Digital X.500 Directory Synchronizer Release Notes V1.4-1

Software Version: V1.4-1

Date: January 1997

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 ©1997 by Digital Equipment Corporation.

All rights reserved.

Printed in U.S.A.

Trademarks

The following are trademarks of Digital Equipment Corporation: AXP, DEC, DECnet, Digital, OpenVMS, VAX, VMS and the DIGITAL logo.

The following are trademarks of other companies: OSF and OSF/1 are registered trademarks of Open Software Foundation, Inc. OSI is a registered trademark of CA Management, Inc. UNIX is a registered trademark in the United States and other contries licensed exclusively through X/Open Company Ltd.

Contents

Chapter 1 V1.4-1 New Features

1.1 Reduction of XDSU Pre and Post Processing requirements 1–1
1.1.1 Allow Input File to have Variable Length Fields 1–1
1.1.2 Allow Input File to have Header Lines 1–3
1.1.3 Allow Output File to have Header Lines 1–3
1.1.4 Allow Input and Output File Filtering 1–3
1.2 Identify a record using user-defined key fields (other than DN) 1–4
1.3 Initial-Load-Only Fields 1–4
1.4 User-written Unique Field Generation Programs 1–5
1.5 Maximum Adds and Deletes for IMPORT mode 1–6
1.6 Mark-for-Delete Flag for IMPORT mode 1–7

Chapter 2 Problems fixed in V1.4-1

2.1 Input File now Sorted when using new V1.4 CHANGES mode 2–1
2.2 sort PATH now fully qualified on Digital UNIX 2–1
2.3 Importing Undefined Syntax Values greater than 127 bytes 2–1
2.4 BIND_DSA paramenter not working on Digital Unix 2–1

Chapter 3 V1.4 New Features

3.1 Limited Use License Support Added 3–1
3.2 CHANGES Mode Optionally Compares X.500 against Foreign Directory 3–1
3.3 Case Changes Optionally Recognized 3–2
3.4 More Detail Logs for CHANGES mode Added 3–2
3.5 Substring filter ability for GET_VALUE added 3–2
3.6 MHS-OR-ADDRESS Syntax Comparision Restriction Removed 3–3
3.7 Script Files Added to make running XDSU easier 3–3
3.8 New XDSU subdirectories added 3–3

3.9 XDSU_CONFIG_INSTANCE procedure improved 3–4
3.10 INITIAL_LOAD parameter defaults to NO 3–4
3.11 SYNCH_ID no longer required 3–4
3.12 Example Files moved to a subdirectory 3–4
3.13 commonName can have UNIQUE_FIELD processing 3–4

Chapter 4 Problems fixed in V1.4

4.1 BIND_DSA parameter for RFC1006 addresses 4–1
4.2 Modifying the same records in TRANSACTION mode 4–1
4.3 Problems extracting large amounts of X.500 entries 4–1
4.4 Extra characters added to ENCODE_DELIM processing 4–2

Appendix A XDSU V1.4-1 RDF Syntax

A.1 RDF Syntax A–1
A.1.1 RDF action format for Input Files A–1
A.1.2 RDF action format for Output Files A–2
A.1.3 RDF conditional format A–3
A.1.4 Other RDF syntax Rules A–4
A.1.4.1 Comments A–4
A.1.4.2 Line Continuation A–4
A.1.4.3 Whitespace A–4

Chapter 1

V1.4-1 New Features

1.1 Reduction of XDSU Pre and Post Processing requirements

For XDSU V1.4-1, new features have been added which reduce and may eliminate the need to write custom scripts to pre-process foreign directory files before they are imported with XDSU and post-process foreign directory files after they exported with XDSU. These new features allow:

- an input file to have variable length fields (separated by a string) instead of requiring fix-lengthed, column-aligned fields.
- an input file to have header lines which are skipped.
- an output file to have header lines which are included.
- input and output file filtering (the ability to ignore certain records depending on a condition).

1.1.1 Allow Input File to have Variable Length Fields

For V1.4-1, XDSU now allows input from a foreign directory input file which is not fixed-length and column-aligned but has variable-length fields which are delimited by a field separator. Then individual fields can then in turn be delimited by a sub-field separator.

