILE SUMMARY:
```

Example C–10 Cont'd on next page

Example C–10 (Cont.): PROFS Export Processing

```
synch_type: EXPORT
synch_label: PROFS Export
bind_dsa:
rdf_file: xdsu_export_profs.rdf
output_for_dir_file: export_profs.dat
nrec_subscr: 1
end_run_file: copy_to_profs.com
output_old_metafile:
input_old_metafile:
notify: SYSTEM
verbose:
encode_delim:
search_context: /country=us/organization=digital
synch_id: X5Synch
synch_id_field:
gid_field: description
gid_match: *
gid_exclude: PROFS
country_match:
admd_match:
prmd_match:
orgname_match:
orgunit_match:
orgunit2_match:
orgunit3_match:
orgunit4_match:
get_field:
get_value:

PROFS Export run ended at: Tue Feb 28 17:32:56 1995
```

An examination of the output file, export_profs.dat, (shown in Example C–11) confirms that the information has been extracted from X.500 and formatted correctly.

Example C–11: Exported X.500 Data in Nickname Manager Format

```
Guest           Our           GuestO   MEL   NJUSDCOA
Rotary          Sharon        RotaryS  MEL   NJUSDCOA
Blue            Suzanne       BlueS    MEL   NJUSDCOA
Reed            Mary          ReedM    MEL   NJUSDCOA
ADMIN                         ADMIN    MEL   NJUSDCOA
Smith           Byron         SmithB   MEL   NJUSDCOA
Jones           Joyce         JonesJ   CYO   NJUSDCOA
Peterson        Sam           PetersoS CYO   NJUSDCOA
Edwards         Peterson      EdwardsP CYO   NJUSDCOA
```

C.4.5 Transfer the Synchronizer-500 Export Output File

The final step in the export process is to transfer the output file to the PROFS system so that it can be merged into the Nickname Manager File. This step could be accomplished by the end-of-run file, COPY_TO_PROFS.COM.

Appendix D
X.500 Directory Attributes

The following tables contain the default list of X.500 directory attributes that are accessible through Synchronizer-500 as defined in the Synchronizer-500 layout file (xdsu_layout.dat).

Table D–1 contains the default list of standard X.500 directory attributes in the DEC X.500 Directory Service schema available for use with Synchronizer-500. (See Appendix A of the *DEC X.500 Directory Service Management Guide*). These represent attributes contained in the Person, organizationalPerson, organizationalRole, residentialPerson, and decMailUser Object Classes.

Table D–2 lists the attributes in the emailInfo auxiliary object class which is supplied with Synchronizer-500.

The RDF keyword, description, length, syntax, and number of RDN and attribute values is given for each attribute.

The syntaxes listed in the following tables is how Synchronizer-500 defines the syntax for the X.500 attributes. This many not necessarily be how the attribute syntax is listed in the *DEC X.500 Management Guide*. When Synchronizer-500 adds an attribute to X.500, the value will be specified according to the syntax listed in the table.

The #rdn and #attr values in the following tables show the default number of values that Synchronizer-500 will support per attribute. If additional RDN or attribute values are needed for a particular attribute, the Synchronizer-500 X.500 layout file (xdsu_layout.dat) will need to be customized. See Appendix H.

Note

If there are multiple values for an attribute, either under the #rdn values or #attr values columns, Synchronizer-500 appends "_n" to the keyword for values #2 through #n. For example, four values are allowed for the attribute "title". The RDF keywords would be "title", "title_2", "title_3", and "title_4". When multiple values are stored in an attribute, the order of the values cannot be guaranteed. Therefore, if the order is important, use multiple attributes to store the values instead of storing all values in one attribute.

Table D–1: Standard X.500 Directory Attributes

Keyword	Description	Len	Syntax	#rdn	#attr
country	country entries represent countries	2	printableString	1	0
organization	organization entries represent organizations	64	printableString	1	0
organizationalUnit	organizationalUnit entries represent subdivisions of organizations, such as departments, offices, or branches	64	printableString	4	0
locality	Locality entries represent a geographical locality, such as a state, a province, a city, or a building	128	printableString	4	0
commonName	Name by which an object is usually known. (e.g., John Smith)	64	teletexString	1	3
countryName	International code that represents a country's name	2	printableString	0	1
organizationName	Name of an organization	64	printableString	0	1
organizational UnitName	Name of an organization unit, such as team, group, department, or committee	64	printableString	0	4
localityName	Name of a locality or region, such as a city or county	128	printableString	0	3
surname	Surname or family name of a person	64	teletexString	0	2
givenName	Given name of a person	64	printableString	0	2
initials	Initials of a person	64	printableString	0	2
generationalQualifier	Generational Qualifier of a person	64	printableString	0	2
description	Descriptive information about an object	1024	printableString	0	1
seeAlso	Names of other directory entries that are in some way related to the entry of which this is an attribute	512	distinguishedName	0	1
telephoneNumber	Telephone number	32	printableString	0	4
title	Position, function, or occupational qualification of an object	64	printableString	0	4
userPassword	Password	128	octetString	0	1
stateOrProvinceName	Name of a state or province	128	printableString	0	4
streetAddress	Site for the local distribution and physical delivery in a postal address, road name and house number	128	printableString	0	4
physicalDelivery OfficeName	Name of the city, town, or village where the physical delivery office is situated	128	printableString	0	4

Table D–1 (Cont.): Standard X.500 Directory Attributes

Keyword	Description	Len	Syntax	#rdn	#attr
postalAddress	Address information required for the physical delivery of postal messages by the postal authority of the named object	185	postalAddress	0	4
postalCode	Postal code or zip code	40	printableString	0	4
postOfficeBox	Post Office Box by which the object receives physical postal delivery	40	printableString	0	4
destinationIndicator	Country and city associated with the named object, the addressee, required to provide the Public Telegram Service, according to CCITT Recommendations F.1 and F.3	128	printableString	0	4
facsimiletelephone Number	Telephone number for a facsimile terminal associated with the named object	32	facsimileTelephone Number	0	4
internationalISDN number	International ISDN Number associated with the named object	16	numericString	0	4
preferredDelivery Method	Preferred methods of physical message delivery	20	integerList	0	4
registeredAddress	Address associated with an object at a particular city location	185	postalAddress	0	4
teletexTerminal Identifier	Teletex terminal identifier for a teletex terminal	1024	teletexTerminal Identifier	0	4
telexNumber	Telex number, country code, and answerback code of a telex terminal	14	telexNumber	0	4
x121Address	Network address of the named object	15	numericString	0	4
roleOccupant	Distinguished name of an entry that represents the occupant of an organizational role	512	distinguishedName	0	1
businessCategory	Information concerning the occupation or field of interest of an object	128	numericString	0	4
mhs-or-addresses	X.400 messaging originator/recipient address	256	mhs-or-address	0	1
rfc822Mailbox	RFC 822 Mailbox Address	256	ia5String	0	1
decMRAddress	Message Router Address	256	teletexString	0	1
decPMAddress	Postmaster Mail Address	256	teletexString	0	1
decPreferred MailAddress	Preferred Mail Type Address	256	objectIdentifier	0	1
decMAILworks UserName	MAILworks User Name	256	teletexString	0	1

Table D–2: emailInfo Auxiliary Class X.500 Directory Attributes

Keyword	Description	Len	Syntax	#rdn	#attr
emailX400Addr84	1984 X.400 address	256	teletexString	0	1
emailX400Addr88	1988 X.400 address	256	teletexString	0	1
emailRFC822Addr	RFC822 address	256	teletexString	0	1
emailCCmailAddr	cc:Mail address	256	teletexString	0	1
emailMicrosoftAddr	Microsoft Mail address	256	teletexString	0	1
emailMAILbusAddr	MAILbus address	256	teletexString	0	1
emailQuickmailAddr	Quickmail address	256	teletexString	0	1
emailAddr08	Electronic mail alias address #8	256	teletexString	0	1
emailAddr09	Electronic mail alias address #9	256	teletexString	0	1
emailAddr10	Electronic mail alias address #10	256	teletexString	0	1
emailAddr11	Electronic mail alias address #11	256	teletexString	0	1
emailAddr12	Electronic mail alias address #12	256	teletexString	0	1
emailPCOUNTRY	X.400 Printable COUNTRY code	2	printableString	0	1
emailNCOUNTRY	Numeric COUNTRY code	3	numericString	0	1
emailGENERATION	Generation, e.g., Jr, Sr, III	3	printableString	0	1
emailINITIALS	Initials	5	printableString	0	1
emailLANGUAGE	Language preference	2	printableString	0	1
emailPRLOCATION	IBM PROFS NODE	8	printableString	0	1
emailPRUSERNAME	IBM PROFS USERID	8	printableString	0	1
emailSNLOCATION	IBM SNADS subscriber group	8	printableString	0	1
emailSNUSERNAME	IBM SNADS user id	8	printableString	0	1
emailMRX_FLAGS	Denies access to MRX for this subscriber if "1" 8		numericString	0	1
emailVMS NODENAME	VMS NODE part of VMSORADDRESS	8	printableString	0	1
emailADMDNAME	X.400 Administration domain name	16	printableString	0	1
emailNADMDNAME	Numeric Administrative domain code	16	numericString	0	1
emailGIVENNAME	First Name	16	printableString	0	1
emailGIVENNAME2	Nickname	16	printableString	0	1
emailINTLOCATION	Internal location code	20	teletexString	0	1
emailINTPHONE	Internal Telephone number	32	teletexString	0	1

Table D–2 (Cont.): emailInfo Auxiliary Class X.500 Directory Attributes

Keyword	Description	Len	Syntax	#rdn	#attr
emailLOCATION	Location	32	teletexString	0	1
emailMAIL ADDRESS	Internal postal mail address	20	teletexString	0	1
emailMHS MAILBOX	MHSMAILBOX part of MAILbus address	32	teletexString	0	1
emailMHS MAILBOX2	MHSMAILBOX part of MAILbus address #2	32	teletexString	0	1
emailADDRPART	Node part of MAILbus address #1	32	teletexString	0	1
emailADDRPART2	Node part of MAILbus address #2	32	teletexString	0	1
emailORGUNIT	Organization unit or workgroup	32	teletexString	0	1
emailORGUNIT2	Additional organization unit	32	teletexString	0	1
emailORGUNIT3	Additional organization unit	32	teletexString	0	1
emailORGUNIT4	Additional organization unit	32	teletexString	0	1
emailNPRMDNAME	Numeric Private domain code	16	numericString	0	1
emailPRMDNAME	X.400 Private domain name	16	printableString	0	1
emailPROXY	32 digit hex DDSID of the owning User Agent	32	printableString	0	1
emailTELEPHONE	Telephone number	32	teletexString	0	1
emailTERMINALID	Telex answerback number	24	teletexString	0	1
emailTITLE	Title, Example: Dr., Manager, etc.	32	teletexString	0	1
emailSUBID	Unique identifier	20	teletexString	0	1
emailUAID	Unique User Agent identifier	32	teletexString	0	1
emailVMS ADDRPART	VMS addrpart part of VMSORADDRESS	32	teletexString	0	1
emailVMS USERNAME	VMS username part of VMSORADDRESS	32	teletexString	0	1
emailX121 ADDRESS	Telex or FAX number	15	numericString	0	1
email1UNUSED1	Available for site specific use	40	teletexString	0	1
email1UNUSED2	Available for site specific use	40	teletexString	0	1
email1UNUSED3	Available for site specific use	40	teletexString	0	1
email1UNUSED4	Available for site specific use	40	teletexString	0	1
email2UNUSED1	Available for site specific use	40	teletexString	0	1

Table D–2 (Cont.): emailInfo Auxiliary Class X.500 Directory Attributes

Keyword	Description	Len	Syntax	#rdn	#attr
email2UNUSED2	Available for site specific use	40	teletexString	0	1
email2UNUSED3	Available for site specific use	40	teletexString	0	1
email2UNUSED4	Available for site specific use	40	teletexString	0	1
email3UNUSED1	Available for site specific use	40	teletexString	0	1
email3UNUSED2	Available for site specific use	40	teletexString	0	1
email3UNUSED3	Available for site specific use	40	teletexString	0	1
email3UNUSED4	Available for site specific use	40	teletexString	0	1
emailFF POSTADDR	Free Form Postal Address	40	teletexString	0	1
emailFF POSTADDR2	Additional Free Form Postal Address Information	40	teletexString	0	1
emailFF POSTADDR3	Additional Free Form Postal Address Information	40	teletexString	0	1
emailFF POSTADDR4	Additional Free Form Postal Address Information	40	teletexString	0	1
emailNAME	Common name	64	teletexString	0	1
emailORGNAME	Company Name or Organization name	64	teletexString	0	1
emailPOSNROLE	Position or Role (job title, e.g., Project Manager)	40	teletexString	0	1
emailSURNAME	Last name or Family name	40	printableString	0	1
emailSURNAME2	Last name or Family name #2	40	printableString	0	1
emailUSE1	User defined field 1	80	teletexString	0	1
emailUSE2	User defined field 2	80	teletexString	0	1
emailUSE3	User defined field 3	80	teletexString	0	1
emailUSE4	User defined field 4	80	teletexString	0	1
emailUSE5	User defined field 5	80	teletexString	0	1
emailUSE6	User defined field 6	80	teletexString	0	1
emailDDA	Domain-Defined Attribute	139	teletexString	0	1
emailDDA2	Domain-Defined Attribute #2	139	teletexString	0	1
emailDDA3	Domain-Defined Attribute #3	139	teletexString	0	1
emailDDA4	Domain-Defined Attribute #4	139	teletexString	0	1

Table D–2 (Cont.): emailInfo Auxiliary Class X.500 Directory Attributes

Keyword	Description	Len	Syntax	#rdn	#attr
emailMHSUSERID	MHSUSERID part of MAILbus address	256	teletexString	0	1
emailMHSUSERID2	MHSUSERID part of MAILbus address #2	256	teletexString	0	1
emailREVERSE	MAILbus reverse lookup	256	teletexString	0	1
emailREVERSE2	MAILbus reverse lookup #2	256	teletexString	0	1

Appendix E
Synchronizer-500 Error Messages

This appendix lists Synchronizer-500 run-time error messages that may appear during Synchronizer-500 processing. Error Messages are logged to standard output (SYS$OUTPUT on VMS, stdout on UNIX) and if FATAL, written to the log file (if IMPORT or TRANSACTION mode) and included in the Mail Message (if NOTIFY is specfied).

The format for a Synchronizer-500 error message is:

