HP DCE for OpenVMS Alpha and OpenVMS I64

Reference Guide

Order Number: BA361-90003

January 2005

This guide provides reference information for HP Distributed Computing Environment (DCE) for OpenVMS Alpha and OpenVMS I64.

Revision/Update Information:	This guide supersedes the *Compaq DCE for OpenVMS VAX and OpenVMS Alpha Reference Guide Version 3.0.*
Operating System:	OpenVMS Alpha Version 7.3-2 or higher OpenVMS I64 Version 8.2
Software Version:	HP DCE for OpenVMS Version 3.2

Hewlett-Packard Company
Palo Alto, California

Printed in the US

ZK6533

The HP OpenVMS documentation set is available on CD-ROM.

Contents

Part IV IDL Reference

8 Additional APIs for Authenticated RPC

9 Enhancements to Existing Authenticated RPC APIs

10 New API for G_Float/IEEE_Float Support

11 DCL Command Interfaces to DCE Tools

Index

Preface

The *HP DCE for OpenVMS Alpha and OpenVMS I64 Reference Guide* provides users of the HP Distributed Computing Environment (DCE) Kit with reference information necessary to use HP DCE Version 3.2 on OpenVMS Alpha and OpenVMS Industry Standard 64 (I64) systems. This guide should be used with the documents listed under Associated Documents.

Intended Audience

This guide is written for:

- Experienced programmers who want to write client/server applications.
- Experienced programmers who want to port existing applications to DCE.
- System managers who manage the distributed computing environment.
- Users who want to run distributed applications.

Document Structure

This guide contains the following parts and chapters:

- Part 1, Integrated Login Reference
  - Chapter 1 provides reference pages for the Integrated Login UAF commands.
  - Chapter 2 provides reference pages for the Integrated Login IMPORT commands.
  - Chapter 3 provides reference pages for the Integrated Login EXPORT commands.
  - Chapter 4 lists Integrated Login status messages.
- Part 2, CDS Subtree Reference
  - Chapter 5 provides reference pages for the CDS subtree commands.
- Part 3, XDS Reference
  - Chapter 6 provides reference pages for the X/Open Directory Services (XDS) API functions.
  - Chapter 7 provides reference pages for the X/Open Object Management (XOM) API functions.
- Part 4, IDL Reference
  - Chapter 8 describes the additional APIs for authenticated RPC.
  - Chapter 9 describes enhancements to existing APIs for authenticated RPC.

- Chapter 10 describes the G-Float/IEEE_Float API support for I64 and Alpha platforms.
- Chapter 11 provides DCL syntax and usage information for the Interface Definition Language (IDL) compiler and the Universal Unique Identifier Generator (UUIDGEN) utility.

Related Documents

For additional information about HP OpenVMS products and services, visit the following World Wide Web address:

`http://www.hp.com/go/openvms/`

Reader's Comments

HP welcomes your comments on this manual. Please send comments to either of the following addresses:

Internet	**openvmsdoc@hp.com**
Postal Mail	Hewlett-Packard Company OSSG Documentation Group, ZKO3-4/U08 110 Spit Brook Rd. Nashua, NH 03062-2698

How To Order Additional Documentation

For information about how to order additional documentation, visit the following World Wide Web address:

`http://www.hp.com/go/openvms/doc/order/`

Conventions

VMScluster systems are now referred to as OpenVMS Cluster systems. Unless otherwise specified, references in this document to OpenVMS Clusters or clusters are synonymous with VMSclusters.

The following conventions are used in this guide:

Ctrl/*x*	A sequence such as Ctrl/*x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
italic text	Italic text indicates important information, complete titles of manuals, or variables. Variables include information that varies in system output (Internal error *number*), in command lines (/PRODUCER=*name*), and in command parameters in text (where *device-name* contains up to five alphanumeric characters).
UPPERCASE TEXT	Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
`Monospace type`	Monospace type indicates code examples and interactive screen displays. In the C programming language, monospace type in text identifies the following elements: keywords, the names of independently compiled external functions and files, syntax summaries, and references to variables or identifiers introduced in an example.

Case-sensitivity	OpenVMS operating system commands do not differentiate between uppercase and lowercase. However, many DCE commands do make this distinction. In particular, the system configuration utility interprets names in a case-sensitive manner.

Part I

Integrated Login Reference

1
Integrated Login DCE$UAF Commands

This chapter contains reference information on the Integrated Login UAF commands discussed in the *HP DCE for OpenVMS Alpha and OpenVMS I64 Product Guide.*

1.1 Running the DCE$UAF Utility

Integrated Login provides two methods of running the DCE$UAF utility:

- By invoking the DCE$UAF utility using a predefined symbol such as:

  ```
  $ DCE$UAF
  DCEUAF>
  ```

  You can also specify a single DCE$UAF command on the command line. Control returns to DCL after the command is executed.

  ```
  $ DCE$UAF command
  $
  ```

  SYS$COMMON:[SYSMGR]DCE$DEFINE_REQUIRED_COMMANDS.COM defines the DCE symbol DCE$UAF, which is used to invoke the DCE$UAF utility. If this symbol is not defined in your environment, you can define it as follows:

  ```
  $ DCE$UAF :== $SYS$SYSTEM:DCE$UAF
  ```

- By issuing the RUN command:

  ```
  $ RUN SYS$SYSTEM:DCE$UAF
  DCEUAF>
  ```

1.2 Initialization File

The Integrated Login DCE$UAF utility has the capability to execute an initialization file. By default, the file is called DCE$UAF_INIT.COM.

The DCE$UAF initialization file is a command procedure that the DCE$UAF utility automatically executes at startup. Every time you start the DCE$UAF utility, the commands contained in the file are automatically executed.

An initialization file contains any command lines you might always enter when you start the DCE$UAF utility. For example, you might include DEFINE/KEY commands in the initialization file.

If you use a file other than DCE$UAF_INIT.COM as your DCE$UAF utility initialization file (for example, START.COM), define it with the following command:

```
$ DEFINE DCE$UAF_INIT WORK:[SMITH]START.COM
```

If you include your startup commands in DCE$UAF_INIT.COM, there is no need to separately define it.

@

Executes a DCE$UAF utility command procedure.

Synopsis

@ FILE-SPEC/OUTPUT=*file-spec*

Parameters

FILE-SPEC
Specifies the command procedure to be executed. A command procedure is a file containing a sequence of DCE$UAF commands. When the command procedure is executed, the commands in the file are processed in order as if each command had been entered individually from the terminal or workstation.

If you do not specify a file type, the system uses the default file type of .COM.

No wildcard characters are allowed in the file specification.

Qualifiers

/OUTPUT=file-spec
Requests that all output directed to the logical device SYS$OUTPUT be written to the file or device specified. System responses and error messages are written to SYS$COMMAND as well as to the specified file.

The default output file type is .LIS.

No wildcard characters are allowed in the output file specification.

Description

The @ command directs the DCE$UAF utility to read commands from the specified command file.

ADD

Adds new records to the DCE authorization file (DCE$UAF).

Synopsis

ADD [USERNAME] [DCE-NAME] */[NO]ALL /FLAGS=flags /[NO]LOG*

Parameters

USERNAME
OpenVMS username for the DCE authorization record to be added. This argument is not required if /ALL is specified. In all other cases it is required.

DCE-NAME
DCE principal name to be associated with the OpenVMS account USERNAME. If the principal exists in a different DCE cell from the current DCE cell, specify the DCE name in the format principal@cell. If the DCE name contains lowercase characters, spaces, or other special characters, enclose the entire string in quotes.

This argument is not required if /ALL is specified. This argument is optional if /ALL is not specified. If it is not specified, then USERNAME, converted to lowercase, is the default.

Qualifiers

/ALL

/NOALL (default)
If present, specifies that a DCE authorization record be created for all accounts in the system authorization file (SYSUAF) that do not currently have an authorization record. In this case the USERNAME and DCE-NAME arguments should not be specified as the DCE name is derived from the OpenVMS username. (Refer to the description of DCE-NAME for more information.) If not present, the record to be added is specified by the USERNAME argument.

/FLAGS=([NO]keyword[,...])
Specifies special attributes that are to be stored with the DCE$UAF record. The keyword you can specify is as follows:

- DCE

  The OpenVMS account defined has a corresponding DCE account. The default is /FLAGS=DCE.

- NO_DCE

  The OpenVMS account defined does not have a corresponding DCE account. In this case DCE-NAME is ignored.

/LOG (default)

/NOLOG
Controls whether or not a message is displayed after a record has been added.

Description

The ADD command adds new records to the DCE authorization file (DCE$UAF).

Example

```
DCE$UAF> ADD SMITH
%DCE-S-UAF_ADDED, created entry SMITH, principal is "smith"
DCE$UAF> ADD SMITH "John Smith"
%DCE-S-UAF_ADDED, created entry SMITH, principal is "John Smith"
DCE$UAF> ADD SMITH "smith@othercell"
%DCE-S-UAF_ADDED, created entry SMITH, principal is "smith@othercell"
```

ANALYZE

Checks the continuity between the OpenVMS system authorization file (SYSUAF) and the DCE Authorization file (DCE$UAF).

Synopsis

ANALYZE /[NO]DCE$UAF /OUTPUT=*output* /[NO]SYSUAF /[NO]VERBOSE

Qualifiers

/DCE$UAF (default)

/NODCE$UAF
Specifies that the DCE authorization file (DCE$UAF) be read record by record and any record found that does not have a matching account in the OpenVMS authorization file (SYSUAF) be reported.

Any discrepancies found can be corrected using the /PURGE command.

/OUTPUT=(*output*)
Determines where the output is written. The default is SYS$OUTPUT:.

/SYSUAF (default)

/NOSYSUAF
Specifies that the OpenVMS system authorization file (SYSUAF) be read account by account and any account found that does not have a matching record in the DCE Authorization file (DCE$UAF) be reported.

Any discrepancies found can be corrected using the ADD command.

/VERBOSE

/NOVERBOSE (default)
Specifies that all accounts/records be displayed. By default only those without matching records/accounts are displayed.

Description

The ANALYZE command checks the continuity between the OpenVMS authorization file (SYSUAF) and the DCE Authorization file (DCE$UAF).

Example

```
DCEUAF> ANALYZE
%DCE-I-UAF_SYSCHKBEG, starting scan of SYSUAF file
SYSUAF entry ALAN does not exist in DCE$UAF
SYSUAF entry BILL does not exist in DCE$UAF
SYSUAF entry DCE$SERVER does not exist in DCE$UAF
SYSUAF entry DEFAULT does not exist in DCE$UAF
SYSUAF entry FAL$SERVER does not exist in DCE$UAF
SYSUAF entry JAMES does not exist in DCE$UAF
SYSUAF entry MAIL$SERVER does not exist in DCE$UAF
SYSUAF entry NML$SERVER does not exist in DCE$UAF
SYSUAF entry OPERATOR does not exist in DCE$UAF
SYSUAF entry PHONE$SERVER does not exist in DCE$UAF
SYSUAF entry SYSTEM does not exist in DCE$UAF
SYSUAF entry WALLY does not exist in DCE$UAF
```

```
12 out of 97 records do not have a DCE$UAF entry
%DCE-I-UAF_SYSCHKEND, completed scan of SYSUAF file
%DCE-I-UAF_DCECHKBEG, starting scan of DCE$UAF file
0 out of 24 records do not have a SYSUAF entry
%DCE-I-UAF_DCECHKEND, completed scan of DCE$UAF file
DCEUAF>
```

ATTACH

Switches control of your terminal from your current process to another process.

Synopsis

ATTACH [PROCESS-NAME] /IDENTIFICATION=*pid*

Parameters

PROCESS-NAME

Specifies the name of a parent process or spawned subprocess to which control passes. The process must already exist, be part of your current job, and share the same input stream as your current process. However, the process cannot be your current process or a subprocess created with the /NOWAIT qualifier.

Process names can contain from 1 to 15 alphanumeric characters. If a connection to the specified process cannot be made, an error message is displayed.

The **PROCESS-NAME** argument is incompatible with the /IDENTIFICATION qualifier.

Qualifiers

/IDENTIFICATION=*pid*

Specifies the process identification (PID) of the process to which terminal control will be transferred. Leading zeros can be omitted. The /IDENTIFICATION qualifier is incompatible with the **PROCESS-NAME** argument.

If you omit the /IDENTIFICATION qualifier, you must specify a process name.

Description

The **ATTACH** command switches control of your terminal from your current process to another process. This command allows you to move between processes that you create with the *SPAWN* command. For example, while you are editing a file, use the **SPAWN** command to move to a subprocess such as the DCE$UAF utility. Then enter **ATTACH** to move back to the editing session. If you want to return to the DCE$UAF utility, enter the **ATTACH** command to move back to the DCE$UAF subprocess you already created.

Examples

1. `DCE$UAF> ATTACH JONES_2`

   Transfers the terminal's control to the subprocess JONES_2.

2. `DCE$UAF> ATTACH/IDENTIFICATION=30019`

   The ATTACH command switches control from the current process to a process having the PID 30019. Notice that because the /IDENTIFICATION qualifier is specified, the **PROCESS-NAME** argument is omitted.

DEFINE/KEY

Associates an equivalence string and a set of attributes with a key on the terminal keyboard.

Synopsis

DEFINE/KEY KEY-NAME EQUIVALENCE-STRING /[NO]ECHO /[NO]ERASE /[NO]IF-STATE /[NO]LOCK-STATE /[NO]LOG /[NO]SET-STATE /[NO]TERMINATE

Parameters

KEY-NAME
Specifies the name of the key that you are defining. All definable keys on VT52 terminals are located on the numeric keypad. On VT100-series terminals, you can define the left and right arrow keys as well as all the keys on the numeric keypad. On terminals with LK201 keyboards, the following types of keys can be defined:

- Keys on the numeric keypad
- Keys on the editing keypad (except the up and down arrow keys)
- Keys on the function key row across the top of the keyboard. (Note that you cannot define function keys F1 to F5.)

Some definable keys are enabled for definition all the time. Others, including KP0 to KP9, Period, Comma, and Minus, must be enabled for definition purposes. Before using these keys, enter either the SET TERMINAL/APPLICATION command or the SET TERMINAL/NONUMERIC command.

On LK201 keyboards, you cannot define the up and down arrow keys or function keys F1 to F5. The left and right arrow keys and the F6 to F14 keys are reserved for command line editing. You must enter the SET TERMINAL/NOLINE_EDITING command before defining these keys. You can also press Ctrl/V to enable keys F7 to F14. Note that Ctrl/V will not enable the F6 key.

EQUIVALENCE-STRING
Specifies the character string to be processed when you press the key. Enclose the string in quotation marks (" ") to preserve spaces and lowercase characters.

Qualifiers

/ECHO (default)

/NOECHO
Displays the equivalence string on your screen after the key has been pressed. You cannot use the /NOECHO qualifier with the /NOTERMINATE qualifier.

/ERASE

/NOERASE (default)
Determines whether the current line is erased before the key translation is inserted.

/IF_STATE=(state-name,...)

/NOIF_STATE
Specifies a list of one or more states, one of which must be in effect for the key definition to work. The /NOIF_STATE qualifier has the same meaning as /IF_STATE=current_state. The state name is an alphanumeric string. States are established with the /SET_STATE qualifier or the SET KEY command. If you specify only one state name, you can omit the parentheses. By including several state names, you can define a key to have the same function in all the specified states.

/LOCK_STATE

/NOLOCK_STATE (default)
Specifies that the state set by the /SET_STATE qualifier remain in effect until explicitly changed. (By default, the /SET_STATE qualifier is in effect only for the next definable key you press or the next read-terminating character that you type.) Can only be specified with the /SET_STATE qualifier.

/LOG (default)

/NOLOG
Displays a message indicating that the key definition has been successfully created.

/SET_STATE=state-name

/NOSET_STATE (default)
Causes the specified state name to be set when the key is pressed. (By default, the current locked state is reset when the key is pressed.) If you have not included this qualifier with a key definition, you can use the SET KEY command to change the current state. The state name can be any alphanumeric string; specify the state as a character string enclosed in quotation marks.

/TERMINATE

/NOTERMINATE (default)
Specifies whether the current equivalence string is to be processed immediately when the key is pressed (equivalent to entering the string and pressing the Return key). By default, you can press other keys before the definition is processed. This allows you to create key definitions that insert text into command lines, after prompts, or into other text that you are entering.

Description

The DEFINE/KEY command associates an equivalence string and a set of attributes with a key on the terminal keyboard. The /KEY qualifier is required.

EXIT

Exits the DCE$UAF utility.

Synopsis

EXIT

Description

The EXIT command allows you to leave the DCE$UAF utility and return to DCL. You can also exit DCE$UAF by entering Ctrl/Z.

MODIFY

Modifies an existing record in the DCE Authorization file (DCE$UAF).

Synopsis

MODIFY USERNAME [DCE-NAME] /FLAGS=*flags* /[NO]LOG

Parameters

USERNAME
DCE$UAF record to be modified.

DCE-NAME
New DCE principal name to be associated with USERNAME. If the principal exists in a different DCE cell from the current DCE cell, specify the DCE name in the format *principal@cell*.

If the DCE name contains lowercase characters, spaces, or other special characters, enclose the entire string in quotes. If this argument is not specified, the DCE principal is not changed.

Qualifiers

/FLAGS=([NO]keyword[,...])
Changes the special attributes that are stored with the DCE$UAF record. The keywords you can specify are documented in the ADD command.

Only those flags that are specified are changed; all other flags remain unchanged. To remove a flag, use the negated form.

If you specify the dce flag, you must specify a DCE name for this entry in DCE-NAME.

/LOG (default)

/NOLOG
Controls whether or not a message is displayed after a record has been modified.

Description

The MODIFY command modifies existing records in the DCE authorization file (DCE$UAF).

PURGE

Removes entries from the DCE Authorization file (DCE$UAF) that do not have a matching entry in the System Authorization file (SYSUAF).

Synopsis

PURGE /[NO]CONFIRM /[NO]LOG /[NO]WARNING

Qualifiers

/CONFIRM

/NOCONFIRM (default)
Controls whether or not the user is asked for confirmation before a DCE$UAF record is deleted.

/LOG (default)

/NOLOG
Controls whether or not a message is displayed after each record has been deleted.

/WARNING (default)

/NOWARNING
Controls whether or not the user is asked the following:

```
Do you really want to delete entries from the DCE$UAF file?
```

Description

The PURGE command removes entries from the DCE Authorization file (DCE$UAF) that do not have a matching entry in the System Authorization file (SYSUAF).

Note that the ANALYZE/DCEUAF command lists the records that are candidates for purging.

REMOVE

Deletes a record from the DCE Authorization file (DCE$UAF).

Synopsis

REMOVE USERNAME /[NO]CONFIRM /[NO]LOG

Parameters

USERNAME
OpenVMS username of the DCE$UAF record that is to be deleted. Full OpenVMS wildcarding is supported.

Qualifiers

/CONFIRM

/NOCONFIRM (default)
Controls whether or not the user is asked for confirmation before the deletion occurs.

/LOG (default)

/NOLOG
Controls whether or not a message is displayed after the record has been deleted.

Description

The REMOVE command deletes a record from the DCE Authorization file (DCE$UAF). Full OpenVMS wildcarding is supported. After calling this command, the specified user can no longer use Integrated Login.

SET VERIFY

Controls whether command lines in command procedures are displayed at the terminal or are printed in a batch job log.

Synopsis

SET [NO]VERIFY

Description

The SET VERIFY command controls whether command lines in command procedures are displayed at the terminal or are printed in a batch job log. The information displayed by the SET VERIFY command can help you in debugging command procedures.

SHOW

Displays records from the DCE Authorization file (DCE$UAF).

Synopsis

SHOW NAME /DCENAME /FORMAT=ADD_COMMAND /FORMAT=REMOVE_COMMAND
/OUTPUT=*output* /VMSNAME /[NO]WILD

Parameters

NAME

OpenVMS username or the DCE name of the DCE$UAF record(s) that is to be displayed. Full OpenVMS wildcarding is allowed.

By default, the name is assumed to be an OpenVMS username. If NAME is to be interpreted as a DCE name, specify /DCENAME. If the DCE name contains lowercase characters, spaces, or other special characters, enclose the entire string in quotes.

Qualifiers

/DCENAME

Specifies that NAME is to be interpreted as a DCE name.

/FORMAT=ADD_COMMAND

When used with the /OUTPUT=output qualifier, produces output that can later be entered into DCE$UAF as valid ADD commands. See the Example section for more information.

/FORMAT=REMOVE_COMMAND

When used with the /OUTPUT=output qualifier, produces output that can later be entered into DCE$UAF as valid REMOVE commands.

You can use this qualifier and a command procedure to remove wildcarded entries from the DCE$UAF file. (The REMOVE command does not support wildcards.) For example, to delete all of the entries that belong to server accounts in your DCE$UAF file, enter the following:

```
DCE$UAF> SHOW *SERVER* /FORMAT=REMOVE_COMMAND /OUTPUT=REM.COM
DCE$UAF> @REM
```

See the Example section for more information.

/OUTPUT=output

Determines where the output is written.

The default is SYS$OUTPUT:.

/VMSNAME (default)

Specifies that NAME is to be interpreted as an OpenVMS account name. This is the default.

/WILD (default)

/NOWILD
Specifies whether or not standard OpenVMS wildcarding is to be applied to NAME. The default is /WILD which means a NAME of "SM*" is interpreted as meaning "match any name starting SM". If /NOWILD is specified the NAME "SM*" is searched for.

Description

The SHOW command displays the DCE$UAF record for the specified user(s).

Example

```
DCE$UAF> SHOW S*
  SMITH                          "smith@/.../othercell.dce.dec.com"
  SILVER                         "silver@/.../othercell.dce.dec.com"
DCE$UAF> SHOW S* /FORMAT=ADD_COMMAND
ADD SMITH "smith@/.../othercell.dce.dec.com"
ADD SILVER "silver@/.../othercell.dce.dec.com"
DCE$UAF> SHOW S* /FORMAT=REMOVE_COMMAND
REMOVE SMITH ! "smith@/.../othercell.dce.dec.com"
REMOVE SILVER ! "silver@/.../othercell.dce.dec.com"
DCE$UAF>
```

SHOW/KEY

Displays the key definitions created with the DEFINE/KEY command.

Synopsis

SHOW/KEY [KEY-NAME] /ALL /[NO]BRIEF /DIRECTORY /[NO]FULL /[NO]STATE

Parameters

KEY-NAME
Specifies the name of the key whose definition you want displayed. For a list of valid key names, see the DEFINE/KEY command.

Qualifiers

/ALL
Displays all key definitions in the current state (or the state specified with the /STATE qualifier). If you use the /ALL qualifier, do not specify a key name.

/BRIEF (default)

/NOBRIEF
Displays only the key definition and state. The /BRIEF and /NOFULL qualifiers are equivalent.

/DIRECTORY
Displays the names of all states for which keys have been defined. If you have not specified a state with a key definition, the SHOW/KEY/DIRECTORY command displays DEFAULT for the state.

You cannot use the /DIRECTORY qualifier with any of the other SHOW/KEY qualifiers.

/FULL

/NOFULL (default)
Displays all qualifiers associated with a definition. By default, only the state of the definition and the definition itself are displayed. The /NOFULL and /BRIEF qualifiers are equivalent.

/STATE=(state-name[,...])

/NOSTATE
Displays the key definitions for the specified state. If you specify only one state name, you can omit the parentheses. State names can be any appropriate alphanumeric string. State names are created with the DEFINE/KEY command.

If you omit the /STATE qualifier or use the /NOSTATE) qualifier, key definitions in the current state are displayed.

Description

The SHOW/KEY command displays the key definitions created with the DEFINE /KEY command.