For example, the following input file records can now be processed:

```
Smith,John,123-4567,,England,MS:AA/BB/Smith%SMTP:smith@boat.com
Pocohontas,,123-456-7890,Virginia,,SMTP:poco@land.com
```

instead of having to be pre-processed to the following:

```
smith       john  123-4567             England  smith@boat.com
pocohontas        123-456-7890 Virginia         poco@land.com
```

To support this option, one new XDSU_CONFIG.DAT parameter for IMPORT, TRANSACTION, and CHANGES mode (when using an input file) has been added:

```
RDF_FIELD_SEPARATOR=  ! set to 1 or 2 double quoted strings
                      ! separated by a comma if necessary
```

To use variable length fields, set RDF_FIELD_SEPARATOR = "field_separator_string" , "sub-field_separator_string"

The sub-field separator string is optional.

To use RDF fields, the syntax in the RDF file changes from:

```
    rec.column.length
```

to:

```
    rec-fieldno                    or
    rec-fieldno.column.length

      'fieldno' is the field# (1-N)
      Use the second method to allow field substrings.
      'length' can be set to # - which means use the
      remaining length of the field.
```

and to use sub-fields:

```
    rec-fieldno-subfieldno                    or
    rec-fieldno-subfieldno.column.length

      'subfieldno' is the sub-field# (1-N) within a field
```

Therefore, the above input file, which has fields separated by a comma and subfields separated by a percent sign (RDF_FIELD_SEPARATOR = ",", "%") has the following RDF file:

```
    surname = 1-1
    givenname = 1-2
    commonName = 1-2
    ? 1-2 : commonName = " "
    commonName = 1-1
    telephoneNumber = 1-3
    stateOrProvince = 1-4
    countryName = 1-5
    ? 1-6-1.1.5 == "SMTP:" : rfc822addr = 1-6-1.6.#
    ? 1-6-2.1.5 == "SMTP:" : rfc822addr = 1-6-2.6.#
```

instead of (which requires a pre-processing program):

```
    surname = 1.1.15
    givenname = 1.16.15
    commonName = 1.16.15
    ? 1.16.15 : commonName = " "
    commonName = 1.1.15
    telephoneNumber = 1.31.15
    stateOrProvince = 1.46.15
    countryName = 1.61.15
    rfc822addr = 1.76.30
```

See Appendix A for a complete description of all RDF Syntax rules and usage.

1.1.2 Allow Input File to have Header Lines

For V1.4-1, XDSU now allows input from a foreign directory input file to have header lines to be skipped before processing the input data.

To support this option, one new XDSU_CONFIG.DAT parameter for IMPORT, TRANSACTION, and CHANGES mode (when using an input file) has been added:

```
NUM_INPUT_HEADER_LINES=   ! set to number of lines
```

Set this value equal to the number of header lines in the input file which are to be skipped. The default is 0 - no header lines. Note: this parameter has no relationship to the NREC_ SUBSCR parameter.

For example, the following input file can now be processed:

```
obj-class,last,first,...
Mailbox,Smith,Jeff, ...
Mailbox,Smith,John, ...
```

By setting NUM_INPUT_HEADER_LINES=1, the first line is skipped.

1.1.3 Allow Output File to have Header Lines

For V1.4-1, XDSU now allows a foreign directory output file to have header lines to be included before processing the output data.

To support this option, one new XDSU_CONFIG.DAT parameter for EXPORT and CHANGES mode has been added:

```
OUTPUT_HEADER_FILE=   ! set to filespec of the header file
```

If a value is specified for this parameter (the default is no header file), the header file is copied to the output file before the data records are written.

1.1.4 Allow Input and Output File Filtering

For V1.4-1, XDSU now allows input from a foreign directory input file to be "filtered" so that certain records which meet user-defined conditions can be "ignored". This can be especially useful when using IMPORT mode since the foreign directory input file contains a full dump of a foreign directory which may contain records which are "local" and should not be synchronized.

To support this option, a new RDF syntax has been implemented:

```
? condition : IGNORE
```

This causes any input file which matched this condition to not be used.

For example, the following will skip any records that contain the substring "admin" in field 2 in the input record.

```
? "admin" << 1-2 : IGNORE
```

In addition, to reduce post-processing requirements, this syntax can be used when EXPORTing records as well, which causes X.500 extracted records to not be written to the output foreign directory file.