```
%XDSU-<Severity>-<Code>, <Message>

    where:

        <Severity> = F (for FATAL) which will cause XDSU to terminate or
                     W (for WARNING) which allows XDSU to continue
        <Code>     = Error Code - See below
        <Message>  = Error Message
```

Synchronizer-500 Error Messages:

BADVALUE, for 'syntax', 'field'='value'"

Explanation: A field contained a bad value for a syntax.

User Action: Check Section H.2 or see the X.500 Directory Services Management Guide to check the legal values for this syntax. Update the field's value to conform to the syntax or change the syntax in the layout file to match the desired field value. It's only recommended that 'printableStringSyntax' be changed to 'teletexStringSyntax'. Any other syntax changes may also require a change to the X.500 schema.

CONFIGOPEN, error opening configuration file: 'filename'

Explanation: The configuration file could not be opened.

User Action: Assure that 'filename' is a valid file specification for the config file and that the user has read access. The default value for the configuration file (xdsu_config.dat in the default directory) can be changed by passing the location of the config file as the first argument in the xdsu command line.

DATATRUNC, data truncated for 'DN' attribute 'field'

Explanation: Data read from X.500 for 'DN', attribute 'field', is too long to fit into the layout record.

User Action: Increase the size of the field in the layout record or shorten the value in the X.500 directory. Note that the length of the field in the layout file refers to the ascii representation of the value.

DELFAIL, failed to delete pre-sorted metafile: 'filename'

Explanation: After a sort has completed, the old (pre-sorted) metafile could not be deleted.

User Action: Check access to the file.

EMPTYRDF, RDF file doesn't contain any valid records

Explanation: An RDF file specified in the config file did not have any non-comment lines.

User Action: Specifiy a valid RDF file.

EXTRALINE, 'num' extra line(s) in input file ignored

Explanation: The number of records in the input file was not evenly divisible by the number of records per subscriber (nrec_subscr).

User Action: Remove extra records or add records to make the number of records evenly divisible by nrec_subscr.

FDFCREATE, error creating foreign directory file: 'filename'

Explanation: The output foreign directory file could not be created.

User Action: Assure that 'filename' is a valid file specification and that sufficient privileges and disk space exist to create the file. The filename is determined by a configuration file parameter.

FDFOPEN, error opening foreign directory file: 'filename'

Explanation: The input foreign directory file could not be read.

User Action: Assure that 'filename' is a valid file specification and that sufficient privileges exist to read the file.

FDFWRITE, error writing foreign directory file

Explanation: Error writing data to the output foreign directory file.

User Action: Check for sufficient free disk space for the file.

INPTRUNC, data truncated for Field 'field' in Record 'num'

Explanation: In record number 'num' in the input file, the value for 'field' was too long, as specified in the RDF file, to fit into the meta file record.

User Action: Change the RDF and the input file to use less data for this field or increase the length of the field in the layout file.

INTERNAL_ERROR, 'message'

Explanation: An internal software error has resulting in a data inconsistency.

User Action: Contact your Digital Software Support Contact as specified under the terms of your warranty.

INVLAYOUTREC, invalid schema file record: 'record'

Explanation: An invalid record in the layout file was found

User Action: Check the syntaxes rules for a layout record in Appendix H.

INVRDFCOL, invalid RDF location column number: 'num'

Explanation: An invalid starting column number in the RDF was found.

User Action: Verify the column number is a an integer from 1-4096 or a '*' if in EXPORT or CHANGES mode.

INVRDFCOND, invalid RDF conditional record syntax: 'record'

Explanation: An invalid RDF conditional record was found.

User Action: Verify that the conditional part of the RDF record is terminated by a colon (':').

INVRDFKWD, invalid RDF keyword: 'keyword'

Explanation: An invalid RDF keyword was found.

User Action: Verify that the keyword is in the layout file.

INVRDFLEN, invalid RDF location field length: 'num'

Explanation: An invalid field length in the RDF was found.

User Action: Verify the field length is a an integer from 1-4096 or a '#' if in EXPORT or CHANGES mode. Also verify that the field length plus the starting column number does not exceed 4096.

INVRDFLOC, invalid RDF location syntax in record: 'num'

Explanation: An invalid RDF location (n.n.n) syntax was found.

User Action: Verify the location is 3 fields separated by 2 periods ('.'). Also, the fields must be integers with the exception that in the EXPORT and CHANGES mode, the second field can be a '*' and the third field can be a '#'.

INVRDFNUM, invalid RDF location record number: 'num'

Explanation: An invalid RDF record number was found.

User Action: Verify that the record number is an integer from 1 to the nrec_subscr parameter for this RDF.

INVRDFREC, invalid RDF record syntax: 'record'

Explanation: An invalid RDF record was found.

User Action: Verify that the RDF record is two field strings separated by an equal sign ('=') with an optional conditional prefix string.

INVRDFSTR, invalid RDF string: 'string'

Explanation: An invalid RDF string was found.

User Action: Verify that the RDF string is enclosed by double quote characters.

KEYWVALDUP, duplicate config kewyword: 'record'

Explanation: A configuration file keyword was repeated.

User Action: Remove duplicate definitions of a keyword. A keyword may only appear once.

KEYWVALINV, invalid config record value: 'record'

Explanation: An invalid value for an configuration file keyword was found.

User Action: Check Appendix F for configuration file keyword information.

KEYWVALREQ, value required for config keyword: 'record'

Explanation: A required configuration file keyword was not found or did not have a value.

User Action: Supply a value for the required keyword.

LAYOUTOPEN, error opening schema data file: 'filename'

Explanation: The schema (layout) file could not be opened.

User Action: Assure that 'filename' is a valid file specification for the layout file and that the user has read access. The default value for the layout file (xdsu_ layout.dat in the parent directory) can be changed by passing the location of the layout file as the second argument on the xdsu command line.

LOGOPEN, error creating log file: 'filename'

Explanation: The log file could not be created.

User Action: Assure that 'filename' is a valid file specification and that sufficient privileges and disk space exist to create the file. The filename is determined by the log_file configuration file parameter.

MAXFAIL, maximum fail transactions reached at 'num' transactions

Explanation: The program has stopped after reaching the specified number of failed Add, Modify, and Delete Transactions.

User Action: This parameter is set by the 'max_fail_transactions' configuration file keyword in IMPORT or TRANSACTION mode.

MEMALLOC, error allocating memory

Explanation: The program has stopped due to insufficient memory.

User Action: See the system manager to allow more memory allocation. If none is available, see Chapter 4 on suggestions for reducing XDSU's memory requirements.

METACREATE, error creating metafile: 'filename'

Explanation: The metafile 'filename' could not be created.

User Action: Assure that 'filename' is a valid file specification and that sufficient privileges and disk space exist to create the file. If a disk space problem, See Chapter 4 on suggestions for spreading out metafiles over multiple disks.

METAOPEN, error opening metafile: 'filename'

Explanation: The metafile 'filename' could not be read.

User Action: Assure that 'filename' is a valid file specification and that sufficient privileges exist to read the file.

METAWRITE, error writing metafile

Explanation: Error writing data to the metafile.

User Action: Check for sufficient free disk space for the file.

NOSYNCHTYPE, first config keyword must be SYNCH_TYPE

Explanation: The first non-comment line in the configuration file must contain the SYNCH_TYPE keyword.

User Action: Assure that the first non-comment line in the configuration file contains the SYNCH_TYPE keyword.

NOVALGID, GID 'string' not found in 'filename'

Explanation: The GID_MATCH value 'string' was not found in the valid id file 'filename'.

User Action: Change the INITIAL_LOAD value to YES if this is the first time entries for this GID are being IMPORTed. Otherwise, change the GID_MATCH value to a previously used GID.

PROCFAIL, error running command procedure: 'filename'

Explanation: The command procedure 'filename' could not be run.

User Action: Assure that 'filename' is a valid file specification and that sufficient privileges exist to execute the file as a subprocess.

PROFSCONFIG, gen_profs_alias config error

Explanation: The configuration file specified gen_profs_alias=YES but required layout file flags were not found.

User Action: Assure that the PRL and PRU special purpose flags are specified in the layout file. See Appendix H.

RDFOPEN, Error opening RDF file: 'filename'

Explanation: The rdf file 'filename' could not be opened.

User Action: Assure that 'filename' is a valid file specification and that sufficient privileges exist to read the file.

RENAMEFAIL, failed to rename sorted metafile 'filename' to 'filename'

Explanation: The sorted metafile could not be renamed to the pre-sorted metafile name.

User Action: Assure that the 'filenames' are valid file specifications and that sufficient privileges exist to rename the file.

SENDFAIL, error sending mail to 'username'

Explanation: A mail notification message could not be sent to 'username'.

User Action: Assure that 'username' is a valid mail address and that sufficient privileges exist to send mail to that user exist.

SNADSCONFIG, gen_snads_alias config error

Explanation: The configuration file specified gen_snads_alias=YES but required layout file flags were not found.

User Action: Assure that the SNL and SNU special purpose flags are specified in the layout file. See Appendix H.

SORTFAILED, sort failed of metafile: 'filename'

Explanation: An error sorting the metafile 'filename' occurred.

User Action: Assure that 'filename' and 'filename'_SRT are valid file specifications and that sufficient privileges and disk space exist to sort the file.

SUMMOPEN, error opening summary file: 'filename'

Explanation: The temporary summary file 'filename' could not be opened.

User Action: Assure that sufficient privileges and disk space exist to create the file.

UNIQNOTINRDF, unique field 'field' must be specified in RDF

Explanation: The configuration file keyword 'unique_field' was specified but an RDF value was not.

User Action: Specify a value for the 'unique_field' keyword in the RDF file.

VALGIDEXISTS, GID 'string' already exists in 'filename' during an initial load

Explanation: The GID_MATCH value 'string' was found in the valid id file 'filename'.

User Action: Change the INITIAL_LOAD value to NO in the configuration file.

VALIDIDOPEN, error opening valid id file: 'filename'

Explanation: The valid id file 'filename' could not be opened.

User Action: Assure that 'filename' is a valid file specification and that sufficient privileges and disk space exist to read and write the file and create it if it doesn't exist.

X500EXIT, error exiting X.500 Directory Service

Explanation: An error occurred during a ds_unbind or ds_shutdown X.500 Directory Services call.

User Action: Set VERBOSE=YES and rerun to get a trace of the error and then consult the X.500 Documentation for further information if needed.

X500FETCH, error reading an entry from X.500 Directory Service

Explanation: An error occurred during a ds_read X.500 Directory Services call after a ds_search call returned the matching DN's.

User Action: Set VERBOSE=YES and rerun to get a trace of the error and then consult the X.500 Documentation for further information if needed. Also, check that the layout file OID parameters match the compiled X.500 schema.

X500INIT, error initializing X.500 Directory Service

Explanation: An error occurred during a ds_initialize or ds_bind X.500 Directory Services call.

User Action: Set VERBOSE=YES and rerun to get a trace of the error and then consult the X.500 Documentation for further information if needed.

X500READ, error reading 'DN'

Explanation: An error occurred during a ds_read X.500 Directory Services call of an existing MODIFY record in TRANSACTION Mode.

User Action: Set VERBOSE=YES and rerun to get a trace of the error and then consult the X.500 Documentation for further information if needed. Also, verify that the entry for 'DN' exists. If not, the transaction must be moved from the MODIFY file to the ADD file.

X500SEARCH, error searching X.500 Directory Service

Explanation: An error occurred during a ds_search X.500 Directory Services call or during the setup of a the filter object using OSI Object Manipulation (OM) calls.

User Action: Set VERBOSE=YES and rerun to get a trace of the error and then consult the X.500 Documentation for further information if needed. Also, one of the filter objects may have had a syntax error. Check that the values for GID_ MATCH, GID_EXCLUDE, SYNCH_ID, and the various *_MATCH values do not violate the syntax for their assocaiated fields.

Appendix F
Configuration File Templates

This appendix presents Synchronizer-500's *import, transaction, export,* and *changes* configuration file templates and definitions for terms used in each of the files.

F.1 Import Configuration File Template

The *import* configuration file template is shown in Example F–1. The terms used in the file are defined following the template example.

Example F–1: Import Configuration File Template