SHOW/VERSION

Displays the version number of the DCE$UAF utility.

Synopsis

SHOW/VERSION

Description

The SHOW/VERSION command displays the version number of the DCE$UAF utility.

SPAWN

Creates a subprocess of the current process.

Synopsis

SPAWN [COMMAND-STRING] /[NO]CARRIAGE_CONTROL /[NO]CLI /INPUT /[NO]KEYPAD /[NO]LOG /[NO]LOGICAL_NAMES /[NO]NOTIFY /OUTPUT /PROCESS /PROMPT /[NO]SYMBOLS /TABLE /[NO]WAIT

Parameters

COMMAND-STRING
Specifies a DCL command string of less than 132 characters that is to be executed in the context of the created subprocess. When the command completes execution, the subprocess terminates and control returns to the parent process. If both a command string and the /INPUT qualifier are specified, the specified command string executes before additional commands are obtained from the /INPUT qualifier.

Qualifiers

/CARRIAGE_CONTROL

/NOCARRIAGE_CONTROL
Determines whether carriage-return and line-feed characters are prefixed to the subprocess's prompt string. By default, SPAWN copies the current setting of the parent process.

/CLI=cli-filespec

/NOCLI
Specifies the name of a command language interpreter (CLI) to be used by the subprocess. The default CLI is the same as the parent process (defined in SYSUAF). If you specify the /CLI qualifier, the attributes of the parent process are copied to the subprocess.

The CLI you specify must be located in SYS$SYSTEM and have the file type EXE.

/INPUT=filespec
Specifies an input file containing one or more DCL commands to be executed by the spawned subprocess. File type defaults to COM and no wildcards are allowed in the file specification. Once processing of the input file is complete, the subprocess is terminated. If both a command string and the /INPUT qualifier are specified, the specified command string executes before additional commands are obtained from the /INPUT qualifier. If neither is specified, SYS$INPUT is assumed (in which case a SPAWN/NOWAIT command is aborted if Ctrl/Y is pressed to abort something running in your parent process).

You cannot explicitly specify non-record-oriented, process-permanent files (NRO PPFs) with the /INPUT qualifier. The system displays an error message when it encounters such a file as the value for the /INPUT qualifier.

If SYS$INPUT is a terminal, it cannot have an associated terminal mailbox.

/KEYPAD (default)

/NOKEYPAD
Copies keypad key definitions and the current keypad state from the parent process. By default, if you have established key definitions or states with the DEFINE/KEY or the SET/KEY command, these settings are copied to the subprocess. Use the /NOKEYPAD qualifier if you do not want the key settings to be copied.

/LOG (default)

/NOLOG
Displays the assigned subprocess name and any messages indicating transfer of control between processes.

/LOGICAL_NAMES (default)

/NOLOGICAL_NAMES
Copies process logical names and logical name tables to the subprocess. By default, all process logical names and logical name tables are copied to the subprocess except those explicitly marked CONFINE or created in executive or kernel mode.

/NOTIFY

/NONOTIFY (default)
Controls whether a message is broadcast to your terminal notifying you that your subprocess has completed or aborted. This qualifier should not be used unless you specify the /NOWAIT qualifier. The /NOTIFY qualifier cannot be specified when the SPAWN command is executed from within a noninteractive process.

Note that messages broadcast as a result of using the /NOTIFY qualifier are considered to be DCL messages. Therefore, if SET BROADCAST=NODCL is in effect, all such notification messages are suppressed.

/OUTPUT=filespec
Specifies the output file to which the results of the SPAWN operation are written. No wildcards can be used in the file specification. (Do not specify SYS$COMMAND as a file specification for the /OUTPUT qualifier when using the /NOWAIT qualifier; both parent and subprocess output will be displayed simultaneously on your terminal.)

You cannot explicitly specify non-record-oriented, process-permanent files (NRO PPFs) with the /OUTPUT qualifier. The system displays an error message when it encounters such a file as the value for the /OUTPUT qualifier.

If you omit the /OUTPUT qualifier, output is written to the current SYS$OUTPUT device.

/PROCESS=subprocess-name
Specifies the name of the subprocess to be created. If you omit the /PROCESS qualifier, a unique process name is assigned with the same base name as the parent process and a unique number. The default subprocess name format is username_n. If you specify a process name that already exists, an error message is displayed. If the /LOG qualifier has been specified, the assigned name of the subprocess is displayed.

/PROMPT=string
Specifies the prompt string for DCL to use in the subprocess. The default is the prompt of the parent process.

The string can consist of more than one character. All valid ASCII characters can be used in the string. The string must be enclosed in quotation marks (" ") if it contains spaces, special characters, or lowercase characters. Otherwise, letters are automatically converted to uppercase, and leading and trailing spaces are removed.

If no string is specified, the DCL default prompt string " $ " is used for the subprocess.

/SYMBOLS (default)

/NOSYMBOLS
Determines whether global and local symbols (except $RESTART, $SEVERITY, and $STATUS) are passed to the subprocess. $RESTART, $SEVERITY, and $STATUS symbols are never passed to the subprocess.

/TABLE=command-table
Specifies the name of an alternate command table to be used by the subprocess.

/WAIT (default)

/NOWAIT

Requires that you wait for the subprocess to terminate before you enter another DCL command. The /NOWAIT qualifier allows you to enter new commands while the subprocess is running. (Use the /OUTPUT qualifier with the /NOWAIT qualifier to avoid displaying both parent and subprocess output on the terminal simultaneously.)

Note that specifying the /NOWAIT qualifier causes both input and output to be shared with the parent process. If the input device is a terminal, control characters, such as Ctrl/T or Ctrl/Y, also affect all subprocesses sharing the input device. The Ctrl/Y control character, for example, interrupts all such subprocesses. This problem may be avoided by specifying /INPUT=NL:.

Description

The SPAWN command creates a subprocess of the current process. The context of the subprocess is copied from the current process. You can use the SPAWN command to leave the DCE$UAF temporarily, perform other functions (such as displaying a directory listing or printing a file), and then return to the DCE$UAF utility.

VERIFY

Verifies that DCE accounts referenced by the DCE Authorization file (DCE$UAF) still exist in the DCE registry.

Synopsis

VERIFY NAME /DCENAME /OUTPUT=output /VMSNAME /[NO]WILD

Parameters

NAME
OpenVMS username or the DCE name of the DCE$UAF record(s) that is to be verified. Full OpenVMS wildcarding is allowed.

By default the name is assumed to be an OpenVMS username. If NAME is to be interpreted as a DCE name, specify /DCENAME. If the DCE name contains lowercase characters, spaces, or other special characters, enclose the entire string in quotes.

Qualifiers

/DCENAME
Specifies that NAME is to be interpreted as a DCE name.

/OUTPUT=output
Determines where the output is written.

/VMSNAME (default)
Specifies that NAME is to be interpreted as an OpenVMS account name.

/WILD (default)

/NOWILD
Specifies whether or not standard VMS wildcarding is to be applied to NAME. The default is /WILD which means a NAME of "SM*" is interpreted as meaning "match any name starting SM". If /NOWILD is specified, the NAME "SM*" is searched for.

Description

The VERIFY command verifies that DCE accounts referenced by the DCE Authorization file (DCE$UAF) still exist in the DCE registry.

2

Integrated Login IMPORT Commands

This chapter contains reference information on the Integrated Login IMPORT commands discussed in the *HP DCE for OpenVMS Alpha and OpenVMS I64 Product Guide.*

2.1 Running the DCE Registry IMPORT Utility

Integrated Login provides two methods of running the DCE IMPORT utility, as follows:

- By invoking the DCE IMPORT utility using a predefined symbol:

  ```
  $ DCE$IMPORT
  IMPORT>
  ```

  You can also specify a single DCE IMPORT command on the command line. Control returns to DCL after the command is executed.

  ```
  $ DCE IMPORT command
  $
  ```

  SYS$COMMON:[SYSMGR]DCE$DEFINE_REQUIRED_COMMANDS.COM defines the DCE symbol DCE$IMPORT, which is used to invoke the DCE IMPORT utility. If this symbol is not defined in your environment, you can define the symbol as follows:

  ```
  $ DCE$IMPORT :== $SYS$SYSTEM:DCE$IMPORT
  ```

- By issuing the RUN command:

  ```
  $ RUN SYS$SYSTEM:DCE$IMPORT
  ```

ADD/EXCLUDE

Adds an OpenVMS username to the IMPORT exclude list.

Synopsis

ADD/EXCLUDE [USERNAME]

Parameters

USERNAME
Specifies the name of the OpenVMS account to be added to the IMPORT exclude list.

Description

The ADD/EXCLUDE command adds an OpenVMS username to the IMPORT exclude list. This command performs the same function as IMPORT/EXCLUDE.

DELETE/EXCLUDE

Deletes an OpenVMS username from the IMPORT exclude list.

Synopsis

DELETE/EXCLUDE [USERNAME]

Parameters

USERNAME
Specifies the name of the OpenVMS account to be deleted from the IMPORT exclude list.

Description

The DELETE/EXCLUDE command deletes an OpenVMS username from the IMPORT exclude list.

EXIT

Synopsis

EXIT

Exits the IMPORT utility.

Description

The EXIT command exits the IMPORT utility and returns you to DCL. You can also exit IMPORT by pressing Ctrl/Z.

IMPORT

Creates DCE accounts based on OpenVMS accounts from an existing System Authorization File (SYSUAF).

Synopsis

IMPORT [VMS-USERNAME] /[NO]CONFIRM /DCE_LOGIN /[NO]EXCLUDE /[NO]INFORM
/[NO]INTERACTIVE /OUTPUT=output /[NO]RECAP /[NO]TEST_ONLY
/[NO]EXPIRATION_DATE=date /FLAGS=flags /GOOD_SINCE_DATE=date
/GROUP=group /HOME_DIRECTORY=string /LIFETIME=hours
/LOGIN_SHELL=string /MISCELLANEOUS=string
/ORGANIZATION=organization /PASSWORD=passwd /PRINCIPAL=principal
/RENEWABLE_LIFETIME=hours

Parameters

VMS-USERNAME
Specifies the name of the OpenVMS account to be imported.

If an asterisk is specified in place of VMS-USERNAME, all accounts from the OpenVMS system authorization file are selected.

Command Qualifiers

/CONFIRM

/NOCONFIRM
Controls whether the IMPORT command asks for confirmation before creating a DCE principal or account, or both.

In interactive mode the default is /CONFIRM. In noninteractive mode the default is /NOCONFIRM.

/DCE_LOGIN=(keyword=value[,...])
Provides account details of a DCE account authorized to create principals and accounts in the DCE registry. Valid keywords for the DCE_LOGIN qualifier are:

PRINCIPAL — The principal name to be used for authentication purposes when creating accounts and/or principals in the DCE registry. If you do not specify a principal using this qualifier, you are prompted for one interactively.

PASSWORD — The password associated with the principal name that was specified by the PRINCIPAL keyword. If you do not specify a password with this qualifier, you are prompted for one interactively. If you are an interactive user, if you do not specify the PASSWORD keyword and allow IMPORT to prompt you, the password is not echoed and does not appear on your terminal.

/EXCLUDE

/NOEXCLUDE (default)
Determines whether the OpenVMS account is imported to the DCE registry. If the OpenVMS account is not imported then the DCE account is not created. Instead, an entry is created in the IMPORT exclude file for the specified OpenVMS account. IMPORT/EXCLUDE performs the same function as ADD /EXCLUDE.

/INFORM

/NOINFORM (default)
Determines whether the user is informed of OpenVMS accounts that would have been selected for import, but are not selected because they have already been imported (they have an entry in the DCE$UAF) or they have an entry in the IMPORT exclude file.

/INTERACTIVE (default)

/NOINTERACTIVE
Controls whether an interactive or noninteractive import is performed.

In interactive mode, a series of questions is asked and the user's responses are used to determine the account details. This mode is well suited to interactive users.

In noninteractive mode, all input is supplied through the data qualifiers, and any missing or conflicting data causes the DCE account not to be created. This mode is well suited to command files and batch jobs.

Data qualifiers can be specified in interactive mode. In this case the data they provide is used to provide the default answers to the relevant questions. All questions are asked.

/OUTPUT=output
Specifies the location at which all program output is written. The default is SYS$OUTPUT:.

/RECAP

/NORECAP (default)
If you specify /RECAP, details of the DCE account are displayed before it is actually created. When /CONFIRM is also specified, the account details are displayed immediately before the confirmation request.

/TEST_ONLY

/NOTEST_ONLY (default)
If you specify /TEST_ONLY, DCE accounts and DCE$UAF entries are not actually created.

Data Qualifiers

/EXPIRATION_DATE=date

/NOEXPIRATION_DATE (default)
Specifies the expiration date for the DCE account. If this qualifier is not specified, or if /NOEXPIRATION_DATE is specified, then the DCE account is created without an expiration date.

/FLAGS=([no]keyword[,...])
Specifies attributes of the DCE account. The keywords you can specify are as follows:

ACCOUNT_VALID — Determines account validity. An account without this flag set is invalid and cannot log in. The default is ACCOUNT_VALID.

CLIENT — Indicates whether the account is for a principal that can act as a client. The default is CLIENT.

DUPLICATE_KEYS — Determines if tickets issued to the account's principal can have duplicate keys. The default is NODUPLICATE_KEYS.

FORWARDABLE_CERTIFICATES — Determines whether a new ticket-granting ticket with a network address that differs from the present ticket-granting ticket network address can be issued to the account's principal. (The PROXIABLE CERTIFICATE flag performs the same function for service tickets.) The default is FORWARDABLE_CERTIFICATES.

PASSWORD_VALID — Determines whether the current password is valid. If this flag is not set, the next time the principal logs in to the DCE account, the system prompts the principal to change his or her password. The default is PASSWORD_ VALID.

POSTDATED_CERTIFICATES — Determines if tickets with a future start time can be issued to the account's principal. The default is NOPOSTDATED_ CERTIFICATES.

PROXIABLE_CERTIFICATE — Determines whether a new ticket with a different network address than the present ticket can be issued to the account's principal. (The FORWARDABLE CERTIFICATE flag performs the same function for ticket-granting tickets.) The default is NOPROXIABLE_CERTIFICATE.

RENEWABLE_CERTIFICATE — Determines if the ticket-granting ticket issued to the account's principal can be renewed. If this flag is set, the authentication service renews the ticket-granting ticket if its lifetime is valid. The default is RENEWABLE_CERTIFICATE.

SERVER — Indicates whether the account is for a principal that can act as a server. The default is SERVER.

TGT_AUTHENTICATION — Determines whether tickets issued to the account's principal can use the ticket-granting ticket authentication mechanism. The default is TGT_AUTHENTICATION.

/GOOD_SINCE_DATE=date
Specifies the date and time that the account was known to be in an uncompromised state. If not specified, this date is set to the current date and time.

/GROUP=group
Specifies the name of an existing DCE group that is associated with the account being created. If the group does not exist, it is not created by IMPORT. The default group name is "none".

/HOME_DIRECTORY=string
Specifies the directory in which the principal is placed at login. If not specified, the DCE account is created without a home directory.

/LIFETIME=hours
Specifies the maximum amount of time, in hours, that a ticket can be valid. If not specified, the maximum certificate lifetime defined as registry authorization policy is used.

/LOGIN_SHELL=string
Specifies the shell that is executed when a principal logs in. If not specified, the DCE account is created without a login shell.

/MISCELLANEOUS=string
Specifies a text string that is typically used to describe the use of the account. If not specified, the DCE account is created without a miscellaneous value.

/ORGANIZATION=organization
Specifies the name of an existing DCE organization that is associated with the account being created. If the organization does not exist, it is not created by IMPORT. The default organization name is none.

/PASSWORD=passwd
Specifies the password to be assigned to the DCE account. If not specified, the DCE account is created without a valid DCE password.

/PRINCIPAL=(keyword[,...])
Specifies the principal that is associated with the DCE account that is being created. The keywords you can specify are as follows:

ALIAS — Specifies that the principal defined by the NAME keyword is an alias. By default the name is considered a primary principal.

FULL_NAME=string — An optional string that is used to more fully qualify a primary name. If the name contains spaces, lowercase characters, or any other special characters, enclose the string in quotes. The default is no full name.

NAME=name — The standard name (primary or alias) that is associated with the DCE account. If the name contains spaces, lowercase characters, or any other special characters, enclose the string in quotes. The default is to take the username from the system authorization file (SYSUAF) record, edit it according to the CASE keyword, and then use this as the principal name.

OBJECT_CREATION_QUOTA=number — The number of registry objects that can be created by the principal. If you do not specify this keyword, then no quota is established and the principal can create an unlimited number of registry objects.

UNIX_ID=number — The required UNIX identifier that is associated with the principal. If a primary principal is being created, you can omit the UNIX ID and one is generated automatically. If an alias principal is being created, you must specify the UNIX ID of the corresponding primary principal.

CASE=keyword — Specifies how the principal name should be formatted. For example, to specify that the principal name should be all lowercase, use */PRINCIPAL=CASE=LOWERCASE*.

noedit — This is the default and indicates that no formatting should be performed.

lowercase[=n1,[n2]] — Convert the principal name so that the first n1 characters and last n2 are lowercase, and the remainder are uppercase. If you do not specify a value for n1, the entire principal is converted to lowercase. If you do not specify a value for n2, 0 is used.

uppercase[=n1,[n2]] — Convert the principal name so that the first n1 characters and last n2 are uppercase, and the remainder are lowercase. If you do not specify a value for n1, the entire principal is converted to uppercase. If you do not specify a value for n2, 0 is used.

/RENEWABLE_LIFETIME=hours
Specifies the amount of time, in hours, before a principal's ticket-granting ticket expires and that principal must log in to the system again to reauthenticate and obtain another ticket-granting ticket.

If not specified, the maximum certificate renewable lifetime defined as registry authorization policy is used.

Description

The DCE IMPORT command creates DCE accounts, and optionally principals, based on existing VMS account information. It also creates entries in the DCE$IMPORT exclude file.

The DCE IMPORT function reads the specified record(s) from the OpenVMS system authorization file (SYSUAF) and for each selected account performs the following:

- If a DCE$UAF record for this OpenVMS account already exists, the account is not imported. (An existing DCE$UAF record is an indication that this OpenVMS account has already been imported.)
- If an entry for this OpenVMS account exists in the IMPORT exclude file, the account is not imported. (An entry in the IMPORT exclude file signifies that this OpenVMS account should not be imported.)
- Otherwise, an attempt is made to create the DCE principal and account. If the principal and account are successfully created, then the matching DCE$UAF record is also created.

Although the DCE principal and account are created if they do not already exist, the group and organization entries are not created. This is done purposely to eliminate the risk of creating erroneous groups and organizations.

If either the DCE principal or account already exists, it is treated as a success and the corresponding DCE$UAF entry is created. Use the DCE$UAF utility if you want to create DCE$UAF entries for existing principals and accounts.

DCE IMPORT has two modes, interactive and noninteractive. Refer to the description of the /INTERACTIVE qualifier for details.

If you do not specify /DCE_LOGIN, you are prompted for your principal name and password (nonechoed) before any account processing begins. This is true in interactive and noninteractive mode.

Examples

This section shows the dialog during an interactive IMPORT session. The dialog is very similar to RGY_EDIT create account dialog; the order of questions and the defaults are often the same.

Each question requires input from the user (note that in this context the user is probably the system administrator), and most questions offer a default. Some defaults vary depending upon the answers to previous questions, and some vary depending upon how you answered the same question before. This second feature is known as **sticky input** and reduces the amount of input the user must type. Some defaults are reset each time you start on a new OpenVMS account while others are carried forward to the next account; this is **intra-account sticky input** and **inter-account sticky input**, respectively.

All text comparisons are made case-blind. All nonquoted input obtained from the command line qualifiers is converted to uppercase. Input obtained from interactive questions is not converted to uppercase.

The OpenVMS account details are displayed for the first (or current, or next) account as follows:

1.

```
OpenVMS Account Details:

Username:  SMITH
Owner:     John Smith
Account:   OVMS

c - create DCE account using regular script
a - create DCE account using abbreviated script
x - add this OpenVMS account to the IMPORT exclude list
s - skip this OpenVMS account
e - exit IMPORT

Enter option (c/a/x/s/e) [c]:)
```

Default: c

Sticky Input: Inter-Account

Valid Responses: c a x s e

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

The OpenVMS account details are displayed for the first (or current) account and the user is asked which action is required for this account.

If the user enters c or a then the dialog continues from step 2.

If the user enters x then an entry is created in the IMPORT exclude file for this account and the dialog continues from step 1.

If the user enters s then the current OpenVMS account is not processed any further, the next OpenVMS account (if any) is selected and the dialog continues from step 1.

If the user enters e the IMPORT utility terminates.

2.

```
Enter DCE account details:

Principal [smith]:
```

Default: The username from the current system authorization (SYSUAF) record, converted to lowercase.

Sticky Input: Intra-Account

Valid Responses: Any string, except null

Case-Sensitive: Yes

Invalid Response causes question to be re-asked: No

The user either enters a different principal name for the account or accepts the default.

If the principal is already is use, an error is displayed and the dialog restarts from step 1.

An invalid response causes the dialog to restart from step 1.

3.

```
The principal "smith" does not exist in the DCE registry.
Do you want to create the principal (y/n) [y]:
```

Default: y
Sticky Input: Inter-Account
Valid Responses: y n
Case-Sensitive: No
Invalid Response causes question to be re-asked: Yes
If the user enters n then restart from step 1, otherwise continue.

4.

```
Enter details for DCE Principal "smith":
Alias (y/n) [n]:
```

Default: n
Sticky Input: Inter-Account
Valid Responses: y n
Case-Sensitive: No
Invalid Response causes question to be re-asked: Yes

5.

```
UNIX number (-1 means auto-assign) [-1]:
```

Default: -1
Sticky Input: Intra-Account
Valid Responses: Integer in range -1 through 65535
Case-Sensitive: No
Invalid Response causes question to be re-asked: Yes

6.

```
Full Name [John Smith]:
```

Default: The owner from the current system authorization (SYSUAF) record.
Sticky Input: Intra-Account
Valid Responses: Any string, including null string
Case-Sensitive: Yes
Invalid Response causes question to be re-asked: Yes

7.

```
Object Creation Quota (-1 means unlimited) [-1]:
```

Default: -1
Sticky Input: Inter-Account
Valid Responses: -1, 0 or Positive Integer.
Case-Sensitive: No
Invalid Response causes question to be re-asked: Yes

8.

```
OK to create this principal now (y/n) [y]:
```

Default: y

Sticky Input: No

Valid Responses: y n

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

If the user enters *n* then the dialog restarts from step 1.

If the user enters *y*, an attempt is made to create the principal. If the principal creation fails, an error message is displayed and the dialog restarts from step 1. Otherwise, the principal is successfully created and the dialog continues.

9.

```
Group [none]:
```

Default: none

Sticky Input: Inter-Account

Valid Responses: Any string, excluding null string

Case-Sensitive: Yes

Invalid Response causes question to be re-asked: Yes

A check is made to see if the group exists. If the group does not exist, then an error message is displayed and the question is repeated.

10.

```
Organization [none]:
```

Default: none

Sticky Input: Inter-Account

Valid Responses: Any string, excluding null string

Case-Sensitive: Yes

Invalid Response causes question to be re-asked: Yes

A check is made to see if the organization exists. If the organization does not exist, then an error message is displayed and the question is repeated.

11.

```
Enter Password (null means no valid password) []:
```

Default: Null string

Sticky Input: No

Valid Responses: Any string, including null string

Case-Sensitive: No

Invalid Response causes question to be re-asked: No

The response is not echoed as the user enters it.

If a null string is entered, IMPORT does not set a valid password on the DCE account and the account user is only able to log in using his or her OpenVMS password.

