IBM Distributed Computing Environment Version 3.1
for AIX and Solaris:

Administration Commands Reference
IBM Distributed Computing Environment Version 3.1
for AIX and Solaris:

Administration Commands Reference
First Edition (August 1999)
This edition applies to Version 3.1 of the IBM Distributed Computing Environment for AIX and Solaris and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. Send your comments to the following address:

International Business Machines Corporation
Department VLXA
11400 Burnet Road
Austin, Texas
78758

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

This documentation and the software to which it relates are derived in part from materials supplied by the following:

Copyright © 1995, 1996 Open Software Foundation, Inc.
Copyright © 1990, 1991 Siemens Nixdorf Informationssysteme AG
Copyright © 1988, 1989, 1995 Massachusetts Institute of Technology
Copyright © 1995, 1996 Hitachi, Ltd.

Licensee agrees that it will comply with and will require its Distributors to comply with all then applicable laws, rules and regulations (i) relating to the export or re-export of technical data when exporting or re-exporting a Licensed Program or Documentation, and (ii) required to limit a governmental agency’s rights in the Licensed Program, Documentation or associated technical data by affixing a Restricted Rights notice to the Licensed Program, Documentation and/or technical data equivalent to or substantially as follows: “Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in DFARS 52.227-7013(c)(1)(ii); FAR 52.227-19; and FAR 52.227-14, Alternate III, as applicable or in the equivalent clause of any other applicable Federal government regulations.”

© Copyright International Business Machines Corporation 1990, 1999. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
### Contents

**Preface** ......................................................................................................................... ix
**Audience** .......................................................................................................................... ix
**Applicability** .................................................................................................................... ix
**Purpose** ............................................................................................................................. ix
**Document Usage** ............................................................................................................. ix
**Related Documents** .......................................................................................................... ix
**Typographic and Keying Conventions** .............................................................................. x
**Problem Reporting** .......................................................................................................... xi

### Chapter 1. DCE Commands

- **sams** ............................................................................................................................. 2
- **svcdumplog** ................................................................................................................... 10
- **dce_intro** ....................................................................................................................... 11
- **account** .......................................................................................................................... 14
- **acl** ................................................................................................................................ 27
- **attrlist** .............................................................................................................................. 41
- **aud** .................................................................................................................................. 46
- **audevents** ........................................................................................................................ 52
- **audfilter** ............................................................................................................................ 56
- **audtrail** ................................................................................................................................ 62
- **cds** .................................................................................................................................... 66
- **cdsalias** ............................................................................................................................ 70
- **cdscache** ........................................................................................................................... 74
- **cdsclient** ............................................................................................................................ 80
- **cell** ..................................................................................................................................... 84
- **cellalias** .................................................................................................................................. 91
- **clearinghouse** ................................................................................................................... 94
- **clock** .................................................................................................................................. 104
- **csrc** .................................................................................................................................. 110
- **dceagtd (AIX only)** ......................................................................................................... 114
- **dceagtd (Solaris only)** ..................................................................................................... 116
- **dcecp** .................................................................................................................................. 118
- **dced** ................................................................................................................................... 136
- **directory** ............................................................................................................................ 139
- **dts** ....................................................................................................................................... 154
- **ems** .................................................................................................................................... 167
- **emsconsumer** ................................................................................................................... 170
- **emsevent** ............................................................................................................................ 175
- **emsfilter** ............................................................................................................................ 179
- **emslog** .................................................................................................................................. 183
- **emsd** ................................................................................................................................... 186
- **endpoint** .............................................................................................................................. 187
- **getcellname** ...................................................................................................................... 195
- **getip** .................................................................................................................................... 196
- **group** .................................................................................................................................... 197
- **host** ..................................................................................................................................... 207
- **hostdata** ............................................................................................................................. 215
- **keytab** .................................................................................................................................. 223
- **link** ..................................................................................................................................... 232
- **log** ....................................................................................................................................... 238
- **name** .................................................................................................................................... 243
- **object** .................................................................................................................................. 247
- **organization** ...................................................................................................................... 253

© Copyright IBM Corp. 1990, 1999
Chapter 2. Remote Procedure Call Commands

rpc_intro........................................... 366
idl.................................................. 368
idl –spmi........................................... 375
uuidgen............................................. 377
rpc_intro........................................... 380
rpccp............................................... 381
add element....................................... 390
add entry........................................... 393
add mapping....................................... 395
add member........................................ 398
exit.................................................. 400
export............................................... 401
help.................................................. 404
import............................................... 406
quit.................................................. 409
remove element.................................... 410
remove entry....................................... 412
remove group....................................... 413
remove mapping.................................... 414
remove member..................................... 416
remove profile...................................... 418
rpccprotses........................................ 419
rpccresolve......................................... 420
show entry.......................................... 422
show group......................................... 424
show mapping....................................... 426
show profile........................................ 429
show server......................................... 432
unexport............................................ 434