1.2 Identify a record using user-defined key fields (other than DN)

When updating existing X.500 records with a TRANSACTION Mode Modify or Delete File, or when comparing X.500 records against a foreign directory file using CHANGES Mode, XDSU currently requires the DN be contained in (or generated from) the input file so it knows which record to modify, delete, or compare.

In XDSU V1.4-1, alternative "index" fields may be specified - instead of the DN - which can identify the X.500 record to be modified, deleted, compared.

This requires the following new XDSU_CONFIG.DAT field:

```
KEY_FIELDS=    ! list X.500 field
               ! if more than one - separate by commas
```

If KEY_FIELDS is specified, therefore, the DN_SEQUENCE is not used, but the key fields are used to identify the X.500 record to be modified, deleted, or compared. The key fields must produce a single X.500 record to be used.

This can be used in the following example.

- An input file needs to update existing X.500 entries with additional information from another source using a TRANSACTION Mode Modify File.
- The input file cannot identify the record using the DN.
- The input file does contain the badge number for each record which is unique for each user and contained in the existing X.500 entries.
- To use the input file, therefore, set KEY_FIELDS=badgeNumber and XDSU will read X.500 using the badge number from the input file ("SEARCH where badgeNumber=value") to get the DN before comparing and updating the X.500 record.

1.3 Initial-Load-Only Fields

For V1.4-1, using IMPORT mode, XDSU now allows fields to be entered in X.500 only during the initial ADD of the record. This allows XDSU to provide an initial value for a field but it no longer "owns" (or updates) the field value on subsequent imports. Therefore, if the value of the field is changed by another DUA, XDSU would not set it back to its initial value during the next IMPORT.

This requires the following new XDSU_CONFIG.DAT field for IMPORT mode:

```
ADD_ONLY_FIELDS=  !if more than 1 field, separate by commas
```

An example use for this would be the userPassword field which can be set to some default value when the record is created, but will never again be updated by XDSU, but "owned" by the user through Infobroker.

1.4 User-written Unique Field Generation Programs

For V1.4-1, XDSU now supports calling a user-written procedure with optional parameters to generate a unique field value.

To support this option, two new XDSU_CONFIG.DAT parameters for IMPORT and TRANSACTION mode have been added:

```
MAKE_UNIQUE_PROGRAM= !command/program to run
MAKE_UNIQUE_PARAMS=  !if more than 1 field, separate by commas
```

If MAKE_UNIQUE=Y, and the UNIQUE_FIELD is not unique on the first attempt, MAKE_UNIQUE_PROGRAM is spawned (can be a command procedure or executable) with the fields listed in MAKE_UNIQUE_PARAMS as the parameters. The attempt number will also be passed to the user program as a parameter. The user program will then return another unique field value to try. This will be repeated until a successful unique value is generated.

The way the MAKE_UNIQUE_PROGRAM returns a unique value differs between the OpenVMS and Digital Unix versions of XDSU.

On OpenVMS (Vax or Alpha), the value is returned through the job logical XDSU_UNIQUE_FIELD_VALUE. Here is an OpenVMS example:

```
XDSU_CONFIG.DAT parameters:

 UNIQUE_FIELD = emailUSE3
 MAKE_UNIQUE = YES
 MAKE_UNIQUE_PROGRAM = @MAKE_UNIQUE.COM
 MAKE_UNIQUE_PARAMS =  surname,givenname

MAKE_UNIQUE.RDF

 emailUSE3 = surname

MAKE_UNIQUE.COM

 $!
 $! MAKE_UNIQUE.COM
 $!
 $!    p1 - Attempt Number (0 to N)
 $!    p2 - Last Name
 $!    p3 - First Name
 $!
 $ if P1 .eqs. "0"
 $ then
 $    val = P2 + f$extract(0,1,P3)    ! LAST + FIRST ONE
 $ else
 $ if P1 .eqs. "1"
 $ then
 $    val = P2 + f$extract(0,2,P3)    ! LAST + FIRST TWO
 $ else
 $    val = P2 + f$extract(0,2,P3) + P1 ! LAST + FIRST TWO + N
 $ endif
 $ endif
 $!
 $ define/job XDSU_UNIQUE_FIELD_VALUE "''val'"
 $ exit
```