```
!
! XDSU_CONFIG.IMPORT_TEMPLATE
!
synch_type=IMPORT
synch_label=
!
! Files and Run Options
!
bind_dsa=
rdf_file=
input_for_dir_file=
nrec_subscr=
log_file=
end_run_file=
old_metafile=
new_metafile=
initial_load=
notify=
verbose=
encode_delim=
max_transactions=
max_fail_transactions=
max_trans_file=
!
! Data Options
!
dn_sequence=
create_root=
search_context=
synch_object=
synch_id=
synch_id_field=
gid_field=
gid_match=
gen_profs_alias=
gen_snads_alias=
admd_space=
unique_field=
unique_field_len=
make_unique=
truncate=
unique_field_separator=
unique_search_context=
```

SYNCH_TYPE=IMPORT indicates that data from a foreign system is to be imported into X.500.

SYNCH_LABEL indicates the label to use for Synchronizer-500 output (mail messages and end-of-run synopses). The label value is specified as a string. Use a value that will help the Directory Administrator identify the synchronization run, e.g. SYNCH_LABEL=Import Microsoft Mail. A default of XDSU is provided, if the value for this record is not specified.

BIND_DSA specifies a presentation address for the DSA to which you wish to bind. This feature is not enabled at this time. RESERVED FOR FUTURE USE.

RDF_FILE points to the record description file (RDF) that describes the format of the foreign directory file being imported. The RDF controls how information from the foreign directory file is mapped into X.500. The RDF_FILE is required and must be a valid file specification.

INPUT_FOR_DIR_FILE points to the input foreign directory file (FDF) that contains the list of directory entries to import into X.500. The INPUT_FOR_DIR_FILE is required and must be a valid file specification.

NREC_SUBSCR specifies the number of records (lines) to be read per subscriber (entry) in the input foreign directory file. In most cases, each subscriber has only one record. Examine the foreign directory file to determine the number of records per subscriber. NREC_SUBSCR is required.

LOG_FILE indicates the location of Synchronizer-500 transaction log file. The default, if not specified, is xdsu_log.log in the default directory. This file contains all the ADD, MODIFY, and DELETE activity that was performed during an import run.

END_RUN_FILE provides a means of calling a user-written procedure when the import operation finishes. Examples of uses for this feature include report generation or calling a file that starts an export run.

OLD_METAFILE and **NEW_METAFILE** indicate the file specification of the *old* and *new* metafiles. By default, these files are given a random name and created in the default directory when Synchronizer-500 is executed. If the Directory Administrator wants to spread these files over multiple disks (See Section 2.1.2), then file specifications should be given for these keywords.

INITIAL_LOAD is a flag (YES/NO) that indicates whether or not this is the first time information from this particular foreign directory is being loaded into X.500. The GROUP ID (GID_MATCH= , described below) is used to associate entries in X.500 with the particular foreign directory from which they were loaded.

If INITIAL_LOAD=YES, Synchronizer-500 records the GROUP ID in the file xdsu_valid_ id.dat and the entries are then processed and loaded into X.500.

If INITIAL_LOAD=YES, the GROUP ID specified in GID_MATCH= must be unique (not previously used to identify another directory group) or Synchronizer-500 will halt with a fatal error (VALGIDEXISTS). Once a foreign directory has been initially loaded, the INITIAL_ LOAD flag must be set to NO for subsequent runs.

If INITIAL_LOAD=NO, then the GROUP ID (GID_MATCH=) must already be recorded in the file xdsu_valid_id.dat. (By default, xdsu_valid_id.dat resides in the XDSU root directory.)

NOTIFY is used to specify the mail address to which the summary messages will be sent. The account can be either a local account (e.g., SYSTEM) or a remote account (e.g., APPLE::MANAGER). This value must be entered in VMSmail addressing format on an OpenVMS system (or be a logical name which translates into a valid address), and Sendmail addressing format on a UNIX system.

VERBOSE controls whether extra information is displayed to the transaction log file (log_ file=) and to sys$output/stdout. If VERBOSE=NO or not specified, no extra information is displayed. If VERBOSE=YES, each transaction is logged with a full listing of attributes and their values. Also, all X.500 transactions are logged as they occur to sys$output/stdout and any X.500 errors are logged containing the X.500 error codes.

ENCODE_DELIM indicates a character to use as a delimiter when Synchronizer-500 needs to encode characters not available in the T.61 (teletexString) character set. These characters are {, }, ©, ¹, and \.

When Synchronizer-500 encounters one of these characters on an import, it will convert the character to its hex value and preface it with the character specified in the field ENCODE_ DELIM. On an export, Synchronizer-500 will convert the hex number back to the correct character for output into the data file.

If this field is left blank and Synchronizer-500 encounters a character which needs to be encoded, the transaction is rejected with a status of FAILURE.

MAX_TRANSACTIONS and **MAX_TRANS_FILE** are used together to permit calling a user-written procedure when the specified number of X.500 transactions have been executed. (An X.500 transaction is defined as an ADD/MODIFY/DELETE transaction.) MAX_ TRANSACTIONS specifies the point at which to call the procedure named by MAX_TRANS_ FILE. For example, MAX_TRANSACTIONS=1000 means that after every 1000 X.500 transactions have been performed, Synchronizer-500 will execute the MAX_TRANS_FILE procedure. Before calling the procedure, XDSU will unbind and shutdown from X.500 and then reinitialize and rebind to X.500 after the procedure is done. If MAX_TRANS_FILE is not specified, XDSU still does the unbind/shutdown/reinitialize/rebind sequence.

MAX_FAIL_TRANSACTIONS provides a method of stopping Synchronizer-500 after a number of failures have occurred. If you are importing a large number of entries into X.500, this parameter will allow you to stop Synchronizer-500 after a certain number of failures without having to finish processing the entire file. MAX_FAIL_TRANSACTIONS=500 means that after 500 failed transactions have occurred, Synchronizer-500 writes an error to the log file and stops executing.

DN_SEQUENCE specifies the order of the relative distinguished names (RDNs) that make up an entry's distinguished name (e.g. /country/organization/organizationalUnit/commonName). If multiple values of the same RDN attribute are desired in the DN, only specify the RDN Value once. For example, if the DN is /C=US /O=DIGITAL /OU=SALES /OU=GOVERNMENT /CN="Fred Smith", the DN_SEQUENCE is /C/O/OU/CN.

CREATE_ROOT specifies the level of superior entries in the DIT that should be created for this import, if necessary (e.g., organizationalUnit).

SEARCH_CONTEXT specifies the part of the naming context to which Synchronizer-500 should limit its searches, (e.g., /country=us/organization=digital/organizationalUnit=sales).

SYNCH_OBJECT specifies the X.500 object classes into which you are importing data (e.g., Person,organizationalPerson,emailInfo). Separate multiple object classes by a comma. The object classes must be specified as OBJ records in the layout file. This field is only used when an entry is added. Changing the value of this field will not affect existing entries (i.e. existing fields won't be modified).

SYNCH_ID_FIELD specifies the X.500 attribute in which the SYNCH_ID value is stored. This can be any valid attribute in the Layout File.

SYNCH_ID is used to tag each entry with an identifier that tells Synchronizer-500 it owns the entry. Synchronizer-500 does not try to modify or delete an entry that it does not own. The SYNCH_ID field value can be, for example, the name of the node on which Synchronizer-500 is running. If no value is specified, the default value 'XDSU' is used.

GID_FIELD specifies the X.500 attribute in which the Group ID (GID) value is stored. This can be any valid attribute in the Layout File.

GID_MATCH specifies the value for the Group ID. During the import process, the Group ID is used for comparing the foreign directory file with the existing entries in X.500 belonging to that foreign directory so that Synchronizer-500 can determine updates needed to X.500. During the export process, the Group ID is used to select which entries in X.500 Synchronizer-500 will write to the output file.

The value chosen for the Group ID is strictly arbitrary; that is, the name chosen has no affect on how the entries are handled. For example, a choice of MICROSOFT does not mean that the entries will be handled as MICROSOFT entries. It is a good idea, however, to choose a Group ID that clearly identifies the foreign directory file to which the entries belong. Microsoft NEW YORK or Engineering ALL-IN-1 are examples of good GROUP ID identifiers.

GEN_PROFS_ALIAS indicates whether or not (YES/NO) to generate PROFS aliases. The default for this field is NO. See Section 1.3.8 for more information.

GEN_SNADS_ALIAS indicates whether or not (YES/NO) to generate SNADS aliases. The default for this field is NO. See Section 1.3.8 for more information.

ADMD_SPACE indicates whether or not (YES/NO) the attribute emailADMDNAME should have a BLANK as its value. The default value is "NO". If there is a value for emailADMD-NAME in the RDF (as a constant or from the input data file), Synchronizer-500 will use the value even if ADMD_SPACE is set to "YES". A blank will be generated only if there is no ADMD value and if ADMD_SPACE is set to "YES".

UNIQUE_FIELD is the X.500 attribute that Synchronizer-500 will check for uniqueness. Optionally, Synchronizer-500 can modify this field to make it unique. (See MAKE_UNIQUE described below.)

Synchronizer-500 allows the Directory Administrator to specify an X.500 attribute that Synchronizer-500 guarantees to be unique (up to a selected length, see UNIQUE_FIELD_ LEN described below) in X.500 when using the import process to add data to X.500.

This option checks the uniqueness of the selected X.500 attribute. If the value is not unique and if MAKE_UNIQUE is not set to "NO", Synchronizer-500 generates a unique key field for the specified X.500 attribute for a specified length (UNIQUE_FIELD_LEN). It does this by acquiring a random number and appending an appropriate number of characters from that number to the data that already exists in the X.500 attribute. If the value is not

unique and if MAKE_UNIQUE is set to "NO", the transaction is rejected with a status of USERDEFNOTUNIQ. See Section 1.3.7 for more information.

Note

Uniqueness checking and unique value generation is performed only during import ADD transactions. Once this field has been entered by the ADD transaction, Synchronizer-500 will not modify the field, even if modification is defined in the RDF.

UNIQUE_FIELD can be set to any valid attribute in the Layout File. This field can be left blank if uniqueness checking is not needed.

UNIQUE_FIELD_LEN is the length to guarantee uniqueness for UNIQUE_FIELD (see description above). If a length is specified it must have a minimum value of eight.

MAKE_UNIQUE indicates whether or not (YES/NO) Synchronizer-500 should modify the UNIQUE_FIELD to make it unique. If the value is "YES", Synchronizer-500 generates a unique key value for the specified attribute (UNIQUE_FIELD) for a specified length (UNIQUE_FIELD_LEN). It does this by acquiring a random number and appending an appropriate number of characters from that number to the data that ready exists in the X.500 attribute. If the value is "NO", the transaction is rejected with a status of USERDEFNOTUNIQ. See Section 1.3.7 for more information.

"YES" is the default value (make the field unique). The default is used if no value was specified.

The **TRUNCATE** keyword is used in combination with the UNIQUE_FIELD and MAKE_UNIQUE keywords. A value of "NUMBER" is allowed. When an attribute has been designated for uniqueness checking and it should be modified if not unique, some special processing goes on inside of Synchronizer-500. After the value has been formed from its component RDF records, it is checked "as-is" for uniqueness in X.500. If unique, it is used in its original form.

If not unique, Synchronizer-500 modifies it to become unique. It does this by generating a 5 digit random number and appending it to the existing data. Synchronizer-500 will continue to do this until the field is unique.

TRUNCATE=NUMBER comes into play as follows: After a unique number has been generated, it is checked to see if the width of the attribute is sufficient to include the 5 digits. If not, Synchronizer-500 will truncate the existing data in order to accommodate the entire 5 digit number. If TRUNCATE=NUMBER is specified in the configuration file, the 5 digit number will be truncated instead of the data.

For example: the attribute emailPRUSERNAME is selected for uniqueness processing (UNIQUE_FIELD=emailPRUSERNAME). The width of the attribute is 8. The RDF has selected a 5 character name SMITH to be used for emailPRUSERNAME. There is already an X.500 entry with a emailPRUSERNAME of SMITH. Synchronizer-500 generates a random number 12345. If TRUNCATE=NUMBER is not used, the new emailPRUSERNAME will be SMI12345. If TRUNCATE=NUMBER is used, the new PRUSERNAME will be SMITH123.

Care must be exercised when using this option. Sufficient space must be left for Synchronizer-500's use to resolve ties. For example, if an X.500 attribute is 8 characters and 7 characters are consistently fetched from the input data file, this leaves room for 10 ties ("0" - "9"). If more than 10 ties exist, Synchronizer-500 will be unable to generate a unique name.

UNIQUE_FIELD_SEPARATOR is an optional keyword which can be used to specify a string to separate the original unique field value and any random prefix which may be appended if the field needs to be made unique. Specify the value in double quotes (e.g to specify a hyphen, set UNIQUE_FIELD_SEPARATOR="-"). The default is a null string ("").

UNIQUE_SEARCH_CONTEXT specifies the part of the naming context to which Synchronizer-500 should limit its searches during Unique Field processing, standard uniqueness checking, and PROFS and SNADS Alias genertation.

F.2 Transaction Configuration File Template

The *transaction* configuration file template is shown in Example F–2. The terms used in the file are defined following the template example.

Example F–2: Transaction Configuration File Template