12.

```
Retype password:
```

Default: No default

Sticky Input: No

Valid Responses: Any string, including null string

Case-Sensitive: No

Invalid Response causes question to be re-asked: No

The user reenters the password for verification. If the verification check fails then an error message is displayed and the dialog continues from step 11.

This question is skipped if a password was not entered in step 11.

If the create abbreviated option was taken in step 1, the dialog now jumps to step 31, assuming that all further questions had been answered with a RETURN to accept their defaults.

13.

```
Enter misc info []:
```

Default: Null string

Sticky Input: Inter-Account

Valid Responses: Any string, including null string

Case-Sensitive: Yes

Invalid Response causes question to be re-asked: Yes

User inputs optional miscellaneous data.

14.

```
Enter home directory [/]:
```

Default: /

Sticky Input: Intra-Account

Valid Responses: Any string, including null string

Case-Sensitive: Yes

Invalid Response causes question to be re-asked: Yes

15.

```
Enter shell []:
```

Default: Null string

Sticky Input: Inter-Account

Valid Responses: Any string, including null string

Case-Sensitive: Yes

Invalid Response causes question to be re-asked: No

16.

```
Password valid (y/n) [y]:
```

Default: y

Sticky Input: Inter-Account

Valid Responses: y n

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

This question is omitted if a password was not provided in step 11.

17.

```
Enter expiration date (standard VMS time format or none) [none]:
```

Default: none

Sticky Input: Inter-Account

Valid Responses: OpenVMS standard date/time or none

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

If the date/time is in the past then it is considered invalid.

18.

```
Allow account to be client principal (y/n) [y]:
```

Default: y

Sticky Input: Inter-Account

Valid Responses: y n

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

19.

```
Account valid for login (y/n) [y]:
```

Default: If the disuser flag from the current system authorization (SYSUAF) record is set, the default is n; otherwise, the default is y.

Sticky Input: Intra-Account

Valid Responses: y n

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

20.

```
Allow account to obtain post-dated certificates (y/n) [n]:
```

Default: n

Sticky Input: Inter-Account

Valid Responses: y n

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

21.

```
Allow account to obtain forwardable certificates (y/n) [y]:
```

Default: y

Sticky Input: Inter-Account

Valid Responses: y n

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

22.

```
Allow certificates to this account to be issued
       via TGT authentication (y/n) [y]:
```

Default: y

Sticky Input: Inter-Account

Valid Responses: y n

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

23.

```
Allow account to obtain renewable certificates (y/n) [y]:
```

Default: y

Sticky Input: Inter-Account

Valid Responses: y n

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

24.

```
Allow account to obtain proxiable certificates (y/n) [n]:
```

Default: n

Sticky Input: Inter-Account

Valid Responses: y n

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

25.

```
Allow account to obtain duplicate session keys (y/n) [n]:
```

Default: n

Sticky Input: Inter-Account

Valid Responses: y n

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

26.

```
Good since date (standard VMS time) [current-date-time]:
```

Default: Current date/time

Sticky Input: Intra-Account

Valid Responses: OpenVMS standard date/time

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

27.

```
Create/Change authorization policy for this account (y/n) [n]:
```

Default: n

Sticky Input: Inter-Account

Valid Responses: y n

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

If the user answers **n** the dialog continues from step 31.

28.

```
Enter maximum certificate lifetime in hours (0 means forever) [8]:
```

Default: Taken from registry authorization policy

Sticky Input: Intra-Account

Valid Responses: Positive integer, including 0

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

29.

```
Enter maximum certificate-renewable lifetime in hours
         (0 means forever) [168]:
```

Default: Taken from registry authorization policy

Sticky Input: Intra-Account

Valid Responses: Positive integer, including 0

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

30.

```
OK to create DCE account based on above (y/n) [y]:
```

Default: y

Sticky Input: No

Valid Responses: y n

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

If /NOCONFIRM was specified, then this question is not asked.

If the /RECAP qualifier was specified, then immediately before this question details of the account about to be created are displayed.

If the user answers n, an `account not created` message is displayed and the dialog starts again, for the same OpenVMS account, from step 1.

If the user answers y or if /NOCONFIRM was specified, then an attempt is made to create the DCE account. If the account creation succeeds, then a success message is displayed and the dialog starts for the next OpenVMS account from step 1. If the DCE account creation fails, then an error message is displayed and the dialog starts again, for the same OpenVMS account, from step 1.

Following is an example of an interactive IMPORT command:

```
IMPORT> IMPORT SMITH

OpenVMS Account Details:

 Username:  SMITH
 Owner:     John Smith
 Account:   OVMS
 c - create DCE account using regular script
 a - create DCE account using abbreviated script
 x - add this OpenVMS account to the IMPORT exclude list
 s - skip this OpenVMS account
 e - exit IMPORT

 Enter option (c/a/x/s/e) [c]: c

 Enter DCE account details:

 Principal [smith]:

 The principal "smith" does not exist in the DCE registry.
 Do you want to create the principal (y/n) [y]:

 Enter details for DCE Principal "smith":

   Alias (y/n) [n]:
   UNIX number (-1 means auto-assign) [-1]:
   Full Name [John Smith]:
   Object Creation Quota (-1 means unlimited) [-1]:

 OK to create this principal now (y/n) [y]:

 Principal "smith" successfully created.

 Group [none]: DCE
 Organization [none]: OpenVMS
 Enter Password (null means no valid password) []:
 Retype password:
 Enter misc info []:
 Enter home directory [/]:
 Enter shell []:
 Password valid (y/n) [y]:
 Enter expiration date (standard VMS time format or none) [none]:
 Allow account to be server principal (y/n) [y]:
 Allow account to be client principal (y/n) [y]:
 Account valid for login (y/n) [y]:
 Allow account to obtain post-dated certificates (y/n) [n]:
 Allow account to obtain forwardable certificates (y/n) [y]:
 Allow certificates to this account to be issued
 via TGT authentication (y/n) [y]:
 Allow account to obtain renewable certificates (y/n) [y]:
 Allow account to obtain proxiable certificates (y/n) [n]:
 Allow account to obtain duplicate session keys (y/n) [n]:
 Good since date (standard VMS time) [current-date-time]:
 Create/Change authorization policy for this account (y/n) [n]:

 OK to create DCE account based on above (y/n) [y]:

 DCE Account successfully created.

IMPORT>
```

SHOW/EXCLUDE

Displays OpenVMS usernames in the IMPORT exclude list.

Synopsis

SHOW/EXCLUDE [USERNAME] /ALL /OUTPUT=output

Parameters

USERNAME
Specifies the name of the OpenVMS account to be displayed from the IMPORT exclude list. Full OpenVMS wildcarding is allowed. If you specify a value or values for the USERNAME parameter, you cannot specify the /ALL qualifier.

Qualifiers

/ALL
Specifies that all IMPORT exclude entries are to be displayed. If you do not specify a username, then this qualifier is assumed.

/OUTPUT=output
Specifies the location at which the output is written. The default is SYS$OUTPUT:.

Description

The SHOW/EXCLUDE command displays OpenVMS usernames from the IMPORT exclude list.

3
Integrated Login EXPORT Commands

This chapter contains reference information on the Integrated Login EXPORT commands discussed in the *HP DCE for OpenVMS Alpha and OpenVMS I64 Product Guide.*

3.1 Running the DCE Registry EXPORT Utility

The DCE EXPORT utility allows system administrators to create an OpenVMS authorization file (SYSUAF) based on existing accounts in the DCE registry.

Integrated Login provides two methods of running the DCE EXPORT utility:

- By invoking the DCE EXPORT utility using a predefined symbol:

  ```
  $ DCE$EXPORT
  EXPORT>
  ```

  You can also specify a single DCE EXPORT command on the command line. Control returns to DCL after the command is executed.

  ```
  $ DCE EXPORT command
  $
  ```

  SYS$COMMON:[SYSMGR]DCE$DEFINE_REQUIRED_COMMANDS.COM defines the DCE symbol DCE$EXPORT, which is used to invoke the EXPORT utility. If this symbol is not defined in your environment, you can define the symbol as follows:

  ```
  $ DCE$EXPORT :== $SYS$SYSTEM:DCE$EXPORT
  ```

- By issuing the RUN command:

  ```
  $ RUN SYS$SYSTEM:DCE$EXPORT
  ```

ADD/EXCLUDE

Adds a DCE principal name to the EXPORT exclude list.

Synopsis

ADD/EXCLUDE [PRINCIPAL]

Parameters

PRINCIPAL
Specifies the DCE principal name to be added to the EXPORT exclude list. Lowercase principal names must be enclosed in quotes.

Description

The ADD/EXCLUDE command adds a DCE principal name to the EXPORT exclude list. This command performs the same function as EXPORT/EXCLUDE.

DELETE/EXCLUDE

Deletes a DCE principal name from the EXPORT exclude list.

Synopsis

DELETE/EXCLUDE [PRINCIPAL]

Parameters

PRINCIPAL
Specifies the DCE principal name to be deleted from the EXPORT exclude list. Lowercase principal names must be enclosed in quotes.

Description

The DELETE/EXCLUDE command deletes a DCE principal name from the EXPORT exclude list.

EXIT

Exits the EXPORT utility.

Synopsis

EXIT

Description

The EXIT command exits the EXPORT utility and returns you to DCL. You can also exit EXPORT by pressing Ctrl/Z.

EXPORT

Creates OpenVMS accounts based on existing DCE accounts in the DCE registry.

Synopsis

EXPORT [DCE-ACCOUNT-NAME] /[NO]ADD_IDENTIFIERS /[NO]CONFIRM /DCE_LOGIN
/[NO]EXCLUDE /[NO]INFORM /[NO]INTERACTIVE /OUTPUT=output
/[NO]RECAP /[NO]TEST_ONLY /[NO]WILD
/[NO]ACCOUNT=account /DEVICE=device /DIRECTORY=directory
/GROUP_UIC=group_uic /LIKE=vms_account
/MEMBER_UIC=member_uic /[NO]OWNER=owner
/PASSWORD=passwd /[NO]QUOTA=n /USERNAME=username

Parameters

DCE-ACCOUNT-NAME
Specifies the name of the DCE account that is to be exported. If the DCE account name contains lowercase characters, spaces or other special characters, enclose the name in quotes.

If you specify an asterisk for this argument, all accounts from the registry are selected.

Command Qualifiers

/ADD_IDENTIFIERS

/NOADD_IDENTIFIERS (default)
Adds identifiers for the username and account name to the OpenVMS rights database.

/CONFIRM

/NOCONFIRM
Controls whether the EXPORT command asks for confirmation before creating the OpenVMS account.

In interactive mode, the default is /CONFIRM. In noninteractive mode, the default is /NOCONFIRM.

/DCE_LOGIN=(keyword=value[,...])
Provides account details of a DCE account authorized to create principals and accounts in the DCE registry. Valid keywords for the DCE_LOGIN qualifier are:

PRINCIPAL — The principal name to be used for authentication purposes when creating accounts and/or principals in the DCE registry. If you do not specify a principal using this qualifier, you are prompted for one interactively.

You must enclose the principal name in quotes to maintain lowercase.

PASSWORD — The password associated with the principal name that was specified by the PRINCIPAL keyword. If you do not specify a password via this qualifier, you are prompted for one interactively. If you are an interactive user, if you do not specify the PASSWORD keyword and allow EXPORT to prompt you, the password is not echoed and does not appear on your terminal.

This information has to be entered only once per session, on the first EXPORT command. Subsequent EXPORT commands within the same session do not require you to reenter this information.

/EXCLUDE

/NOEXCLUDE (default)
Determines whether the DCE account is exported to OpenVMS. If the DCE account is not exported, then the OpenVMS account is not created. Instead, an entry is created in the EXPORT exclude file for the specified DCE account. EXPORT/EXCLUDE performs the same function as ADD/EXCLUDE.

/INFORM

/NOINFORM (default)
Determines whether the user is informed of DCE accounts that would have been selected for export, but are not selected because they have already been exported (they have an entry in the DCE$UAF) or they exist in the EXPORT exclude file.

/INTERACTIVE (default)

/NOINTERACTIVE
Controls whether an interactive or noninteractive export is performed.

In interactive mode a series of questions is asked and the user's responses are used to determine the account details. This mode is well suited to interactive users.

In noninteractive mode all input is supplied through the data qualifiers, and any missing or conflicting data causes the OpenVMS account to not be created. This mode is well suited to command files and batch jobs.

Data qualifiers can be specified in interactive mode. In this case the data they provide is used to provide the default answers to the relevant questions. All questions are asked.

/OUTPUT=output
Specifies the location at which all program output is written. The default is SYS$OUTPUT:.

/RECAP

/NORECAP (default)
If you specify /RECAP, details of the OpenVMS account are displayed before it is actually created. When /CONFIRM is also specified, the account details are displayed immediately before the confirmation request.

/TEST_ONLY

/NOTEST_ONLY (default)
If you specify /TEST_ONLY, OpenVMS accounts, identifiers, and DCE$UAF entries are not actually created.

/WILD (default)

/NOWILD
Specifies whether standard VMS wildcarding is applied to DCE-ACCOUNT-NAME. The default is /WILD, which means that a DCE-ACCOUNT-NAME of "SM*" is interpreted as "export any account starting with SM". If /NOWILD is specified, the DCE-ACCOUNT-NAME "SM*" is exported.

Data Qualifiers

/ACCOUNT=account (default)

/NOACCOUNT
Specifies the account string for the OpenVMS account (same as /ACCOUNT in AUTHORIZE). The account is a string of 1 to 8 alphanumeric characters.

If this qualifier is not specified, the DCE account's group name, truncated to 8 characters if necessary, is used. Specify /NOACCOUNT if no account field is required.

/DEVICE=device
Specifies the name of the OpenVMS account's default device at login. The device name is a string of 1 to 31 alphanumeric characters. If you omit the colon from the device name value, a colon is automatically appended.

The default device is copied from the device field from the account specified by the /LIKE qualifier.

/DIRECTORY=directory
Specifies the default directory name for the DIRECTORY field of the OpenVMS SYSUAF record. The directory name can be 1 to 63 alphanumeric characters. If you do not enclose the directory name in brackets, EXPORT adds the brackets for you.

The default directory name is [username], where username is the OpenVMS account's username.

/GROUP_UIC=group_uic
Specifies the group part of the UIC for the OpenVMS account. You can specify this qualifiier as an octal group UIC code or as an existing group UIC identifier. If specified as an octal number, it must be in the range 1 to 37776 (octal).

The default is to take the OpenVMS account's ACCOUNT field, convert it to uppercase, and interpret this as a group UIC identifier. If such an identifier does not exist, a similar translation is attempted for the DCE account's group name. If neither identifiers exist then the group UIC is derived from the OpenVMS account specified by the LIKE qualifier.

/LIKE=vms-account
Specifies an existing OpenVMS account to be used as the basis for the OpenVMS account being created. Any fields not specified on the EXPORT command line, as well as all quotas, privileges, and so on, are inherited from the /LIKE account. The default is DEFAULT (as it is in AUTHORIZE).

This qualifier is useful if you want to specify SYSUAF flags on a newly created account that are different from the default. In that case, use the /LIKE qualifier and specify an existing account with the desired SYSUAF flags.

/MEMBER_UIC=member_uic
Specifies the member part of the UIC for the OpenVMS account. /MEMBER_UIC should be specified as an octal number within the range 0 to 177776 (octal).

The default is to use the first available member UIC within the group UIC (as specified by /GROUP_UIC). For example, if the selected group is 150 and that group has members 1, 2, 5 and 6 already defined, then the new UIC is [150,3].

/OWNER=owner (default)

/NOOWNER
Specifies the owner string for the OpenVMS account (same as /OWNER in AUTHORIZE). The owner is a string of 1 to 31 characters.

If you do not specify this qualifier, the DCE account's principal name, truncated to 31 characters if necessary, is used. Specify /NOOWNER if no owner field is required.

/PASSWORD=passwd
Specifies the password for the OpenVMS account. Passwords can be from 0 to 32 characters in length and can include alphanumeric characters, dollar signs, and underscores. They are not case-sensitive.

If you do not specify a password, the account is created without a valid OpenVMS password.

/QUOTA=quota (default)

/NOQUOTA
Specifies the disk quota for the device specified by /DEVICE to be given to the OpenVMS account (if quotas are enabled on that volume).

The default is 1000 blocks. If quotas are not enabled on the device specified by /DEVICE or if /NOQUOTA is specified, then no quota is given.

/USERNAME=username
Specifies the username for the OpenVMS account. The username is a string of 1 to 12 alphanumeric characters. The string can contain underscores.

If you do not specify this qualifier, the DCE account's principal name, truncated to 12 characters and uppercased, is used.

Description

The DCE EXPORT command is used to create accounts in the OpenVMS system authorization file (SYSUAF) based on existing accounts in the DCE registry.

The DCE EXPORT function reads the specified account(s) from the DCE registry and for each selected account performs the following:

- If a DCE$UAF record for this DCE account already exists, the account is not exported. (An existing DCE$UAF record is an indication that this DCE account has already been exported.) Note that orphaned DCE$UAF entries can be detected via the DCE$UAF ANALYZE command.
- If an entry for this DCE account exists in the EXPORT exclude file, the account is not exported. (An entry in the EXPORT exclude file signifies that this DCE account should not be exported.)

- If a DCE$UAF record does not exist, the DCE EXPORT utility attempts to create an OpenVMS account. If the account is successfully created, the matching DCE$UAF record is also created.

DCE EXPORT has two modes, interactive and noninteractive. Refer to the description of the /INTERACTIVE qualifier for details.

If the OpenVMS account already exists, it is treated as a success and the corresponding DCE$UAF entry is created.

If you want to specify SYSUAF flags that are different from the default on a newly created account, use the /LIKE qualifier and specify an existing account with the desired SYSUAF flags.

Examples

This section shows the dialog during an interactive EXPORT session.

Each question requires input from the user (note that in this context the user is probably the system administrator), and most questions offer a default. Some defaults vary depending upon the answers to previous questions, and some vary depending upon how you answered the same question before. This second feature is known as **sticky input** and reduces the amount of input the user must type. Some defaults are reset each time you start on a new OpenVMS account while others are carried forward to the next account; this is **intra-account sticky input** and **inter-account sticky input**, respectively.

All text comparisons are made case-blind. All nonquoted input is converted to uppercase before being stored in SYSUAF.

The DCE account details are displayed for the first (or current, or next) account as follows:

1.

```
DCE Account Details:

Principal:     smith (John Smith)
Group:         ETG
Organization:  OVMS

c - create OpenVMS account
x - add this DCE account to the EXPORT exclude file
s - skip this DCE account
e - exit IMPORT

Enter option (c/x/s/e) [c]:
```

Default: c

Sticky Input: Inter-Account

Valid Responses: c x s e

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

The DCE account details are displayed for the first (or current) account and the user is asked which action is required for this account.

If the user enters c then the dialog continues from step 2.

If the user enters x then an entry for this DCE account is created in the EXPORT exclude file and the dialog continues from step 1.

If the user enters s then the current DCE account is not processed any further, the next DCE account (if any) is selected, and the dialog continues from step 1.

If the user enters e, then the EXPORT utility terminates.

2.

```
Enter OpenVMS account details:
OpenVMS Username [SMITH]:
```

Default: DCE registry principal name, forced to uppercase, truncated to 12 characters.

Sticky Input: Intra-Account

Valid Responses: Any string up to 12 characters

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

The user either enters a different OpenVMS username for the account or accepts the default.

The EXPORT utility checks if the account already exists. If the account exists, an error message is displayed and the dialog continues from step 1.

3.

```
This OpenVMS account is to be based upon [DEFAULT]:
```

Default: DEFAULT

Sticky Input: Inter-Account

Valid Responses: Any string up to 12 characters

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

4.

```
OpenVMS Password (null means no valid password) []:
```

Default: Null string

Sticky Input: No

Valid Responses: Any string, including null string

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

The response is not echoed as the user enters it. If a null string is entered, EXPORT does not set a valid password on the OpenVMS account and the account user is only able to log in via his or her DCE password.

5.

```
Retype password:
```

Default: No default

Sticky Input: No

Valid Responses: Any string, including null string

Case-Sensitive: No

Invalid Response causes question to be re-asked: NO

User reenters the password for verification. If the verification check fails then an error message is displayed and the dialog continues from step 4.

This step is skipped if a password was not entered in step 4.

6.

```
OpenVMS account string [ETG]:
```

Default: DCE account's group name

Sticky Input: Intra-Account

Valid Responses: 1 to 8 alphanumeric characters, or null string

Case-Sensitive: Yes, if quoted

Invalid Response causes question to be re-asked: Yes

A null string means do not add an account field.

7.

```
Enter UIC group (octal number or existing identifier) [ETG]:
```

Default: OpenVMS account's ACCOUNT field. If null, the DCE account's GROUP name. Uppercased.

Sticky Input: Intra-Account

Valid Responses: Octal number in range 1 to 37776, or string up to 31 chars maximum

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

If a string is entered but it is not an existing group identifier, an error message is issued and the question is asked again.

8.

```
Enter UIC member (octal number) [22]:
```

Default: Next available UIC member within the selected group

Sticky Input: No

Valid Responses: Octal number in range 0 to 177776

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

9.

```
Create UIC identifiers if they don't already exist (y/n) [y]:
```

Default: y

Sticky Input: Inter-Account

Valid Responses: y n

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

10.

```
Account Owner ["John Smith"]:
```

Default: DCE account principal's full name if it exists, otherwise DCE account principal's name, truncated to 8 chars

Sticky Input: No

Valid Responses: ASCII string, up to 8 chars

Case-Sensitive: No, unless quoted

Invalid Response causes question to be re-asked: Yes

11.

```
Default Device [USER$:]
```

Default: Default device from the /LIKE account

Sticky Input: Intra-Account

Valid Responses: ASCII string, up to 32 chars

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

12.

```
Default Directory [SMITH]:
```

Default: The account's username

Sticky Input: Intra-Account

Valid Responses: ASCII string, up to 32 chars

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

13.

```
Disk quota (if quotas are enabled) [1000]:
```

Default: 1000

Sticky Input: Inter-Account

Valid Responses: Any positive integer

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

14.

```
OK to create OpenVMS account based on above (y/n) [y]:
```

Default: y

Sticky Input: No

Valid Responses: y n

Case-Sensitive: No

Invalid Response causes question to be re-asked: Yes

If /NOCONFIRM was specified, then this question is not asked.

If the /RECAP qualifier was specified, details of the account about to be created are displayed immediately before this question is asked.

If the user answers n then an `account not created` message is displayed and the dialog starts again, for the same DCE account, from step 1.

If the user answers y, or if /NOCONFIRM was specified, then an attempt is made to create the OpenVMS account. If the account create succeeds, then a success message is displayed and the dialog starts for the next DCE account from step 1. If the OpenVMS account create fails, then an error message is displayed and the dialog starts again, for the same DCE account, from step 1.

Following is an example of an interactive EXPORT command:

```
EXPORT> EXPORT "smith"

DCE Account Details:

 Principal:     smith (John Smith)
 Group:         DCE
 Organization:  OpenVMS

 c - create OpenVMS account
 x - add this DCE account to the EXPORT exclude file
 s - skip this DCE account
 e - exit IMPORT

 Enter option (c/x/s/e) [c]: c

 Enter OpenVMS account details:

 OpenVMS Username [SMITH]:
 This OpenVMS account is to be based upon [DEFAULT]:
 OpenVMS Password (null means no valid password) []:
 Retype password:
 OpenVMS account string [ETG]:
 Enter UIC group (octal number or existing identifier) [ETG]:
 Enter UIC member (octal number) [22]:
 Create UIC identifiers if they don't already exist (y/n) [y]:
 Account Owner ["John Smith"]:
 Default Device [USER$:]
 Default Directory [SMITH]:
 Disk quota (if quotas are enabled) [1000]:

 OK to create OpenVMS account based on above (y/n) [y]:

 OpenVMS Account successfully created.

EXPORT>
```

SHOW/EXCLUDE

Displays DCE principal names in the EXPORT exclude list.