On Digital UNIX, the value is returned through the stdout pipe. Here is a Digital UNIX example:

```
xdsu_config.dat parameters:

 UNIQUE_FIELD = emailUSE3
 MAKE_UNIQUE = YES
 MAKE_UNIQUE_PROGRAM = ./make_unique
 MAKE_UNIQUE_PARAMS =  surname,givenname

make_unique.rdf

 emailUSE3 = surname

make_unique  (must be executable)

 #!/usr/bin/ksh
 #
 # $1 - attempt number (0-N)
 # $2 - surname
 # $3 - givenName
 #
 if [[ $1 -eq 0 ]]
 then
   typeset -L1 s=$3
   echo "$2$s"        # last + first one
 elif [[ $1 -eq 1 ]]
 then
   typeset -L2 s=$3
   echo "$2$s"        # last + first two
 else
   typeset -L2 s=$3
   echo "$2$s$1"      # last + first two + N
 fi
```

1.5 Maximum Adds and Deletes for IMPORT mode

In XDSU V1.4-1, when running full IMPORT mode, an optional threshold on the maximum number of deletes and adds allowed has been added. Using this option helps ensure that XDSU has a valid input file or to prevent too many changes from being entered into X.500. The thresholds are tested before any adds or deletes are applied.

This requires two new xdsu_config.dat parameters for IMPORT mode:

```
  MAX_ADD_TRANSACTIONS=      !(default is no limit)
  MAX_DELETE_TRANSACTIONS=   !(default is no limit)
```

If a MAX ADD or DELETE limit is specified, XDSU will compare the metafiles to determine the #adds and the #deletes that would be generated. If a maximum is exceeded, XDSU exits with an error before doing any changes of any kind to X.500.

Note that MAX_ADD_TRANSACTIONS should not be set until after an initial load.

1.6 Mark-for-Delete Flag for IMPORT mode

In XDSU V1.4-1, when running full IMPORT mode, instead of deleting records permanently from X.500, a way to "mark records for delete" has been added. This will keep the record in X.500 but with a modified attribute that flags the record. This is useful if manual intervention is needed to check on records that are to be deleted.

This requires three new xdsu_config.dat parameters for IMPORT mode:

```
MARK_FOR_DELETE =             !(YES or NO - default is NO)
MARK_FOR_DELETE_FIELD =       ! X.500 attribute to use as flag
MARK_FOR_DELETE_FIELD_VALUE= ! value to put in flag field when
                              ! a record is marked for delete
```

If MARK_FOR_DELETE = YES, and a record in X.500 is to be deleted using IMPORT mode (because it is no longer in the foreign directory input file), the record is modified so that MARK_FOR_DELETE_FIELD = MARK_FOR_DELETE_FIELD_VALUE instead of being deleted from X.500.

If a record is "marked-for-delete" and it reappears in the foreign directory input file on a subsequent import, the record is modified to remove the MARK_FOR_DELETE_FIELD attribute.

Chapter 2

Problems fixed in V1.4-1

2.1 Input File now Sorted when using new V1.4 CHANGES mode

Using the new CHANGES mode (which compares X.500 against the foreign directory directly without using an old metafile - see section 3.2), the foreign directory was not properly sorted before the compare. This resulted in incorrect changes being generated, unless the foreign directory was sorted prior to running XDSU.

This has fixed in XDSU V1.4-1.

2.2 sort PATH now fully qualified on Digital UNIX

When sorting metafiles, XDSU uses the unix sort command in /usr/bin. XDSU, however, was assuming the PATH is properly set up to run the sort.

In XDSU V1.4-1, the sort command is now explicitly specified as /usr/bin/sort in case the PATH does not contain /usr/bin.

2.3 Importing Undefined Syntax Values greater than 127 bytes

In V1.4-1, XDSU now supports importing Undefined Syntax Values greater than 127 bytes. The limit is now 4096 bytes - which is the maximum XDSU supported field length for any X.500 attribute.

2.4 BIND_DSA paramenter not working on Digital Unix

BIND_DSA parameter causes a core dump on Digital Unix. This has been identified as a problem in the X.500 API and will be fixed in a version of X.500 subsequent to X.500 V3.0 ECO 3.

Meanwhile, this problem can be worked around by specifying the presentation address in the dua.defaults file instead of using the BIND_DSA parameter.

Chapter 3
V1.4 New Features

3.1 Limited Use License Support Added

XDSU V1.4 can now support multiple license levels based on the maximum number of X.500 entries that XDSU can import or export. This number is defined in the "Product Token" field of the license PAK. A value in this field determines the maximum number of entries that can be imported or exported. No value in this field means an unlimited number of entries can be imported or exported (as in previous versions of XDSU).