```
!
! XDSU_CONFIG.TRANSACTION_TEMPLATE
!
synch_type=TRANSACTION
synch_label=
!
! Files and Run Options
!
bind_dsa=
rdf_file=
input_add_file=
input_mod_file=
input_del_file=
nrec_subscr=
log_file=
end_run_file=
add_metafile=
del_metafile=
mod_metafile=
initial_load=
notify=
verbose=
encode_delim=
max_transactions=
max_fail_transactions=
max_trans_file=
!
! Data Options
!
dn_sequence=
create_root=
synch_object=
synch_id=
synch_id_field=
gid_field=
gid_match=
gen_profs_alias=
gen_snads_alias=
admd_space=
unique_field=
unique_field_len=
make_unique=
truncate=
unique_field_separator=
unique_search_context=
```

All keywords in the *Transaction* Configuration File are the same as in the *Import* Configuration File with the following exceptions:

SYNCH_TYPE=TRANSACTION indicates that X.500 data is to be updated from a foreign system.

INPUT_ADD_FILE, **INPUT_MOD_FILE**, and **INPUT_DEL_FILE** are used (instead of INPUT_FOR_DIR_FILE) which point to the input foreign directory files (FDF) that contains the list of directory entries to added, modified, and deleted into X.500. These keywords are optional and must be a valid file specification if specified.

ADD_METAFILE, **MOD_METAFILE**, and **DEL_METAFILE** are used (instead of OLD_METAFILE and NEW_METAFILE) to indicate the file specifications of the *add*, *mod* and *del* metafiles. By default, these files are given a random name and created in the default directory when Synchronizer-500 is executed. If the Directory Administrator wants to spread these files over multiple disks (See Section 2.1.2), then file specifications should be given for these keywords.

These is no SEARCH_CONTEXT keyword for the Transaction Configuration File.

F.3 Export Configuration File Template

The *export* configuration file template is shown in Example F–3. The terms used in the file are defined following the template example.

Example F–3: Export Configuration File Template