Synopsis

SHOW/EXCLUDE [PRINCIPAL] */ALL /OUTPUT=output*

Parameters

PRINCIPAL
Specifies the DCE principal name to be displayed from the EXPORT exclude list. Full OpenVMS wildcarding is allowed.

Qualifiers

/ALL
Specifies that all EXPORT exclude entries are to be displayed. If you do not specify a principal name, then this qualifier is assumed.

/OUTPUT=output
Specifies the location at which the output is written. The default is SYS$OUTPUT:.

Description

The SHOW/EXCLUDE command displays DCE principal names in the EXPORT exclude list.

4
Integrated Login Status Messages

This chapter provides Integrated Login status messages. You can receive Integrated Login messages from the Integrated Login procedure or utilities. The prefix of the message denotes the source, as follows:

The Integrated Login procedure messages (IL_messagetext)
The DCE User Authorization (DCE$UAF) utility (UAF_messagetext)
DCE IMPORT utility messages (IMP_messagetext)
DCE EXPORT utility messages (EXP_messagetext)

The messages are listed in alphabetical order.

4.1 Integrated Login Procedure Messages

IL_DCECERT, certified DCE login for <USERNAME> as principal "<PRINCIPAL>"

Explanation: The user successfully logged in to OpenVMS and DCE. The credentials are certified.

User Action: None.

IL_DCENOCERT, noncertified DCE login for <USERNAME> as principal "<PRINCIPAL>"

Explanation: User successfully logged in to OpenVMS and DCE. The credentials are not certified.

User Action: None.

IL_DCEPWDEXP, your DCE password has expired and must be reset

Explanation: The DCE password has expired.

User Action: Change your DCE password by entering the CHPASS command.

IL_ERRVMSPWD, error synchronizing OpenVMS password with DCE password

Explanation: The user logged in specifying a DCE password. The user's DCE password and OpenVMS password are different. The OpenVMS password was not sucessfully synchronized to match the DCE password because an error occurred during synchronization.

User Action: Set the DCE password to a value that is a valid OpenVMS password.

IL_INVPWDLEN, password must be between 'number' and 32 characters

Explanation: Your DCE password could not be set to the same value as your OpenVMS password because its length was invalid.

User Action: Set your DCE password to a value that is the length range of the error message.

IL_NOCREDMOD, unable to modify owner field of credential files

Explanation: The credential files could not be set up correctly. The user will not see this message.

User Action: Submit a Problem Tracking and Reporting (PTR).

IL_NONETCRED, network credentials not obtained

Explanation: A problem in the network prevented Integrated Login from obtaining information about the principal from the DCE registry; therefore, network credentials were not given.

User Action: Try logging in later with DCE_LOGIN. If you still receive this error, ask your cell administrator to troubleshoot the network.

IL_RGYNOTTHERE, unable to access DCE registry

Explanation: The DCE registry was not available when you logged in. You do not have DCE credentials; you are logged in to OpenVMS only.

User Action: Ask your system manager to correct the problem.

IL_VMSONLY, DCE login as principal "<PRINCIPAL>" failed, OpenVMS login to <USERNAME> successful

Explanation: The DCE login failed but the OpenVMS login was successful. This occurs when the user's DCE and OpenVMS passwords are different and the user specifies the OpenVMS password at the password prompt.

User Action: No action is required if you want to be logged in to OpenVMS only. If you want to be logged in to DCE as well as OpenVMS, perform a manual DCE login using the DCE_LOGIN command or log out and then log in specifying your DCE password at the password prompt.

IL_VMSPWDSYNC, OpenVMS password synchronized with DCE password

Explanation: The user logged in specifying a DCE password. The user's DCE password and OpenVMS password are different. The OpenVMS password was successfully synchronized to match the DCE password.

User Action: None.

4.2 IMPORT Utility Messages

IMP_ACCEXISTS, account for <PRINCIPAL> already exists in DCE

Explanation: An attempt has been made to recreate an account for <PRINCIPAL> in the DCE registry.

User Action: None. This is a warning indicating that this suboperation in the IMPORT operation was previously performed.

IMP_ADDDCE, username <USERNAME> successfully imported into DCE

Explanation: A DCE account has been successfully created for OpenVMS username <USERNAME>.

User Action: None.

IMP_ADDDCEACC, account for <PRINCIPAL> successfully added to DCE

Explanation: A DCE account was successfully created for <PRINCIPAL>.

User Action: None. This is an informational message displayed only if /INFORM is specified on the DCE IMPORT command line.

IMP_ADDDCEPRN, principal <PRINCIPAL> successfully added to DCE

Explanation: Principal <PRINCIPAL> record successfully created in the DCE registry.

User Action: None. This is an informational message displayed only when /INFORM is specified on the DCE IMPORT command line.

IMP_ADDDCEUAF, username <USERNAME> successfully added to DCE$UAF

Explanation: Username <USERNAME> successfully added to the DCE$UAF file.

User Action: None. This is an informational message displayed only if /INFORM is specified on the DCE IMPORT command line.

IMP_BINDERR, error binding to DCE security registry

Explanation: Unable to bind to the DCE security server.

User Action: Note accompanying DCE error message for more details.

IMP_CREDCEUAF, created new DCE$UAF file

Explanation: A new DCE$UAF file was created.

User Action: None.

IMP_DCEERR, <DCE ERROR MESSAGE>

Explanation: Accompanying DCE error message supplied with other DCE IMPORT error messages.

User Action: Use this message to determine the cause of the problem.

IMP_DCELOGIN, error in DCE login

Explanation: An error occurred during DCE login.

User Action: Enter a valid DCE username and password when prompted by DCE IMPORT.

IMP_DCEUAFERR, error searching DCE$UAF

Explanation: An error occurred while searching the DCE$UAF file.

User Action: Note the accompanying error message for more details.

IMP_DELACC, account for principal <PRINCIPAL> deleted from DCE

Explanation: DCE account for <PRINCIPAL> was deleted from the DCE registry. This occurs when an atomic IMPORT operation fails during one of its suboperations. Such failure prompts a backout of all suboperations successfully performed during this IMPORT operation. Deleting the account is one such backout operation.

User Action: None. This is an informational message displayed only when /INFORM is specified on the DCE IMPORT command line.

IMP_DELDCEUAF, username <USERNAME> successfully deleted from DCE$UAF

Explanation: Username <USERNAME> deleted from DCE$UAF file.

User Action: None. This is an informational message displayed only if /INFORM is specified on the DCE IMPORT command line.

IMP_DELFRGRP, principal <PRINCIPAL> from group <GROUP>

Explanation: Principal <PRINCIPAL> was deleted from <GROUP> in the DCE registry. This occurs when an atomic IMPORT operation fails during one of its suboperations. Such failure prompts a backout of all suboperations successfully performed during this IMPORT operation. Deleting the principal from the group is one such backout operation.

User Action: None. This is an informational message displayed only when /INFORM is specified on the DCE IMPORT command line.

IMP_DELFRORG, principal <PRINCIPAL> deleted from organization <ORGANIZATION>

Explanation: Principal <PRINCIPAL> was deleted from <ORGANIZATION> in the DCE registry. This occurs when an atomic IMPORT operation fails during one of its suboperations. Such failure prompts a backout of all suboperations successfully performed during this IMPORT operation. Deleting the principal from the organization is one such backout operation.

User Action: None. This is an informational message displayed only when /INFORM is specified on the DCE IMPORT command line.

IMP_DELPRN, principal <PRINCIPAL> deleted from DCE

Explanation: Principal <PRINCIPAL> was deleted from the DCE registry. This occurs when an atomic IMPORT operation fails during one of its suboperations. Such failure prompts a backout of all suboperations successfully performed during this IMPORT operation. Deleting the principal is one such backout operation.

User Action: None. This is an informational message displayed only when /INFORM is specified on the DCE IMPORT command line.

IMP_ERRACCEXC, error accessing DCE IMPORT exclude file

Explanation: Could not access the DCE IMPORT exclude file.

User Action: Note the accompanying error message for more details.

IMP_ERRADDEXC, adding username to DCE IMPORT exclude file

Explanation: Could not add the requested username to the DCE IMPORT exclude file.

User Action: Note the accompanying error message for more details.

IMP_ERRADDGRP, error adding principal <PRINCIPAL> to group <GROUP>

Explanation: Could not add <PRINCIPAL> to <GROUP> in the DCE registry.

User Action: Note the accompanying DCE error message for more details.

IMP_ERRADDORG, error adding principal <PRINCIPAL> to organization <ORGANIZATION>

Explanation: Could not add <PRINCIPAL> to <ORGANIZATION> in DCE registry.

User Action: Note the accompanying DCE error message for more details.

IMP_ERRADDUAF, error adding username to DCE$UAF file

Explanation: Could not add the imported username to the DCE$UAF file.

User Action: Note the accompanying error message for more details.

IMP_ERRCHGAUT, error changing account authorization policy

Explanation: Could not change the DCE account's authorization policy.

User Action: Note the accompanying DCE error message for more details.

IMP_ERRCRACC, error creating account for <PRINCIPAL>

Explanation: Could not create a DCE account for <PRINCIPAL> .

User Action: Note the accompanying DCE error message for more details.

IMP_ERRCRDCEUAF, error creating DCE authorization file

Explanation: An error occurred while attempting to create the DCE$UAF file.

User Action: Note the accompanying error message for more details.

IMP_ERRCRPRN, error creating principal <PRINCIPAL>

Explanation: Could not create a principal in the DCE registry.

User Action: Note the accompanying DCE error message for more details.

IMP_ERRDCEUAF, error accessing DCE authorization file <GROUP>

Explanation: An error occurred while attempting to access the DCE$UAF file.

User Action: Note the accompanying error message for more details.

IMP_ERRDELACC, error deleting account for <PRINCIPAL>

Explanation: Unable to delete account for <PRINCIPAL> from DCE registry.

User Action: See accompanying DCE error message for more details.

IMP_ERRDELEXC, error deleting username from DCE IMPORT exclude file

Explanation: Could not remove requested username from the DCE IMPORT exclude file.

User Action: Note the accompanying error message for more details.

IMP_ERRDELFRGRP, error deleting principal <PRINCIPAL> from group <GROUP>

Explanation: An error occurred while deleting <PRINCIPAL> from <GROUP> in the DCE registry. This delete operation is performed if the overall IMPORT operation failed and a backout of changes applied to the DCE registry is required.

User Action: Note the accompanying error message for more details.

IMP_ERRDELFRORG, error deleting principal <PRINCIPAL> from organization <ORGANIZATION>

Explanation: An error occurred while deleting <PRINCIPAL> from <ORGANIZATION> in the DCE registry. This delete operation is performed if the overall IMPORT operation failed and a backout of changes applied to the DCE registry is required.

User Action: Note the accompanying error message for more details.

IMP_ERRDELPRN, error deleting principal <PRINCIPAL>

Explanation: Unable to delete <PRINCIPAL> from DCE registry

User Action: See accompanying DCE error message for more details.

IMP_ERRDELUAF, error deleting username from DCE$UAF file

Explanation: Could not delete a username from the DCE$UAF file.

User Action: Note the accompanying error message for more details.

IMP_ERRSPAWN, error spawning subprocess

Explanation: An error occurred while spawning a subprocess on the SPAWN command.

User Action: Refer to appropriate OpenVMS documentation for resolution.

IMP_ERRSYSUAF, error accessing SYSUAF file

Explanation: Could not access the OpenVMS SYSUAF file.

User Action: See accompanying OpenVMS or RMS error message for more details.

IMP_EXCADD, username <USERNAME> added to DCE IMPORT exclude list

Explanation: Username <USERNAME> successfully added to the DCE IMPORT exclude file. A DCE account will not be created for this username.

User Action: None.

IMP_EXCDEL, username <USERNAME> removed from DCE IMPORT exclude list

Explanation: Username <USERNAME> successfully removed from DCE IMPORT exclude file. A subsequent IMPORT session could be used to create a DCE account for this username.

User Action: None.

IMP_EXCLUDED, username <USERNAME> has been excluded from DCE

Explanation: Username <USERNAME> cannot be imported since it has been excluded from the DCE registry.

User Action: None. This is an informational message displayed when /INFORM is specified on the DCE IMPORT command line.

IMP_INDCE, username <USERNAME> already imported into DCE

Explanation: An import operation was attempted on an already imported OpenVMS username.

User Action: None. This is an informational message displayed only when /INFORM is specified on the DCE IMPORT command line.

IMP_INDCEUAF, user <USERNAME> already in DCE$UAF

Explanation: Username <USERNAME> already exists in the DCE$UAF.DAT file.

User Action: None. This is a warning indicating that this suboperation in the IMPORT operation was previously performed.

IMP_INEXCLUDE, username <USERNAME> already in DCE IMPORT exclude file

Explanation: Username <USERNAME> has previously been added to the DCE IMPORT exclude file.

User Action: None. This informational message is displayed when an exclude operation is attempted on an already excluded username and is displayed only when /INFORM is specified on the DCE IMPORT command line.

IMP_INITERROR, initialization error

Explanation: An error occurred during DCE IMPORT's initialization phase.

User Action: Note error messages accompanying or directly preceding this message.

IMP_INITWAIT, initializing.....

Explanation: DCE IMPORT is in initialization mode.

User Action: None.

IMP_INVDATETM, invalid date/time

Explanation: Date/time entered has invalid format.

User Action: Enter date/time in standard format (dd-MMM-yyyy hh:mm:ss).

IMP_INTINPDEV, internal error opening input device

Explanation: Error opening SYS$INPUT.

User Action: Verify user run-time environment. See the appropriate OpenVMS documentation for more details.

IMP_INPREQ, input required!

Explanation: Input not entered where input was mandatory.

User Action: Provide required input.

IMP_INTERROR, internal error

Explanation: DCE IMPORT internal error occurred.

User Action: Contact your support engineer or submit a Problem Tracking and Reporting (PTR).

IMP_INVPASSWD, password validation failed. Please retry

Explanation: The password entered when prompted for a retype does not match the originally entered password.

User Action: Enter correct password for original and retype entry.

IMP_NODCEUAF, unable to open DCE authorization file

Explanation: Error occurred while attempting to open the DCE$UAF file.

User Action: See accompanying message for details.

IMP_NOEXCUSR, no excluded users

Explanation: No users listed in DCE IMPORT exclude file.

User Action: None.

IMP_NOPRIN, principal <PRINCIPAL> does not exist in DCE Registry

Explanation: Principal <PRINCIPAL> does not exist in the DCE Registry. This means that <PRINCIPAL> does not have a corresponding OpenVMS username /account.

User Action: None.

IMP_NOSCHPRM, corresponding primary principal not found in DCE

Explanation: The DCE principal name specified as the primary principal while attempting to create an alias principal name is nonexistent in the DCE registry.

User Action: Use the correct DCE principal name. Use the DCE tool RGY_EDIT to view the DCE registry.

IMP_NOSCHUSR, OpenVMS username <USERNAME> does not exist on this system

Explanation: An attempt was made to import a nonexistent OpenVMS user.

User Action: Choose a valid OpenVMS user.

IMP_NOSUCHEXC, no such username in exclude file

Explanation: Username specified does not exist in DCE IMPORT's exclude file.

User Action: Specify username that exists in DCE IMPORT's exclude file. Enter command SHOW/EXCLUDE to display the entire exclude list.

IMP_NOSUCHGRP, no group <GROUP>. Please choose a valid group

Explanation: The group name specified is nonexistent in the DCE registry.

User Action: Choose a valid group name. Use the DCE tool RGY_EDIT to search the DCE registry for group names.

IMP_NOSUCHORG, no organization <ORGANIZATION>. Please choose a valid organization

Explanation: The organization name specified is nonexistent in the DCE registry.

User Action: Choose a valid organization name. Use the DCE tool RGY_EDIT to search the DCE registry for organization names.

IMP_OUTOPNERR, error opening alternate output

Explanation: Could not access output medium.

User Action: If /OUTPUT was specified, verify the file name supplied with /OUTPUT. If /OUTPUT was not specified, check user run-time environment. See appropriate OpenVMS documentation for more details.

IMP_PREXISTS, principal <PRINCIPAL> already exists in DCE

Explanation: An attempt has been made to add <PRINCIPAL> to the DCE registry.

User Action: None. This is a warning indicating that this suboperation in the IMPORT operation was previously performed.

IMP_PRINGRP, principal <PRINCIPAL> already exists in group <GROUP>

Explanation: An attempt was made to add <PRINCIPAL> to DCE group <GROUP> when it already was a member of the group. This action was attempted during an import operation.

User Action: None. This is an informational message displayed only when /INFORM is specified on the DCE IMPORT command line.

IMP_PRINORG, principal <PRINCIPAL> already exists in organization <ORGANIZATION>

Explanation: An attempt was made to add <PRINCIPAL> to DCE organization <ORGANIZATION> when it was already a member of that organization. This action was attempted during an import operation.

User Action: None. This is an informational message displayed only when /INFORM is specified on the DCE IMPORT command line.

IMP_PRINUSE, principal <PRINCIPAL> in use by another OpenVMS username

Explanation: The DCE principal name specified for the OpenVMS username being imported is associated with another OpenVMS username.

User Action: Choose a DCE principal name that is not associated with any OpenVMS username.

IMP_RANGEERR, error in entry! Number must be between 1 and 65535

Explanation: The value entered for quota is not within the desired range.

User Action: Enter a number between 1 and 65535.

IMP_TIMERR, DCE time configuration error

Explanation: Time configuration incorrect on the DCE system.

User Action: Refer to the Troubleshooting chapter in the *HP DCE for OpenVMS Alpha and OpenVMS I64 Product Guide.*

IMP_TOOLONG, input for <QUALIFIER> too long

Explanation: Value of <QUALIFIER> is longer than expected maximum size of value.

User Action: Enter a value that is within the valid size range.

IMP_USERERR, error getting input from user

Explanation: Error occurred while getting user input.

User Action: Provide valid input.

4.3 EXPORT Utility Messages

EXP_ACCEXISTS, OpenVMS account for <PRINCIPAL> already exists

Explanation: Could not export <PRINCIPAL> because it has already been exported.

User Action: None.

EXP_ADDDCEACC, account for <PRINCIPAL> successfully added to OpenVMS

Explanation: An OpenVMS account was successfully created for <PRINCIPAL>.

User Action: Note directly preceding and following messages for warnings.

EXP_ADDDCEUAF, principal <PRINCIPAL> successfully added to DCE$UAF

Explanation: Principal <PRINCIPAL> successfully added to the DCE$UAF file as part of the EXPORT operation. Message displayed only if /INFORM is specified on the EXPORT command line.

User Action: None.

EXP_ADDUAF, principal <PRINCIPAL> successfully exported to OpenVMS

Explanation: An OpenVMS account was successfully created for DCE <PRINCIPAL>.

User Action: Note directly preceding and following messages for warnings.

EXP_BINDERR, error binding to DCE security registry

Explanation: Cannot connect to the DCE security server.

User Action: Note accompanying error message for more details.

EXP_CREDCEUAF, created new DCE$UAF file

Explanation: A new DCE$UAF file was created.

User Action: None.

EXP_DCEERR, <DCE ERROR MESSAGE>

Explanation: This is the accompanying DCE error message.

User Action: Use this message to solve the problem generating the error.

EXP_DCELOGIN, error in DCE login

Explanation: Could not perform a DCE login.

User Action: Enter valid DCE principal and password combination.

EXP_DCEUAFERR, error searching DCE$UAF

Explanation: Error searching or reading DCE$UAF file.

User Action: Note accompanying error message for more details.

EXP_DELDCEUAF, principal <PRINCIPAL> successfully deleted from DCE$UAF

Explanation: Principal <PRINCIPAL> successfully deleted from DCE$UAF as part of larger delete operation. Message is displayed only if /INFORM is specified on the EXPORT command line.

User Action: None.

EXP_DISUSER, <USERNAME> may still be DISUSER-ed

Explanation: OpenVMS account for <USERNAME> was successfully created but could not enable the account.

User Action: Manually remove the DISUSER flag using the AUTHORIZE utility.

EXP_ERRACCEXC, error accessing DCE EXPORT exclude file

Explanation: Could not access DCE EXPORT exclude file.

User Action: Note accompanying error message for more details.

EXP_ERRADDEXC, error adding principal to DCE EXPORT exclude file

Explanation: Could not add principal to DCE EXPORT exclude file.

User Action: Note accompanying error message for more details.

EXP_ERRADDUAF, error adding principal to DCE$UAF file

Explanation: Could not add principal name to DCE$UAF file.

User Action: Note accompanying error message for more details.

EXP_ERRCRACC, error creating OpenVMS account for <USERNAME>

Explanation: Could not create an OpenVMS account for <USERNAME> .

User Action: See accompanying error message for more details.

EXP_ERRCRDCEUAF, error creating DCE authorization file

Explanation: An error occurred while attempting to create the DCE$UAF file.

User Action: Note accompanying error message for more details.

EXP_ERRCREUAF, error creating OpenVMS account for <USERNAME>— see following messages

Explanation: Could not create the OpenVMS account for <USERNAME>.

User Action: Note accompanying error messages for more details.

EXP_ERRDCEUAF, error accessing DCE authorization file

Explanation: An error occurred while attempting to access the DCE$UAF file.

User Action: Note accompanying error message for more details.

EXP_ERRDELEXC, error deleting principal from DCE EXPORT exclude file

Explanation: Could not delete principal from DCE EXPORT exclude file.

User Action: Note accompanying error message for more details.

EXP_ERRDELUAF, error deleting principal from DCE$UAF file

Explanation: Could not delete principal from DCE$UAF file.

User Action: Note accompanying error message for more details.

EXP_ERRENAUSR, error enabling user <USERNAME>

Explanation: Could not remove DISUSER flag from <USERNAME>'s account.

User Action: Manually remove the flag using the AUTHORIZE utility.

EXP_ERRQUOTA, error assigning disk quota to username <USERNAME> — see following messages

Explanation: Error(s) occurred while attempting to set up disk quota for <USERNAME>.

User Action: Note the messages following this message.

EXP_ERRSETPW, error setting password for <USERNAME>

Explanation: Could not set password for OpenVMS <USERNAME>.

User Action: Manually set password using the AUTHORIZE utility.

EXP_ERRSPAWN, error spawning subprocess

Explanation: Error spawning subprocess with the SPAWN command.

User Action: Check user run-time configuration. Refer to appropriate OpenVMS documentation for more details.

EXP_ERRSYSUAF, error accessing SYSUAF file

Explanation: Could not access the SYSUAF file.

User Action: Note accompanying error message for more details.

EXP_ERRUAFGET, error getting SYSUAF information

Explanation: Error accessing information in the SYSUAF file.

User Action: Note accompanying error message for more information.

EXP_EXCADD, principal <PRINCIPAL> added to DCE EXPORT exclude list

Explanation: Principal <PRINCIPAL> successfully added to the DCE EXPORT exclude list.

User Action: None.

EXP_EXCDEL, principal <PRINCIPAL> removed from DCE EXPORT exclude list

Explanation: Principal <PRINCIPAL> successfully deleted from the DCE EXPORT exclude list.

User Action: None.

EXP_EXCLUDED, principal <PRINCIPAL> has been excluded from OpenVMS

Explanation: Unable to export <PRINCIPAL> because it is on the DCE EXPORT exclude list. This message is displayed only if /INFORM is specified on the EXPORT command line.