As of this writing, the different levels and prices have not yet been determined, although the unlimited license price and part number has not changed.

3.2 CHANGES Mode Optionally Compares X.500 against Foreign Directory

In export CHANGES mode, the OLD_METAFILE is a copy of what was in X.500 during the prior changes run. This is compared against the current state of X.500, which is extracted into the NEW_METAFILE, and changes export files are generated. For the next XDSU run, the NEW_METAFILE is renamed to the OLD_METAFILE, etc.

However, XDSU has no knowledge that the changes generated will get correctly applied to the foreign directory or database. XDSU would have to compare X.500 against the foreign directory directly to see which changes should be generated.

In XDSU V1.4, therefore, export CHANGES mode can now accept an OPTIONAL foreign directory extract file, which can be compared against X.500 to generate changes the to be exported back to the foreign directory. This could be used for foreign directories which are not reliably guaranteed to properly accept changes or can fall out of synch somehow with X.500 due to external factors. The foreign directory, however, needs to able to generate a full export of entries which were previously imported from XDSU.

To use this option, four new XDSU_CONFIG.DAT parameters for CHANGES mode have been added: RDF_FILE, INPUT_FOR_DIR_FILE, NREC_SUBSCR, and DN_SEQUENCE. These will be used to create the OLD_METAFILE. The OLD_METAFILE and NEW_METAFILE

parameters can be left blank as these will now be temporary files that will not be needed from run to run.

If not using this new option (as before), RDF_FILE must be blank and values for OLD_METAFILE and NEW_METAFILE must be specified.

3.3 Case Changes Optionally Recognized

In previous versions of XDSU, when XDSU imported data into X.500, an attribute was not modified if all that was different was the case of the value. This was consistent with X.500 attributes which are usually case insensitive.

However, because other foreign directories or databases, which XDSU may be synching X.500 with, may be case sensitive, XDSU may now be able to make case changes if desired.

A new, optional XDSU_CONFIG.DAT parameter, RECOGNIZE_CASE_CHANGES, has been added for V1.4. The default will be NO, which will maintain the current functionality of ignoring case changes. If YES, however, in IMPORT and TRANSACTION mode, any case changes will result in the modification of the attribute. And if the distinguished value RDN (i.e. the last RDN value in the DN) changes case, the X.500 entry will be deleted and added with the new case.

In EXPORT and CHANGES mode, setting this parameter to YES will cause case changes to be recognized and exported to foreign directories or databases.

3.4 More Detail Logs for CHANGES mode Added

In V1.4, if VERBOSE=YES in CHANGES mode, XDSU will write to stdout/sys$output the DN of each entry that is used in the DEL, ADD, or MOD files. This is similar to the verbose output used in IMPORT mode.

This allows XDSU to provide easier auditing and reporting of activity during CHANGES mode, which has been requested by some customers.

3.5 Substring filter ability for GET_VALUE added

In V1.4, in EXPORT or CHANGES mode, XDSU now allows specifying a wildcard for the GET_VALUE parameter. For example:

```
GET_FIELD=surname
GET_VALUE=A*
```

This would return all entries where surname begins with 'A'.

```
GET_VALUE=*A*B*
```

This would return all entries where surname contains an 'A' followed by a 'B'

Also, specifying a approximate name match is supported.

```
GET_VALUE=~SMITH
```

This would return all entries where surname is approxiamately SMITH.

This feature has also been implemented for the *_MATCH parameters (e.g. ORGNAME_ MATCH, ORGUNIT_MATCH, etc.) in EXPORT and CHANGES mode. The attributes used for GET_VALUE and the *_MATCH parameters must support substring and approximate match to utilize this feature.

3.6 MHS-OR-ADDRESS Syntax Comparision Restriction Removed

In previous versions of XDSU, if the format for specifying MHS-OR-ADDRESSES SYNTAX Attributes in the input file did not exactly match what was extracted a modification was made during each import - even if the value encoded to the same internal value.

In V1.4, XDSU will now compare the internal values so needless modifies will no longer be performed.

