HP Tru64 UNIX Enterprise Directory

Release Notes

Revision/Update Information: Version 5.6

This document was prepared using DECdocument, Version 3.3-1B.

Contents

1 Before You Start

Note that the product now available as the HP Tru64 UNIX Enterprise Directory Version 5.6 was previously available with these names:

HP Tru64 UNIX Enterprise Directory V5.5 ECO1
HP Tru64 UNIX Enterprise Directory V5.5
HP Tru64 UNIX Enterprise Directory V5.4
Tru64 UNIX Enterprise Directory V5.3
Tru64 UNIX Enterprise Directory for eBusiness V5.2 and earlier
Compaq X.500 Directory Service V4.0
Digital X.500 Directory Service V3.1 and earlier

This section documents important information about the Enterprise Directory and about installing and uninstalling the Enterprise Directory software.

1.1 New Features in This Release

This kit is an upgrade of HP Tru64 UNIX Enterprise Directory V5.5 ECO1 and includes the following new features:

- Password Policy

  HP Enterprise Directory Version 5.6 implements the password policy as defined in Draft Behera. The policy provides the ability to ensure the secure read and update access to directory information throughout the network. The policy is applied on a password that is used to authenticate. The policy also includes the following:

  - Whether and when passwords expire
  - Whether failed bind attempts cause the account to be locked and
  - If and how users are able to change their passwords.

  The password policy can be applied to any attribute holding a users password used for an authenticated LDAP bind operation. Here the term user represents any LDAP client application that has an identity in the directory.

  For details on usage and applicability of password policy as defined in Draft Behera see Section 11.

- Password Hashing

  HP Enterprise Directory Version 5.6 implements hashing of the users password. Here the term user represents any LDAP client application that has an identity in the directory. The directory server provides options of crypt and the algorithms supported by OpenSSL_add_all_digests( ) function of OpenSSL library with which the users password can be hashed.

For details on usage and applicability of password hashing see Section 12.

- Sample Cluster Application Availability Script on Tru64 UNIX

  A sample Cluster Application Availability (CAA) script for achieving HP Enterprise Directory failover capability on Tru64 UNIX cluster (TruCluster) environment and steps to register and invoke CAA script have been added to Section 13.

- Enabling Memory Tracing Facility

  Memory tracing facility is added to this version of DSA, which can be turned on as required for faster diagnostics of DSA. For details on enabling memory tracing facility see Section 14.

- Support for Samba Schema 3.0

  Samba 3.0 Schema elements are supported in this version of HP Enterprise Directory. For details on Samba 3.0 schema elements see Section 15.

1.2 Upgrading from Previous Versions of the Tru64 UNIX Enterprise Directory

On Tru64 UNIX, Version 3.0 and later of the Directory, the DSA uses memory image files instead of snapshot files. Memory image files are specific to the version of the kit. This applies to all versions (SSB, ECO kits, and FT kits). Therefore, your current memory image files cannot be used by this kit. To use your existing database with this kit, create a snapshot file of your database before you upgrade.

To create a snapshot file, disable and delete the DSA using the following NCL commands:

ncl> disable dsa

ncl> delete dsa to snapshot

After installing this release, use the following NCL commands to recreate the DSA:

ncl> create dsa from snapshot

ncl> enable dsa

When the DSA has been recreated and enabled successfully, you can delete the snapshot file from the DSA system as follows:

rm /var/dxd/DSA-information-tree.snapshot*

The DSA does not need this file any longer, so you can save disk space by deleting it. Do not delete any of the other database files.

1.3 Installing on Tru64 UNIX DCE Systems

If you are installing the Enterprise Directory on a Distributed Computing Environment (DCE) system, then you must install it after installing DCE, or reinstall the Enterprise Directory after installing DCE.

A DCE installation overwrites some files installed with the Enterprise Directory without checking to see whether the existing files are more recent versions. If this happens, you need to reinstall the Directory Service to make sure that the latest versions of the files are present on the system.

1.4 UNIX Cluster Support

This information applies to UNIX cluster systems only.

You can run more than one DSA on a UNIX cluster provided that `/var/dxd` is a host-specific directory. If the directory `/var/dxd` is common to all hosts, you can run only one DSA on the cluster.

Although you can run more than one DSA on a UNIX cluster, only one DSA can access a database at any one time. When a DSA loads the database, the database is locked to prevent another DSA accessing it.

If the database is being used by another DSA, you can identify the DSA by using the following command:

```
cat /var/dxd/DSA-information-tree.lock
```

If you try to create a DSA when the database is being used by another DSA, the NCL create dsa command fails, for example:

```
ncl> create dsa

Node 0 DSA
AT 1998-07-10-10:25:11.697+01:00I0.749

FAILED IN DIRECTIVE: Create
DUE TO: Error specific to this entity's class

REASON: Cannot open database
Description: The DSA cannot open the database as it is being used by another DSA.
```

When this error occurs, the Create Failure event is also generated, for example:

```
Event: Create Failure
       from: Node DEC:.reo.mrdsa DSA
       at  : 1998-07-10-10:25:11.692+01:00I0.749

       Reason                      = Database already in use by another DSA.
```

Note that DSA Events will only be generated on clusters that have DECnet–Plus installed.

2 Additional Documentation for Directory Functions

This section deals with new functions added to Enterprise Directory since the last full release of the main documents.

2.1 Running without DECnet–Plus

Prior to Tru64 UNIX Enterprise Directory V5.3 release, the Directory Service had a pre-requisite that DECnet–Plus was installed. This restriction has been removed, but for customers who wish to use an OSI Transport, the use of DECnet–Plus is still optionally available.

In order to remove the DECnet–Plus dependency, the DSA and DUAs, now have their own implementation of RFC1006 transport.

Directory User Agent (DUA)

In this section there are references to a DUA, which stands for Directory User Agent. This reference will apply to DXIM and applications linked against the supplied XDS library.

The DSA and DUAs will use their implementation of RFC1006 transport if the DECnet–Plus OSAK library is not available. If the OSAK library is available, then by default the DSA and DUAs will use the OSAK library for transport. This will be the case for the DSA unless the DSA characteristic `PROHIBIT DECNET TRANSPORT` is set to `TRUE`. Similarly this will be the case for a DUA unless the environment variable dxd_no_decnet is defined as `TRUE` before the DUA is activated.

If DECnet–Plus is installed then there is an issue with the use of standard port numbers. This is described in Section 2.1.1.

2.1.1 Resolving a conflict with Port 102

If you wish to configure an RFC1006 presentation address on a system which has DECnet–Plus installed, then it is recommended that you use the DECnet implementation of RFC1006, which will always use port 102.

If you wish to configure a DSA to have an RFC1006 presentation address on a system which does not have DECnet–Plus installed, then you must use the DSA's and DUAs' private RFC1006 implementation, and it is recommended that in this case you also use port 102 (however, see note 4 below).

If you wish to use the DSA's or a DUA's private RFC1006 implementation on a system which has DECnet–Plus installed, the following section explains why there is a conflict, and how to resolve it.

The RFC1006 specification reserves port 102 for applications using the RFC1006 protocol. The DECnet OSAK library provides CLNS, CONS and RFC1006 transports: its RFC1006 implementation allocates TCP/IP port 102, and then automatically and transparently diverts any incoming requests to the appropriate RFC1006 application (such as the DSA).

Because the DSA's private RFC1006 implementation does not use DECnet, it cannot use port 102 if DECnet–Plus is installed and already using it.

So to use the DSA's private RFC1006 implementation on a system that has DECnet–Plus installed, you need to configure the DSA to listen on a port number other than 102 for incoming RFC1006 connections (and also set the DSA characteristic `PROHIBIT DECNET TRANSPORT` to `TRUE`). For example, the RFC1006 address below specifies port 5050:

```
"DSA"/"DSA"/"DSA"/RFC1006+foo.bar.hp.com+5050
```

Note however, that:

1. DECnet–Plus will only ever listen to port 102 for RFC1006 communication. So for a system which has DECnet–Plus installed, a DSA configured with the presentation address shown above and with the DSA characteristic `PROHIBIT DECNET TRANSPORT` set to `FALSE`, the port number (5050) will be ignored by DECnet–Plus in favour of port 102. This restriction applies to both OpenVMS and Tru64 UNIX.
2. On OpenVMS, when DECnet–Plus makes outgoing connections to another system using RFC1006 protocol, it always ignores the specified port number and uses port 102. So if you configure a DSA or DUA with the presentation address shown above on a system which does not have DECnet–Plus installed, then it will use port 5050, but no other OpenVMS DSA will be able to communicate with it. This restriction does not apply to Tru64 UNIX.
3. The DSA's and DUAs' implementation of RFC1006 respect the port number specified in presentation addresses for both incoming and outgoing connections.
4. The DSA's implementation of RFC1006 does not have the same arbitration between multiple RFC1006 applications that DECnet–Plus provides. So if you configure the DSA on a system without DECnet to use port 102, this will prevent other applications which also want to use port 102 from being able to run.
5. If an RFC1006 presentation address, that does not have a port number, is set or displayed then the default port number 102 will be used.

2.2 DSA NCL commands

Management of the DSA can now be performed from the HP Administrator for Enterprise Directory program. In previous versions management had to be performed from the Network Control Language (NCL) command line interface, and System Managers may have written command procedures using NCL commands in order to manage the Directory. So that these command procedures may continue to be used on systems where DECnet–Plus is not installed, and hence NCL is not available, version 5.3 or later of the Enterprise Directory includes an NCL Emulator.

The NCL Emulator accepts commands in exactly the same syntax as NCL, with the restriction that it will only accept NCL commands for the "DSA" entity, that is it can only be used to manage an Enterprise Directory.

To start the NCL Emulator invoke the program at `/usr/sbin/dxd_ncl`.