User Action: If incorrectly excluded, use DELETE/EXCLUDE to remove it from the DCE EXPORT exclude list and reexport.

EXP_GRPUICFULL, no member UIC available in specified group

Explanation: No more members available in the specified group.

User Action: Use another group UIC if possible.

EXP_INDCEUAF, principal <PRINCIPAL> already in DCE$UAF

Explanation: Could not add already existing principal name to DCE$UAF.

User Action: None.

EXP_INEXCLUDE, principal <PRINCIPAL> already in DCE EXPORT exclude file

Explanation: An attempt was made to add an already existing principal name to the DCE EXPORT exclude file.

User Action: None.

EXP_INITERROR, initialization error

Explanation: Error during initialization phase for DCE EXPORT.

User Action: Note accompanying error message for more details.

EXP_INITWAIT, initializing.....

Explanation: DCE EXPORT in initialization phase.

User Action: None.

EXP_INPREQ, input required!

Explanation: Input not entered where mandatory.

User Action: Provide input.

EXP_INTERROR, internal error

Explanation: Internal error in DCE EXPORT.

User Action: Note accompanying error message for more details or submit a Problem Tracking and Reporting (PTR).

EXP_INTINPDEV, internal error opening input device

Explanation: Error accessing SYS$INPUT.

User Action: Check user run-time configuration. Refer to appropriate OpenVMS documentation for more information.

EXP_INVGRPUIC, invalid group UIC

Explanation: Group UIC entered is invalid (format if value, name if identifier).

User Action: Enter valid group UIC.

EXP_INVMEMUIC, invalid member UIC

Explanation: Member UIC entered is out of range or of invalid format.

User Action: Enter valid member UIC.

EXP_INVMS, principal <PRINCIPAL> already exported to OpenVMS

Explanation: A record for <PRINCIPAL> already exists in the DCE$UAF file indicating that is has already been exported.

User Action: None.

EXP_INVPASSWD, password validation failed. Please retry

Explanation: Password validation failed while entering password for the OpenVMS account to be created.

User Action: Enter valid password.

EXP_INVPWDLEN, password length must be between <MINIMUM> and <MAXIMUM> characters

Explanation: The user-specified password for the OpenVMS account is outside of the defined range.

User Action: Respecify password of valid length.

EXP_NAMEINUSE, OpenVMS username <USERNAME> already mapped to another DCE principal

Explanation: OpenVMS username specified is already associated with another DCE principal in the DCE$UAF file.

User Action: Specify a username that is not associated with a DCE principal. Use the DCE$UAF utility to search the DCE$UAF file for usernames and associated DCE principal names.

EXP_NODCEUAF, unable to open DCE authorization file

Explanation: Error occurred while attempting to open the DCE$UAF file.

User Action: Note accompanying error message for more details.

EXP_NOEXCUSR, no excluded users

Explanation: No principal names listed in the DCE EXPORT exclude file.

User Action: None.

EXP_NOSCHUSR, no principal <PRINCIPAL> in DCE registry

Explanation: Principal <PRINCIPAL> requested for export does not exist in the DCE registry.

User Action: Use valid DCE principal name. Use the DCE tool RGY_EDIT to view DCE principal names.

EXP_NOSUCHEXC, no such principal in DCE EXPORT exclude file

Explanation: Requested principal does not exist in DCE EXPORT exclude file.

User Action: Use the SHOW/EXCLUDE command to list names in the exclude file.

EXP_NOSUCHPR, no DCE account <PRINCIPAL>

Explanation: An attempt was made to export a nonexistent DCE principal.

User Action: Specify a valid DCE principal name. Use the DCE tool RGY_EDIT to view the DCE principals.

EXP_NOTINEXC, principal <PRINCIPAL> not in DCE EXPORT exclude file

Explanation: An attempt was made to access a nonexistent record in the DCE EXPORT file.

User Action: Use SHOW/EXCLUDE to see the contents of the exclude file.

EXP_NOVMSUSR, no OpenVMS user <USERNAME>

Explanation: A nonexistent OpenVMS username was specified with the /LIKE qualifier.

User Action: Specify a valid OpenVMS username.

EXP_NXTMEMUIC, error finding next available member UIC

Explanation: Could not find the next available member UIC in the group specified.

User Action: Note the accompanying error message for more details.

EXP_OUTOPNERR, error opening alternate output

Explanation: Could not access file name specified with /OUTPUT qualifier.

User Action: Note accompanying error message for more details.

EXP_SEEFILE, see file <FILE NAME> for error messages

Explanation: Error(s) occurred while creating the OpenVMS account but EXPORT was unable to display the error messages. The user is asked to read the file <FILE NAME> for the error messages.

User Action: Read the file <FILE NAME> for error messages.

EXP_TIMERR, DCE time configuration error

Explanation: Time configuration incorrect on the DCE system.

User Action: Refer to the troubleshooting chapter in the *HP DCE for OpenVMS Alpha and OpenVMS I64 Product Guide.*

EXP_TOOLONG, input for <QUALIFIER> too long

Explanation: Value of <QUALIFIER> is longer than expected maximum size of value.

User Action: Enter a value that is within the valid size range.

EXP_USERERR, error getting input from user

Explanation: User entered invalid input.

User Action: Enter valid input.

4.4 DCE$UAF Utility Messages

UAF_ADDED, created entry <USERNAME>, principal is "<PRINCIPAL>"

Explanation: The ADD command completed successfully.

User Action: None.

UAF_ANOTADDED, error creating entry <USERNAME>

Explanation: The ADD command could not be performed. The accompanying error messages explain the reason.

User Action: Make sure the command is correct, or take action based on an accompanying message of the following form:

```
SEVERITY ERROR
BASE 3800
```

UAF_CONNREG, connecting to registry "<REGISTRY>"

Explanation: UAF_CONNREG, connecting to registry "<REGISTRY>" Indicates that the DCEUAF utility is establishing a connection to the specified DCE registry. (A connection to the DCE registry is made only for a VERIFY command.)

User Action: None.

UAF_CREDCEUAF, created new DCE$UAF file

Explanation: A new DCE$UAF file has been successfully created.

User Action: None.

UAF_DCECHKBEG, starting scan of DCE$UAF file

Explanation: The ANALYZE command is starting the scan of the DCE$UAF file.

User Action: None.

UAF_DCECHKEND, completed scan of DCE$UAF file

Explanation: The ANALYZE command has completed the scan of the DCE$UAF file.

User Action: None.

UAF_DCEERR, <TEXT>

Explanation: Secondary message that is displayed after another message and that provides more information about the problem.

User Action: Refer to the User Action for the main message.

UAF_DCENAMEREQ, DCE name required for this function

Explanation: While modifying an existing DCE$UAF record, the "NO_DCE" attribute was cleared, but a DCE principal name was not specified.

User Action: Enforce the "NO_DCE" attribute, or specify a DCE principal name.

UAF_DELETED, deleted entry <USERNAME>

Explanation: The DELETE command completed successfully.

User Action: None.

UAF_ERRATTACH, error attaching to process

Explanation: The user issued an ATTACH command with no parameters or qualifiers, which means attach to parent. The current process is not a subprocess so there is no parent to which to attach.

User Action: Use the SPAWN command to access the DCL prompt without exiting DCE$UAF.

UAF_ERRCLSOUT, error closing /OUTPUT file

Explanation: The RMS file system returned an error when attempting to close the output file specified by the /OUTPUT qualifier.

User Action: Take action based on the accompanying RMS message.

UAF_ERRCREUAFL, error creating DCE authorization file

Explanation: The attempt to create a new DCE$UAF file failed.

User Action: Take action based on the accompanying RMS message.

UAF_ERRDCEUAF, error accessing DCE authorization file

Explanation: The DCE$UAF file could not be accessed.

User Action: Take action based on the accompanying RMS message.

UAF_ERRDEFKEY, error defining key

Explanation: User issued a DEFINE/KEY command which could not be performed.

User Action: Take action based on the accompanying error message.

UAF_ERRDKEY, error deleting key <KEY> definition

Explanation: User issued a DELETE/KEY command which could not be performed.

User Action: Take action based on the accompanying error message.

UAF_ERROPNCMF, error opening command file <FILENAME>

Explanation: The command file specified could not be opened.

User Action: Take action based on the accompanying error message.

UAF_ERROPNOUT, error opening /OUTPUT file

Explanation: The RMS file system returned an error when attempting to open the output file specified by the /OUTPUT qualifier.

User Action: Take action based on the accompanying RMS message.

UAF_ERRSPAWN, error spawning subprocess

Explanation: Unable to create the spawned subprocess.

User Action: Check the qualifier values to make sure they are valid. Check whether the process quotas have been exceeded.

UAF_ERRSYSUAF, error accessing SYSTEM authorization file

Explanation: The SYSTEM authorization file could not be read.

User Action: Take action based on the accompanying RMS message.

UAF_ERRWRTOUT, error writing to /OUTPUT file

Explanation: The RMS file system returned an error when attempting to write to the output file specified by the /OUTPUT qualifier.

User Action: Take action based on the accompanying RMS message.

UAF_FILEXISTS, file already exists

Explanation: An attempt was made to create DCE$UAF.DAT, but the file already exists.

User Action: If a new file is desired, delete the old file before attempting to create a new one.

UAF_ILLCELLNAME, cell name lexically invalid

Explanation: A DCE cell name was specified that contained illegal characters or exceeded a length of 1024 characters.

User Action: Specify a name containing any ASCII printable character that is within the length limit of 1024 characters.

UAF_ILLPRINCNAME, principal name lexically invalid

Explanation: A DCE principal name was specified that contained illegal characters or exceeded a length of 1024 characters.

User Action: Specify a name containing any ASCII printable character that is within the length limit of 1024 characters.

UAF_ILLVMSNAME, OpenVMS name lexically invalid

Explanation: An OpenVMS username was specified that contained illegal characters or exceeded a length of 32 characters.

User Action: Specify a name containing only alphanumeric characters, dollar-sign ('$') or underscore ('_') that is within the length limit of 32 characters.

UAF_INTERROR, internal error <NUMBER>, please submit a PTR

Explanation: An error in the software has been detected.

User Action: Submit a Problem Tracking and Reporting (PTR).

UAF_INVALIDCTX, invalid READ_ALL context

Explanation: The context ID exchanged between user applications and the DCE$UAF.DAT file-control library was ill-formed, or superseded by a newer instance of that ID. This message usually indicates a programming error.

User Action: Exit the application and run it again.

UAF_KEYDEFD, key has been successfully defined

Explanation: The DEFINE/KEY command completed successfully.

User Action: None.

UAF_KEYDEL, key <KEY> definition has been deleted

Explanation: The DELETE/KEY command completed successfully.

User Action: None.

UAF_KEYNOTF, key <KEY> definition not found

Explanation: An attempt was made to use the SHOW/KEY command to show the definition of a key that is not defined.

User Action: None.

UAF_MAXDEPEXC, maximum command file depth exceeded

Explanation: Command files can only be nested to a maximum depth of 8.

User Action: Change the structure of your command files so as not to exceed the maximum nested depth of 8.

UAF_MODED, modified entry <USERNAME>

Explanation: The MODIFY command completed successfully.

User Action: None.

UAF_NODCEUAF, unable to open DCE authorization file

Explanation: The DCE$UAF file does not exist. The user will be asked if a new DCE$UAF file should be created.

User Action: If a DCE$UAF file that you want to use exists in another directory, exit the DCE$UAF utility and SET DEFAULT to that directory or define the logical name DCE$UAF to point to the required DCE$UAF file and then restart the DCE$UAF utility.

UAF_NODELETE, unable to delete record

Explanation: A REMOVE command was issued, but the delete operation failed. The accompanying message explains the reason the operation failed.

User Action: Take action based on the accompanying error message.

UAF_NOFREECTX, no free READ_ALL context

Explanation: The DCE$UAF_READ_ALL_INIT function was unable to find a free context to return to the application. This may be a result of an error in the user application with regard to freeing a context when no longer needed.

User Action: Exit the application and run it again.

UAF_NOKEYF, no key definition found

Explanation: A SHOW/KEY * command was issued but there are no keys currently defined.

User Action: None.

UAF_NOMODIFY, unable to modify record

Explanation: The specified MODIFY command could not be completed.

User Action: Take action based on the accompanying error message.

UAF_NOPARENT, there is no parent to which to attach

Explanation: The user issued an ATTACH command with no parameters or qualifiers, which means attach to parent. The current process is not a subprocess so there is no parent to which to attach.

User Action: Use the SPAWN command to access the DCL prompt without exiting DCE$UAF.

UAF_NOSHOW, unable to display record

Explanation: The SHOW command could not be completed.

User Action: Take action based on the accompanying error message.

UAF_NOSUCHUSER, username !AS does not exist in the DCE$UAF

Explanation: The specified OpenVMS username does not exist, so the requested action could not be performed.

User Action: Specify a username that exists in the DCE$UAF. Use the SHOW command to see the entries, if necessary.

UAF_NOSUCHPRIN, principal "!AZ" does not exist in the DCE$UAF

Explanation: The specified DCE principal does not exist, so the requested action could not be performed.

User Action: Specify a principal that exists in the DCE$UAF. Use the SHOW command to see the entries, if necessary.

UAF_NOTADDED, error creating entry <USERNAME>

Explanation: The ADD command failed.

User Action: Take action based on the accompanying error message.

UAF_NOTDELETED, error deleting entry <USERNAME>

Explanation: The DELETE command failed.

User Action: Take action based on the accompanying error message.

UAF_NOTMODED, error modifying entry <USERNAME>

Explanation: The MODIFY command failed.

User Action: Take action based on the accompanying error message.

UAF_NOVERIFY, unable to verify account

Explanation: A VERIFY command could not be completed. The accompanying message explains the reason.

User Action: Take action based on the accompanying error message.

UAF_NOVREC, no version record found in file

Explanation: Each DCE$UAF.DAT file contains a special version record that is added when the file is created and read each time the file is opened. If no version record is found, then the file-control library will abort any attempt to read or write authorization data from or to the file. This message indicates a possible problem in the way that the file was created, or possible damage to the file after it was created.

User Action: Delete the current file and create a new one.

UAF_NYI, this feature is not yet implemented

Explanation: The specified command (or part of the command) is not implemented in this field test version of the software.

User Action: This feature will be implemented in an update to the field test software.

UAF_RECEXISTS, record already exists

Explanation: The ADD command cannot be performed because the specified username or principal already exists in the DCE$UAF file.

User Action: Check the username and principal name and reenter the command if incorrect.

UAF_SYSCHKBEG, starting scan of SYSUAF file

Explanation: The ANALYZE command is starting the scan of the SYSUAF file.

User Action: None.

UAF_SYSCHKEND, completed scan of SYSUAF file

Explanation: The ANALYZE command has completed the scan of the SYSUAF file.

User Action: None.

UAF_VERSIONSKEW, version skew between library and UAF file

Explanation: Each DCE$UAF.DAT file contains a special version record that is added when the file is created and read each time the file is opened. If the version information in the file record is not compatible with the internal version of the file-control library, then this message is returned, and any attempt to read or write authorization data from or to the file will be aborted. The message indicates an incompatibility between the DCE$UAF.DAT file and the file-control library. This can only happen as a result of the installation of newer software on a system with an existing DCE$UAF.DAT file.

User Action: Convert the older-format file to the format required by the new software.

UAF_VERUSERERR, error accessing registry

Explanation: An error occurred while attempting to access the DCE registry. The accompanying text explains the reason. (A connection to the DCE registry is made only for a VERIFY command.)

User Action: Take action based on the accompanying error message.

UAF_VRECFAIL, write of version record failed

Explanation: Each DCE$UAF.DAT file contains a special version record that is added when the file is created. This message indicates that the record could not be written to the file, in which case the newly created file is deleted. This message indicates a possible problem with the file system, the file device, or other low-level problem that is generally not within the user's power to correct.

User Action: If multiple attempts to create the file correctly fail, then further diagnostic work by the system administrator may be necessary to find the source of the problem.

Part II

CDS Subtree Reference

5

CDS Subtree Commands

This chapter contains reference information on the CDS subtree commands that are in the CDS control program (cdscp). The CDS subtree commands are equivalent to their counterparts in the DCE control program (dcecp). The dcecp subtree commands are documented in the Open Group documentation.

delete subtree(8cds)

Deletes a subtree of directories and their contents or an individual directory and its contents.

Synopsis

cdscp delete subtree *tree-name* [norecurse] [exclude *entry-type*]

Parameters

tree-name
The name of the uppermost directory in the subtree you intend to delete. You can use the optional norecurse keyword to restrict the deletion only to the directory (and contents) you specify in *tree-name*.

entry-type
One or more of the following types of entries to exclude from deletion: all objects, soft links, or specific directories. You can exclude multiple entry-types in a single command. Use any combination of the following *entry-type* specifiers, separating multiple arguments with commas. You must leave a blank character space after each comma and after each *directory-name* specification.

objects
links
directory *directory-name*

Description

The delete subtree command deletes a subtree of directories and their contents. You can use the optional norecurse keyword to restrict the deletion to only the directory you specify in *tree-name*. If that directory has child directories (and you use the optional norecurse keyword) the command deletes only the directory's contents.

You can use the optional exclude directory argument to specify one or more directories to exclude from deletion. Specify multiple directories in the following format:

```
exclude directory directory-name, directory
directory-name, directory directory-name
```

Permissions Required

You must have delete permission to all affected directories and their contents. (Delete and administer permission to all affected directories is also sufficient.) You also need write permission to the clearinghouse that stores the master replica of the directory you specify in *tree-name*.

Example

The following command deletes the /.:/admin/site03 directory and its contents as well as all of its child directories and their contents.

```
cdscp>  delete subtree /.:/admin/site03
```

See Also

delete_directory
dump_subtree
merge_file
merge_subtree

dump subtree(8cds)

Dumps a subtree of directories and their contents into an interim file.

Synopsis

cdscp dump subtree *tree-name* [norecurse] [to, into] file *filename* [exclude *entry-type*]

Parameters

tree-name
The name of the uppermost directory in the subtree you intend to dump to the interim file. You can use the optional norecurse keyword to restrict the dump only to the directory (and contents) you specify in *tree-name*.

filename
The name of the interim file to which the subtree is dumped.

entry-type
One or more of the following types of entries to exclude from the dump: all ACLs, object entries, soft links, or specific directories. You can exclude multiple entry types in a single command. Use any combination of the following *entry-type* specifiers, separating multiple arguments with commas. You must leave a blank character space after each comma and after each *directory-name* specification.

acls
objects
links
directory *directory-name*

Description

The dump subtree command dumps a subtree of directories and their contents into an interim file. You can use the optional norecurse keyword to dump only the directory (and contents) you specify in *tree-name*. You can use the optional [exclude] keyword to omit all ACLs, object entries, soft links, or specific directories from the interim file. Use the file extension .dat as a convention for interim filenames.

This command is useful for backing up individual directories or subtrees and can also be used as the first step of a directory merge operation. If you intend to merge a subtree into the namespace of a foreign cell (or a reconfigured cell), use the optional exclude keyword to exclude the ACLs from the interim file.

ACLs that reference the source cell name will have no meaning in the target cell and will convey no access.

Permissions Required

You must have read permission to all affected directories and their contents.

Example

The following command creates an interim file named /.:/admin/site03.dat that contains the /.:/admin/site03 directory, its contents, and all its subdirectories and their contents.

```
cdscp>  dump subtree /.:/admin/site03 into file site03.dat
```

See Also

```
delete_subtree
merge_file
merge_subtree
```

merge file(8cds)

Merges the contents of an interim file (created with the dump subtree command) into an existing subtree.

Synopsis

cdscp merge file *ifile* [to, into] subtree *tree-name* failures [to] file [=] *filename*

Parameters

ifile
The name of an interim file that contains a directory and its contents, or a subtree of directories and their contents.

tree-name
The name of the uppermost directory in the target subtree.

filename
The name of a failures file that contains names that could not be merged.

Description

The merge file command merges the contents of an interim file created with the dump subtree command into an existing subtree whose uppermost directory you specify in *tree-name*. If the target directory *tree-name* does not exist, you must use the create directory or recreate directory command to create the target directory before you proceed.

The existing values of the CDS_Convergence, CDS_InChName, and CDS_ UpgradeTo attributes associated with the directory you specify in *tree-name* are overwritten with the corresponding attribute values associated with the directory that was specified as *tree-name* in the dump subtree command used to create the interim file.

If you did not exclude source subtree ACLs from the interim file, the ACLs of new entries created in the target subtree as a result of the merge will contain the original source subtree ACL entries as well as any ACL entries that may propagate from the new parent directory in the target subtree of the uppermost directory in the interim file. The principal who executes the merge file command is granted full permission to all new entries in the target subtree.

You use the failures to file = *filename* argument to specify the name of a file that will contain the names of any directories, object entries, or soft links (including their ACLs) that could not be merged. You can use this file if you perform subsequent merge file operations to merge failed names.

Permissions Required

You must have read and insert permission to the target directory (and contents) you specify in *tree-name*. You also need write permission to the clearinghouse that stores the master replica of the uppermost directory in the target subtree.

Example

The following command merges the interim file branch01.dat with the /.:/admin/site03 directory:

```
cdscp> merge file branch01.dat into subtree /.:/admin/site03 failures to
file = branch01failed.dat
```

See Also

dump_subtree
merge_subtree

merge subtree(8cds)

Dumps a directory or subtree and its contents into an interim file and then merges the contents of that file into an existing directory.

Synopsis

cdscp merge subtree *old-tree-name* [norecurse] [to, into] subtree *new-tree-name* [exclude *entry-type*]

Parameters

old-tree-name
The name of the uppermost directory in the subtree whose directories and contents you intend to merge. If you use the optional norecurse keyword, you can restrict the dump and merge operation to the particular directory (and contents) you specify in *old-tree-name*.

new-tree-name
The name of the uppermost directory in the target subtree.

entry-type
One or more of the following types of entries to exclude from the dump and merge operation: ACLs, object entries, soft links, or specific directories. You can exclude multiple entries in a single command. Use any combination of the following *entry-type* specifiers, separating multiple arguments with commas. You must leave a blank character space after each comma and after each *directory-name* specification.

acls
objects
links
directory *directory-name*

Description

The merge subtree command combines the operations performed by the dump subtree and merge file commands into a single operation. This command dumps a subtree into an interim file and then merges the contents of the file into another existing subtree. If you use the optional norecurse keyword, the command dumps and merges only the directory you specify in *old-tree-name*. You can use the optional exclude keyword to omit all ACLs, object entries, soft links, and specific directories form the dump and merge operation. The target directory you specify in *new-tree-name* must already exist. If it does not, the command returns an error and you must use the create directory or recreate directory command to create the target directory before you proceed.

This command is especially useful when all clearinghouses are available for every directory in both subtrees, no duplicate names exist in source and target subtrees, and when the permissions required to create entries in the target subtree have already been granted. If a duplicate name is detected, or if any affected clearinghouse cannot be reached while the merge subtree command is in progress, the command completes what it can. No failures file is created.

Permissions Required

You must have read permission to all affected directories and contents in the source subtree (*old-tree-name*). You also need read and insert permission to the target directory (and contents) you specify in *new-tree-name*, and write permission to the clearinghouse that stores the master replica of the target directory.

Example

The following command merges the contents of a subtree beginning with the `/.:/admin/site01` directory with the `/.:/admin/site02` directory.

```
cdscp>  merge subtree /.:/admin/site01 into subtree /.:/admin/site02
```

See Also

`dump_subtree`
`merge_file`

recreate directory(8cds)

Recreates an existing directory (in a source subtree) as a new directory (in a target subtree).

Synopsis

cdscp recreate directory *directory-name* [as] directory *newdirectory-name* [exclude acls]

Parameters

directory-name
The full name of the directory you intend to recreate.

newdirectory-name
The name of the recreated directory.