3.7 Script Files Added to make running XDSU easier

In V1.4, for the Digital UNIX version of XDSU only, several optional script (command) files have been added to the XDSU root directory which can be used to make running xdsu instances easier:

```
do_import   - script to do a single xdsu import
do_trans    - script to do a single xdsu transaction mode import
do_export   - script to do a single xdsu export
do_change   - script to do a single xdsu changes mode export
do_change1  - script to do a single xdsu changes mode export
              where X.500 is compared against a foreign dir.
do_archive  - script to archive data files (called by others)
do_send_mail- script to send mail (called by others)
do_synch    - example master synch script, which calls
              do_import, do_export,  etc.
```

The output from the do_* scripts produces a good overview of the synch process. So this output could be captured as a synchronization summary report. The detaled XDSU output is also captured in the individual directories (xdsu.log) if needed.

See the example do_synch script and the comments in the do_* scripts for more information on how to use the optional procedures.

3.8 New XDSU subdirectories added

In XDSU V1.4, for the Digital UNIX version of XDSU only, three general purpose subdirectories are created during installation which help organize XDSU data.

```
/var/dxd/xdsu/xfer_in - input files (exports from other systems)
/var/dxd/xdsu/xfer_out - output files (imports to other systems)
/var/dxd/xdsu/logs - central logging directory
```

Use of these directories is optional but are helpful if using the do_* script files. The xfer_in directory provides a central place for all foreign directory exports to be placed which are then used as import files into X.500. The xfer_out directory provides a central place for all X.500 exports to be placed which are then used as foreign directory imports. The logs directory provides a central place to collect all log files.

3.9 XDSU_CONFIG_INSTANCE procedure improved

In V1.4, the XDSU_CONFIG_INSTANCE command procedure has been improved to automatically edit the XDSU_CONFIG.DAT file with default values based on the XDSU mode type (IMPORT, TRANSACTION, EXPORT, or CHANGES) and the name of subdirectory. Also, if an IMPORT or TRANSACTION mode instance, the Group ID value will be added to XDSU_VALID_ID.DAT.

3.10 INITIAL_LOAD parameter defaults to NO

During IMPORT or TRANSACTION mode, if the INITIAL_LOAD=YES, then the GroupID is added to XDSU_VALID_ID.DAT. Since the Group ID can now be entered into the XDSU_VALID_ID.DAT using the XDSU_CONFIG_INSTANCE procedure, the INITIAL_LOAD parameter is no longer required and defaults to NO.

3.11 SYNCH_ID no longer required

The SYNCH_ID and SYNCH_ID_FIELD parameters in XDSU_CONFIG.DAT are now optional.

Although it is recommended that all X.500 entries that XDSU imports be tagged with a SYNCH_ID, it isn't really necessary since there is also a group id attribute that is stored with every entry.

3.12 Example Files moved to a subdirectory

In XDSU V1.4, for the Digital UNIX version of XDSU only, the example files are stored in a subdirectory of the XDSU root directory (/var/dxd/xdsu/examples), instead of in the XDSU root directory.

If upgrading from a previous version of XDSU, old example files in the xdsu root directory should be removed.

3.13 commonName can have UNIQUE_FIELD processing

In XDSU V1.4, the Distinguished Value RDN (DVAL) attribute, which is, by default commonName, can be used in UNIQUE_FIELD processing. Prior to V1.4, only attributes which were not RDN values in XDSU_LAYOUT.DAT could be used in UNIQUE_FIELD processing.

Chapter 4
Problems fixed in V1.4

4.1 BIND_DSA parameter for RFC1006 addresses

In V1.4, specifying a RFC1006 presentation address in XDSU_CONFIG.DAT for the bind_dsa parameter will not work correctly. An example of an RFC1006 address is:

```
"DSA"/"DSA"/"DSA"/RFC1006+16.36.56.76,RFC1006
```

There are still some presentation addresses that are not correctly handled by the BIND_DSA parameter (e.g. non-Digital DSA's). These addresses should be specified in the dua defaults file instead and leave the BIND_DSA blank in XDSU_CONFIG.DAT.

4.2 Modifying the same records in TRANSACTION mode

In previous versions of XDSU, in TRANSACTION Mode during the Modify Pass, if the same entry occured more than once in the Modify Input File, only the initial change was definitely made while the subsequent changes may have been ignored.

In V1.4, XDSU will now compare an entry in the modify input file with what is current in X.500 instead of what X.500 was like at the start of the XDSU run.