There are some points to note if the NCL Emulator is used to manage a DSA on another node:

- The NCL Emulator can only be used to manage version V5.3 or later of an Enterprise Directory.
- With the original NCL it was necessary to set up OpenVMS proxies in order to gain the authority to manage a remote DSA running on OpenVMS. These are not required by the NCL Emulator, and instead a new command called AUTHENTICATE is supplied exclusively by the NCL Emulator. The syntax is:

```
AUTHENTICATE NODE <node> DSA [PORT <port>] [PASSWORD <password>]
```

  The AUTHENTICATE directive is used to bind to a remote DSA, in order to issue NCL commands to the remote DSA. An AUTHENTICATE directive must be issued before any remote commands to that DSA are attempted. It remains in force for the duration of the NCL session. The password required is the Management Password required to access the HP Administrator for Enterprise Directory Utility on the remote system. For details on how to set this password see Section 3.3. Note that remote commands require a TCP/IP network to be configured.

Note

On Tru64 UNIX, if you use the key combination <control>/c to leave the NCL emulator, the emulator will not release message queues resources. You can eventually run out of message queues resources doing this.

The message queues resources can be released by re-booting the system or by using the utilities `ipcs` and `ipcrm` with care.

2.3 Schema Extensions

New schema objectclasses and attributes have been defined to support HP OpenVMS LDAP SYS$ACM Authentication Agent. This product was first supplied as an Early Adopters' Kit in OpenVMS V7.3-2, and enables OpenVMS authentication information to be stored in an LDAP Directory.

In addition the Internet standard inetOrgPerson LDAP Object Class has been added to the schema. This is described in RFC 2798 and is a general purpose object class that holds attributes about people.

2.4 Extended Syntax Support

DXIM will now display passwords as octet strings if they contain non-ISO Latin-1 characters. Octet strings can now be used to enter passwords.

Support has been added for an "approximate match" when comparing attributes that have a syntax of bitstring. An approximate match is true if all the bits set in the assertion are also set in the value, and false if any of the bits set in the assertion are not set in the value.

For example, the following would be considered matches:

'1000'b would match '101100001'b
'111'b would match '1111111'b

The following would not be considered matches:

'1000'b would not match '0100110110'b
'1010'b would not match '1101110'b

Note that bit 0 is the left-most bit.

2.5 Selective Shadowing

This release supports selective shadowing; that is, it is now possible to specify which attributes can and cannot be shadowed to a consumer DSA.

The shadowing filter is controlled by the shadowingAttributeSelection attribute in the shadow agreement subentry. Thus every shadowing agreement has its separate filter. If a shadow agreement subentry does not have the shadowingAttributeSelection attribute then all user attributes are shadowed to the consumer DSA.

The shadowing filter is performed on the supplier DSA. Time stamp and security attributes are always shadowed to the consumer DSA. See X.525 for more information.

The shadowingAttributeSelection attribute is multivalued. Each value is a classAttributeSelection and consists of an object class and either an "include" or "exclude" list of attributes or "all attributes".

The object class in the classAttributeSelection can either be a specific object class or all object classes. In the "exclude" form, attributes not in the exclude list are implicitly included.

A classAttributeSelection is applicable to an entry if allObjectClasses is specified or if the specified object class matches an object class of the entry. If more than one classAttributeSelection filter applies to an entry, an "include" filter takes priority over an "exclude" filter and an "exclude" filter takes priority over "all attributes" and implicitly included attributes. If no classAttributeSelection is applicable to an entry then only the time stamp and security attributes are shadowed.

The command-line DXIM syntax for the shadowingAttributeSelection value is:

```
<objectclassselection>  ::= "objectClass" "=" <value>   |
                            "allObjectClasses"

<attributefilter>       ::= "include" "=" "{" <attr>, <attr>... "}" |
                            "allAttributes" |
                            "exclude" "=" "{" <attr>, <attr>... "}"

<classAttributeSelection> ::=
                        "[" <objectclassselection> "," <attributefilter> "]"
```

For example:

```
dxim> modify /c=us/o=acme/cn="supplier /c=us/o=acme 3" -
_dxim> add attribute shadowingAttributeSelection = -
_dxim> [objectclass=person, -
_dxim> exclude={description,title,telephone number}]
```

Note

In Section A.1.35 of the V5.0 Management Guide, the attribute shadowingAttributeSelection (together with this new information) replaces shadowingAttributes.

2.6 LDAP Support Enhancements

2.6.1 LDAP Controls

The DSA supports the ManageDSAIT Control, as defined in RFC 3296, to indicate that an operation is intended to manage knowledge information within the DSA Information Tree.

2.6.2 LDAP Extensions

The DSA supports the Start TLS extension to LDAP for use with SSL (Section 2.6.3), as defined in RFC 2830.

2.6.3 Secure Sockets Layer

The Directory can receive commands over a secure line using LDAPv3 (provided that the client is using the Start_TLS extension). The Directory uses the same LDAP port to receive both secure and unsecured communication.

The Directory supports SSLv23 (the default), SSLv3 and TLSv1 protocols, but only one can be supported at any time. The user can select the protocol by disabling the DSA and setting the protocol using the NCL command:

```
SET DSA LDAP SECURITY PROTOCOL <"SSLv3"/"SSLv23"/"TLSv1">
```

To use this support, you must provide a Certificate file and a Private Key for the DSA. These are stored as (/var/dxd/DSA-certificate.pem) and (/var/dxd/DSA-private-key.pem). The DSA's Pass Phrase must be used as the PEM passphrase. The DSA's Pass Phrase is stored in the DSA Characteristic "PRIVATE KEY PASSPHRASE".

The DSA does not provide a default certificate or private key. For more information on certificates and keys, see http://www.openssl.org.

The Directory can be placed in one of three security states — no security, selectable security or mandatory security. This is chosen by setting the attribute SSL State to OFF, ON, or MANDATORY using the NCL (or NCL Emulator) command "SET DSA SSL State <*state*>", for example SET DSA SSL State ON .

If the SSL State is set to OFF, SSL will not be available. If the SSL State is set to ON, SSL will be available to any LDAP client that can issue an ldap_tls_start request. If the SSL State is set to MANDATORY, LDAP connections will only be accepted from clients that either use anonymous credentials (and therefore have the lowest possible access level), or use SSL to encrypt the connection.

The SSL cipher suites used by the DSA can be specified by setting the characteristic LDAP CIPHERSUITES. If it is not specified then the implementation's default for SSL is used.

2.7 Administrative Options for RFC1006 Listener

There are two new DSA options provided for tuning the RFC1006 Listener in a pure TCP/IP environment:

1. Option Name : rfclistener_threads

   Description : Number of rfclistener threads brought up by the DSA.

   Maximum Value : 10

   Minimum Value : 2

   Default Value : 2

2. Option Name : rfclistener_timeout

   Description : Rfclistener Timeout value while waiting for authentication information to arrive from the client over a new connection.

   Maximum Value : 75 seconds

   Minimum Value : 0 seconds [equivalent to infinite wait]

   Default Value : 0 seconds [equivalent to infinite wait]

When faced with a situation where some clients connecting to the DSA do not send the required authentication information to the DSA, the administrator can configure a suitable value for 'rfclistener_timeout' to timeout such clients. This value should generally take into account the amount of time required for the authentication information to be received from a XDS/LLAPI based DSA client operating over the slowest link at the site where these changes are being made. Based on this timeout value the administrator might want to increase the number of RFC1006 Daemon threads to provide for a better response time to the other valid clients.

The change in the value of 'rfclistener_threads' only takes effect after the DSA has been disabled and enabled subsequent to the change in value of 'rfclistener_threads' DSA option. Thus to change this value first disable the DSA, then change the value, and then enable the DSA to bring up the required number of RFC1006 Listener threads.

The value of the DSA option 'rfclistener_timeout' can be changed at any point in time when the DSA is enabled. The changed value only applies to the new client connection requests that come-in after the value of this option has been changed.

Example commands to set these option values:

```
Ncl> test dsa command "set option rfclistener_threads=4"
Ncl> test dsa command "set option rfclistener_timeout=6"
```

2.8 Events in a DECnetless environment

A new Event logging interface utilizing the syslog facility provided on Tru64 UNIX has been built into the DSA to log events in a pure TCP/IP environment. All the events earlier being logged by the DSA in a DECnet environment are now also logged by the DSA in a pure TCP/IP environment. These events are logged in the /var/adm/syslog.dated/current/user.log file on the system. Detailed documentation about each of the events logged by the DSA can be found in the 'Problem Solving' Guide for Enterprise Directory. The format for the Event Messages in a pure TCP/IP environment is:

```
event-name,
from: Node node-identity DSA
at : date-and-time

message
```

where:

- event-name is the text identifying the event.
- node-identity is the unique node identity of the node that issued the event. This contains the node's namespace, its location code and the node name.
- date and time is the date and time. This is formatted as yyyy-mm-dd-hh-mmss. nnn+nnxn.nnn, where yyyy-mm-dd-hh-mm-ss is the date and time when the event is issued and nnn+nnxn.nnn is internal timing information.
- message is further information provided with the event.

Detailed documentation about each of the events logged by the DSA can be found in Section 10 of the 'Problem Solving' Guide for Enterprise Directory. For quick reference the events logged by the DSA in a pure TCP/IP environment are:

- Accounting Disabled
- Accounting Enabled
- Accounting File Rollover
- Accounting File Access Failure
- Accounting Records Discarded
- Authentication Failure
- Changes of State
- Communication Failure Event
- Create Failure

- Distributed Operation Failure
- Failure To Start Accounting Facility
- Internal Error
- Listen Failure
- Communication Failure Event
- Resource Exhausted
- Shadow Agreement Update Complete
- Shadow Agreement Update Failure
- Shadow Update Complete
- Shadow Update Failure

2.9 Starting the DSA on a Tru64 UNIX System

In order to remove the dependency on DECnet–Plus, it has been necessary to provide new startup scripts for the DSA on a Tru64 UNIX system. This section provides some details about these scripts, which are located in `/var/dxd/scripts` except where stated. They are:

- dxd_servers

  This is a new script to start and stop the Directory servers, and is located in /sbin/init.d. It takes the parameter "start" or "stop". It calls /var/dxd/scripts/dxd_common_startup and /var/dxd/scripts/dxd_common_shutdown. The script is invoked automatically when UNIX starts up or shuts down. This script should not be modified.