```
!
! XDSU_CONFIG.EXPORT_TEMPLATE
!
synch_type=EXPORT
synch_label=
!
! Files and Run Options
!
bind_dsa=
rdf_file=
output_for_dir_file=
nrec_subscr=
end_run_file=
output_old_metafile=
input_old_metafile=
notify=
verbose=
encode_delim=
!
! Data Options
!
search_context=
synch_id=
synch_id_field=
gid_field=
gid_match=
gid_exclude=
country_match=
admd_match=
prmd_match=
orgname_match=
orgunit_match=
orgunit2_match=
orgunit3_match=
orgunit4_match=
get_field=
get_value=
```

SYNCH_TYPE=EXPORT indicates that data from X.500 is written to an output file.

SYNCH_LABEL indicates the label to use for Synchronizer-500 output (mail messages and end-of-run synopses). The label value is specified as a string. Use a value that will help the Directory Administrator identify the synchronization run, e.g. SYNCH_LABEL=Export Microsoft Mail. A default of XDSU is provided, if the value for this record is not specified.

BIND_DSA specifies a presentation address for the DSA to which you wish to bind. This feature is not enabled at this time. RESERVED FOR FUTURE USE.

RDF_FILE points to the record description file (RDF) that describes the format of the foreign directory file being exported. The RDF controls how information from X.500 is output in a format suitable for use by the foreign directory. The RDF_FILE is required and must be a valid file specification.

OUTPUT_FOR_DIR_FILE points to the output foreign directory file (FDF) to which to write the export X.500 data. The OUTPUT_FOR_DIR_FILE is required and must be a valid file specification.

NREC_SUBSCR specifies the number of records (lines) to be written per subscriber (entry) in the output foreign directory file. NREC_SUBSCR is required.

END_RUN_FILE provides a means of calling a user-written procedure when the export operation finishes. Examples of uses for this feature include calling a procedure to mail the output file to the foreign system for import into the foreign directory.

OUTPUT_OLD_METAFILE permits saving the metafile by specifying a file specification for it. If OUTPUT_OLD_METAFILE keyword has no value assigned to it, the metafile created during the export process is deleted at the end of the run. OUTPUT_OLD_METAFILE needs to specified when a baseline for CHANGES mode is being established. OUTPUT_ OLD_METAFILE must be a valid file specification.

INPUT_OLD_METAFILE is provided in case the Directory Administrator wants to use a previous extract of X.500 data. If a value is supplied for this keyword, Synchronizer-500 will *not* extract data from X.500 into the old metafile. It will instead use this existing file.

NOTIFY is used to specify the mail address to which the summary messages will be sent. The account can be either a local account (e.g., SYSTEM) or a remote account (e.g., APPLE::MANAGER). This value must be entered in VMSmail addressing format on an OpenVMS system (or be a logical name which translates into a valid address), and Sendmail addressing format on a UNIX system.

VERBOSE controls whether extra information is displayed to sys$output/stdout. If VERBOSE=NO or not specified, no extra information is displayed. If VERBOSE=YES, all X.500 transactions are logged as they occur to sys$output/stdout and any X.500 errors are logged containing the X.500 error codes.

ENCODE_DELIM indicates the character used to encode t.61 characters during the Import or Transaction Run.

SEARCH_CONTEXT specifies the part of the naming context to which Synchronizer-500 should limit its searches, (e.g., /country=us/organization=digital/organizationalUnit=sales).

SYNCH_ID_FIELD specifies the X.500 attribute in which the SYNCH_ID value is stored. This can be any valid attribute in the Layout File.

SYNCH_ID specifies the value for the SYNCH_ID_FIELD in which to extract data from X.500. This field is optional.

GID_FIELD specifies the X.500 attribute in which the Group ID (GID) value is stored. This can be any valid attribute in the Layout File. This field is required.

GID_MATCH specifies the value(s) for the GID_FIELD in which to extract data from X.500. Multiple values can be specified and are separated by commas (e.g. GID_ MATCH=GROUP1,GROUP2). All GID_FIELD values can be specified by setting GID_ MATCH=*. This field is required.

GID_EXCLUDE specifies the value(s) for the GID_FIELD in which NOT to extract data from X.500. Multiple values can be specified and are separated by commas (e.g. GID_ EXCLUDE=GROUP1,GROUP2). This field may only be specified when GID_MATCH=*.

Several X.400 standard attributes can be used to further qualify the subset of entries to be exported from X.500. The following configuration file terms can be used in conjunction with SEARCH_CONTEXT, SYNCH_ID, GID_MATCH, and GID_EXCLUDE to limit the export:

COUNTRY_MATCH
ADMD_MATCH
PRMD_MATCH
ORGNAME_MATCH
ORGUNIT_MATCH
ORGUNIT2_MATCH
ORGUNIT3_MATCH
ORGUNIT4_MATCH.

GET_FIELD and **GET_VALUE** provide one more filter to be used when extracting data from X.500. These keywords allow the directory administrator to select a group of entries from X.500 based on a value in a specific attribute. GET_FIELD can be any valid attribute in the Layout File and GET_VALUE is a value to match on in X.500. The wild card, asterisk * character, is acceptable.

For example: GET_FIELD=emailSNLOCATION and GET_VALUE=NACGWY1, will extract all entries with an attribute of emailSNLOCATION that has a value of NACGWY1.

F.4 Changes Configuration File Template

The *changes* configuration file template is shown in Example F–4. The terms used in the file are defined following the template example.

Example F–4: Changes Configuration File Template

```
!
! XDSU_CONFIG.CHANGES_TEMPLATE
!
synch_type=CHANGES
synch_label=
!
! Files and Run Options
!
bind_dsa=
add_rdf_file=
mod_rdf_file=
del_rdf_file=
output_add_file=
output_mod_file=
output_del_file=
nrec_subscr_add=
nrec_subscr_mod=
nrec_subscr_del=
end_run_file=
input_old_metafile=
output_new_metafile=
input_new_metafile=
notify=
verbose=
encode_delim=
!
! Data Options
!
search_context=
synch_id=
synch_id_field=
gid_field=
gid_match=
gid_exclude=
country_match=
admd_match=
prmd_match=
orgname_match=
orgunit_match=
orgunit2_match=
orgunit3_match=
orgunit4_match=
get_field=
get_value=
```

All keywords in the *Changes* Configuration File are the same as in the *Export* Configuration File with the following exceptions:

SYNCH_TYPE=CHANGES indicates that only data which has changed in the X.500 DIT should be written to the output files.

ADD_RDF_FILE, **MOD_RDF_FILE**, and **DEL_RDF_FILE** are used (instead of RDF_FILE) which point to the RDF that describes the format of the ADD, MOD, and DEL output foreign directory files. These must be valid file specifications. ADD_RDF_FILE and DEL_RDF_FILE are required.

Note

If a value for MOD_RDF_FILE is omitted, all MODIFY transactions will be implemented as DELETEs and ADDs. In other words, instead of a MODIFY record being written to the MODIFY file, a DELETE record is written to the DELETE file and an ADD record is written to the ADD file. When these files are applied to the foreign directory, the DELETE must be performed before the ADD in order for the intended results to occur.

OUTPUT_ADD_FILE, **OUTPUT_MOD_FILE**, and **OUTPUT_DEL_FILE** are used (instead of OUTPUT_FOR_DIR_FILE) to give the name of the file to which to write the exported X.500 ADD, MOD, and DEL data. These must be valid file specifications. OUTPUT_ADD_FILE and OUTPUT_DEL_FILE are required as is OUTPUT_MOD_FILE if MOD_RDF_FILE is specified.

NREC_SUBSCR_ADD, **NREC_SUBSCR_MOD**, and **NREC_SUBSCR_DEL** are used (instead of NREC_SUBSCR) to specify the number of records per subscriber to be written in the output ADD, MOD, and DEL file.

INPUT_OLD_METAFILE, **OUTPUT_NEW_METAFILE**, and **INPUT_NEW_METAFILE** are used in CHANGES mode (instead of OUTPUT_OLD_METAFILE and INPUT_OLD_METAFILE in EXPORT mode).

INPUT_OLD_METAFILE is the *old* metafile against which Synchronizer-500 will compare the current contents of X.500 to determine the changes that have occurred. INPUT_OLD_METAFILE is required and must be a valid file specification. If this record contains a null value, Synchronizer-500 will assume that there is no previous metafile to compare to and will generate all Change mode transactions as ADDs.

OUTPUT_NEW_METAFILE permits saving the *new* metafile by specifying a file specification for it. If OUTPUT_NEW_METAFILE keyword has no value assigned to it, the metafile created during the changes process is deleted at the end of the run. OUTPUT_NEW_METAFILE is a valid file specification. Normally, the OUTPUT_NEW_METAFILE is specified and the END_RUN_FILE procedure renames it to be the INPUT_OLD_METAFILE for the next run.

INPUT_NEW_METAFILE is provided in case the Directory Administrator wants to compare two *old* metafiles to determine the changes that have occurred between the two. If a value is supplied for this keyword, Synchronizer-500 will *not* extract data from X.500 into the new metafile. It will instead use this existing file.

Appendix G
Synchronizer-500 Transaction Status Codes

During an IMPORT or TRANSACTION type synchronization run, all transactions are logged to the Synchronizer-500 transaction log file. Each transaction contains a completion status (NORMAL or FAILED) and a status code.

The Synchronizer-500 transaction status codes and their meanings are shown in Table G–1.

Table G–1: Synchronizer-500 Transaction Status Codes

Status Code	Completion Status	Meaning and Possible Actions
SUCCESS	NORMAL	transaction complete.
KEYFLDMOD	NORMAL	transaction complete, UNIQUE_FIELD modified to be unique (digits added).
KEYFLDTRUNC	NORMAL	transaction complete, UNIQUE_FIELD too long, truncated back to UNIQUE_FIELD_ LEN.
BADFLDVAL	FAILED	Synchronizer-500 has detected a invalid field value for the indicated syntax for this field. This error is accompanied by a BADVALUE warning message which gives details on which field has the bad value.
CCMNOTUNIQ	FAILED	emailCCmailAddr (CCM) attribute not unique. Check X.500 for existence of X.500 entry with the same emailCCmailAddr attribute.
FAILURE	FAILED	X.500 transaction failed. Returned from XDS or XOM API. If VERBOSE=YES, this error is accompanied by a trace dump of the X.500 API error.

Table G–1 (Cont.): Synchronizer-500 Transaction Status Codes

Status Code	Completion Status	Meaning and Possible Actions
MBUSNOTUNIQ	FAILED	emailMAILbusAddr (MBUS) attribute not unique. Check X.500 for existence of X.500 entry with the same emailMAILbusAddr attribute.
MSMNOTUNIQ	FAILED	emailMicrosoftAddr (MSM) attribute not unique. Check X.500 for existence of X.500 entry with the same emailMicrosoftAddr attribute.
PRUNOTUNIQ	FAILED	PROFS USERID/NODE (PROF) attributes not unique. Check X.500 for existence of X.500 entry with the same PROFS USERID/NODE values.
QMNOTUNIQ	FAILED	emailQuickmailAddr (QM) attribute not unique. Check X.500 for existence of X.500 entry with the same emailQuickmailAddr attribute.
RFCNOTUNIQ	FAILED	emailRFC822Addr (RFC) attribute not unique. Check X.500 for existence of X.500 entry with the same emailRFC822Addr attribute.
SNUNOTUNIQ	FAILED	SNADS USERID/NODE (SNAD) attributes not unique. Check X.500 for existence of X.500 entry with the same SNADS USERID/NODE values.
UNIQFAILED	FAILED	An XDS or XOM API error occurred while performing uniqueness checking on this entry. If VERBOSE=YES, this error is accompanied by a trace dump of the API error.
UQ08NOTUNIQ	FAILED	emailAddr08 (UQ08) attribute not unique. Check X.500 for existence of X.500 entry with the same emailAddr08 attribute.
UQ09NOTUNIQ	FAILED	emailAddr09 (UQ09) attribute not unique. Check X.500 for existence of X.500 entry with the same emailAddr09 attribute.
UQ10NOTUNIQ	FAILED	emailAddr10 (UQ10) attribute not unique. Check X.500 for existence of X.500 entry with the same emailAddr10 attribute.
UQ11NOTUNIQ	FAILED	emailAddr11 (UQ11) attribute not unique. Check X.500 for existence of X.500 entry with the same emailAddr11 attribute.

Table G–1 (Cont.): Synchronizer-500 Transaction Status Codes

Status Code	Completion Status	Meaning and Possible Actions
UQ12NOTUNIQ	FAILED	emailAddr12 (UQ12) attribute not unique. Check X.500 for existence of X.500 entry with the same emailAddr12 attribute.
USERDEFNOTUNIQ	FAILED	User defined UNIQUE_FIELD not unique. Check X.500 for existence of X.500 entry with the same UNIQUE_FIELD attribute value.
X484NOTUNIQ	FAILED	X.400 1984 (X484) attributes not unique. Check X.500 for existence of X.500 entry with the same X.400 1984 standard attributes
X488NOTUNIQ	FAILED	X.400 1988 (X488) attributes not unique. Check X.500 for existence of X.500 entry with the same X.400 1988 standard attributes
84XNOTUNIQ	FAILED	emailX400Addr84 (84X) attribute not unique. Check X.500 for existence of X.500 entry with the same emailX400Addr84 attribute.
88XNOTUNIQ	FAILED	emailX400Addr88 (88X) attribute not unique. Check X.500 for existence of X.500 entry with the same emailX400Addr88 attribute.

Appendix H
Synchronizer-500 X.500 Layout File

The X.500 layout file describes to Synchronizer-500 the information it needs about the X.500 object classes and attributes on which it will be operating. This information includes items such as the label, length, object identifiers (OIDs), syntaxes, number of values allowed, etc. This file is named xdsu_layout.dat and is located in the Synchronizer-500 root directory.

The default X.500 layout file which is supplied with Synchronizer-500 contains all the attributes listed in Appendix D.

The X.500 layout file also contains special purpose flags. One set of flags is used to define the uniqueness checking performed by Synchronizer-500. Another set of flags identifies attributes on which Synchronizer-500 performs special processing.

The X.500 layout file also defines the metafile format.

H.1 Layout File Records

The X.500 layout file consists of three kinds of records: object class (OBJ), relative distinguished name (RDN), and attribute (ATTR). Each will be discussed in the following sections.

H.1.1 Object Records

Object records describe to Synchronizer-500 the X.500-specific information it will need to create entries and their object classes in an X.500 DIB. Object records have three fields delimited by the vertical line "|" character:

1. type (the string "OBJ")
2. label by which this object class is referenced
3. object class OID (OID format)

Note

Object identifier (OID) format is the integer representation of the OID surrounded by left and right braces ("{" and "}"). Each integer (from the root to the end object) is separated by a space (" "). For example, the object identifier for the organizationalPerson in OID format is {2 5 6 7}.

Following is an example of the object records contained in the Synchronizer-500 default X.500 layout file:

```
OBJ|Top|{2 5 6 0}
OBJ|Person|{2 5 6 6}
OBJ|organizationalPerson|{2 5 6 7}
OBJ|organizationalRole|{2 5 6 8}
OBJ|residentialPerson|{2 5 6 10}
OBJ|emailInfo|{1 3 24 30 2 2 1}
OBJ|decMailUser|{1 3 24 10 4 16}
```

The object record data is referenced in the Import and Transaction configuration file (xdsu_config.dat) record "synch_object=". This record describes the object classes of the entries that Synchronizer-500 creates, for example:

```
synch_object=Person,organizationalPerson,emailInfo
```

This tells Synchronizer-500 that the primary objects that it is synching are person (OID {2 5 6 6}), organizationalPerson (OID {2 5 6 7}), and emailInfo (OID {1 3 24 30 2 2 1}). It uses these OIDs when creating entries.

The object class for entries is only set on Add transactions. Synchronizer-500 will not modify an existing entry's objectClass attributes if synch_object is changed.

H.1.2 Relative Distinguished Name Records

Relative distinguished name (RDN) records describe to Synchronizer-500 the X.500-specific information it will need to create the distinguished name for an entry. RDN records have ten fields delimited by the vertical line "|" character:

1. type (the string "RDN")
2. label by which this RDN is referenced
3. maximum length of a value (decimal)
4. attribute OID (OID format or hexadecimal[1])
5. syntax (syntax string (see Section H.2) or hexadecimal[1]),
6. number of RDN values allowed (decimal)
7. number of attribute values allowed (decimal)
8. uniqueness check flags, see Section H.3
9. special purpose flags, see Section H.4
10. Object Class OID for Parent Entries (OID format or hexadecimal[1])

[1] - hexadecimal formats retained for V1.0 compatibility.

The following is an example of the RDN records contained in the Synchronizer-500 default X.500 layout file:

```
RDN|distinguishedName|512|000000|00|1|0||DNAM|000000
RDN|country|2|{2 5 4 6}|printableStringSyntax|1|0|||{2 5 6 2}
RDN|organization|64|{2 5 4 10}|printableStringSyntax|1|0|||{2 5 6 4}
RDN|organizationalUnit|64|{2 5 4 11}|printableStringSyntax|4|0|||{2 5 6 5}
RDN|locality|128|{2 5 4 7}|printableStringSyntax|4|0|||{2 5 6 3}
RDN|commonName|64|{2 5 4 3}|printableStringSyntax|1|3||DVAL|000000
```

RDN record labels are referenced in the configuration file (dn_sequence, create_root, and search_context) and also in the RDF files.

Note the first record in the list (distinguishedName). This is not an actual X.500 RDN. The purpose of defining this "attribute" is to allocate space in the metafile format for Synchronizer-500 to generate a distinguishedName key. This key is then used to sort the metafile and for comparing metafile records. This is a *mandatory* X.500 layout file record which must be the first RDN record and the special purpose flag must be DNAM.

Note also the last record in the list (commonName). The last RDN field must have the DVAL (distinguished value) special purpose flag and contain only 1 RDN value. This is also the only RDN field where more than 0 attribute values allowed can be specified.

H.1.3 Attribute Records

Attribute records describe to Synchronizer-500 the X.500-specific information it will need to create attributes for an entry. Attribute records have ten fields delimited by the vertical line "|" character:

1. type (the string "ATTR")
2. label by which this attribute is referenced
3. maximum length of a value (decimal)
4. attribute OID (OID format or hexadecimal[1])
5. syntax (syntax string (see Section H.2) or hexadecimal[1]),
6. number of RDN values allowed (decimal) (must be 0 for ATTR records)
7. number of attribute values allowed (decimal)
8. uniqueness check flags, see Section H.3
9. special purpose flags, see Section H.4
10. Object Class OID - not used for ATTR records

[1] - hexadecimal formats retained for V1.0 compatibility.

The following is an example of the attribute records contained in the default X.500 layout file:

```
ATTR|countryName|2|{2 5 4 6}|printableStringSyntax|0|1|||
ATTR|organizationName|64|{2 5 4 10}|printableStringSyntax|0|1|||
ATTR|organizationalUnitName|64|{2 5 4 11}|printableStringSyntax|0|4|||
ATTR|localityName|128|{2 5 4 7}|printableStringSyntax|0|3|||
ATTR|surname|64|{2 5 4 4}|teletexStringSyntax|0|2|||
ATTR|givenName|64|{2 5 4 42}|printableStringSyntax|0|2|||
ATTR|initials|64|{2 5 4 43}|printableStringSyntax|0|2|||
ATTR|generationalQualifier|64|{2 5 4 44}|printableStringSyntax|0|2|||
ATTR|description|1024|{2 5 4 13}|printableStringSyntax|0|1|||
ATTR|seeAlso|512|{2 5 4 34}|distinguishedNameSyntax|0|1|||
ATTR|telephoneNumber|32|{2 5 4 20}|printableStringSyntax|0|4|||
ATTR|title|64|{2 5 4 12}|printableStringSyntax|0|4|||
```

Attribute record labels are referenced in the RDF files and in the configuration file (get_field, gid_field, synch_id_field, unique_field).

H.2 Syntaxes

As noted above, the X.500 layout file defines the syntaxes for RDNs and attributes. The syntax for an RDN or attribute should match the syntax defined for the attribute in the X.500 schema. One exception is the X.500 schema syntax "stringSyntax" may be either defined in the X.500 layout file as "printableStringSyntax" or "teletexStringSyntax". The following is a list of syntaxes that Synchronizer-500 supports and some examples and rules for each syntax: (See the *DEC X.500 Directory Service Management Guide* for more information.)

1. **distinguishedNameSyntax** - a distinguished name string. For example: "/C=US/O=ACME /OU=SALES". The RDN's fields must be in the layout file.
2. **facsimileTelephoneNumberSyntax** - a string representing a Facsimile Telephone Number.
3. **ia5StringSyntax** - a string of international alphabet 5 characters. Valid characters are 0-9, A-Z, a-z, blank, and !\"#$%&'( )*+,-./:;<=>?@[\]^_‘{ | }~";
4. **integerListSyntax** - a string of integers separated by commas. For example: "1,2,3,4,5".
5. **mhs-or-address-syntax** - a string containing an X.400 originator/recepient address. For example "C=US; A=VAN; P=Acme; O=Acme; G=George; S=Smith; OU1=Finance". This format is a string composed of the elements "mhs-keyword=mhs-value" with each element separated by "; ". Please note that the mhs-keyword must be given in a specified order. The list of possible mhs-keyword values and their order is as follows:

   C
   A
   X.121
   T-ID
   P
   O
   N-ID
   G
   I
   S
   Q
   OU1
   OU2
   OU3
   OU4
   DDA:
   CN

   Note

   If the above format and order is not used when constructing an mhs- or-address, unequal compares could occur at modify time and unnecessary modifications may occur.

6. **numericStringSyntax** - a string of numerics. Valid characters are 0-9 and space. For example: "85 14 7 40 243".
7. **objectIdentifierSyntax** - an object identifier string. An object identifier is an ordered sequence of integers in breaces. For example, "{2 5 6 2}".

8. **octetStringSyntax** - a string of octets. For example: 'A1B2C3'H.

9. **postalAddressSyntax** - a string of up to 6 substrings separated by commas. Each substring must be less that 30 characters. For example: "The Wicket, 14 Ace Avenue, Lower Dingle, nr Tatton, Hampshire, England"

10. **printableStringSyntax** - a string of characters. Valid characters are 0-9, A-Z, a-z, blank, and '( )+,-./:=? This syntax (or teletexStringSyntax) may be used for any attribute which is defined in the X.500 schema as "stringSyntax".

11. **teletexStringSyntax** - a string of T.61 characters. Any character is allowed (including international characters such as é and å) except: "{}©¹\" which may be encoded using the ENCODE_DELIM configuration keyword. This syntax (or printableStringSyntax) may be used for any attribute which is defined in the X.500 schema as "stringSyntax".

12. **teletexTerminalIdentifierSyntax** - a string representing a Teletex Terminal Identifier.

13. **telexNumberSyntax** - a string containing three strings each separated by a single space. For example: "1234 ABACUS US".

H.3 Uniqueness Check Flags

The X.500 layout file defines the uniqueness checks performed by Synchronizer-500 (See Section 1.3.6). This is controlled by field #8 in the RDN records and attribute records. (See Section H.1.2 and Section H.1.3.) In field #8, there are search terms (4 characters each, separated by commas if there are more than one) that define the check to be made. All attributes that contain a common search term will be grouped together in the uniqueness search.

For example, the default X.500 layout file contains the following records:

```
ATTR|emailPCOUNTRY|2|2bce0b0136|13|0|1|X484,X488|COUN|
ATTR|emailADMDNAME|16|2bce0b014A|13|0|1|X484,X488|ADMD|
ATTR|emailORGUNIT|32|2bce0b0156|14|0|1|X484,X488|OU1|
ATTR|emailORGUNIT2|32|2bce0b0157|14|0|1|X484,X488|OU2|
ATTR|emailORGUNIT3|32|2bce0b0158|14|0|1|X484,X488|OU3|
ATTR|emailORGUNIT4|32|2bce0b0159|14|0|1|X484,X488|OU4|
ATTR|emailPRMDNAME|16|2bce0b015B|13|0|1|X484,X488|PRMD|
ATTR|emailNAME|64|2bce0b018120|14|0|1|X488||
ATTR|emailORGNAME|64|2bce0b018121|14|0|1|X484,X488|ORGN|
```

All of these attributes contain an "X488" in field #8's. This means that, in this example, emailPCOUNTRY, emailADMDNAME, emailORGUNIT, emailORGUNIT2, emailORGUNIT3, emailORGUNIT4, emailPRMDNAME, emailNAME, and emailORGNAME will be used for the "X488" (the 1988 X.400) uniqueness check.

Search terms can be deleted to disable a uniqueness check. Conversely, if a custom uniqueness check is desired, additional terms can be edited into the X.500 layout file.

H.4 Special Purpose Flags

There are RDNs and attributes that Synchronizer-500 needs to know about in order for special processing to be performed on them. These records are denoted by the usage of "special" purpose flags in field #9 of the RDN and attribute records. The following table lists these flags and their meanings.

Table H–1: Synchronizer-500 Special Flags

Flag	Meaning
DNAM	distinguishedName - must be the first RDN record
DVAL	distinguished value - must be the last RDN record
COUN	X.400 country - determines field for COUNTRY_MATCH
ADMD	X.400 ADMD - determines field for ADMD_MATCH
PRMD	X.400 PRMD - determines field for PRMD_MATCH
ORGN	X.400 organization - determines field for ORGNAME_MATCH
OU1	X.400 organizationUnit 1 - determines field for ORGUNIT_MATCH
OU2	X.400 organizationUnit 2 - determines field for ORGUNIT2_MATCH
OU3	X.400 organizationUnit 3 - determines field for ORGUNIT3_MATCH
OU4	X.400 organizationUnit 4 - determines field for ORGUNIT4_MATCH
GIVN	X.400 givenname - for SNADS and PROFS username generation
SURN	X.400 surname - for SNADS and PROFS username generation
PRU	PROFS username - for SNADS and PROFS username generation
PRL	PROFS location - for SNADS and PROFS username generation
SNU	SNADS username - for SNADS and PROFS username generation
SNL	SNADS location - for SNADS and PROFS username generation
NONORM	turns off normalized comparison of this attribute

Appendix I
Synchronizer-500 Files and their Locations

The following tables contain a list of Synchronizer-500 files for an OpenVMS system and for a UNIX system.

I.1 Location of Synchronizer-500 Files for an OpenVMS System

Table I–1 lists the files used by Synchronizer-500 and their locations on an OpenVMS system. The XDSU$ROOT subdirectory of the DXD$DIRECTORY directory is referred to as Synchronizer-500's root directory (XDSU root).

Note

If Synchronizer-500 is not installed in the default location (one directory level below the DEC X.500 Directory Service root DXD$DIRECTORY), then the location specified when Synchronizer-500 was installed should be referred to as Synchronizer-500's root directory (XDSU root).

Table I–1: OpenVMS File Locations

Files	Location	Function
XDSU$MAIN.EXE	SYS$SYSTEM	Synchronizer-500 image
XDSU$IVP.COM	SYS$TEST	Synchronizer-500 Installation Verification Procedure
XDSU012.RELEASE_NOTES	SYS$HELP	Release Notes
EMAIL.SC	XDSU root	email auxiliary class schema file
XDSU_CONFIG.CHANGES_TEMPLATE	XDSU root	Changes Configuration File template
XDSU_CONFIG.EXPORT_TEMPLATE	XDSU root	Export Configuration File template

Table I–1 (Cont.): OpenVMS File Locations

Files	Location	Function
XDSU_CONFIG.IMPORT_TEMPLATE	XDSU root	Import Configuration File template
XDSU_CONFIG.TRANSACTION_TEMPLATE	XDSU root	Transaction Configuration File template
XDSU_CONFIG_INSTANCE.COM	XDSU root	Create instance subdirectory procedure
XDSU_LAYOUT.DAT	XDSU root	Synchronizer-500 X.500 layout file
XDSU_LAYOUT_PRE_V20_X500.DAT	XDSU root	Synchronizer-500 Pre V2.0 of X.500 layout file
DSU_EXPORT_DDS.RDF	XDSU root	Sample DDS export RDF file for use with DSU
DSU_CONFIG.DAT_EXPORT_DDS	XDSU root	Sample DDS export configurtion file for use with DSU
GENEXPRT.RDF	XDSU root	generic Export RDF
GENIMPRT.RDF	XDSU root	generic Import RDF
XDSU_CHANGES_MICROSOFT.COM	XDSU root	Sample Microsoft changes end-of-run command file
XDSU_CHANGES_MICROSOFT_ADD.RDF	XDSU root	Sample Microsoft changes add RDF file
XDSU_CHANGES_MICROSOFT_DELETE.RDF	XDSU root	Sample Microsoft changes delete RDF file
XDSU_CONFIG.DAT_CHANGES_MICROSOFT	XDSU root	Sample Microsoft changes configuration file
XDSU_CONFIG.DAT_EXPORT_MICROSOFT	XDSU root	Sample Microsoft export configuration file
XDSU_CONFIG.DAT_EXPORT_PROFS	XDSU root	Sample PROFS export configuration file
XDSU_CONFIG.DAT_EXPORT_SMTP	XDSU root	Sample SMTP export configuration file
XDSU_CONFIG.DAT_IMPORT_DDS	XDSU root	Sample DDS import configuration file
XDSU_CONFIG.DAT_IMPORT_MICROSOFT	XDSU root	Sample Microsoft import configuration file

Table I–1 (Cont.): OpenVMS File Locations

Files	Location	Function
XDSU_CONFIG.DAT_IMPORT_PROFS	XDSU root	Sample PROFS import configuration file
XDSU_EXPORT_MICROSOFT.COM	XDSU root	Sample Microsoft export end-of-run command file
XDSU_EXPORT_MICROSOFT.RDF	XDSU root	Sample Microsoft export RDF file
XDSU_EXPORT_PROFS.RDF	XDSU root	Sample PROFS export RDF file
XDSU_EXPORT_SMTP.COM	XDSU root	Sample SMTP export end-of-run command file
XDSU_EXPORT_SMTP.RDF	XDSU root	Sample SMTP export RDF file
XDSU_IMPORT_DDS.RDF	XDSU root	Sample DDS import RDF file
XDSU_IMPORT_MICROSOFT.COM	XDSU root	Sample Microsoft import end-of-run command file
XDSU_IMPORT_MICROSOFT.RDF	XDSU root	Sample Microsoft import RDF file
XDSU_IMPORT_MICROSOFT.TXT	XDSU root	Sample Microsoft import data file
XDSU_IMPORT_PROFS.COM	XDSU root	Sample PROFS import end-of-run command file
XDSU_IMPORT_PROFS.RDF	XDSU root	Sample PROFS import RDF file
XDSU_IMPORT_PROFS.TXT	XDSU root	Sample PROFS import data file
XDSU012.RELEASE_NOTES	XDSU root	Release Notes

I.2 Location of Synchronizer-500 Files for a UNIX System

Table I–2 lists the files used by Synchronizer-500 and their locations on a UNIX system. The directory /var/dxd/xdsu is referred to as Synchronizer-500's root directory (XDSU root).

Table I–2: UNIX File Locations

Files	Location	Function
xdsu	/usr/sbin	Synchronizer-500 image
xdsu012.release_notes	/usr/doc	Release Notes
email.sc	XDSU root	email auxiliary class schema file
xdsu_config.changes_template	XDSU root	Changes Configuration File template
xdsu_config.export_template	XDSU root	Export Configuration File template
xdsu_config.import_template	XDSU root	Import Configuration File template
xdsu_config.transaction_template	XDSU root	Transaction Configuration File template
xdsu_config_instance	XDSU root	Create instance subdirectory procedure
xdsu_layout.dat	XDSU root	Synchronizer-500 X.500 layout file
xdsu_layout_pre_V20_x500.dat	XDSU root	Synchronizer-500 Pre V2.0 of X.500 layout file
dsu_export_dds.rdf	XDSU root	Sample DDS export RDF file for use with DSU
dsu_config.dat_export_dds	XDSU root	Sample DDS export configurtion file for use with DSU
genexprt.rdf	XDSU root	generic Export RDF
genimprt.rdf	XDSU root	generic Import RDF
xdsu_changes_microsoft.com	XDSU root	Sample Microsoft changes end-of-run command file
xdsu_changes_microsoft_add.rdf	XDSU root	Sample Microsoft changes add RDF file
xdsu_changes_microsoft_delete.rdf	XDSU root	Sample Microsoft changes delete RDF file
xdsu_config.dat_changes_microsoft	XDSU root	Sample Microsoft changes configuration file
xdsu_config.dat_export_microsoft	XDSU root	Sample Microsoft export configuration file

Table I–2 (Cont.): UNIX File Locations

Files	Location	Function
xdsu_config.dat_export_profs	XDSU root	Sample PROFS export configuration file
xdsu_config.dat_export_smtp	XDSU root	Sample SMTP export configuration file
xdsu_config.dat_import_dds	XDSU root	Sample DDS import configuration file
xdsu_config.dat_import_microsoft	XDSU root	Sample Microsoft import configuration file
xdsu_config.dat_import_profs	XDSU root	Sample PROFS import configuration file
xdsu_export_microsoft.com	XDSU root	Sample Microsoft export end-of-run command file
xdsu_export_microsoft.rdf	XDSU root	Sample Microsoft export RDF file
xdsu_export_profs.rdf	XDSU root	Sample PROFS export RDF file
xdsu_export_smtp.com	XDSU root	Sample SMTP export end-of-run command file
xdsu_export_smtp.rdf	XDSU root	Sample SMTP export RDF file
xdsu_import_dds.rdf	XDSU root	Sample DDS import RDF file
xdsu_import_microsoft.com	XDSU root	Sample Microsoft import end-of-run command file
xdsu_import_microsoft.rdf	XDSU root	Sample Microsoft import RDF file
xdsu_import_microsoft.txt	XDSU root	Sample Microsoft import data file
xdsu_import_profs.com	XDSU root	Sample PROFS import end-of-run command file
xdsu_import_profs.rdf	XDSU root	Sample PROFS import RDF file
xdsu_import_profs.txt	XDSU root	Sample PROFS import data file
xdsu012.release_notes	XDSU root	Release Notes

Note

The sample end-of-run DCL command files are intended for use on an OpenVMS system even though they are included in the UNIX kit.

Index

A

acronyms, xii
allocating disk space, 4–2
asterisk *
  export and changes RDF syntax, 3–20
  use in converting metafile data, 2–6
  use with Group ID during export, 1–14
attribute
  definition of, 1–3
  type, 1–3
  value, 1–3
attributes
  descriptions of, D–1
  Directory Group ID, 1–10
  length, D–1
  mandatory, 1–3
  optional, 1–3
  specifying multiple values, 3–21
  syntax, D–1
  values allowed, D–1
auxiliary class, 1–3

C

changes, 1–14
  algorithm, 2–1
  changes operations illustration, 1–15