4.3 Problems extracting large amounts of X.500 entries

In previous versions of XDSU, extracting a large number of entries from X.500 had problems.

The first problem was that if XDSU or the DSA process ran out of resources (usually virtual memory) during the search, XDSU would return with 0 records extracted instead of an error. In V1.4, a fatal error is now generated and the appropriate error message is reported.

The second problem was that XDSU would get an EMS_FETCH error or abort during an extract after extacting 40000+ entries. (Note: the minimum number of entries which caused the problem was variable depending on the number of RDN values in the DN's.)

This problem has been fixed in V1.4, but will require a higher virtual memory quota for the XDSU process.

4.4 Extra characters added to ENCODE_DELIM processing

In addition to \{}¹©, several other 8-bit ascii codes must be encoded when storing values using teletexStringSyntax and ENCODE_DELIM is specified. They are (in hex) 0xA0, 0xA6, 0xAC, 0xAD, 0xAE.

Appendix A
XDSU V1.4-1 RDF Syntax

A.1 RDF Syntax

The format of an RDF statement is one of the following:

```
action
? conditional : action
? conditional : action : else_action
? conditional : IGNORE
```

The format of an action (or else-action) clause differs depending on whether the RDF is being used to input or output a foreign directory file.

A.1.1 RDF action format for Input Files

When the RDF file is used for reading a foreign directory input file (IMPORT, TRANSACTION, or CHANGES mode - when using a foreign directory file for comparison), the format of an action (or else_action) clause is:

```
Destination = Source
```

The Destination field must be an X.500 attribute which is listed in XDSU_LAYOUT.DAT (as an RDN or ATTR). The Source field may be one the following:

- constant string - Double quoted "Constant String" (e.g. "Digital")
- input record location - RECORD_NUMBER.STARTING_COLUMN.WIDTH (e.g 1.1.64)
- input record field - RECORD_NUMBER-FIELD_NUMBER[.STARTING_COLUMN.WIDTH] (e.g. 1-1 or 1-1.1.64 or 1-1.10.#)
- input record sub-field - RECORD_NUMBER-FIELD_NUMBER-SUBFIELD_NUMBER[.STARTING_COLUMN.WIDTH] (e.g. 1-1-1 or 1-1-1.1.64 or 1-1-1.1.#)
- keyword - an X.500 attribute in XDSU_LAYOUT.DAT (e.g. surname)

The meaning of the terms used in a input record location or field are:

- **RECORD_NUMBER** is a number between 1 and the NREC_SUBSCR parameter. 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.
- **STARTING_COLUMN** is the column number in the input foreign directory file where the attribute starts. For an input record field, the the column number is the column number relative to the start of the field. Column numbers start at 1.
- **WIDTH** is the length of the attribute to be read. For example, if a WIDTH of 20 is specified, then XDSU begins reading the location specified by the STARTING_COLUMN parameter and writes the next 20 characters to the indicated attribute in X.500. For an input record field, a width of # can be specified to mean the remainder of the field.
- **FIELD_NUMBER** is the field number from 1 to N between field separator strings. The field separator string is the first string specified by the RDF_FIELD_SEPARATOR parameter in XDSU_CONFIG.DAT.
- **SUBFIELD_NUMBER** is the sub-field number from 1 to N between sub-field separator strings within a field. The sub-field separator string is the second string specified by the RDF_FIELD_SEPARATOR parameter in XDSU_CONFIG.DAT.

Some examples of Input File Action Lines are:

```
emailORGNAME = "Sales"  ! constant string
emailSURNAME = 2.1.10   ! 1st 10 characters of line 2
emailGIVENNAME = 1-1    ! 1st field of line 1
emailUSE1 = 1-2.1.10    ! 1st 10 chars of 2nd field of line 1
emailUSE2 = 1-3-2.1.10  ! 1st 10 chars of 2nd sub-field of
                        !   3rd field of line 1
emailUSE3 = 1-3-2.11.#  ! characters 11 until the end of the 2nd
                        !   sub-field of the 3rd field of line 1
surname = emailSURNAME  ! X.500 keyword
```

A.1.2 RDF action format for Output Files

When the RDF file is used for creating a foreign directory output file (EXPORT, or CHANGES mode), the format of an action (or else_action) clause is:

```
Source = Destination
```

The Destination field must be an output record location of the form RECORD_NUMBER.STARTING_COLUMN.WIDTH (e.g. 1.1.30, 1.31.#, 1.*.#). The Source field may be one the following:

- keyword - an X.500 attribute in XDSU_LAYOUT.DAT (e.g. surname)
- constant string - Double quoted "Constant String" (e.g. "Digital")

The meaning of the terms used in a output record location are:

- **RECORD_NUMBER** is a number between 1 and the NREC_SUBSCR parameter. 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 that data is to be written to in the output Foreign Directory File.