Description

The recreate directory command recreates an existing directory (in a source subtree) as a new directory (in a target subtree). Only the directory itself is duplicated, not its contents. This command does not delete or modify the source directory.

All writable attribute values of the source directory (CDS_Convergence, CDS_InChName, and CDS_UpgradeTo) are retained in the duplicate. The ACL entries associated with the source directory are also preserved unless you use the optional exclude acls keywords.

If you include a wildcard character in your *directory-name* specification, the name you specify as *newdirectory-name* must already exist. Only directories matching the wildcard are recreated in *newdirectory-name*.

Although all original ACL entries are retained, the duplicate directory also inherits ACL entries that may be propagated from its new parent directory in the target subtree. The principal executing this command is granted full access to the new directory.

The following attribute values are updated in the duplicate and may not match the values of the original directory: CDS_AllUpTo, CDS_CTS, CDS_DirectoryVersion, CDS_InCHName, CDS_ParentPointers, CDS_Replicas, and CDS_UTS.

Permissions Required

You must have read and insert permission to the target directory (*newdirectory-name*) in which you intend to recreate the source directory. You also need write permission to the clearinghouse that stores the master replica of the target directory.

Example

The following command recreates the existing directory /.:/sales/quar1 as a new directory named /.:/mkt/quar1.

```
cdscp>  recreate directory /.:/sales/quar1 as directory /.:/mkt/quar1
```

See Also

merge_file
merge_subtree
recreate_link
recreate_object

recreate link(8cds)

Recreates an existing soft link as a new soft link with a new full name.

Synopsis

cdscp recreate link *link-name* [as] link *newlink-name* [exclude acls]

Parameters

link-name
The full name of the soft link.

newlink-name
The new name of the soft link you are recreating. If you specify the name of an existing directory in the target subtree, the soft link is recreated in that directory with its original simple link name.

Description

The recreate link command creates a copy of the specified soft link with a new full name. All of the soft link's writable attribute values are retained in the duplicate. (CDS_CTS and CDS_UTS attribute values are not preserved.) You can use the optional exclude acls keywords to omit the original ACL entries from the duplicate.

Although all original ACL entries are retained, the new soft link also inherits ACL entries that may be propagated from the new parent directory in the target subtree. The principal executing this command is granted full access to the new soft link.

You can use a trailing wildcard character after the *link-name* argument to recreate multiple soft links. In this case, you must specify a directory in the *newlink-name* argument, and all recreated soft links will have the same simple names as their sources.

Permissions Required

You must have insert permission to the directory in which you intend to recreate the soft link.

Example

The following command recreates all soft links that exist in the /.:/sales directory as new soft links in the /.:/mkt directory.

```
cdscp>  recreate link /.:/sales/* as link /.:/mkt
```

See Also

merge_file
merge_subtree
recreate_directory
recreate_object

recreate object(8cds)

Recreates an existing object entry as a new object entry with a new full name.

Synopsis

cdscp recreate object *object-name* [as] object *newobject-name* [exclude acls]

Parameters

object-name
The full name of the object.

newobject-name
The new name of the copy of the object entry. If you specify the name of an existing directory in the target subtree, the object specified in *object-name* is recreated in that directory with its original simple name.

Description

The recreate object command creates a copy of an object entry with a new full name. All of the object's writable attribute values, including its ACL, are retained in the duplicate. (CDS_CTS and CDS_UTS attribute values are not preserved.) You can use the optional exclude acls keywords to omit the original ACL entries from the duplicate.

Although all original ACL entries are retained, the new object entry also inherits ACL entries that may be propagated from the new parent directory in the target subtree. The principal executing this command is granted full access to the new object entry.

You can use a trailing wildcard character after the *object-name* argument to recreate multiple object entries. In this case, you must specify an existing directory in the *newobject-name* argument and all duplicate object entries will have the same simple names as their sources. The command does not modify or delete the source object entry (or entries).

Permissions Required

You must have insert permission to the directory in which you intend to recreate the object entry.

Example

The following command recreates the existing object entry */.:/eng/psprinter03* as a new object entry named */.:/rnd/printer01*.

```
cdscp>  recreate object /.:/eng/psprinter03 as object /.:/rnd/printer01
```

See Also

merge_file
merge_subtree
recreate_directory
recreate_link

replace link(8cds)

Deletes a specified soft link and replaces it with a new soft link to redirect lookups from the original location to the new location.

Synopsis

cdscp replace link *link-name* [with] link *newtree-name*

Parameters

link-name
The full name of the soft link in its old location.

newtree-name
The full name of the directory into which the soft link has moved.

Description

The replace link command deletes a specified soft link and replaces it with a soft link whose link target is the corresponding entry in the directory you specify in the *newtree-name* argument. This command is useful when you need to redirect lookups for only a subset of a directory's contents.

Permissions Required

You must have insert permission to the directory in which you intend to create the soft link. You also need either delete permission to the soft link or administer permission to the directory that stores the soft link.

Example

The following command replaces the soft link /.:/eng/link1 with a new soft link whose link target is the corresponding entry in the /.:/rnd directory. The link target attribute (CDS_LinkTarget) of the new soft link will point to /.:/rnd/eng/link1.

```
cdscp>  replace link /.:/eng/link1 with link /.:/rnd
```

See Also

replace_object
replace_subtree

replace object(8cds)

Deletes a specified object entry and replaces it with a new soft link whose link target is the corresponding entry in a new location.

Synopsis

cdscp replace object *object-name* [with] link *newtree-name*

Parameters

object-name
The full name of the object entry in its old location.

newtree-name
The full name of the directory into which the object entry has moved.

Description

The replace object command deletes a specified object entry and replaces it with a soft link whose link target is the corresponding entry in the directory you specify in *newtree-name*. This command is useful when you need to redirect lookups only for a subset of a directory's contents.

Permissions Required

You must have insert permission to the directory in which you intend to create the soft link. You also need either delete permission to the object entry or administer permission to the directory that stores the object entry.

Example

The following command replaces the object entry */.:/admin/obj2* with a soft link whose link target is the corresponding entry in the directory */.:/sales*. The link target attribute (CDS_LinkTarget) of the new soft link will point to */.:/sales /admin/obj2*.

```
cdscp>  replace object /.:/admin/obj2 with link /.:/sales
```

See Also

replace_link
replace_subtree

replace subtree(8cds)

Deletes the contents of a subtree that has just been merged or appended to a new location and replaces the information with soft links whose link targets are the corresponding entries in the new location.

Synopsis

cdscp replace subtree *tree-name* [with] link *newtree-name* [norecurse] [exclude *entry-type*]

Parameters

tree-name
The full name of the topmost directory in the subtree.

newtree-name
The full name of the topmost directory in the target subtree.

entry-type
One or more of the following types of entries to exclude from the change: object entries, soft links, or directories. Use any combination of the following *entry-type* specifiers, separating multiple arguments with commas. You must leave a blank character space after each comma and after each *directory-name* specification.

acls
objects
links
directory *directory-name*

Description

The replace subtree command deletes the contents of a subtree that has just been merged or appended to a new location and replaces the information with soft links whose link targets are the corresponding entries in the new location. This command is especially useful after you merge or append the CDS namespaces of different cells.

For all entries except clearinghouse object entries, this command deletes the entries in a directory specified in *tree-name* and replaces them with soft links. These soft links redirect lookups of the names from their old (source) locations to their new (target) locations. The replace subtree command preserves both the clearinghouse object entry and its enclosing directory while deleting the directory's contents and replacing each name with an individual soft link. You can use the optional norecurse keyword to restrict the replacement operation to only the directory (and contents) you specify in *tree-name*.

Permissions Required

You must have insert permission to the directory in which you intend to create the soft links. You also need either delete permission to the entries in the source directory or administer permission to that directory.

Example

The following command deletes the entries in the directory /.:/sales/quar1 and replaces them with soft links whose link targets are their corresponding entries in /.:/total/quar1.

```
cdscp>  replace subtree /.:/sales/quar1 with link /.:/total/quar1
```

See Also

replace_link
replace_object

Part III

XDS Reference

This part provides two chapters that contain additional reference pages for the HP X/Open XDS API functions. HP's XDS implementation supports additional directory services functions beyond those supported in OSF DCE. The structure of this part and the additional HP services are described here.

For a description of the standard DCE directory service functions, refer to Chapter 4 in the *OSF DCE Application Development Reference Manual*.

6

XDS Directory Services Reference Pages

This chapter provides additional reference pages for the X/Open Directory Services (XDS) API functions. HP's XDS implementation supports two additional XDS functions, which support asynchronous operations. The functions are as follows:

- The ds_abandon function abandons the outstanding asynchronous function call.
- The ds_receive_result function retrieves completed results of an outstanding asynchronous operation.

ds_intro(3xds)

Introduces the X/Open Directory Services (XDS) functions.

Syntax

```
#include <xom.h>
#include <xds.h>
```

Description

This reference page lists the XDS interface functions supported in the HP X.500 product. XDS provides a C language binding.

Function	Description
ds_abandon	Abandons an outstanding asynchronous operation.
ds_add_entry	Adds a leaf entry to the Directory Information Tree (DIT).
ds_bind	Opens a session with a directory user agent.
ds_compare	Compares a purported attribute value with the attribute value stored in the directory for a particular entry.
ds_initialize	Initializes the interface.
ds_list	Enumerates the immediate subordinates of a particular directory entry.
ds_modify_entry	Performs an atomic modification of a directory entry.
ds_modify_rdn	Changes the Relative Distinguished Name (RDN) of a leaf entry.
ds_read	Queries information on a directory entry by name.
ds_receive_result	Retrieves the result of an asynchronously executed operation.
ds_remove_entry	Removes a leaf entry from the DIT.
ds_search	Finds entries of interest in a portion of the DIT.
ds_shutdown	Shuts down the interface.
ds_unbind	Unbinds from a directory session.
ds_version	Negotiates features of the interface and service.
dsX_trace_object	Displays an explanation of the content of an object.

DCE Notes

The HP X.500 Directory Service supports asynchronous operations, which the Distributed Computing Environment (DCE) XDS interface does not. Thus, the Abandon and Receive Result functions are included in the HP product.

The differences between the X.500 Directory Service and the Cell Directory Service (CDS) are as follows:

- All functions operate on the X.500 name space.
- CDS does not support the Modify RDN or Search functions. The Service-Error `unwilling-to-perform` is returned if either function is attempted on CDS.

- CDS does not support the X.500 schema. Therefore, CDS does not have:
  - The concept of an object class
  - Mandatory attributes for a given object
  - A set of attributes expressly permitted for a given object
  - A predefined definition of single and multivalued attributes

  The absence of the schema means that the usual errors, which are returned by X.500 for breach of the rules, are not returned by CDS.
- The CDS naming Directory Information Tree (DIT) is modeled on a typical file system architecture, in which directories are used to store objects and can contain subdirectories. Leaf objects in the CDS DIT are similar to X.500 naming objects. However, subtree objects are called directories as in a file system directory. All new objects must be added to an existing directory. CDS directory objects cannot be added, removed, modified, or compared using the XDS programming interface.
- In CDS, the naming attribute of an object is not stored in the object. Consequently, in CDS the Read operation never returns this attribute, and the Compare operation applied to this attribute returns with the Attribute-Error \fIconstraint-violation\fP.

See the notes in the relevant reference page for function-specific differences.

ds_abandon(3xds)

Abandons an outstanding asynchronous operation.

Syntax

Status = ds_abandon(*Session, Invoke-ID*)

Argument	Data Type	Access
Session	OM_private_object	read
Invoke_ID	Integer	write
Status	DS_status	

C Binding

DS_status ds_abandon*(session, invoke_id)*

DS_status ds_abandon (
OM_private_object *session*
OM_sint *invoke_id*)

Arguments

Session

The Session OM private object that was returned by the Bind function, identifying the directory session in which the operation was submitted to the directory.

Invoke-ID

Identifies the operation that is to be abandoned. You can only abandon interrogatory operations (Compare, List, Read, and Search).

The value of Invoke-ID must be that which was returned by the function call that initiated the asynchronous directory operation that is now to be abandoned.

Description

This function abandons the outstanding asynchronous function call. The asynchronous function is no longer outstanding after the Abandon function returns, and the results of the asynchronous function will never be returned by the Receive-Result function.

Note

The DCE XDS interface does not support asynchronous operations.

Return Values

Possible return values are as follows:

Return	Description
DS_SUCCESS	The operation completed successfully.
DS_NO_WORKSPACE	A workspace has not been set up by a call to the Initialize function.

If neither of these constants is returned, then the function returns a pointer to an error object of one of the classes listed below.

Errors

This function can return pointers to the following error objects:

- Abandon-Failed
- Communications-Error
- Library-Error, with problem attribute values of *bad-session* or *miscellaneous*

The result of the asynchronous operation will not be returned even if an Abandon-Failed error is returned.

Example

The following code extract shows an example call to the Abandon function.

```
OM_private_object  bound_session;
OM_sint            invoke_id;
{
    DS_status          status;
    status = ds_abandon(bound_session, invoke_id);
    if (status == DS_SUCCESS)
    {
      printf("ABANDON was successful\n");
    }
    else
    {
      printf("ABANDON failed\n");
    }
}
```

The abandon function abandons the results of the asynchronous operation identified by the Invoke-ID argument.

ds_receive_result(3xds)

Retrieves the result of an asynchronously executed operation.

Syntax

Status = ds_receive_result(*Session, Completion-Flag, Operation-Status, Result, Invoke-ID*)

Argument	Data Type	Access
Session	OM_private_object	read
Completion-Flag	Unsigned Integer	write
Operation-Status	DS_status	write
Result	OM_private_object	write
Invoke-ID	Integer	write
Status	DS_status	

C Binding

DS_status ds_receive_result*(session, completion_flag, operation_status, result, invoke_id)*

DS_status ds_receive_result (
OM_private_object *session*
OM_uint *context*<completion_flag_returned>(indent\ 3) DS_status *operation_status_return*
OM_private_object *result_return*
OM_sint *invoke_id_return*)

Arguments

Session

The Session OM private object that was returned by the Bind function, identifying the directory session in which the operation was performed.

Completion-Flag

One of the following values to indicate the status of outstanding asynchronous operations:

- Completed-Operation. At least one outstanding asynchronous operation is completed and its result is available.
- Outstanding-Operations. There are outstanding asynchronous operations but none is completed.
- No-Outstanding-Operation. There are no outstanding asynchronous operations.

  The result of the Completion-Flag parameter is valid if Status has the value Success.

Upon successful return with Completion-Flag having the value `completed-operation`, Status and Invoke-ID parameter values for the completed operation are returned.

Operation-Status	Takes an error value if an error occurred during the execution of the asynchronous directory operation. If no error occurred, then it takes the value success. The possible error values are listed for each individual operation in the corresponding function description. This result is valid only if the status has the value success and Completion-Flag has the value completed-operation.
Result	The result of the completed asynchronous operation. Its value is the constant Null-Result if the operation was one that does not return a result (Add-Entry, Modify-Entry, Modify-RDN, or Remove-Entry). Otherwise, it is an OM object of the appropriate OM class for the result of the asynchronous operation. You can check the class of the Result by using the OM functions. This result is valid only if the following conditions are true: • Status has the value *success* • Completion-Flag has the value *completed-operation* • Operation-Status has the value *success*
Invoke-ID	The Invoke-ID of the operation whose result is being returned. This result is valid if the Status has the value *success* and Completion-Flag has the value *completed-operation*.

Description

This function is used to retrieve the completed results of an outstanding asynchronous operation.

The function results include two status indications. One, called Status, indicates that the function call itself was successful and is always returned. The other, called Operation-Status, is used to return the status of the completed asynchronous operation and is only returned if there is one. See HP X.500 Directory Service Programming for information about calling functions asynchronously.

Return Values

Possible return values are as follows:

Return	Description
DS_SUCCESS	The operation completed successfully.
DS_NO_WORKSPACE	A workspace has not been set up by a call to the Initialize function.

If neither of these constants is returned, then the function returns a pointer to an error object of one of the classes listed below.

Errors

This function can return pointers to the following error object:

- Library-Error, with problem attribute values of `bad-session` or `miscellaneous`
- Any errors related to the completed asynchronous operation are reported in Operation-Status as described above.

Examples

The following code extract shows an example call to the Receive Result function. The Receive Result function is used to obtain the result of an outstanding asynchronous operation.

```
{
    /* Call the Modify Entry function asynchronously using the   */
    /* changes object as a parameter. The Asynchronous attribute */
    /* on the OM Context object has value True                   */
    status = ds_modify_entry(session,context,name,changes,&invoke_id);
    if (status == DS_SUCCESS)
    {...}
    else
    {...}
    /* now wait for the response... */
    completion_flag = DS_OUTSTANDING_OPERATIONS;
    /* loop around calls to receive_result() until we get one back */
    while ((status == DS_SUCCESS) &&
           (completion_flag == DS_OUTSTANDING_OPERATIONS))
    {
        status = ds_receive_result(bound_session, &completion_flag,
                                   &operation_status,
                                   &modify_entry_result,
                                   &invoke_id);
        if (status == DS_SUCCESS)
        {
            switch (completion_flag)
            {
                case DS_COMPLETED_OPERATION:
                /* operation is complete */
                break;
                case DS_OUTSTANDING_OPERATIONS:
                ...
                break;
                case DS_NO_OUTSTANDING_OPERATION:
                ...
                break;
```

```
            }
        }
    }
}
```

The Receive Result function uses, as input, the Invoke-ID argument output from the asynchronous function.

dsX_trace_object(3xds)

Displays an explanation of the content of an object on the current output device.

Syntax

(*void*) dsX_trace_object(*Object*)

Argument	Data Type	Access
Object	OM_object	read

C Binding

dsX_trace_object*(object)*

dsX_trace_object*object*
OM_object *object*

Arguments

OM_Object
The object whose content you want to inspect.

Description

This function displays on the current output device information about the content of an OM object, as follows:

- A full expansion of a public object
- The type of a private object
- Details of the content of an error object
- For a name object or AVA encoded in ASN.1, both the ASCII and hexadecimal representations of the ASN.1 encoding

The routine also checks for null pointers.

Errors

None.

Examples

The following code extract shows an example call to the Trace Object function:

```
{
  OM_workspace workspace;
  OM_return_code status;
  OM_object session = NULL;

  status = om_create(DS_C_SESSION,OM_TRUE,workspace,&session);
  if (status == OM_SUCCESS)
    {
      dsX_trace_object(session);
    }
}
```

7

XOM Reference Pages

This chapter provides reference pages for the X/Open Object Management (XOM) API functions.

HP's XOM implementation supports two additional ways to create private objects. The functions are as follows:

- The om_encode function uses the encoding rules you specify to create a private object independent of the original private object.
- The om_decode function decodes the ASN.1 of the original object to create the new one. You must specify the class of the existing object and the rules used to encode it.

For a description of the standard DCE XOM functions, refer to Chapter 4 of the *OSF DCE Application Development Reference Manual*.

om_intro(3xom)

Introduces the OM API functions.

Synopsis

#include <XOM.H>

Description

This reference page defines the functions of the C interface in the HP X.500 product.

Function	Description
om_copy	Copies a private object.
om_copy_value	Copies a string between private objects.
om_create	Creates a private object.
om_decode	Creates a new private object that decodes an existing ASN.1 private object.
om_delete	Deletes a private or service-generated object.
om_encode	Creates a new private object that encodes an existing private object.
om_get	Gets copies of attribute values from a private object.
om_instance	Tests an object's class.
om_put	Puts attribute values into a private object.
om_read	Reads a segment of a string in a private object.
om_remove	Removes attribute values from a private object.
om_write	Writes a segment of a string into a private object.

As indicated in the table, the service interface comprises a number of functions whose purpose and range of capabilities are summarized as follows:

Function	Description
om_copy	This function creates an independent copy of an existing private object and all its subobjects. The copy is placed in the original's workspace, or in another specified by the XOM application.
om_copy_value	This function replaces an existing attribute value or inserts a new value in one private object with a copy of an existing attribute value found in another. Both values must be strings.
om_create	This function creates a new private object that is an instance of a particular class. The object can be initialized with the attribute values specified as initial in the class definition. The service does not permit the API user to explicitly create instances of all classes, but rather only those indicated by a package's definition as having this property.
om_decode	This function creates a new private object by decoding the ASN.1 of the original object.

Function	Description
om_delete	This function deletes a service-generated public object, or makes a private object inaccessible.
om_encode	This function creates a new private object, the encoding, which exactly and independently encodes an existing private object, the original.
om_get	This function creates a new public object that is an exact but independent copy of an existing private object. The client can request certain exclusions, each of which reduces the copy to a part of the original. The client can also request that values be converted from one syntax to another before they are returned. The copy can exclude: attributes of types other than those specified, values at positions other than those specified within an attribute, the values of multivalued attributes, copies of (not handles for) subobjects, or all attribute values (revealing only an attribute's presence).
om_instance	This function determines whether an object is an instance of a particular class. The client can determine an object's class simply by inspection. This function is useful because it reveals that an object is an instance of a particular class, even if the class is an instance of a subclass of that class.
om_put	This function places or replaces in one private object copies of the attribute values of another public object or private object. The source values can be inserted before any existing destination values, before the value at a specified position in the destination attribute, or after any existing destination values. Alternatively, the source values can be substituted for any existing destination values or for the values at specified positions in the destination attribute.
om_read	This function reads a segment of a value of an attribute of a private object. The value must be a string. The value can first be converted from one syntax to another. The function enables the client to read an arbitrarily long value without requiring that the service place a copy of the entire value in memory.
om_remove	This function removes and discards particular values of an attribute of a private object. The attribute itself is removed if no values remain.
om_write	This function writes a segment of a value of an attribute to a private object. The value must be a string. The segment can first be converted from one syntax to another. The written segment becomes the value's last segment because any elements beyond it are discarded. The function enables the client to write an arbitrarily long value without having to place a copy of the entire value in memory.

In the C interface, the functions are realized by macros. The function prototype in the synopsis of a function's specification shows the client's view of the function.

The intent of the interface definition is that each function be atomic; that is, either it carries out its assigned task in full and reports success, or it fails to carry out even a part of the task and reports an exception. However, the service does not guarantee that a task is always carried out in full.

om_decode(3xom)

Creates a new private object that decodes an existing ASN.1 private object.

Synopsis

OM_return_code om_decode*(encoding, original)*

Argument	Data Type	Access
encoding	OM_private_object	read
original	OM_private_object	write
return_code	OM_return_code	

C Binding

OM_return_code om_decode*(encoding, original)*

OM_private_object encoding,
OM_private_object #original

Arguments

Encoding
The encoded object that you want to decode. It must be an instance of the Encoding class.

Original
An object that is the decoded version of the encoding. The service creates this object in the workspace in which the encoding is located. The service returns this argument if the Return Code of the function is OM_SUCCESS.

Description

This function creates a new private object by decoding the ASN.1 of the original object.

In the Encoding argument, you specify the class of the existing object and the rules used to encode it. In the current version of the OM API, you must specify ASN.1 BER.

Return Values

Possible return values are as follows:

Return	Description
OM_SUCCESS	The function has completed its task successfully.
OM_ENCODING_INVALID	The value of the Object Encoding is invalid.
OM_FUNCTION_ INTERRUPTED	The function was aborted by external intervention.