- dxd_common_startup

  This common startup script starts the DSA and the HP Administrator for Enterprise Directory Management Utility. The DSA is started if the scripts /var/dxd/scripts/dxd_dsa_startup and /var/dxd/scripts/dxd_dsa_process exist. The Management Utility is started if the script /var/dxd/scripts/caed_utility_startup exists. This script should not be modified.

- dxd_common_shutdown

  This common shutdown script stops the DSA and the HP Administrator for Enterprise Directory Management Utility. The DSA is stopped if the scripts /var/dxd/scripts/dxd_dsa_shutdown and /var/dxd/scripts/dxd_dsa_process exist. The Management Utility is stopped if the script /var/dxd/scripts/caed_utility_shutdown exists. This script should not be modified.

- dxd_dsa_startup

  This startup script checks that the DSA process is not running and removes the file dxd_stop_server if it exists. It starts the DSA process dxd_dsa_process and then executes the NCL script dxd_dsa_startup.ncl using the NCL Emulator. The first parameter to the script should be "dsa" and if a second parameter of "nonclscript" is present then the NCL script is not executed. This script should not be modified.

- dxd_dsa_shutdown

  The first parameter to this script should be "dsa" and there is an optional second parameter of "snapshot" which causes the DSA to delete to snapshot. It checks that the DSA process is running and stops the DSA by creating the file dxd_stop_server in the DSA directory and executing the NCL script dxd_dsa_shutdown.ncl. If the optional parameter "snapshot" is supplied then the shutdown script is replaced by a disable command and a delete DSA to snapshot command. This script should not be modified.

- dxd_dsa_process

  This is the process script for the DSA. It runs the DSA continuously if the file dxd_stop_server does not exist. This script may be modified if required.

- dxd_dsa_startup.ncl

  This file contains the DSA startup NCL Emulator commands for the DSA. This script may be modified if required.

- dxd_dsa_shutdown.ncl

  This file contains the DSA shutdown NCL Emulator commands for the DSA. This script may be modified if required.

2.10 Running in an IPv6 Environment

In a DECnetless environment the DSA can now run in a pure IPv4, pure IPv6 or a dual stack (IPv4,IPv6) environment.

The supported servers in the DSA have been modified to discover the configured protocols (IPv4, IPv6) on the system. In a DECnetless environment, if IPv6 is configured on a particular system, the servers initialize to listen on IPv6, else they fall back to listening on IPv4.

Similarly the supported clients (and client APIs) in the DSA have been modified to connect to the server making use of the user provided or stored address of the server host. If this attempt fails, then all the configured addresses of the server host are locally resolved. Then an attempt is made to establish a connection on each of the resolved address (IPv4, IPv6) successively, till either a successful connection is established or the resolved addresses are exhausted.

In addition to this when the user sets a presentation address in the DSA, the DSA functions as follows:

If either only an IPv4 or IPv6 address can be resolved on the system for the node specified in the presentation address, then the resolved address of the node will be used. In case both an IPv4 and an IPv6 address is resolved for the node, the DSA prefers to use the IPv4 address by default. This is because the address choosen by the DSA is then encoded and stored in the DSA. This encoded address is passed during replication to other peer DSAs. Since the peer DSA involved in replication may not be IPv6 aware (lower versions of DSA), or may not have support for IPv6 environment (DSAs running in a DECnet environment), the DSA by default tries to resolve and use an IPv4 address. This is because if the DSA is setup for replication with peer DSAs that might be running a lower version of DSA than DSA V5.5 or may be running in a RFC1006 DECnet environment, then it is necessary that the DSA use an IPv4 address in the presentation that is passed to such peer DSAs during replication agreement setup for maximum interoperability.

If the user wants to override the above described default behaviour and force the DSA to try and resolve an IPv6 address by default (in a DUAL stack environment where both an IPv4 and IPv6 addresses have been configured), then the user needs to perform the following steps:

For the DSA edit the script /var/dxd/scripts/dxd_dsa_startup and set the environment variable dxd_prefer_ipv6 to TRUE and restart the DSA. Similarly for DUA define the environment variable dxd_prefer_ipv6 as TRUE before the DUA is activated.This should be done only when the user is sure that DSA instance will replicate only with other DSA instances that are V5.5 or above and are running in DECnetless environment.

If the user specifically provides the dotted decimal IPv4 address or the colon separated hexadecimal IPv6 address in the presentation address, then the DSA uses this address and ignores the setting of the dxd_prefer_ipv6 environment variable. For example if the user enters the following command:

```
ncl> set dsa pre add  -
_ncl>     '"DSA"/"DSA"/"DSA"/RFC1006+16.138.73.12+6001,RFC1006'
```

Then the DSA will encode and store the IPv4 address irrespective of the value of the dxd_prefer_ipv6 environment variable.

Similarly if the user enters the following command:

```
ncl> set dsa pre add  -
_ncl> '"DSA"/"DSA"/"DSA"/RFC1006+fe80::a00:2bff:fee5:51d3+6001,R FC1006'
```

Also for the GUI the user can specify an IPv6 address of the DSA node to run the GUI in an IPv6 environment. Note when specifying an IPv6 address in the GUI the user should also specify the Interface-id.

For example, if the IPv6 address of the DSA node is 'fe80::230:6eff:fef3:acd9' then when performing 'Add New Server Details' operation specify 'fe80::230:6eff:fef3:acd9%<interface-id>' in the 'IP Node Name' field of the server details window. This interface-id should be the one over which the ping6 utility when provided this address 'fe80::230:6eff:fef3:acd9%<interface-id>' should be able to ping the DSA node successfully.

3 HP Administrator for Enterprise Directory Management Utility

The HP Administrator for Enterprise Directory GUI has a Utility for controlling certain aspects of the behaviour of a directory server (DSA) over an SSL-secured network connection, that requires TCP/IP. This section deals with the GUI and the Utility.

3.1 Background

For the most part, the HP Administrator for Enterprise Directory GUI communicates directly with the directory server (DSA) that is running in order to manage the DSA. However, there are certain extended management operations (for example, restarting the DSA) which cannot be performed by the DSA itself. In these cases, the GUI communicates with a detached process that runs on the same node as the DSA. This process is the HP Administrator for Enterprise Directory Utility.

3.2 HP Administrator for Enterprise Directory Utility Functions

The GUI communicates with the Utility in order to perform the following operations:

- displaying a list of the available schema files
- creating a new schema file
- editing an existing schema file
- recompiling the schema
- restarting the DSA
- editing the DSA Characteristics

The system manager can decide whether or not the HP Administrator for Enterprise Directory Utility should be enabled on any given server. The GUI only requires that the Utility be available in order to carry out the functions listed above; other administrative operations (such as viewing and updating the contents of the DSA) do not require the Utility to be running.

3.3 Security

As part of the process to configure the HP Administrator for Enterprise Directory Utility, the system manager specifies a management password: this password must be supplied to the Utility by any client attempting to connect to it. This means that users of the GUI do not automatically have the ability to perform extended management operations; they must first supply the Utility's password.

The Utility's password is distinct from any DSA-based password, and should only be disclosed to trusted users.

3.4 Utility Configuration (Tru64 UNIX)

If you install the Utility, the directory /var/dxd/caed_native will be created. This directory will contain files used by the Utility.

The Utility relies on a configuration file called administrator.ini which resides in /var/dxd/caed_native. This file contains an encoded version of the Utility password, as well as information relating to the IP port numbers that it should use.

By default, the GUI expects to use port 389 for LDAP connections to the DSA, and 907 for SSL connections between the GUI and Utility process. You can override these values by editing the configuration file.

To find out what port number is being used for LDAP access to the Enterprise Directory, you can use the NCL utility, for example:

```
ncl> show dsa ldap port

Node 0 DSA
AT 2002-01-01-10:05:02.110+00:00Iinf

Characteristics

    LDAP Port                          = 389
```

If port 907 is already in use by an application on your system, then you will need to choose another port number in the range 1..1023.

3.5 Creating the Configuration File (Tru64 UNIX)

Before starting the Utility for the first time, you need to create this file, which can be done using the command: # /var/dxd/scripts/caed_utility_configure

This script will prompt you to provide the password for the Utility, and then create administrator.ini for you. Initially, the port numbers that will be used are 389 for LDAP connections to the DSA, and 907 for connections between the GUI and the Utility.

Once the command procedure has created the configuration file, it gives you the option of starting the Utility. If you want to change the port numbers that the utility uses, then you can reply NO to this question. In this case, you can edit the configuration file before starting the utility.

3.6 Editing the Configuration File (Tru64 UNIX)

The version of `administrator.ini` that is provided the first time you run the configuration command procedure looks like this:

```
[<nodename>]
Ae Title=<nodename>
DSA Entity Name=DSA
Port Number=389
[General]
ServerPort=907
---Do not edit any data below this line---
HashedPassword=<some-value>
```

Where *nodename* will be the name of the server you are using, and *some-value* is a string of digits representing an encoded version of the password that you supplied. Note that the `Ae Title` field will not necessarily reflect the Ae Title of your DSA. To change either of the port numbers, edit the configuration file and modify the appropriate line(s).

3.7 Changing the Utility Password (Tru64 UNIX)

The Utility's password is stored in the configuration file in an encoded form using a hashing algorithm internal to the daemon. The only way to change this password is to request that the daemon generate a new one, which is what happens when you run the configuration command procedure.