  configuration file, 3–12
  configuration file template, F–13
  definition of, 1–2
  example Microsoft Mail changes process, 5–31 to 5–37
  file size estimates, 4–3
  processing, 2–6 to 2–7
  RDF conditional statement syntax, 3–19
  RDF formatting operator characters, 3–20
  synchronization activities, 3–3
  use of foreign directory files, 1–10
class, 1–3
components, 1–6
configuration files
  changes configuration file, 3–12
  changes keyword descriptions, F–13
  changes template, F–13
  example DDS import configuration file, A–11
  example Microsoft Mail changes configuration file, 5–32
  example Microsoft Mail export configuration file, 5–27
  example Microsoft Mail import configuration file, 5–10
  example PROFS export configuration file, C–20
  example PROFS import configuration file, C–11
  example SMTP export configuration file, B–5
  export configuration file, 3–10
  export keyword descriptions, F–10
  export template, F–10
  import configuration file, 3–6
  import keyword descriptions, F–2
  import template, F–2
  introduction of, 1–6
  location, 3–5
  setting up, 3–4
  transaction configuration file, 3–8
  transaction keyword descriptions, F–8
  transaction template, F–8
creating an instance subdirectory, 3–3

D

DDS synchronization
  example, A–1
  export configuration file on the MAILbus system, A–6
  import configuration file illustration, A–11
  import log file illustration, A–14
  import process, A–1

DDS synchronization (cont'd)
initial load import processing illustration, A–13
MAILbus system activities, A–2
RDF for DDS export on the MAILbus system, A–3
sample X.500 entry illustration, A–16
setting up the import RDF, A–7
X.500 system activities, A–7
directory entries
definition of, 1–2
directory Group ID
See Group ID
Directory Information Tree, 1–1, 1–3
definition of, 1–2
Directory User Agent, 1–1
distinguished name
definition of, 1–3
generation of, 1–12
minimizing name clashes, 4–2
DIT
See Directory Information Tree
DUA
See Directory User Agent

E

emailInfo auxiliary class, 1–3
attribute definitions, D–4
attribute descriptions, D–4
attribute lengths, D–4
attribute syntax, D–4
attribute values allowed, D–4
definition of, xii
including in the DEC X.500 Directory Service schema, 3–1
uniqueness checking, 1–16
X.400 attributes for export, 1–13
error messages, E–1
export, 1–13
algorithm, 2–1
configuration file, 3–10
configuration file template, F–10
definition of, 1–2
example Microsoft Mail export process, 5–23 to 5–31
example PROFS export process, C–19 to C–22
example SMTP export process, B–2 to B–8
export operations illustration, 1–13
file size estimates, 4–3
processing, 2–4 to 2–6
RDF conditional statement syntax, 3–19
RDF formatting operator characters, 3–20
export (cont'd)
selection criteria, 1–13
synchronization activities, 3–3
use of foreign directory files, 1–10

F

features and capabilities, 1–4
files
OpenVMS system, I–1
UNIX system, I–4
foreign directory, 1–5
foreign directory files
introduction of, 1–9
mapping to X.500 attributes for import and transaction illustration, 3–15
mapping X.500 attributes on export and changes illustration, 3–16
transferring, 1–21
use in changes, 1–10
use in export, 1–10
use in import, 1–9
use in transaction, 1–9

G

GDIs
See global domain identifiers
GID
See Group ID
global domain identifiers, 5–6, 5–26
Group ID, 1–10
asterisk * character during export, 1–14
usage during import initial load, 1–12
use during X.500 extract for export, 2–4
use during X.500 extract for import, 2–2

I

import, 1–11
algorithm, 2–1
concatenated RDF statements, 3–17
configuration file, 3–6
configuration file template, F–2
definition of, 1–2
example DDS import process, A–1 to A–16
example Microsoft Mail import process, 5–4 to 5–22
example PROFS import process, C–5 to C–18
file size estimates, 4–3
import operations illustration, 1–11
initial load definition, 1–11
initial load operation, 1–12

import (cont'd)
processing, 2–2 to 2–4
RDF conditional statement syntax, 3–18
synchronization activities, 3–2
update definition, 1–11
update operation, 1–12
use of foreign directory files, 1–9
initial load, 5–18
instances, 1–5
introduction, 1–1

L

layout file
description, H–1
introduction of, 1–9
RDN and attribute definitions, D–1
log files
example DDS import, A–14
example Microsoft Mail import, 5–16
example Microsoft Mail update import, 5–20
example PROFS import, C–15
example PROFS update import, C–18
monitoring, 4–4

M

metafiles, 1–5
introduction of, 1–9
spreading out, 4–4
usage during changes, 2–7
usage during export, 2–4 to 2–6
usage during import, 2–2 to 2–4
usage during transaction, 2–4
Microsoft Mail synchronization
changes configuration file illustration, 5–33
changes process, 5–31
changes processing illustration, 5–35
changes subdirectory creation, 5–31
changes X.500 "ADD" data illustration, 5–36
changes X.500 "DELETE" data illustration, 5–37
example, 5–1
example configuration and addressing, 5–3
export configuration file illustration, 5–27
exported X.500 data illustration, 5–30
export process, 5–23
export processing illustration, 5–29
export subdirectory creation, 5–23
import configuration file illustration, 5–10
import log file illustration, 5–16
import OpenVMS mail message illustration, 5–15
Microsoft Mail synchronization (cont'd)
import process, 5–4
import subdirectory creation, 5–4
initial load import processing illustration, 5–13
running the initial load import, 5–13
running the update import, 5–18
sample X.500 entry illustration, 5–2
synchronization process overview, 5–2
typical file illustration, 5–2
update import log file illustration, 5–20
update import processing illustration, 5–18
writing the changes RDFs, 5–32
writing the export RDF, 5–24
writing the import RDF, 5–5
X.500 attribute mapping illustration, 5–5
X.500 to Microsoft Mail mapping illustration, 5–24

O

organizationalPerson class, 1–3
organizationalRole class, 1–3

P

pound sign #
export and changes RDF syntax, 3–20
use in converting metafile data, 2–6
PROFS alias generation, 1–20
PROFS synchronization
activities, C–1
example, C–1
export configuration file illustration, C–20
exported X.500 data illustration, C–22
export process, C–19
export processing illustration, C–21
export RDF illustration, C–19
export subdirectory creation, C–19
import configuration file illustration, C–11
import log file illustration, C–15
import process, C–5
import subdirectory creation, C–5
initial load import processing illustration, C–13
messaging network illustration, C–3
running the initial load import, C–12
running the update import, C–16
sample X.500 entry illustration, C–16
typical Nickname Manager file illustration, C–3
update import log file illustration, C–18
update import processing illustration, C–16
writing the import RDF, C–6
X.500 attribute mapping illustration, C–5

Q

question mark ?
  export and changes RDF syntax, 3–19
  import RDF syntax, 3–18
  usage in an export RDF, 3–17
  usage in an import RDF, 3–17
  use in converting metafile data, 2–6

R

RDF
  See record description files
RDN
  See relative distinguished name
record description files
  concatenated statements, 3–17
  conditional statements, 3–18
  definition of, 1–4
  example DDS import RDF, A–8
  example Microsoft Mail changes RDFs, 5–32
  example Microsoft Mail export RDF, 5–24
  example Microsoft Mail import RDF, 5–5
  example PROFS export RDF, C–19
  example PROFS import RDF, C–6
  example SMTP export RDF, B–3
  export and changes formatting operator characters, 3–20
  introduction of, 1–7
  keywords listing for DEC X.500 Directory Service, D–1
  keywords listing for emailInfo auxiliary class, D–4
  mapping into X.500 illustration, 1–8
  record formats, 3–17
  rules, 3–21
  setting up, 3–14
  syntax, 3–17
relative distinguished name
  definition of, 1–3
  descriptions of, D–1
residentialPerson class, 1–3
root directory, 3–5
  definition of, xii

S

schema, 3–1
Search Context
  use during X.500 extract for import, 2–3, 2–5
set up, 3–1
SMTP synchronization
  example, B–1
  export configuration file illustration, B–5
  exported X.500 data illustration, B–8
  export process, B–2
  export processing illustration, B–6
  export RDF illustration, B–3
  export subdirectory creation, B–3
  messaging network illustration, B–2
SNADS alias generation, 1–20
starting, 3–21
status codes, G–1
subdirectory, creating, 3–3
suggestions for running, 4–1
superior entries
  created by Synchronizer-500, 1–4
Synch ID
  use during X.500 extract for export, 2–5
  use during X.500 extract for import, 2–2
SYNCH_ID, 1–10

T

terms, xii
tilde ~
  export and changes RDF syntax, 3–19
  import RDF syntax, 3–18
  usage in an export RDF, 3–17
  usage in an import RDF, 3–17
  use in converting metafile data, 2–6
transaction
  algorithm, 2–1
  concatenated RDF statements, 3–17
  configuration file, 3–8
  configuration file template, F–8
  file size estimates, 4–3
  processing, 2–4
  RDF conditional statement syntax, 3–18
  synchronization activities, 3–3
transaction mode, 1–12
  definition of, 1–2
  operation, 1–12
  use of foreign directory files, 1–9
transaction status codes, G–1

U

uniqueness checking, 1–4
  failure status codes, 1–18
  standard, 1–16
  user defined, 1–18
usage with X.500, 1–2

V

verbose, 5–17

X

X.500, 1–1, 1–2, 1–5
  attribute listing, D–1
  auxiliary class, 1–3
  class, 1–3
  directory entries, 1–2
  Directory Information Tree, 1–2
  organizationalPerson class, 1–3
  organizationalRole class, 1–3
  RDN and attribute definitions, D–1
  residentialPerson class, 1–3
  schema, 3–1
X.521 Recommendations, 1–1