Return	Description
OM_MEMORY_INSUFFICIENT	There is not enough memory to complete the function.
OM_NETWORK_ERROR	The service cannot use the underlying network.
OM_NO_SUCH_CLASS	There is an undefined class identifier.
OM_NO_SUCH_OBJECT	You have specified a nonexistent object, or an invalid Handle for an object.
OM_NO_SUCH_RULES	There is an undefined rules identifier.
OM_NOT_AN_ENCODING	There is an object that is not an instance of the Encoding class.
OM_NOT_PRIVATE	There is a public object where there should be a private object.
OM_PERMANENT_ERROR	The service encountered a permanent problem for which there is no defined error code.
OM_POINTER_INVALID	An invalid pointer was supplied as a function argument.
OM_SYSTEM_ERROR	The service cannot use the operating system.
OM_TEMPORARY_ERROR	The service encountered a temporary problem for which there is no defined error code.
OM_TOO_MANY_VALUES	An implementation limit prevents the addition to an object of another attribute value.
OM_WRONG_VALUE_LENGTH	There is an attribute with a value that violates the value length constraints in force.
OM_WRONG_VALUE_MAKEUP	There is an attribute with a value that violates a constraint of its syntax.
OM_WRONG_VALUE_NUMBER	There is an attribute with a value that violates the value number constraints in force.
OM_WRONG_VALUE_SYNTAX	There is an attribute value with an illegal syntax.
OM_WRONG_VALUE_TYPE	There is an attribute value with an illegal type.

Examples

The following example shows the decoding of the object encoded in the code example from OSI-Abstract-Data Manipulation. The encoded object is `encoding`, and the decoded object is `decoded_object`.

```
OM_return_code      result;
OM_private_object   encoding,
                    decoded_object;
     result = om_decode (encoding,
                         /* object to be decoded */
                        &decoded_object);
                         /* decoded object */
```

om_encode(3xom)

Creates a new private object that encodes an existing private object.

Synopsis

OM_return_code om_encode*(original, rules, encoding)*

Argument	Data Type	Access
original	OM_private_object	read
rules	OM_object_identifier	read
encoding	OM_private_object	write
return_code	OM_return_code	

C Binding

OM_return_code(om_encode)*(original, rules, encoding)*

OM_private_object original,
OM_object_identifier rules,
OM_private_object #encoding

Arguments

Original
The object you want to encode.

Rules
The set of rules that the service must follow to produce an encoding. In this version of the OM API, you can only specify ASN.1 BER.

Encoding
An object that is the encoded version of the original. The service creates this object in the workspace in which the original is located. The service returns this argument if the Return Code of the function is OM_SUCCESS. The returned object is an instance of the Encoding class.

Description

This function creates a new private object, the encoding, which exactly and independently encodes an existing private object, the original. When you apply this function to a private object, the function uses the encoding rules you specify to create a new private object. The new encoded private object is independent of the original private object.

Return Values

Possible return values are as follows:

Return	Description
OM_SUCCESS	The function has completed its task successfully.

Return	Description
OM_FUNCTION_ DECLINED	The function does not apply to the object to which it is addressed.
OM_FUNCTION_ INTERRUPTED	The function was aborted by external intervention.
OM_MEMORY_ INSUFFICIENT	There is not enough memory to complete the function.
OM_NETWORK_ERROR	The service cannot use the underlying network.
OM_NO_SUCH_OBJECT	You have specified a nonexistent object, or an invalid Handle for an object.
OM_NO_SUCH_RULES	There is an undefined rules identifier.
OM_NOT_PRIVATE	There is a public object where there should be a private object.
OM_PERMANENT_ERROR	The service encountered a permanent problem for which there is no defined error code.
OM_POINTER_INVALID	An invalid pointer was supplied as a function argument.
OM_SYSTEM_ERROR	The service cannot use the operating system.
OM_TEMPORARY_ERROR	The service encountered a temporary problem for which there is no defined error code.

Examples

The following example shows the encoding of an object of the MH class Report encodable_object. The object is encoded according to the rules CWOM_BER, and the encoded object is encoding.

```
OM_return_code      result;
OM_private_object   encodable_object,
                    encoding;
     result = om_encode (encodable_object,
                         /* object to be encoded */
                        OM_BER,
                         /* encoding rules */
                        &encoding);
                         /* encoded object */
```

Part IV

IDL Reference

8

Additional APIs for Authenticated RPC

The following APIs are available from DCE Version 3.0 to create client credentials and to support server impersonation of a client. This means that the server runs with the security credentials of the client, and all of the capabilities of the client belong to the server. For additional information on RPC APIs, see the *DCE Application Reference Manual* and the *DCE Administration Guide*.

rpc_winnt_set_auth_identity

This function is called by the client RPC application to allocate and populate a WINNT auth_identity structure to be used as a parameter to rpc_binding_set_ auth_info( ). The caller must use the rpc_winnt_free_auth_identity( ) function to free the WINNT auth_identity. The strings that are passed in may be ASCII or Unicode (UCS-2) strings. The input flag will tell which type of strings they are.

Syntax

```
#include <DCE/RPC.H>

PUBLIC void rpc_winnt_set_auth_identity (
       rpc_winnt_auth_string_p_t        Username;
       rpc_winnt_auth_string_p_t        Password;
       rpc_winnt_auth_string_p_t        Domain;
       unsigned32                       CharacterSetFlag;
       rpc_auth_identity_handle_t       *auth_identity;
       unsigned32                       *stp)
```

Arguments

INPUT

username — Pointer to null terminated string containing username
password — Pointer to null terminated string containing password
domain — Pointer to null terminated string containing domain

CharacterSetFlag

SEC_WINNT_AUTH_IDENTITY_UNICODE — 2 byte Unicode (UCS-2)
SEC_WINNT_AUTH_IDENTITY_ANSI — ASCII (ISO8859-1)

OUTPUT

auth_identity — Pointer to a pointer to a WINNT auth_identity structure
stp — Pointer to returned status

Note

Be sure to allocate space for three strings (username, password, domain). The string variables will probably be pointers of type unsigned_char_t if the strings are ASCII or pointers of type wchar_t if the strings are Unicode (UCS-2).

If the domain string is a valid empty string, then the domain of the computer will be used.

rpc_winnt_free_auth_identity

This function is called by the client RPC application to free a WINNT auth_ identity structure that was previously allocated by a call to rpc_winnt_set_auth_ identity( ).

Syntax

```
#include <DCE/RPC.H>

PUBLIC void rpc_winnt_free_auth_identity (
       rpc_auth_identity_handle_t  *auth_identity,
       unsigned32                        *stp)
```

Arguments

INPUT

auth_identity — Pointer to a pointer to a WINNT auth_identity structure. On output, auth_identity will be set to NULL.

OUTPUT

stp — Pointer to returned status

Examples

The following code extract sets and frees the WINNT auth_identity.

```
#include <dce/rpc.h>

int main ()
{
/*                                              */
/* Declare variables to be used                 */
/*                                              */

rpc_auth_identity_handle_t      auth_identity;
unsigned_char_t                 username[255];
unsigned_char_t                 domain[255];
unsigned_char_t                 password[255];
error_status_t                  status;

/*                                              */
/* Initialize input arguments                   */
/*                                              */

printf("Enter username: ");
gets(username);
printf("Enter password: ");
gets(password);
printf("Enter domain: ");
gets(domain);

/*                                              */
/* Try to set the WINNT auth_identity           */
/* for use in rpc_binding_set_auth_info()       */
/*                                              */
```

APIs for Authenticated RPC
rpc_winnt_free_auth_identity

```
rpc_winnt_set_auth_identity(username, password, domain,
                            SEC_WINNT_AUTH_IDENTITY_ANSI,
                            (rpc_auth_identity_handle_t*)&auth_identity,
                             &status);

if (status != rpc_s_ok)
{
 printf ("*** Can't set winnt auth identity\n");
}
else
{
 rpc_winnt_free_auth_identity((rpc_auth_identity_handle_t*)&auth_identity, &status);
}

return(1);
}
```

rpc_impersonate_client

This function is called by the server application to allow the current server thread to run with all of the client privileges.

Syntax

```
#include <DCE/RPC.H>

void rpc_impersonate_client(
       rpc_binding_handle_t binding_handle,
       unsigned32 *status)
```

Arguments

INPUT

binding_handle — Specifies a server-side call handle for this RPC which represents the client to impersonate.

OUTPUT

status — Specifies a pointer to an unsigned 32-bit integer that holds a status code.

rpc_revert_to_self

This function is called by the server application to revert back to its original security context after impersonating a client.

Syntax

```
#include <DCE/RPC.H>
       rpc_revert_to_self(st)
```

Arguments

INPUT
None.

OUTPUT
st — Specifies a pointer to an unsigned 32-bit integer that holds a status code.

rpc_revert_to_self_ex

This function is called by the server to revert back to its original security context after impersonating a client. This acts as a call to rpc_revert_to_self( );

Syntax

```
#include <DCE/RPC.H>

       rpc_revert_to_self_ex(
       rpc_binding_handle_t        binding_handle,
       unsigned32                  *status)
```

Arguments

INPUT

call handle — This parameter is ignored.

OUTPUT

status — Specifies a pointer to an unsigned 32-bit integer that holds a status code.

9

Enhancements to Existing Authenticated RPC APIs

rpc_binding_set_auth_info

This function sets authentication and authorization information for a server binding handle. It is used by client applications.

Syntax

```
#include <dce/rpc.h>
#include <dce/sec_login.h>

void rpc_binding_set_auth_info(
        rpc_binding_handle_t binding,
        unsigned_char_t *server_princ_name,
        unsigned32 protect_level,
        unsigned32 authn_svc,
        rpc_auth_identity_handle_t auth_identity,
        unsigned32 authz_svc,
        unsigned32 *status);
```

Arguments

INPUT

binding
Specifies the server binding handle to set the authentication and authorization information.

server_princ_name
Specifies the principal name of the server referenced by binding. The content of the name and its syntax is defined by the authentication service in use.

A client that does not know the server principal name but wishes to know it, can call the rpc_mgmt_inq_server_princ_name() routine to obtain the principal name of a server that is registered for the required authentication service. Using a principal name obtained in this way means that the client is interested in one-way authentication. In other words, it means that the client does not care which server principal received the remote procedure call request. The server, though, still verifies that the client is who the client claims to be.

protect_level
Specifies the protection level for remote procedure calls made using binding. The protection level determines the degree to which authenticated communications between the client and the server are protected by the authentication service specified by authn_svc.

If the RPC runtime or the RPC protocol in the bound protocol sequence does not support a specified level, the level is automatically upgraded to the next higher supported level. The possible protection levels are as follows:

- rpc_c_protect_level_default

  Uses the default protection level for the specified authentication service. The default protection level for the DCE shared-secret key authentication service is rpc_c_protect_level_pkt_integ.

  The default protection level for the WINNT authentication service is rpc_c_ authn_level_connect.

- rpc_c_protect_level_none

  Performs no authentication: tickets are not exchanged, session keys are not established, client PACs or names are not certified, and transmissions are in the clear. Note that although uncertified PACs should not be trusted, they may be useful for debugging, tracing, and measurement purposes.

- rpc_c_protect_level_connect

  Performs protection only when the client establishes a relationship with the server.

- rpc_c_protect_level_call

  Performs protection only at the beginning of each remote procedure call when the server receives the request.

  This level does not apply to remote procedure calls made over a connection-based protocol sequence (that is, ncacn_ip_tcp). If this level is specified and the binding handle uses a connection-based protocol sequence, the routine uses the rpc_c_protect_level_pkt level instead.

- rpc_c_protect_level_pkt

  Ensures that all data received is from the expected client. rpc_c_protect_ level_pkt_integ Ensures and verifies that none of the data transferred between client and server has been modified.

  This is the highest protection level that is guaranteed to be present in the RPC runtime.

- rpc_c_protect_level_pkt_privacy

  Performs protection as specified by all of the previous levels and also encrypt each remote procedure call argument value. This is the highest protection level, but it may not be available in the RPC runtime.

  The rpc_c_authn_winnt authentication service does not currently support rpc_c_protect_level_pkt_privacy *protect_level*.

authn_svc

Specifies the authentication service to use. The exact level of protection provided by the authentication service is specified by the *protect_level* parameter. The supported authentication services are as follows:

- rpc_c_authn_none

  No authentication: no tickets are exchanged, no session keys established, client PACs or names are not transmitted, and transmissions are in the clear. Specify rpc_c_authn_none to turn authentication off for remote procedure calls made using binding.

- rpc_c_authn_dce_secret

  DCE shared-secret key authentication.

- rpc_c_authn_default

  DCE default authentication service. The current default authentication service is DCE shared-secret key; therefore, specifying rpc_c_authn_default is equivalent to specifying rpc_c_authn_dce_secret

- rpc_c_authn_dce_public

  DCE public key authentication (reserved for future use).

- rpc_c_authn_winnt

  Microsoft's NTLM authentication protocol.

  Note

  Microsoft NTLM functionality is available as of OpenVMS Version 7.2-1 on Alpha only.

auth_identity
Specifies a handle for the data structure that contains the client's authentication and authorization credentials appropriate for the selected authentication and authorization services.

When using the rpc_c_authn_dce_secret authentication service and any authorization service, this value must be a sec_login_handle_t obtained from one of the following routines:

- sec_login_setup_identity( )
- sec_login_get_current_context( )
- sec_login_newgroups( )

These routines are described in the *DCE Application Development Guide*.

Specify NULL to use the default security login context for the current address space.

When using the rpc_c_authn_winnt authentication service, this value must be a rpc_auth_identity_handle_t obtained from the rpc_winnt_set_auth_identity( ) routine.

Specify NULL to use the default security login context for the current address space.

authz_svc
Specifies the authorization service implemented by the server for the interface of interest. The validity and trustworthiness of authorization data, like any application data, is dependent on the authentication service and protection level specified. The supported authorization services are as follows:

- rpc_c_authz_none

  Server performs no authorization. This is valid only if the authn_svc parameter is rpc_c_authn_none, specifying that no authentication is being performed.

  If the rpc_c_authn_winnt authentication service is selected then the value rpc_c_authz_none MUST be specified as the authorization service.

- rpc_c_authz_name

  Server performs authorization based on the client principal name. This value cannot be used if authn_svc is rpc_c_authn_none.or rpc_c_authn_winnt.

- rpc_c_authz_dce

  Server performs authorization using the client's DCE Privilege Attribute Certificate (PAC) sent to the server with each remote procedure call made with binding. Generally, access is checked against DCE Access Control Lists

(ACLs). This value cannot be used if authn_svc is rpc_c_authn_none. or rpc_c_authn_winnt.

OUTPUT

status
Returns the status code from this routine. This status code indicates whether the routine completed successfully or, if not, why not.

The possible status codes and their meanings are as follows:

- rpc_s_ok

  Success.

- rpc_s_invalid_binding

  Invalid binding handle.

- rpc_s_wrong_kind_of_binding

  Wrong kind of binding for operation.

- rpc_s_unknown_authn_service

  Unknown authentication service.

- rpc_s_authn_authz_mismatch

  Requested authorization service is not supported by the requested authentication service.

- rpc_s_unsupported_protect_level

  Requested protection level is not supported.

Description

The rpc_binding_set_auth_info( ) routine sets up the specified server binding handle so that it can be used to make authenticated remote procedure calls that include authorization information.

Unless a client calls rpc_binding_set_auth_info( ) with the parameters to set establish authentication and authorization methods, all remote procedure calls made on the *binding* handle are unauthenticated. Some authentication services (*authn_svc*) may need to communicate with the Security service to perform this operation. Otherwise, they may receive the rpc_s_comm_failure status.

The *authn_svc* parameter specifies the authentication service to use. If authentication is chosen, the *protect_level* parameter can specify a variety of protection levels, ranging from no authentication to the highest level of authentication and encryption. If the *protect_level* parameter is set to rpc_c_protect_level_none, no authentication is performed, regardless of the authentication service chosen.

If the *authn_svc* parameter is WINNT and the *protect_level* parameter is set to rpc_c_protect_level_none, no authentication information will be set on the binding. This means that calls to rpc_binding_inq_auth_info( ) will fail with the rpc_s_binding_has_no_auth status.

The *authz_svc* parameter specifies the authorization service to use. If no authentication has been chosen (*authn_svc* of rpc_c_authn_none), then no authorization (*authz_svc* of rpc_c_authz_none) must be chosen as well. If authentication will be performed, and the *authn_svc* is rpc_c_authn_dce_ secret you have two choices for authorization: name-based authorization and

DCE authorization. The use of name-based authorization, which provides a server with a client's principal name, is not recommended. DCE authorization uses PACs, a trusted mechanism for conveying client authorization data to authenticated servers. PACs are designed to be used with the DCE ACL facility. If authentication will be performed, and the *authn_svc* is rpc_c_authn_winnt you have one choice for authorization: rpc_c_authz_none.

The *server_princ_name* parameter specifies the principal name of the server referenced by *binding*. If the *server_princ_name* parameter is NULL, the RPC runtime will call the rpc_mgmt_inq_server_princ_name() routine on the users behalf. The resulting server principal name will get stored in the *binding* for future use by the client application if desired. The designer of the client application needs to decide if this is the desired behavior. Some authentication services such as rpc_c_authn_dce_secret needs the server´s principal name for a successful authentication. The rpc_c_authn_winnt authentication service has no dependency on the server's principal name for a successful authentication. If the client application has no need to know the server's principal name and would not like to incur the overhead of automatically obtaining it, then the *server_princ_name* parameter should be non-null.

Whether the call actually wakes up in the server manager code or is rejected by the runtime depends on following conditions:

- If the client specified no authentication, then none is attempted by the RPC runtime. The call wakes up in the manager code whether the server specified authentication or not. This permits both authenticated and unauthenticated clients to call authenticated servers. When the manager receives an unauthenticated call, it needs to make a decision about how to proceed.
- If the client specified DCE secret key authentication and the server specified no authentication, then the runtime will fail the call, and it will never reach the manager routine.
- If the client specified WINNT authentication and the server specified no authentication, then the runtime will fail the call, and it will never reach the manager routine.
- If both client and server specified DCE secret key authentication or if both client and server specified WINNT authentication, then authentication will be carried out by the RPC runtime transparently. Whether the call reaches the server manager code or is rejected by the runtime depends on whether the authentication succeeded.

Although the RPC runtime is responsible any authentication that is carried out, the fact that the runtime will always permit unauthenticated clients to reach the manager code means that a manager access function typically does need to make an authentication check. When the manager access routine calls rpc_binding_inq_auth_client() it needs to check for a status of rpc_s_binding_has_no_auth. In this case, the client has specified no authentication and the manager access function needs to make an access decision based on this fact. Note that in such a case, no meaningful authentication or authorization information is returned from rpc_binding_inq_auth_client().

Return Values

No value is returned.

See Also

rpc_binding_inq_auth_client(3rpc)
rpc_binding_inq_auth_info(3rpc)
rpc_mgmt_inq_dflt_protect_level(3rpc)
rpc_mgmt_inq_server_princ_name(3rpc)
sec_login_get_current_context(3sec)
sec_login_newgroups(3sec)
sec_login_setup_identity(3sec)
rpc_winnt_set_auth_identity
rpc_winnt_free_auth_identity

rpc_binding_inq_auth_info

Returns authentication and authorization information from a server binding handle. It is used by client applications.

Syntax

```
#include <dce/rpc.h>
#include <dce/sec_login.h>

void rpc_binding_inq_auth_info(
        rpc_binding_handle_t binding,
        unsigned_char_t **server_princ_name,
        unsigned32 *protect_level,
        unsigned32 *authn_svc,
        rpc_auth_identity_handle_t *auth_identity,
        unsigned32 *authz_svc,
        unsigned32 *status);
```

Arguments

INPUT

binding

Specifies the server binding handle from which to return the authentication and authorization information.

OUTPUT

server_princ_name

Returns a pointer to the expected principal name of the server referenced by *binding*. The content of the returned name and its syntax is defined by the authentication service in use.

Specifying NULL prevents the routine from returning this parameter. In this case, the caller does not have to call the rpc_string_free( ) routine.

protect_level

Returns the protection level used for remote procedure calls made with *binding*. The protection level determines the degree to which authenticated communications between the client and the server are protected.

Note that the returned level may be different from the level specified for *protect_level* on the call to rpc_binding_set_auth_info( ). If the RPC runtime or the RPC protocol in the bound protocol sequence does not support a specified level, the level is automatically upgraded to the next higher supported level.

Specifying NULL prevents the routine from returning this parameter. The possible protection levels are as follows:

- rpc_c_protect_level_default

  Uses the default protection level for the specified authentication service.

- rpc_c_protect_level_none

  Performs no protection.

- rpc_c_protect_level_connect

Performs protection only when the client establishes a relationship with the server.

- rpc_c_protect_level_call

  Performs protection only at the beginning of each remote procedure call when the server receives the request.

- rpc_c_protect_level_pkt

  Ensures that all data received is from the expected client.

- rpc_c_protect_level_pkt_integ

  Ensures and verifies that none of the data transferred between client and server has been modified.

- rpc_c_protect_level_pkt_privacy

  Performs protection as specified by all of the previous levels and also encrypt each remote procedure call parameter value.

authn_svc
Returns the authentication service used for remote procedure calls made with *binding*. Specifying NULL prevents the routine from returning this argument.

The possible authentication services are as follows:

- rpc_c_authn_none

  No authentication.

- rpc_c_authn_dce_secret

  DCE shared-secret key authentication.

- rpc_c_authn_dce_public

  DCE public key authentication (reserved for future use).

- rpc_c_authn_default

  DCE default authentication service.

- rpc_c_authn_winnt

  Microsoft´s NTLM authentication protocol.

auth_identity
Returns a handle for the data structure that contains the client's authentication and authorization credentials. This parameter must be cast as appropriate for the authentication and authorization services established via rpc_binding_set_ auth_info( ).

When using the rpc_c_authn_dce_secret authentication service and any authorization service, this value must be a sec_login_handle_t obtained from one of the following routines:

- sec_login_setup_identity( )
- sec_login_get_current_context( )
- sec_login_newgroups( )

These routines are described in the *DCE Application Development Guide.*

Specifying NULL prevents the routine from returning this parameter. When using the rpc_c_authn_winnt authentication service, this value must be a rpc_ auth_identity_handle_t obtained from the rpc_winnt_set_auth_identity( ) routine. Specify NULL prevents the routine from returning this parameter.

authz_svc
Returns the authorization service used for remote procedure calls made with *binding*.

Specifying NULL prevents the routine from returning this parameter. The possible authorization services are as follows:

- rpc_c_authz_none

  Server performs no authorization. This is valid only if the authn_svc parameter is rpc_c_authn_none.

- rpc_c_authz_name

  Server performs authorization based on the client principal name.

- rpc_c_authz_dce

  Server performs authorization using the client's DCE Privilege Attribute Certificate (PAC) sent to the server with each remote procedure call made with *binding*. Generally, access is checked against DCE Access Control Lists (ACLs).

status
Returns the status code from this routine. This status code indicates whether the routine completed successfully or, if not, why not.

The possible status codes and their meanings are as follows:

- rpc_s_ok

  Success.

- rpc_s_invalid_binding

  Invalid binding handle.

- rpc_s_wrong_kind_of_binding

  Wrong kind of binding for operation.

- rpc_s_binding_has_no_auth

  Binding has no authentication information.

Description

The rpc_binding_inq_auth_info( ) routine returns authentication and authorization information associated with the specified server binding handle. The calling client associates the authentication and authorization data with the server binding handle by a prior call to the rpc_binding_set_auth_info( ) routine.

The RPC runtime allocates memory for the returned *server_princ_name* parameter. The caller is responsible for calling the rpc_string_free( ) routine for the returned parameter string.

Return Values

No value is returned.

See Also

rpc_binding_set_auth_info(3rpc)
rpc_string_free(3rpc)

rpc_server_register_auth_info

This routine registers authentication information with the RPC runtime. It is used by server applications.