You can run the `caed_utility_configure` script at any time to change the password. When a new password is generated, the other contents of the `administrator.ini` file are left unchanged.

3.8 Starting the Utility (Tru64 UNIX)

After running `caed_utility_configure`, you have the option of starting the Utility. You can also start the utility using the command `#/var/dxd/scripts/caed_utility_startup`.

Note that this procedure will fail if you have not created a valid `administrator.ini`, or if the Utility is already running.

The system automatically invokes the Utility startup script as part of the Enterprise Directory startup if you have installed the Utility on your system. If you have installed the Utility but do not want it to be run, then ensure that no copy of the file `/var/dxd/caed_native/administrator.ini` exists; if this file is not found, then the Utility will not be started.

3.9 Shutting down the Utility (Tru64 UNIX)

If you wish to shut down the Utility process, a script is supplied in `/var/dxd/scripts`, which can be invoked using the command `# /var/dxd/scripts/caed_utility_shutdown`.

3.10 Diagnosing problems with the Utility (Tru64 UNIX)

As it runs, the Utility process writes a log file at `/var/dxd/caed_native/caed_utility_output.log`. This log file contains information about requests from any client that attempts to make a connection, as well as messages that may help diagnose problems that occur if the utility fails to start properly.

4 Lightweight Directory Access Protocol

DSA supports LDAP V2 and V3 protocols allowing LDAP clients to access the Directory.

4.1 LDAP String Syntaxes Not Implemented

Enterprise Directory implements both the LDAP protocols (V2 and V3). The following syntaxes are not supported as string syntaxes but you can use all of them as binary syntaxes:

- Teletex Terminal Identifier
- Presentation Address
- Guide (search guide)
- User Certificate
- CA Certificate
- Certificate Revocation List
- Cross Certificate Pair
- Other Mailbox
- Distribution List Submit Permission

4.2 Restrictions on LDAP V3 UTF-8 LDAP Strings

The LDAP V3 implementation only supports UTF-8 characters that can be mapped to T.61 characters.

5 Restrictions in DECnet–Plus

Some restrictions in DECnet–Plus can affect the behaviour of the Enterprise Directory. It is therefore helpful to be aware of DECnet restrictions, as listed in the DECnet–Plus release notes for your system. Particular areas of interest are NCL restrictions and restrictions related to OSI Transport.

The following sections list restrictions in DECnet–Plus that are known to affect the Enterprise Directory. See Section 6 for information about NCL restrictions.

5.1 DSA on Tru64 UNIX Systems Handling of Multiple Bind Requests

There is a known problem with the OSI transport software that the HP DSA uses on Tru64 UNIX systems. The problem occurs when an HP DSA receives multiple bind requests in quick succession from other vendors' DSAs or directory applications, or from previous versions of the products.

An HP DSA can support multiple concurrent bindings, but when many new bind requests are received in quick succession from other vendors' products or from previous versions of the products, the DSA has a problem queuing the requests. The DSA therefore rejects some of the requests, and OSI transport congestion events are produced. For this reason, you might find that bind requests are successful intermittently, although the majority of requests will be successful. If a bind request fails, try again.

If you have problems connecting to DSAs on Tru64 UNIX systems, but the failures do not cause OSI transport congestion events, then you have encountered an unknown problem. If *Enterprise Directory — Problem Solving* does not help you solve the problem, you should report it to HP.

6 NCL Restrictions

This section describes restrictions on the use of the NCL director to manage the DSA.

6.1 Creating and Deleting the DSA Quickly

If you use the NCL CREATE DSA and DELETE DSA directives repeatedly in quick succession you might see the following error message in response to one of the DELETE DSA directives:

```
No Such Entity Instance exists
```

This occasionally happens even if the preceding CREATE DSA directive appeared to succeed. In fact, the preceding CREATE DSA directive failed, but NCL lost the error response.

Note that this restriction does not apply to the NCL Emulator.

6.2 Restriction on Using Zero Length Passwords

The schema provided with this version of the Directory Service states that the `userPassword` attribute can have a minimum length of zero characters.

However, the NCL commands for the DSA entity and the Accessor entity, which both have Password attributes, do not support zero length passwords. Do not assign a zero length password to a directory entry if you also need to configure the same password in either a DSA or Accessor entity.

6.3 Limited NCL Command Buffer Size

The NCL command input buffer size is 2048 bytes. This limits the number of characters that can be entered in one NCL directive. When you interactively enter a long directive, the NCL command input buffer can overflow, causing the directive to fail.

Note that the NCL Emulator program can accept input commands up to a maximum of 4096 bytes.

6.4 NCL Command Filtering

If you use a filter in an NCL command, the command fails for most of the attributes of the DSA entity. NCL does not fully support filtering for the types of attribute used by the Directory Service.

7 DSA Information and Known Problems

7.1 Restrictions on Removing Shadowing Agreements

You should remove shadowing agreements in the reverse of the order you used to set them up. That is, the agreement at the end of the shadowing chain should be removed first. If it is not, the DSA will continue to hold shadow naming contexts that are not updated by a shadowing agreement.

For example, if DSA A shadows a naming context to DSA B, which then further shadows the naming context to DSA C, you should remove the last agreement first: remove the consumer access point on DSA B before removing the consumer access point on DSA A.

7.2 Clarification on Updating Shadowing Agreements

The DSA does not update the supplier's agreement shadowingflags relating to the replication strategy when changes are made on the consumer's shadowingFlags. You must change these settings on the supplier.

7.3 Year 2000 Conformance

In Version 3.1, the UTC time matching rules used by the DSA were changed so that dates beyond the year 2000 are recognized by the DSA. Time syntaxes, such as UTC Time, represent the year as a two digit number For example, 10.30am GMT on 5th November 1999 is expressed in UTC Time as 991105103000Z.

With the new time matching rules, a time value with a year that is less than 50 is assumed by the DSA to be after 2000. For example, a time value containing the year 01 is assumed by the DSA to be 2001.

A time value with the year equal to or greater than 50 is assumed by the DSA to be between 1950 and 1999. For example, a time value with the year 99 is assumed by the DSA to be 1999.

7.4 Dereferencing of Search Aliases

There is a known problem with the dereferencing of search aliases.

When a DSA is processing a search, it checks beneath the specified search base for subordinate references that it needs to follow. It is during these checks that the DSA can make an error. The following set of circumstances can cause a DSA to create an incomplete list of subordinate references:

i There are alias entries within the search subtree

ii Those alias entries refer to entries that are held on the same DSA

iii There are subordinate references beneath the entries referred to by the aliases

Those subordinate references are not followed during the search. The search of subtrees beneath aliased entries can therefore be incomplete.

7.5 Memory Exhaustion During Replication Can Cause the DSA to Terminate

If replication fails because of memory exhaustion, the consumer DSA will exit if it is unable to roll back the changes. The DSA exits to avoid corrupting its database. This generates a Resource Exhausted event, with a Reason of Fatal Memory Exhaustion.

Another possible cause for the DSA to run out of memory is as a result of processing many thousands of entry modifications in rapid succession. If the DSA permits volatile modifications, then the last few modifications may be lost. If the DSA does not permit volatile modifications, the last modification may be lost. However, in either case, the vast majority of the modifications are safe.

Restarting the DSA increases the virtual memory available to the consumer DSA.

If you need to make a lot of changes to a naming context that is replicated to other DSAs, and you cannot provide more virtual memory, you can use the scheduling attributes of the shadowing agreement to make replication more frequent. For example, you might cause an unscheduled update to occur after a specific number of modifications so that the consumer DSA's memory resources are not overloaded. See *Enterprise Directory — Management* for details of shadowing agreement management.

7.6 DSA Handling of Temporary Files

During normal operation the DSA uses temporary files. When the DSA exits, these temporary files are normally deleted. If the DSA exits abnormally, some temporary files may remain.

If the DSA is not running, you can delete any temporary files that remain. Never delete temporary files while the DSA is still running.

On Tru64 UNIX systems, all temporary files are created in /var/dxd/tmp.

7.7 Length Constraints on T.61 Strings

When the DSA checks the lengths of T.61 strings, it counts the number of octets in the string rather than the number of characters that those octets represent. Some T.61 characters require more than one octet to represent. This may confuse users who expect the DSA to accept a string of a certain number of characters, but find that the string is rejected.

7.8 Restriction on the Definition of Labels in the Schema

Enterprise Directory — Management states that you can specify schema definitions within your schema text files in any order. However, there is a known exception to this rule. A label can only be specified after the definition for which it provides a label.

7.9 DSA Handling of Referral Loops

DSAs cannot detect referral loops. For example, a DSA can chain to a second DSA and receive a referral to a third DSA. It can then chain to the third DSA and receive a referral back to the second DSA. This type of looping is not covered by the standard loop detection model.

To prevent a DSA from following such referrals indefinitely, the DSA stops after the tenth referral for a given user request, and instead displays the referral to the user.

Looping should not occur if your DSAs are configured correctly.

7.10 Distributed Access Failure Event

If you see the Distributed Access Failure event with the following additional information, then it is possible that the failure was caused by the remote DSA being unable to verify this DSA's password:

```
Reason                 = Communications Failure
Communications Problem = ACSE User Reject. No Reason Specified
```

The remote DSA should have generated an event that indicates a failure to verify this DSA's password. Distributed operations between DSAs will fail if the DSAs are unable to verify each other's passwords. Refer to *Enterprise Directory — Management* for details of replicating information about your DSAs so that they have copies of each other's passwords.