- **STARTING_COLUMN** is the starting column number in the output foreign directory file record where data is to be written. 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.’’
- **WIDTH** is the length of the source data to be written to the output file. For example, if a WIDTH of 20 is specified, the first 20 characters from the source data are written to the export file, beginning at the spot indicated by STARTING_COLUMN. The field length can be expressed by the pound sign # character, which uses the length of the data found in the source as the field’s width. For X.500 keywords, trailing spaces are truncated.

A.1.3 RDF conditional format

An RDF conditional starts with a question mark (?) character and terminates with a colon (:) character. The question mark can be thought of being an "IF" and the colon an "THEN". (And the second, optional colon an "ELSE").

The conditional logic allows fields to be compared using EQUAL (==), NOT EQUAL (!=), NOT (!), SUBSTRING OF (<<), and NOT A SUBSTRING OF (!<) operators:

```
? field  == "STRING" :  ! True if field equals STRING
? field1 !=  field2  :  ! True if field1 does not equal field2
? field  << "STRING" :  ! True if field is a substring of STRING
? field  !< "STRING" :  ! True if field is not a substring of STRING
? field  :              ! True if field is non-blank
? !field :              ! True if field is blank
```

Fields used in conditionals can be one of the following:

- constant string - Double quoted "Constant String" (e.g. "Digital")
- record location - RECORD_NUMBER.STARTING_COLUMN.WIDTH (e.g 1.1.64)
- input record field - RECORD_NUMBER-FIELD_NUMBER[.STARTING_COLUMN.WIDTH] (e.g. 1-1 or 1-1.1.64 or 1-1.10.#)
- input record sub-field - RECORD_NUMBER-FIELD_NUMBER-SUBFIELD_NUMBER[.STARTING_COLUMN.WIDTH] (e.g. 1-1-1 or 1-1-1.1.64 or 1-1-1.1.#)
- keyword - an X.500 attribute in XDSU_LAYOUT.DAT (e.g. surname)

All comparisons are case insensitive (e.g. "a" equals "A").

Comparisons can be combined by use of logical AND (&&) and OR (||) operators:

```
? field == "A" || field == "B" :  ! True if field equals A or B
? field != "A" && field != "B" :  ! True if field not equal to A and B
? field << "AB" || field = "E" :  ! True if field is a substring of
                                  !  AB or field equals E
```

Parenthesis can also be used for complex conditionals:

```
? (field == "A" || (field != "B" && field1 != field2) ) :
```

The boolean NOT (!) operator can also be used:

```
? !(field == "A" || field == "B") : ! True if field is not A or B
? !field :                          ! True if field is blank
                                    !  (Same as ~field: )
```

An optional ELSE conditional statement after a second colon (':') may be specified:

```
? 1.1.10 : description = 1.1.10 : description = "Default"
```

instead of having to do:

```
? 1.1.10 : description = 1.1.10
~ 1.1.10 : description = "Default"
```

If a conditional is specifed, the action line can be "IGNORE".

This causes any input file which matched this condition to not be used.

For example, the following will skip any records that contain the substring "admin" in field 2 in the input record.

```
? "admin" << 1-2 : IGNORE
```

A.1.4 Other RDF syntax Rules

A.1.4.1 Comments

RDF comments start with a ! character. Anything after a ! character on a line is considered a comment (unless within a double quoted string).

A.1.4.2 Line Continuation

RDF lines may be continued on the next line by ending a line with the hypen ('-') character:

```
? (1-1 == "TEST") || (1-2 != "ORG1" && 1-2 !="ORG2") : -
    organization = "ABCD" : -
    organization = "EFGH"
```

A.1.4.3 Whitespace

Whitespace characters (space, tab) outside of double quotes are ignored. The above examples usually show tokens separated by a single space character but that is only used for clarity and is not required.