Syntax

```
#include <dce/rpc.h>

void rpc_server_register_auth_info(
        unsigned_char_t *server_princ_name,
        unsigned32 authn_svc,
        rpc_auth_key_retrieval_fn_t get_key_fn,
        void *arg,
        unsigned32 *status);
```

Arguments

INPUT

server_princ_name
Specifies the principal name to use for the server when authenticating remote procedure calls using the service specified by *authn_svc*. The content of the name and its syntax is defined by the authentication service in use.

authn_svc
Specifies the authentication service to use when the server receives a remote procedure call request. The following authentication services are supported:

- rpc_c_authn_none

  No authentication.

- rpc_c_authn_dce_secret

  DCE shared-secret key authentication.

- rpc_c_authn_dce_public

  DCE public key authentication (reserved for future use).

- rpc_c_authn_default

  DCE default authentication service.

- rpc_c_authn_winnt

  Microsoft's NTLM authentication protocol.

get_key_fn

Specifies the address of a server-provided routine that returns encryption keys. The following C definition for rpc_auth_key_retrieval_fn_t illustrates the prototype for the encryption key acquisition routine:

```
typedef void (*rpc_auth_key_retrieval_fn_t)
 (
  void            *arg,               /* in  */
  unsigned_char_t *server_princ_name, /* in  */
  unsigned32      key_type,           /* in  */
  unsigned32      key_ver,            /* in  */
  void            **key,              /* out */
  unsigned32      *status             /* out */
 );
```

The RPC runtime passes the *server_princ_name* parameter value specified on the call to rpc_server_register_auth_info( ), as the *server_princ_name* parameter value, to the *get_key_fn* key acquisition routine. The RPC runtime automatically provides a value for the key version (*key_ver*) parameter. For a *key_ver* value of 0 (zero), the key acquisition routine must return the most recent key available. The routine returns the key in the *key* parameter. The *key_type* parameter specifies a Kerberos encryption key type. Because currently the DCE supports only DES encryption, this parameter can be ignored.

If the key acquisition routine, when called from the rpc_server_register_auth_ info( ) routine, returns a status other than rpc_s_ok, the rpc_server_register_ auth_info( ) routine fails and returns the error status to the calling server.

If the key acquisition routine, when called by the RPC runtime while authenticating a client remote procedure call request, returns a status other than rpc_s_ok, the request fails and the RPC runtime returns the error status to the client.

arg
Specifies an argument to pass to the *get_key_fn* key acquisition routine, if specified. (See the description of the *get_key_fn* parameter for details.)

Specify NULL for *arg* to use the default key table file, /krb/v5srvtab. The calling server must be root to access this file.

If *arg* is a key table file name, the file must have been created with the ktadd command. If the specified key table file resides in /krb5, you can supply only the file name. If the file does not reside in /krb5, you must supply the full pathname. You must prepend the file's absolute pathname with the prefix FILE:.

OUTPUT

status
Returns the status code from this routine. This status code indicates whether the routine completed successfully or, if not, why not.

The possible status codes and their meanings are as follows:

- rpc_s_ok

  Success.

- rpc_s_unknown_authn_service

  Unknown authentication service.

- rpc_s_key_func_not_allowed

  authn_svc is rpc_c_authn_default or rpc_c_authn_winnt and a non-null value was supplied for get_key_fn parameter.

Description

The rpc_server_register_auth_info( ) routine registers an authentication service to use for authenticating remote procedure calls to a particular server principal. A server calls this routine once for each authentication service and principal name combination that it wants to register.

The authentication service specified by a client (using the rpc_binding_set_ auth_info( ) routine) must be one of the authentication services registered by the server. If it is not, the client's remote procedure call request fails with an rpc_s_unknown_authn_service status code.

The following table shows the RPC runtime behavior for acquiring encryption keys for each supported authentication service. Note that if *authn_svc* is rpc_ c_authn_default, then *get_key_fn* must be NULL. Also note that if *authn_svc* is rpc_c_authn_winnt, then *get_key_fn* must be NULL.

RPC Key Acquisition for Authentication Services

authn_svc	get_key_fn	arg	Runtime Behavior
rpc_c_authn_default	NULL	NULL	Uses the default method of encryption key acquisition from the default key table.
rpc_c_authn_default	NULL	non-NULL	Uses the default method of encryption key acquisition from the specified key table.
rpc_c_authn_default	non-NULL	Ignored	Error returned.
rpc_c_authn_none	Ignored	Ignored	No authentication performed.
rpc_c_authn_dce_secret	NULL	NULL	Uses the default method of encryption key acquisition from the default key table.
rpc_c_authn_dce_secret	NULL	non-NULL	Uses the default method of encryption key acquisition from the specified key table.
rpc_c_authn_dce_secret	non-NULL	NULL	Uses the specified encryption key acquisition routine to obtain keys from the default key table.
rpc_c_authn_dce_secret	non-NULL	non-NULL	Uses the specified encryption key acquisition routine to obtain keys from the specified key table.
rpc_c_authn_dce_public	Ignored	Ignored	Reserved for future use.
rpc_c_authn_winnt	NULL	Ignored	Uses the default method of encryption key acquisition from the default key table.
rpc_c_authn_winnt	non-NULL	Ignored	Error returned.

Return Values

No value is returned.

See Also

rpc_binding_set_auth_info(3rpc)

rpc_binding_inq_auth_client

This routine returns authentication and authorization information from the binding handle for an authenticated client. It is used by server applications.

Syntax

```
#include <dce/rpc.h>
#include <dce/id_base.h>

void rpc_binding_inq_auth_client(
        rpc_binding_handle_t binding,
        rpc_authz_handle_t *privs,
        unsigned_char_t **server_princ_name,
        unsigned32 *protect_level,
        unsigned32 *authn_svc,
        unsigned32 *authz_svc,
        unsigned32 *status);
```

Arguments

INPUT

binding
Specifies the client binding handle from which to return the authentication and authorization information.

OUTPUT

privs
Returns a handle to the authorization information for the client that made the remote procedure call on *binding*.

The server must cast this handle to the data type specified by *authn_svc* and *authz_svc*.

If the *authn_svc* is rpc_c_authn_winnt the return value must be cast to an (unsigned_char_t *). When the *authn_svc* is rpc_c_authn_winnt the return value is the domain and username of the client that made the request. The string is in the form of *domain_name**user_name*. If the domain name is not obtainable then just the username will be returned (with no slashes).

If the *authn_svc* is rpc_c_authn_dce_secret, the following table shows how to cast the return value.

Casts for Authorization Information

For authz_svc value:	privs contains this data:	Use this cast:
rpc_c_authz_none	A NULL value.	None
rpc_c_authz_name	The calling client's principal name.	(unsigned_char_t *)
rpc_c_authz_dce	The calling client's privilege attribute certificate.	(sec_id_pac_t *)

Note that rpc_c_authz_none is valid only if the *authn_svc* parameter is rpc_c_ authn_none or rpc_c_authn_winnt.

The data referenced by this parameter is read-only and should not be modified by the server. If the server wants to preserve any of the returned data, it must copy the data into server- allocated memory.

Specifying NULL prevents the routine from returning this parameter.

server_princ_name

If *authz_svc* is rpc_c_authn_dce_secret, this parameter returns a pointer to the server principal name specified by the client that made the remote procedure call on *binding*.

If *authz_svc* is rpc_c_authn_winnt, this parameter returns a pointer to the server principal name specified by the server when it called rpc_server_register_auth_ info( ).

The content of the returned name and its syntax is defined by the authentication service in use. Specifying NULL prevents the routine from returning this parameter. In this case, the caller does not have to call the rpc_string_free( ) routine.

protect_level

Returns the protection level requested by the client that made the remote procedure call on *binding*. The protection level determines the degree to which authenticated communications between the client and the server are protected. Specifying NULL prevents the routine from returning this parameter.

The possible protection levels are as follows:

- rpc_c_protect_level_default

  Uses the default protection level for the specified authentication service.

- rpc_c_protect_level_none

  Performs no protection.

- rpc_c_protect_level_connect

  Performs protection only when the client establishes a relationship with the server.

- rpc_c_protect_level_call

  Performs protection only at the beginning of each remote procedure call when the server receives the request.

- rpc_c_protect_level_pkt

  Ensures that all data received is from the expected client.

- rpc_c_protect_level_pkt_integ

  Ensures and verifies that none of the data transferred between client and server has been modified.

- rpc_c_protect_level_pkt_privacy

  Performs protection as specified by all of the previous levels and also encrypt each remote procedure call argument value.

authn_svc
Returns the authentication service requested by the client that made the remote procedure call on *binding*. Specifying NULL prevents the routine from returning this parameter.

The possible authentication services are as follows:

- rpc_c_authn_none No authentication.
- rpc_c_authn_dce_secret

  DCE shared-secret key authentication.
- rpc_c_authn_dce_public

  DCE public key authentication (reserved for future use).
- rpc_c_authn_default

  DCE default authentication service.
- rpc_c_authn_winnt

  Microsoft's NTLM authentication protocol.

authz_svc
Returns the authorization service requested by the client that made the remote procedure call on *binding*. Specifying NULL prevents the routine from returning this parameter.

The possible authorization services are as follows:

- rpc_c_authz_none

  Server performs no authorization. This is valid only if the authn_svc parameter is rpc_c_authn_none.or rpc_c_authn_winnt.
- rpc_c_authz_name

  Server performs authorization based on the client principal name.
- rpc_c_authz_dce

  Server performs authorization using the client's DCE Privilege Attribute Certificate (PAC) sent to the server with each remote procedure call made with binding. Generally, access is checked against DCE Access Control Lists (ACLs).

status
Returns the status code from this routine. This status code indicates whether the routine completed successfully or, if not, why not.

The possible status codes and their meanings are as follows:

- rpc_s_ok

  Success.
- rpc_s_invalid_binding

  Invalid binding handle.
- rpc_s_wrong_kind_of_binding

  Wrong kind of binding for operation.
- rpc_s_binding_has_no_auth

  Binding has no authentication information.

Description

The rpc_binding_inq_auth_client( ) routine returns authentication and authorization information associated with the client identified by *binding*. The calling server manager routine can use the returned data for authorization purposes.

The RPC runtime allocates memory for the returned *server_princ_name* parameter. The server is responsible for calling the rpc_string_free( ) routine for the returned parameter string.

For applications in which the client side uses the IDL auto_handle or implicit_ handle attribute, the server side needs to be built with the IDL explicit_handle attribute specified in the Attribute Configuration File (ACF). Using explicit_ handle provides *binding* as the first parameter to each server manager routine.

Return Values

No value is returned.

See Also

rpc_binding_inq_auth_info(3rpc)
rpc_binding_set_auth_info(3rpc)
rpc_string_free(3rpc)

10

New API for G_Float/IEEE_Float Support

DCE RPC for OpenVMS now supports both G_FLOAT and IEEE floating point types on I64 and Alpha platforms. The default floating-point type on Alpha platform remains G_FLOAT. The default floating-point type on I64 is IEEE_FLOAT. DCE RPC Application developers need to use rpc_set_local_float_drep call in their application for using the non-default floating point type. DCE RPC on VAX platform only supports G_FLOAT type.

Use the floating point types consistently in a single RPC application. Different RPC applications, each using different floating point types, can run on a single system.

rpc_set_local_float_drep

The routine rpc_set_local_float_drep allows the RPC application to set the floating point type being used by the application. Only G_FLOAT and IEEE_FLOAT floating types are supported. This routine if used, must be placed before any other API calls to the RPC runtime. The first parameter float_drep is passed using the macro RPC_APPLICATION_FLOAT_TYPE that is defined in IDLBASE.H header file. This macro is set to the appropriate value based on the /FLOAT compilation option.

Syntax

```
#include dce/rpc.h
void rpc_set_local_float_drep (
                        unsigned8 float_drep,
                        unsigned32 *status);
```

Arguments

INPUT

unsigned8 float_drep — The parameter should always be passed using the macro "RPC_APPLICATION_FLOAT_TYPE". This macro is defined to 0 or 1 based on the compilation option specified for the float type.

OUTPUT

unsigned32 *status — The routine always returns "rpc_s_ok" status.

Note

You can use this routine only on Alpha and I64 systems and is not supported by VAX.

Procedure for building applications on Alpha and I64:

On I64 Systems:

By default DCE uses IEEE_FLOAT type on I64 systems i.e. DCE applications built for I64 systems would use IEEE_FLOAT floating point types.

Use the following steps for using the G_FLOAT floating point type in RPC applications developed on the C and C++ language:

1. Call the new API function rpc_set_local_float_drep(RPC_APPLICATION_ FLOAT_TYPE, &status) before calling any RPC runtime functions. The constant RPC_APPLICATION_FLOAT_TYPE is automatically defined to the floating point type specified on the compiler command line qualifier.
2. Compile the RPC application programs using the compiler qualifier /FLOAT= G_FLOAT.
3. Use the appropriate IDL compile option while building the stubs for:
   - C applications: -CC_CMD "CC/FLOAT=G_FLOAT"
   - C++ applications: -CPP_CMD "CXX/FLOAT=G_FLOAT"

4. Link the RPC applications using the appropriate DCE options file for:
   - C applications: DCE.OPT
   - C++ applications: DCE_CXX.OPT

To use IEEE_FLOAT floating point type in RPC applications developed on the C or C++ language:

1. Compile the RPC application programs using the Compiler qualifier /FLOAT=IEEE_FLOAT (default option).
2. Link the RPC application with DCE.OPT or with DCE_CXX.OPT.

On Alpha Systems:

By default DCE uses G_FLOAT type on Alpha systems i.e. DCE applications built on Alpha systems would use G_FLOAT floating point types.

The following are the details for using the IEEE_FLOAT floating point type in RPC applications developed on the C and C++ language:

1. Call the new API function rpc_set_local_float_drep(RPC_APPLICATION_FLOAT_TYPE, &status) before calling RPC runtime functions. The constant RPC_APPLICATION_FLOAT_TYPE is automatically defined to the floating point type specified on the compiler command line qualifier.
2. Compile the RPC application programs using the compiler qualifier /FLOAT=IEEE_FLOAT.
3. Use the appropriate IDL compile option while building the stubs for:
   - C applications: -CC_CMD "CC/FLOAT=IEEE_FLOAT"
   - C++ applications: -CPP_CMD "CXX/FLOAT=IEEE_FLOAT"
4. Link the RPC applications using the appropriate DCE options file for:
   - C applications: DCE.OPT
   - C++ applications: DCE_CXX.OPT

To use G_FLOAT floating point type in RPC applications developed on the C or C++ language:

1. Compile the RPC application programs using the Compiler qualifier /FLOAT=G_FLOAT (default option).
2. Link the RPC application with DCE.OPT or with DCE_CXX.OPT.

Please also refer the HP C++ Release Notes documentation for any known restrictions or problems with running C++ applications that have been compiled using non-native floating point type.

11

DCL Command Interfaces to DCE Tools

This chapter provides DCL syntax and usage information for the Interface Definition Language (IDL) compiler and the Universal Unique Identifier Generator (UUIDGEN) utility.

11.0.1 IDL Compiler

This section provides DCL syntax for commands to the IDL compiler. Except where noted, IDL DCL command syntax is equivalent to the IDL universal command syntax documented in the **idl(1rpc)** section of the *HP DCE Application Development Reference*. See the reference documentation for a complete description of the IDL universal command syntax.

NAME

IDL — Invokes the Interface Definition Language (IDL) compiler.

SYNOPSIS

IDL *filename* [*qualifier*]...

QUALIFIERS

/CLIENT_FILES [=(*option*[,...])]
/NOCLIENT_FILES

Specify one or more of the following options:

ALL (default)
[NO]AUXILIARY [=*filename*]
NONE
[NO]STUB [=*filename*]

This qualifier is equivalent to the **-client** argument in the universal syntax.

/SERVER_FILES [=(*option*[,...])]
/NOSERVER_FILES

Specify one or more of the following options:

ALL (default)
[NO]AUXILIARY [=*filename*]
NONE
[NO]STUB [=*filename*]

This qualifier is equivalent to the **-server** argument in the universal syntax.

/INCLUDE_DIRECTORY [=*directory*[,...])] (default)
/NOINCLUDE_DIRECTORY

This qualifier is equivalent to the **-I***directory* and **-no_def_idir** arguments in the universal syntax.

/PREPROCESS
/NOPREPROCESS (default)

This qualifier is similar to the **-cpp_cmd** *'c_preprocessor_command_line'* and **-no_cpp** arguments in the universal syntax. However, /PREPROCESS does not accept a value (the compiler to handle the preprocessing), while the **-cpp_cmd** option does accept a value. You cannot use the **/PREPROCESS** qualifier to compile applications requiring the preprocessor on systems without a CC compiler. Use the C++ compiler unless the universal syntax is used.

/DEFINE [=(*identifier*[=*definition*][,...])]
/NODEFINE (default)

This qualifier is equivalent to the **-D** *name[=definition]* argument in the universal syntax.

/UNDEFINE [=(*identifier*[,...])]
/NOUNDEFINE (default)

This qualifier is equivalent to the **-U***name* argument in the universal syntax.

/SYNTAX_ONLY
/NOSYNTAX_ONLY (default)

This qualifier is equivalent to the **-syntax_only** argument in the universal syntax.

/OPTIMIZE [={SPEED | SPACE }]
/OPTIMIZE = SPEED (default)

This qualifier is equivalent to the **-space_opt** argument in the universal syntax.

/OUTPUT_DIRECTORY [=*directory*]
/NOOUTPUT_DIRECTORY (default)

This qualifier is equivalent to the **-out** *directory* argument in the universal syntax.

/HEADER_FILE = *filename*
/HEADER_FILE=*filename*.**H** (default)

This qualifier is equivalent to the **-header** *header_file* argument in the universal syntax.

/KEEP [=*option*]
/NOKEEP

Specify one of the following options:

ALL
C_SOURCE
NONE (equivalent to /NOKEEP)
OBJECT (default)

This qualifier is equivalent to the **-keep** *file_types* argument in the universal syntax.

/CC_COMMAND [=*"command-line"*]
/NOCC_COMMAND
/CC_COMMAND="CC/G_FLOAT/STANDARD=NOPORTABLE" (default)

This qualifier is equivalent to the **-cc_cmd** *'command_line'* argument in the universal syntax.

/CC_QUALIFIERS [=*"command-qualifiers"*]
/NOCC_QUALIFIERS (default)

This qualifier is equivalent to the **-cc_opt** *'command_options'* argument in the universal syntax.

/REPAIR [=(*option*[,...])]
/NOREPAIR

Specify one or more of the following options:

ALL (default)
[NO]BOOLEAN_CONSTANTS
[NO]EXTRA_PAD_BYTES
[NO]MISSING_PAD_BYTES
NONE

This qualifier is equivalent to the **-bug** *n* and **-no_bug** *n* arguments in the universal syntax. The values [NO]MISSING_PAD_BYTES, [NO]EXTRA_ PAD_BYTES, and [NO]BOOLEAN_CONSTANTS correspond to **-bug 1**, **-bug 2**, and **-bug 3**, respectively, in the universal syntax.

/VERIFY
/NOVERIFY (default)

This qualifier is equivalent to the **-confirm** argument in the universal syntax.

/WARNINGS (default)
/NOWARNINGS

This qualifier is equivalent to the **-no_warn** argument in the universal syntax.

/LOG
/NOLOG (default)

This qualifier is equivalent to the **-v** argument in the universal syntax.

/LANGUAGE [={CC | FORTRAN}]
/LANGUAGE=CC (default)
/LANGUAGE=CXX

This qualifier is equivalent to the **-lang** argument in the universal syntax.

/STANDARD [={[NO]PORTABLE | DCE_V10 | DEC_V10 | DCE_V103 | DCE_V13 | DCE_V11 | DCE_V20 | EXTENDED}]
/STANDARD=PORTABLE (default)

This qualifier is equivalent to the **-standard** *[standard_type]* argument in the universal syntax. This universal syntax argument is documented in the section that describes IDL compiler enhancements in the *HP DCE for OpenVMS Alpha and OpenVMS I64 Product Guide*.

/DIAGNOSTICS [=*filename*]
/NODIAGNOSTICS (default)

This qualifier requests that a diagnostic file listing the errors reported by a compilation be generated for LSE. If you do not specify a filename, the compiler uses the basename of the IDL file and appends the **.DIA** extension to it.

/ENTRY_POINT_VECTOR [=(*option*[,...])]
/NOENTRY_POINT_VECTOR
/ENTRY_POINT_VECTOR=(NOCLIENT, MANAGER) (default)

Specify one or more of the following options:

ALL
[NO]CLIENT
[NO]MANAGER
NONE

This qualifier provides a function similar to those of the **-cepv** and **-no_mepv** arguments in the universal syntax.

The /ENTRY_POINT_VECTOR command qualifier controls generation of the client and manager entry point vectors through the keywords CLIENT and MANAGER. In the universal command syntax, two separate **idl** options (**-cepv** and **-no_mepv**) control generation of the client and manager entry point vectors.

The following example generates both client and manager entry point vectors using the universal command syntax:

```
$ idl fpe_server.idl -cepv
```

The equivalent DCL command is as follows:

```
$ idl fpe_server.idl /ENTRY_POINT_VECTOR=(CLIENT,MANAGER)
```

If one or more options are specified, the DCL syntax requires you to specify all required options. Options that are not listed are not enabled.

/TRACE [=(option \ italic)[,...])]
/NOTRACE (default)

Specify one or more of the following options:

[NO]LOG_MANAGER
EVENTS=({ALL | CALLS | CONTEXT_HANDLES | ERRORS | NONE | MISCELLANEOUS}[,...])

This qualifier is equivalent to the **-trace** *value* argument in the universal syntax, which is documented in the *HP DCE for OpenVMS Alpha and OpenVMS I64 Product Guide*.

/VERSION
/NOVERSION (default)

This qualifier is equivalent to the **-version** argument in the universal syntax.

11.0.2 UUID Generator Utility

This section provides DCL syntax for the UUID generator utility. UUIDs may also be generated by using the uuid create command on the dcecp command line. Refer to the *OSF DCE Command Reference Manual* for additional information.

Except where noted, DCL commands are equivalent to the universal command syntax documented in the *HP DCE Application Development Reference*. See the reference documentation for a complete description of the universal command syntax interface to the UUID generator utility.

You can choose to use either the universal interface to the UUID generator utility or the DCL-style alternative.

NAME

IDENTIFIER/TRANSLATE — Translates a DECrpc Version 1 or 1.1 UUID to a DCE RPC UUID.

SYNOPSIS

IDENTIFIER/TRANSLATE *old-style-uuid [qualifier]...*

QUALIFIERS

/OUTPUT=*filename*
/OUTPUT=SYS$OUTPUT (default)

This qualifier, used with a filename, directs output to a file. If you do not specify a filename, the converted UUID goes to SYS$OUTPUT, generally your display terminal.

NAME

IDENTIFIER/GENERATE — Generates one or more DCE RPC UUIDs.

SYNOPSIS

IDENTIFIER/GENERATE *[qualifier]*

QUALIFIERS

/FORMAT=*[option]*

Specify one of the following options.

STRING (default)

STRING Format: *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx*

This is a raw UUID in its readable form.

IDL

IDL Format:

```
[
[uuid(xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx)]
version(1.0)
]
interface INTERFACENAME
{
}
```

This is a UUID as it appears syntactically in an RPC interface definition.

STRUCT

STRUCT Format: This is an initialized C structure declaration, which can be included in C code that is used with DCE RPC.

/COUNT=*n*

This qualifier specifies the number of UUID strings to be generated. If you do not specify a number for *n*, the number **1** is used by default.

/OUTPUT=*filename*
/OUTPUT=SYS$OUTPUT (default)

This qualifier, used with a filename, directs output to a file. If you do not specify a filename, the converted UUID goes to SYS$OUTPUT, generally your display terminal.

Index