7.11 Database Snapshot Failures

The following three events can mean that the DSA has failed to create a new snapshot file. However, the events do not state explicitly that a snapshot failure has occurred. If you see any of these events without a statement of what operation failed, then you can assume that it was an attempt to create a new snapshot file.

```
Resource Exhausted
Insufficient Disk Space.
```

```
Resource Exhausted
Insufficient Memory.
```

```
Internal Error
Unexpected exception in DbMonitor.
```

7.12 DSA Handling of Presentation Address Protocol Identifiers

If a presentation address specifies that a single network service access point (NSAP) can be used for more than one network protocol, the presentation address is not encoded properly by the DSA when it writes the information to disk. The next time you create the DSA, the presentation address is incorrect.

For example, the following presentation address is specified as having two identical NSAPs, but different protocol identifiers:

```
"DSA"/"DSA"/"DSA"/NS+49002AAA004021,CLNS|NS+49002AAA004021,CONS
```

If you delete the DSA, and then recreate the DSA, and then display the presentation address, it appears as follows:

```
"DSA"/"DSA"/"DSA"/NS+49002AAA004021,CLNS|NS+49002AAA004021,CLNS
```

The CONS protocol identifier has been incorrectly encoded, and is displayed as CLNS.

The same encoding error applies to any presentation address attributes within the database, including those in characteristic attributes of the DSA entity and its subentities, and in any directory entries that contain presentation addresses.

7.13 Displaying Entries that are Part of Your Global Prefix

When you create your DIT, you will probably have a global prefix that connects your organization's DIT to the root of the global DIT. The entries named in the global prefix are not part of your organization's DIT, and may not exist at all, except as placeholders for a future connection to a global DIT.

If you try to display entries that are part of your global prefix, the Directory Service returns an error.

For example, in the example used in *Enterprise Directory — Management*, the Abacus organization has a global prefix that includes the name `/c=us`. The entry called `/c=us` does not exist, so if you try to display it, you get an error.

This problem also affects browsing the DIT using the DXIM Browse window; entries that are part of the global prefix cannot be displayed. *Enterprise Directory — Management* gives details of the global prefix. The immediate subordinates of the browse base must be real entries, not part of the global prefix.

8 DXIM Problems

The information in the following sections applies to both the command line interface and the Motif interface to DXIM.

8.1 Handling of Attributes that Have No Matching Rules

If you modify the schema to define an attribute that has no equality matching rule, directory applications, including DXIM, will have difficulty managing that attribute.

This is because the DSA cannot tell whether the value you specify is already present in an entry. Therefore, if you try to add a value to the attribute, the DSA cannot detect value duplications, and if you try to remove a value, the DSA cannot determine whether the value actually exists. This is particularly true of the DXIM Motif interface, which attempts to add and remove values whenever you edit a value input box on the Modify window.

Refer to *Enterprise Directory — Management* for advice about selecting matching rules for attributes.

8.2 Syntaxes Supported by the Directory Service

HP DSAs support the syntaxes and matching rules documented in *Enterprise Directory — Management* and the DXIM online help.

This section details known restrictions with some of the documented syntaxes and matching rules.

- The following syntaxes are supported by the DSA, but are not supported by the DXIM Motif interface:
  - ACIitem
  - Integer

    Although this syntax is supported, the interface cannot handle the specification of values greater than $2 * *31 - 1$
- The following syntaxes are supported by the DSA, but are not fully supported by either DXIM interface:
  - Teletex Terminal Identifier

    Teletex Nonbasic parameters are not supported
  - Facsimile Telephone Number

    G3 Facsimile Nonbasic parameters are not supported

Neither DXIM nor the DSA supports Search Guide syntax.

8.3 DXIM Command Line Interface Problems

The following section describes problems and restrictions in the command line interface version of DXIM. If you create any DXIM scripts, you are recommended to keep them, to help diagnose any problems.

8.3.1 Ambiguous Keyword

The keyword BINDING has two meanings in DXIM: it is used in the SHOW BINDING command and to indicate the Binding service control. This ambiguity means that you may see misleading error messages if you specify an incomplete or incorrect command that includes the keyword BINDING. See the DXIM online help information about the SHOW BINDING command and the Binding service control.

8.4 DXIM Motif Interface Restrictions

8.4.1 Traversing Windows Using the Keyboard

There is a known Motif problem that affects DXIM windows on Tru64 UNIX systems. The problems does not affect DXIM on other operating systems.

You should be able to move the input focus around a DXIM window using either the mouse pointer or the keyboard. In many cases, using the keyboard to move input focus does not work.

8.4.2 Problem Deleting Multiple Entries

If you select a large number of entries, and then select the Delete Entry option, DXIM displays a confirmation window, asking you whether you really want to delete the selected entries.

If you select too many entries, the buttons of the window are not visible on your screen. Press RETURN to cancel the operation. Select fewer entries for deletion, so that the window is small enough to fit on the screen.

8.4.3 Handling of User Passwords

The DXIM Motif interface does not support creation or modification of password attributes, even if they are added to the window definition.

8.4.4 Handling of Search Filters

A DXIM Motif interface user can use the Find window to specify details of entries to search for. If the user specifies a detail that is inappropriate for a given filter field, DXIM ignores that detail. For example, if a user specifies an alphabetic string for a field that requires integer values, DXIM ignores that particular field. Only fields that contain appropriate details are processed.

In most cases, this is the user-friendly way to process user input. However, the effect might be that DXIM returns more entries than the user expects to see, including entries that the user thought would be excluded.

8.4.5 Support of the undefinedSyntax

Attribute values that are of `undefinedSyntax` might be displayed in verbatim format, for example:

```
'14 01 c4'v
```

DXIM uses this format if it cannot convert the value into a user-friendly external representation.

DXIM only accepts the verbatim format on input of such values. You need to know how to represent the value in verbatim format. Where possible, do not use the `undefinedSyntax` for attributes.

8.4.6 Online Help

The online help for the DXIM Motif interface does not support the search for keywords option of the Help widget. If you select Search Keywords... from the Help window, DXIM displays an internal error.

8.4.7 Display of the Base Object

The DXIM Browse and Find windows allow you to show the attributes of an entry by selecting the entry and double clicking on it, or by selecting the Show Attributes option from the View menu.

To see the attributes of an entry you have modified, you must collapse and expand the parent entry of the modified entry.

However, this does not work with the base object of the Browse or Find window. The base object is the entry that is displayed at the top of the window, and appears when you invoke the window. Whenever you use the Show Attributes option on that entry, you see the attributes and values that were in the entry when you invoked the window. To see the attributes of this object you must invoke a new Browse or Find window.

8.4.8 Deleting Entries

When you use the DXIM Motif interface to delete an entry, you have to click on Yes to execute the deletion. If you double click on Yes, the deletion succeeds, but DXIM reports an internal error. You must then exit DXIM.

8.4.9 Cut and Paste Using the Edit Menu

There are known problems with the Motif cut and paste functionality on Tru64 UNIX systems. If you use the Copy option of the DXIM Edit menu, and attempt to copy information from DXIM to another application, the attempt can sometimes fail. Furthermore, subsequent attempts to use cut and paste in any application can fail because Motif is maintaining a lock on behalf of DXIM.

If you find that cut and paste is locked for DXIM, exit DXIM.

8.4.10 No Support for Auxiliary Object Classes

The DXIM Motif interface does not support auxiliary object classes.

8.4.11 Incorrect Integer Values Not Detected

If you specify a hyphen in an integer value when adding a value to an attribute or creating an attribute, DXIM does not return an error. For example if you try to add the value 1-2-3 to an attribute with integer syntax, no error is displayed. When you subsequently display that attribute, the value displayed is 1.

8.4.12 Usability Problems

The DXIM Motif interface is known to have some usability problems.

- The positioning of new input boxes is faulty.

  If you use the Add Value option, a new input box might be placed partially or completely off the window. To workaround this problem, only add one or two values to an attribute at a time. Click on OK. If you want to add more values, select the Modify Entry option again, and you will be able to display one or two more new input boxes. By this method you can gradually add more values.

- Any modifications made to the Directory using the Create or Modify windows are not automatically reflected in the Browse and Find windows. For example, if you change the name of an entry, the Browse window still displays the old name. Similarly, if you create an entry, DXIM does not display the entry in the Browse window.

  If the information in the Browse and Find windows is out of date, use the collapse and expand options to refresh the window. The DXIM Motif interface on a Tru64 UNIX system does not support some of the standard display parameters that you can specify when you invoke the utility. DXIM does support the `-display` parameter and the `-name` parameter. Other standard parameters are ignored, for example, `-background`, `-geometry`, and `-iconic`.

8.5 Application Resources

The DXIM Motif interface does not currently provide an application resource file to define, for example, the colours to use in DXIM windows. You can edit your default resource file to include information that controls the DXIM display. On a Tru64 UNIX system edit $HOME/.Xdefaults and use the resource name `dxd_mainwin`.

For example, to set the foreground colour to red, edit the file $HOME/.Xdefaults and add the following line:

```
dxd_mainwin*Foreground: #ffff00000000
```

9 Problems with the Lookup Client

9.1 Tru64 UNIX V4.0F and the Look-Up Client

The first time you use dxdlu on Tru64 UNIX V4.0F you may get the /sbin/loader error

```
Unresolved symbol in /usr/bin/dxd_lookup: __cxx_call_static_dtors
```

In this case, replace your system's C++ runtime redistribution kit with the one available from:

ftp://ftp.compaq.com/pub/products/C-CXX/tru64/cxx/

This will prevent the error.

9.2 Lookup Client GUI Help Requires Bookreader

The graphical interface of the Lookup Client requires the Bookreader software. If Bookreader is not installed on the Lookup Client system, then attempts to read the online help will have no effect.

9.3 Lookup Client GUI Handling of Naming Attributes

The Lookup Client for Tru64 UNIX systems does not support the management of naming attributes. If you attempt to modify the attribute that is used for naming an entry, for example, to add a value, the Lookup Client displays an error. The error states that the operation is not allowed on an RDN.

The Lookup Client command line interface permits the management of naming attributes, although it does not allow you to modify the value that is actually used in the entry's name. For example, if an entry's distinguished value is John Smith, you could add a value J Smith, but you could not make that new value the distinguished value.

9.4 Lookup Client Display of Modified Entries

Attribute values displayed in Lookup Client displays cannot be guaranteed to have come from master DSAs. This is particularly important when you use the Alter window of the Motif interface. The values displayed in the Alter window may have come from a shadow DSA. After changing an entry, the changes you make might not be visible in the Lookup Client.

This is because the Lookup Client uses a protocol that does not enable applications to specify that master information is required. The only workaround is to make sure that the Lookup Client is connected to the master DSA for the entry you want to change.

10 XDS Problems and Information

10.1 Documentation of Maximum Outstanding Operations Constant

The Directory Service programming documentation states that the value of the DS_MAX_OUTSTANDING_OPERATIONS constant is defined by the DSA implementation. In fact, the value is defined by the XDS API implementation. HP's XDS API defines the constant to be 32. If 32 asynchronous operations are outstanding, XDS will not accept more asynchronous operations.

If you are writing an application that will use another vendor's XDS API implementation, then the constant may have a different value. It may therefore be advisable not to define this constant in your application, because it cannot be set to both values. Instead, your application can detect the Library Error: Too Many Operations if and when the XDS API returns it. If this error is returned, you can use the Receive Result function to reduce the number of outstanding operations.

10.2 Documentation of the Search and Read Functions

Note that the programming documentation is unchanged for this release of Enterprise Directory. *DEC X.500 Directory Service Programming Reference* documents the functions of the XDS API. The discussions of the Search and Read functions, ds_search and ds_read, does not explain how to select the particular attributes that you want returned from the entry or entries to which the function applies. They only explain that you can use the Select-No-Attributes, Select-All-Types, and Select-All-Types-And-Values constants.

If you want to select specific attributes to be returned from these functions, use the Entry Information Selection class. This class is documented in Section 3.12 of *DEC X.500 Directory Service Programming Reference*.

10.3 XDS Example Programs

The API component of the Enterprise Directory includes a set of callable routines that illustrates how to use the XDS and OM routines. You can also use these routines, which are known as the XDSHLI routines, in your application. The set of routines also includes a simple search application, written using the XDSHLI routines. The file /usr/examples/dxd/xdshli.h contains details of the XDSHLI routines and related files.

10.4 Primitive Syntax Attributes in Uninitialized Objects

The sequence OM_CREATE with the initialize argument set to FALSE, followed by OM_PUT using INSERT_AT_END, will return the error WRONG_VALUE_NUMBER, for any single valued primitive attributes. To avoid this, use REPLACE_ALL in OM_PUT.

10.5 XDS is not Thread-safe in a Multithreading Environment

If you are using multithreading, you must lock calls to XDS. This version of XDS does not handle threads correctly in all cases.

10.6 No Support for Boolean Attribute Values

XDS does not support attribute values of Boolean syntax. Such values always appear to be FALSE.

10.7 Incorrect Initial Value for Information Type

The initial value for the Information Type attribute of the Entry Information Selection object should be types-and-values, as stated in the documentation and defined in the standard. However, XDS sets the initial value to types-only.

10.8 Primitive Values Incorrect in a Referral

A restriction in this version of XDS and XOM means that primitive values in Referrals always appear to be present. A value that should be absent has a value of 0.

10.9 Reading the On-line Reference Pages

Your first attempt to read the X.500 reference pages might fail. Repeat the command.

11 Password Policy Feature: Functionality and Usage

DSA Version 5.6 incorporates a new feature that implements the password policy mechanism as defined in Draft Behera. The password policy described in this section will be applied to any attribute holding a users password used for an authenticated LDAP bind operation. The policy assumes that the password attribute holds a single value.

This section explains in general terms each aspect of the password policy as well as the need for each. These policies are subdivided into the general groups of password usage and password modification.

11.1 Password Usage Policy

This section describes policy enforced when a password is used to authenticate. The general focus of this policy is to minimize the threat of intruders once a password is in use.

11.1.1 Password Guessing Limit

In order to prevent intruders from guessing a user's password, a mechanism exists to track the number of consecutive failed authentication attempts, and take action when a limit is reached. This policy consists of five parts:

- A configurable limit on failed authentication attempts.
- A counter to track the number of failed authentication attempts.
- A timeframe in which the limit of consecutive failed authentication attempts must happen before action is taken.
- The action to be taken when the limit is reached. The action will either be nothing, or the account will be locked.
- The number of times the account is locked (if it is to be locked). This can be indefinite.

11.1.2 Password Expiration, Expiration Warning, and Grace Authentications

One of the key properties of a password is the fact that it is not well known. If a password is frequently changed, the chances of that user's account being broken into are minimized. Password policy administrators may deploy a password policy that causes passwords to expire after a specified period of time - thus forcing users to change their passwords periodically.

As a side effect, there needs to be a way in which users are made aware of this need to change their password before actually being locked out of their accounts. One or both of the following methods handle this:

- A warning may be returned to the users sometime before their password is due to expire. If they fail to heed this warning before the expiration time, their accounts will be locked.
- The users may bind to the directory a preset number of times after their password has expired. If they fail to change their password during one of their 'grace' authentications, their accounts will be locked.

11.2 Password Modification Policy

This section describes policy enforced while users are modifying passwords. The general focus of this policy is to ensure that when users add or change their passwords, the security and effectiveness of their passwords is maximized. In this document, the term "modify password operation" refers to any operation that is used to add or modify a password attribute. Often this is done by updating the password attribute during an add or modify operation, but MAY be done by other means such as an extended operation.

11.2.1 Password History

When the Password Expiration policy is used, an additional mechanism may be employed to prevent users from simply re-using a previous password (as this would effectively circumvent the expiration policy).

In order to do this; a history of used passwords is kept. The password policy administrator sets the number of passwords to be stored at any given time. Passwords are stored in this history whenever the password is changed. Users are not allowed to specify any passwords that are in the history list or same as the current password while changing passwords.

11.2.2 Password Minimum Age

Users may circumvent the Password History mechanism by quickly performing a series of password changes. If they change their password enough times, their 'favorite' password will be pushed out of the history list.

This process may be made less attractive to users by employing a minimum age for passwords. If users are forced to wait 24 hours between password changes, they may be less likely to cycle through a history of 10 passwords.

11.2.3 Password Quality and Minimum Length

In order to prevent users from creating or updating passwords that are easy to guess, a password quality policy may be employed. This policy consists of two general mechanisms - ensuring that passwords conform to a defined quality criterion and ensuring that they are of a minimum length.

11.2.4 Password Change after Reset

This policy forces the users to update their password after it has been set for the first time, or has been reset by a password administrator.

This is needed in scenarios where a password administrator has set or reset the password to a well-known value.

11.3 Applying Password Policy (User Perspective)

This section explains the usage of the policy with examples from user perspective.

11.3.1 Creation of pwdPolicy Subentry

Password policy is created as a subentry under a naming context containing pwdPolicy objectclass attribute which contains the following mandatory and optional attributes:

```
Mandatory attribute: pwdAttribute
Optional Attributes: pwdMinAge, pwdMaxAge, pwdInHistory, pwdCheckQuality,
                     pwdMinLength, pwdExpireWarning, pwdGraceAuthNLimit,
                     pwdLockout, pwdLockoutDuration, pwdMaxFailure,
                     pwdFailureCountInterval, pwdMustChange,
                     pwdAllowUserChange, pwdSafeModify
```

The value for the mandatory attribute "pwdAttribute" determines the attribute on which the password policy will be applicable.

Creating a password policy subentry is neither mandatory, nor is it required to specify a password policy immediately. The unit of password policy is the naming context. If you specify a particular password policy for a password attribute (the value specified in pwdAttribute attribute of pwdPolciy objectclass), the password policy applies to all entries in the naming context containing that attribute. Different password policies can be applied to different password attributes within the same naming context.

Note

The only password attributes to which the password policy will apply are the ones that are used by the DSA to authenticate a authenticated bind operation. Currently the DSA uses "userPassword" attribute for authenticating a authenticated bind operation.

There is no default password policy configuration that the DSAs use until the user decides to specify an explicit password policy.

A new DSA NCL characteristic has been defined to configure the password policy and is named as "Password Policy State" which can be set to ON or OFF.

Example 1,

-

  ```
  NCL> set dsa password policy state =on
  ```

  The above command enables the password policy applicability in the DSA.

-

```
NCL> set dsa password policy state = off
```

  The above command disables the password policy applicability in the DSA.

The password policy object is created as a subentry under a naming context.

After the creation of the naming context, say for example, "/C=in/O=hp" and enabling the DSA the subentry object can be created under this naming context using the dxim tool provided along with the DSA.

Example 2,

```
dxim>create /c=in/o=hp/cn=TestPwdPolicy attributes
_dxim>objectclass=(pwdPolicy,subentry), pwdAttribute={2 5 4 35}, -
_dxim>pwdMinAge=500, pwdMaxFailure=5, pwdFailureCountInterval=60, -
_dxim>pwdMustChange=TRUE
```

Once the subentry is created under a naming context all the entries under that naming context, containing the password attribute (value of pwdAttribute in the password policy subentry) are subjected to the password policy decisions provided the NCL attribute Password Policy State is set to ON.

Note

As seen in the above example (example 2), the mandatory attribute of pwdPolicy objectclass "pwdAttribute" accepts OID syntax values. {2 5 4 35} in example(2) corresponds to "userPassword" attribute.

Similarly if the password policy needs to be applied for a different password attribute, other than userPassword the same can also be specified with an OID syntax. Note that these attributes should be used in authentication process during authenticated bind operation. Below example shows the usage of multiple password attributes as values for pwdAttribute.

Example 3,

```
dxim>create /c=in/o=hp/cn=TestPwdPolicy attributes
_dxim>objectclass=(pwdPolicy,subentry), pwdAttribute={{<OID1>, {<OID2>}}-
_dxim>pwdMinAge=500, pwdMaxFailure=5, pwdFailureCountInterval=60, -
_dxim>pwdMustChange=TRUE
```

Note

The DSA enforces that the password attribute subject to a password policy contains one and only one password value.

Password Policy Subentry Attributes

A password policy object class is defined which contains a set of administrative password policy attributes, and a set of operational attributes are defined that hold general password policy state information for each user.

pwdPolicy Object Class

This object class contains the attributes defining a password policy in effect for a set of users under a naming context. This object class defines both mandatory and optional attributes as shown in section 11.3.1.

Attribute Types used in the pwdPolicy ObjectClass

pwdAttribute

This holds the name of the attribute to which the password policy is applied. For example, the password policy may be applied to the userPassword attribute.

pwdMinAge

This attribute holds the number of seconds that must elapse between modifications to the password. If this attribute is not present, 0 seconds is assumed.

pwdMaxAge

This attribute holds the number of seconds after which a modified password will expire. If this attribute is not present, or if the value is 0 the password does not expire. If not 0, the value must be greater than or equal to the value of the pwdMinAge.

pwdInHistory

This attribute specifies the maximum number of used passwords stored in the pwdHistory attribute. If this attribute is not present, or if the value is 0, used passwords are not stored in the pwdHistory attribute and thus may be reused.

pwdCheckQuality

This attribute indicates how the password quality will be verified while being modified or added. If this attribute is not present, or if the value is '0', quality checking will not be enforced. A value of '1' indicates that the server will check the quality, and if the server is unable to check it (due to a hashed password or other reasons) it will be accepted. A value of '2' indicates that the server will check the quality, and if the server is unable to verify it, it will return an error refusing the password.

pwdMinLength

When quality checking is enabled, this attribute holds the minimum number of characters that must be used in a password. If this attribute is not present, no minimum password length will be enforced. If the server is unable to check the length (due to a hashed password or otherwise), the server will, depending on the value of the pwdCheckQuality attribute, either accept the password without checking it ('0' or '1') or refuse it ('2').

pwdExpireWarning

This attribute specifies the maximum number of seconds before a password is due to expire that expiration warning messages will be returned to an authenticating user. If this attribute is not present, or if the value is 0 no warnings will be returned. If not 0, the value must be smaller than the value of the pwdMaxAge attribute.

pwdGraceAuthNLimit

This attribute specifies the number of times an expired password can be used to authenticate. If this attribute is not present or if the value is 0, authentication will fail.

pwdLockout

This attribute indicates, when its value is "TRUE", that the password may not be used to authenticate after a specified number of consecutive failed bind attempts. The maximum number of consecutive failed bind attempts is specified in pwdMaxFailure. If this attribute is not present, or if the value is "FALSE", the password may be used to authenticate when the number of failed bind attempts has been reached.

pwdLockoutDuration

This attribute holds the number of seconds that the password cannot be used to authenticate due to too many failed bind attempts. If this attribute is not present, or if the value is 0 the password cannot be used to authenticate until reset by a password administrator.

pwdMaxFailure

This attribute specifies the number of consecutive failed bind attempts after which the password may not be used to authenticate. If this attribute is not present, or if the value is 0, this policy is not checked, and the value of pwdLockout will be ignored.

pwdFailureCountInterval

This attribute holds the number of seconds after which the password failures are purged from the failure counter, even though no successful authentication occurred. If this attribute is not present, or if its value is 0, the failure counter is only reset by a successful authentication.

pwdMustChange

This attribute specifies with a value of "TRUE" that users must change their passwords when they first bind to the directory after a password is set or reset by a password administrator. If this attribute is not present, or if the value is "FALSE", users are not required to change their password upon binding after the password administrator sets or resets the password. This attribute is not set due to any actions specified by this document, it is typically set by a password administrator after resetting a user's password.

pwdAllowUserChange

This attribute indicates whether users can change their own passwords, although the change operation is still subject to access control. If this attribute is not present, a value of "TRUE" is assumed. This attribute is intended to be used in the absense of an access control mechanism.

12 Password Hashing Feature

HP Enterprise Directory Version 5.6 provides a new feature which allows the administrators to hash the password based on the algorithms: CRYPT, and the algorithms supported by OpenSSL_add_all_digests( ) routine of OpenSSL library. Earlier versions of the DSA stored userPasswords of all entries as plain text which is a security risk. Using Password Hashing feature enhances the security by encrypting the users password.

Two new NCL attributes have been defined for DSA Password Hashing feature:

- PASSWORD HASHING STATE

  The above NCL attribute can take one of the two values ON or OFF. Setting this attribute to ON enables hashing of the password. Setting it to OFF disables the hashing of the password.

  For example,

```
NCL>set dsa password hashing state =on
NCL>set dsa password hashing state =off
```

- PASSWORD HASHING ALGO

  The above NCL attribute can take one of the values. "CRYPT" and the algorithms supported by OpenSSL_add_all_digests( ) routine of OpenSSL library.

  For example,

```
NCL> set dsa password hashing algo ="crypt"
NCL> set dsa password hashing algo ="md5"
NCL> set dsa password hashing algo ="sha"
```

"CRYPT", "MD5" and "SHA" are the algorithms with which the userPassword's will be hashed.

Once the "PASSWORD HASHING STATE" is set to ON, the "userPassword" attribute of the entries being added will be hashed using the algorithm (based on the value in the NCL attribute "PASSWORD HASHING ALGO") as set by the administrator.

13 Sample Cluster Application Availability Script

Below is a sample Cluster Application Availability (CAA) script for achieving HP Enterprise Directory failover capability on Tru64 UNIX cluster (TruCluster) environment.

This sample shows how you can typically put the DSA application under the control of CAA. You can customize it by putting any site specific logic according to your environment.

For detailed steps on how to register and invoke CAA script see Section 13.1 and 13.2.

13.1 Sample CAA Script

```
#!/usr/bin/ksh -p
#
# ****************************************************************
# *                                                              *
# *    Copyright (c) Digital Equipment Corporation, 1991, 1999   *
# *                                                              *
# *   All Rights Reserved.  Unpublished rights  reserved  under  *
# *   the copyright laws of the United States.                   *
# *                                                              *
# *   The software contained on this media  is  proprietary  to  *
# *   and  embodies  the  confidential  technology  of  Digital  *
# *   Equipment Corporation.  Possession, use,  duplication  or  *
# *   dissemination of the software and media is authorized only *
# *   pursuant to a valid written license from Digital Equipment *
# *   Corporation.                                               *
# *                                                              *
# *   RESTRICTED RIGHTS LEGEND   Use, duplication, or disclosure *
# *   by the U.S. Government is subject to restrictions  as  set *
# *   forth in Subparagraph (c)(1)(ii)  of  DFARS  252.227-7013, *
# *   or  in  FAR 52.227-19, as applicable.                      *
# *                                                              *
# *                                                              *
# ****************************************************************
#
#
# @(#)$RCSfile: template.scr,v $ $Revision: 1.1.8.2 $(DEC)
# $Date: 2000/04/14 17:42:58 $
#
#
#
######################################################################
#
# The following section contains variables that can be set to best
# suit your application.
#
# Please review each variable and set as needed.
#
# Set CAA_SCRIPT_DEBUG when invoking the script from command line
# for testing. This will cause all output events to go to the terminal,
# rather than being sent to EVM.
#
######################################################################
#
# Application name - set this variable to a name that describes this
# (mandatory)        application.  Enclose the name in double quotes.
#                    Examples: "apache", "netscape"

SERVICE_NAME="dxd"
```

```
# Associated Processes - the application configured may consist of
# (mandatory             single or multiple processes. Specifying
#                        the names of the processes here allows CAA
#                        to monitor that they are running and allows
#                        CAA to completely clean up when stopping the
#                        application.
#                        Ex:  "proc1 proc2"

PROBE_PROCS="/usr/sbin/dxd_dsad"

# Application Startup Command - CAA will invoke this command when starting
# (mandatory)       the application.  Include the command to execute along
#                   with any flags and arguments needed. Use this variable
#                   along with START_APPCMD2 (see below) when
#                   dealing with a simple application start procedure.
#
#                   If the start procedure is complicated and/or involves
#                   many commands, you may find it easier to disregard
#                   this variable and manually code the commands needed
#                   in the "Start" section of this script (see below).
#
#                   Another alternative for a complicated start procedure
#                   is to create a separate script containing those
#                   commands and specifying that script in this variable.
#
#                   Ex: "/cludemo/avs/avsetup -s"
#
#                   NOTE: if not set, you must manually code the commands
#                   to start the application in the "Start" section of
#                   this script.

START_APPCMD="/var/dxd/scripts/dxd_dsa_startup.ncl"

# Secondary Application Startup Command - used in conjunction with the
# (optional)         Application Startup Command just described above.  Use
#                    if desired to implement a two-step startup process for
#                    your application, if needed.

START_APPCMD2=""
```

```
# Application Stop Command - CAA will invoke this command when stopping
# (optional)         the application.  Include the command to execute along
#                    with any flags and arguments needed.  Use this
#                    variable along with STOP_APPCMD2 (see below) when
#                    dealing with a simple application stop procedure.
#
#                    If the stop procedure is complicated and/or involves
#                    many commands, you may find it easier to disregard
#                    this variable and manually code the commands needed
#                    in the "Stop" section of this script (see below).
#
#                    Another alternative for a complicated stop procedure
#                    is to create a separate script containing those
#                    commands and specifying that script in this variable.
#
#                    Ex: "/cludemo/avs/avsetup -k"
#
#                    NOTE: if not set, you should manually code the commands
#                    to stop the application in the "Stop" section of
#                    this script.  Otherwise, this script will not stop the
#                    application in a graceful manner.

STOP_APPCMD="/var/dxd/scripts/dxd_dsa_shutdown.ncl"

# Secondary Application Stop Command - used in conjunction with the
# (optional)         Application Stop Command just described above.  Use
#                    if desired to implement a two-step stop process for
#                    your application, if needed.

STOP_APPCMD2=""

# Application Directory - If set, this script will change to this directory
# (optional)         prior to executing the start process.  This may allow
#                    you to not have to specify full path names for
#                    commands or files in this directory.
#
#                    Ex:  "/var/opt/product1"

APPDIR=""

export SERVICE_NAME START_APPCMD START_APPCMD2
export APPDIR PROBE_PROCS STOP_APPCMD STOP_APPCMD2

#################################################################
#
# The following section contains variables used by CAA.  We recommend
# leaving them defined as is.
#
#################################################################

DEBUG_PRIORITY=100
INFO_PRIORITY=200
ERROR_PRIORITY=500
```

```
SCRIPT=$0
ACTION=$1                       # Action (start, stop or check)

EVMPOST="/usr/bin/evmpost"      # EVM command to post events
DEBUG=0

if [[ "$CAA_SCRIPT_DEBUG" != "" ]]; then
    DEBUG=1
    EVMPOST="/usr/bin/evmpost -r | /usr/bin/evmshow -d"
fi

export EVMPOST ACTION SCRIPT

######################################################################
#
# The following section contains procedures that are available to
# be used from the start, stop, and check portions of this script.
#
######################################################################

#
# postevent - Posts EVM event with specified parameters
#
# Argument:  $1 - priority (optional)
#            $2 - message  (optional)
#
#

postevent () {
    typeset pri=$1
    typeset msg=${2:-failed}

    typeset evt='event { name sys.unix.clu.caa.action_script '

    if [ ! -z "$pri" ]; then
    evt="$evt priority $pri "
    fi

    evt="$evt var {name name value \\\"$SERVICE_NAME\\\" } "
    evt="$evt var {name script value \\\"$SCRIPT\\\" } "
    evt="$evt var {name action value \\\"$ACTION\\\" } "
    evt="$evt var {name message value \\\"$msg\\\" }"

    evt="$evt }"

    evt="echo $evt | $EVMPOST"

    eval $evt
}

#
# getpid - list PIDs of all processes with supplied name
#
# Modified /sbin/init.d/bin/getpid to list all matches
#
# Arguments:  process name
#
```

```
getpid () {
    if [ -n "$1" ]; then
        GETMYPID=$1
        shift
        /bin/ps -e -o pid,command $* | while read mypid command args
        do
     if [ "$command" = "$GETMYPID" ]; then
                echo "$mypid"
            fi
        done
    fi
}

#
# checkdaemon - return the number of instances of a daemon
#
# Argument:  process name
# Return:    number of instances of the named daemon currently running
#

checkdaemon () {
     if [ -f /var/dxd/dxd_dsad_restart_required ]; then
     rm /var/dxd/dxd_dsad_restart_required
     return 0
     else
     return 1
     fi
}

#
# zapdaemon - kill a given process using brutal force (i.e. -9)
#
# Argument:  list of processes to kill
# Return:    1 - failed to kill some process
#            0 - killed all processes
#

zapdaemon () {
    typeset ret=0
```

```
    for i in ${1}
    do
    checkdaemon ${i}
    if [ $? -ne 0 ]; then
    kill `getpid ${i}
          checkdaemon ${i}
          if [ $? -ne 0 ]; then
            kill -9 `getpid ${i}`
            checkdaemon ${i}
            if [ $? -ne 0 ]; then
        postevent $ERROR_PRIORITY "${i}: stuck - could not kill -KILL"
        ret=1
            else
        postevent $ERROR_PRIORITY "${i}: killed with -KILL"
                  fi
         else
           postevent "" "${i}: killed"
      fi
      fi
    done
    return $ret
}

    done
    return $ret
}

#
# probeapp - Probe process to see if in process list.
#
# This simple form of process probing searches the process list for an
# entry corresponding to the specified process.  If found, all is assumed
# to be well.
#
# More accurate process probing may be available depending upon the nature
# of your application.  For instance, you might invoke a test command that
# the process should respond to and check the returned results.  You should
# consider adding this type of probing to this script, if possible.
#
# Argument:  process name
#
# Return:    1 - process not running
#            0 - process running
#
```

```
probeapp () {
    checkdaemon $1
    if [ $? -ne 0 ]; then
    postevent $DEBUG_PRIORITY "$1 check OK"
    return 0
    else
    postevent $DEBUG_PRIORITY "$1 check failed"
    return 1
    fi
}

#############################################################################
#
# Main section of Action Script - starts, stops, or checks an application
#
# This script is invoked by CAA when managing the application associated
# with this script.
#
# Argument:  $1 - start | stop | check
#
# Returns:   0 - successful start, stop, or check
#            1 - error
#
#############################################################################

#
# Start section - start the process and report results
#
# If the Application Startup Commands (see description above) were used,
# little, if any modifications are needed in this section.  If not used,
# you may replace most of the contents in this section with your own
# start procedure code.
#
# For improved serviceability, preserve the commands below that log
# messages or posts events.
#

case $1 in
'start')
    postevent $DEBUG_PRIORITY "trying to start"
    cd $APPDIR
    if [ "$START_APPCMD" != "" ]; then
        out=`ncl </var/dxd/scripts/create_templates ;
             ncl </var/dxd/scripts/start_dsa_event_dispatching.ncl ;
             ncl < $START_APPCMD`
        if [ $? -ne 0 ]; then
            postevent $ERROR_PRIORITY "start: $out"
         exit 1
        fi
    fi
```

```
  ;;
#
# Stop section - stop the process and report results
#
# If the Application Stop Commands or Associated Processes (see descriptions
# above) were used,little, if any modifications are needed in this section.
# If not used, you may replace most of the contents in this section with
# your own stop procedure code.
#
# For improved serviceability, preserve the commands below that log
# messages or posts events.
#
'stop')
    postevent $DEBUG_PRIORITY "trying to stop"
    cd $APPDIR
    if [ "$STOP_APPCMD" != "" ]; then
    out=`ncl < $STOP_APPCMD`
    if [ $? -ne 0 ]; then
        postevent $ERROR_PRIORITY "stop: $out"
     exit 1
     fi
    fi
#
# Kill stubborn processes and applications that don't have a stop command
#
    for i in ${PROBE_PROCS}; do
    zapdaemon ${i}
    done
    ;;
#
# Check section - check the process and report results
#
# If you specified $PROBE_PROCS (see earlier description), very little,
# if any, changes are needed to have simple process checking.
#
# Your application might allow you to implement more accurate process
# checking.  If so, you may choose to implement that code here.  See the
# description for the probeapp function earlier in this script.
#
'check')
    for i in ${PROBE_PROCS}; do
    postevent $DEBUG_PRIORITY "trying to check $i"
    probeapp $i
    if [ $? -ne 0 ]; then
        postevent "" "check failed for $i"
     exit 1
     fi
    done
```

```
    ;;
*)
    postevent $ERROR_PRIORITY "usage: $0 {start|stop|check}"
    exit 1
    ;;
esac
postevent "" success
exit 0
```

13.2 Configuring the CAA Scripts

To configure CAA Scripts, follow these steps:

- If the HP Enterprise Directory server is not installed, unpack and install the kit.
- Copy the HPED CAA script to `/var/cluster/caa/script/DSA.scr`. For details on how to create CAA script refer to Section 13.1.
- Create a CAA application resource profile by entering the following command:
  `# caa_profile -create DSA -t application`

  The DSA CAA resource profile should look as below:

```
# cat /var/cluster/caa/profile/DSA.cap
NAME=DSA
TYPE=application
ACTION_SCRIPT=DSA.scr
ACTIVE_PLACEMENT=0
AUTO_START=0
CHECK_INTERVAL=60
DESCRIPTION=DSA
FAILOVER_DELAY=0
FAILURE_INTERVAL=0
FAILURE_THRESHOLD=0
HOSTING_MEMBERS=
OPTIONAL_RESOURCES=
PLACEMENT=balanced
REQUIRED_RESOURCES=
RESTART_ATTEMPTS=1
SCRIPT_TIMEOUT=60
```

- Register the DSA server with CAA by entering the following command:

```
# caa_register DSA
# caa_stat DSA
NAME=DSA
TYPE=application
TARGET=OFFLINE
STATE=OFFLINE
```

- Start the DSA server by entering the following command:

```
# caa_start DSA
Attempting to start `DSA` on member `ashwa`
Start of `DSA` on member `ashwa` succeeded.
```

- Check the status of the DSA server by entering the following command:

```
#caa_stat DSA
NAME=DSA
TYPE=application
TARGET=ONLINE
STATE=ONLINE on ashwa
```

- Stop the DSA server by entering the following command:

```
# caa_stop DSA
Attempting to stop `DSA` on member `ashwa`
Stop of `DSA` on member `ashwa` succeeded.
```

14 Enabling Memory Tracing Facility

Memory Tracing Facility is a mechanism which records information about all memory allocations and deallocations performed in the DSA during run time. This information is recorded as in-memory ring buffer and would be available in the DSA process dump files incase of DSA crash. By default, in this version of DSA the Memory Tracing Facility is not enabled.

To enable or disable the memory tracing facility, enter the following command:

-

```
NCL> test dsa command "set option option_memcheck=1"
```

-

```
NCL> test dsa command "set option option_memcheck=0"
```

Note: The traces should be turned ON only on recommendation from HP.

15 Support for Samba Schema 3.0

Samba 3.0 Schema elements are supported in this version of HP Enterprise Directory. Applications that require Samba Schema support can now create entries in Enterprise Directory service using these schema elements.