Chapter 3. Cell Directory Service Commands

cds_intro............................................ 438
add directory...................................... 439
add object......................................... 441
catraverse.......................................... 443
cdsadv............................................... 444
cdsbrowser......................................... 446
cdsclerk............................................ 447
cds cp................................................ 449
cdsd.................................................. 457
cds_dbdump......................................... 459
cdsdel............................................... 460
cds_diag............................................ 461
Chapter 4. Distributed Time Service Commands
disable .................................. 552
dtscp .................................. 553
dtsd .................................. 556
dtsdate .................................. 558
enable .................................. 560
exit .................................. 561
help .................................. 562
quit .................................. 563
set .................................. 564
show .................................. 568
synchronize ......................... 576
unadvertise .......................... 577
update .................................. 578

Chapter 5. Security Service Files and Commands .................................. 579
sec_intro .................................. 580
aud_audit_events ................. 581
cds_audit_events .................. 587
dts_audit_events .................. 600
event_class ......................... 606
group_override ..................... 608
passwd_override ................... 610
sec_audit_events ................ 613
v5srvtab .................................. 645
sec_intro .................................. 646
acl_edit .................................. 648
auditd .................................. 657
dce_login .................................. 659
dcecred_* Files ..................... 662
dceunixd (AIX only) .............. 664
kdestroy .................................. 667
kinit .................................. 668
klist .................................. 671
k5dcelogin .................................. 672
passwd_export ...................... 673
passwd_import ...................... 676
pwd_strengthd ....................... 680
rgy_edit .................................. 683
rmxcred .................................. 695
sec_admin .................................. 697
sec_create_db ....................... 706
secd .................................. 709

Chapter 6. IBM DCE 3.1 for AIX and Solaris Configuration Commands ............ 713
config.dce .................................. 714
start.dce .................................. 728
stop.dce .................................. 730
show.cfg .................................. 732
chpesite .................................. 734
clean_up.dce ......................... 736
dceback .................................. 737
dcesetup (Solaris only) .......... 747
kerberos.dce ......................... 759
mkreg.dce .................................. 761
mkdceweb .................................. 763
ps.dce .................................. 765
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>rmdceweb</td>
<td>766</td>
</tr>
<tr>
<td>rmreg.dce</td>
<td>768</td>
</tr>
<tr>
<td>unconfig.dce</td>
<td>770</td>
</tr>
<tr>
<td><strong>Appendix. Notices</strong></td>
<td>777</td>
</tr>
<tr>
<td><strong>Trademarks</strong></td>
<td>779</td>
</tr>
<tr>
<td><strong>Index</strong></td>
<td>781</td>
</tr>
</tbody>
</table>
Preface

The *IBM DCE Version 3.1 for AIX and Solaris: Administration Commands Reference* provides complete and detailed reference information to help system and network administrators use the correct syntax for International Business Machines (IBM)® Distributed Computing Environment (DCE) administrative commands.

Audience

This reference is written for system and network administrators who have previously administered a UNIX™ environment.

Applicability

This document applies to the IBM DCE Version 3.1 offering and related updates. See your software license for details.

Purpose

The purpose of this reference is to assist system and network administrators with using the correct syntax for DCE administration commands.

Document Usage

This reference is organized into six chapters.

1. For DCE cross-component commands, see "Chapter 1. DCE Commands" on page 1.
2. For DCE remote procedure call (RPC) commands, see "Chapter 2. Remote Procedure Call Commands" on page 365.
4. For DCE Distributed Time Service (DTS) commands, see "Chapter 4. Distributed Time Service Commands" on page 543.
5. For DCE Security Service commands, see "Chapter 5. Security Service Files and Commands" on page 573.
6. For DCE Installation and Configuration commands, see "Chapter 6. IBM DCE 3.1 for AIX and Solaris Configuration Commands" on page 713

Related Documents

For additional information about the Distributed Computing Environment, refer to the following documents:

1. *IBM DCE Version 3.1 for AIX and Solaris: Introduction to DCE*
2. *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide—Introduction to DCE*
3. *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide—Core Components*
This guide uses the following typographic conventions:

**Bold**  **Bold** words or characters represent system elements that you must use literally, such as commands, options, and pathnames.

*Italic*  *Italic* words or characters represent variable values that you must supply. *Italic* type is also used to introduce a new DCE term.

**Constant width**  Examples and information that the system displays appear in constant width typeface.

[ ]  Brackets enclose optional items in format and syntax descriptions.

{}  Braces enclose a list from which you must choose an item in format and syntax descriptions.

|  A vertical bar separates items in a list of choices.

<>  Angle brackets enclose the name of a key on the keyboard.

...  Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.

dcelocal  The OSF variable `dcelocal` in this document equates to the AIX® variable `/opt/dcelocal`.

dceshare  The OSF variable `dceshare` in this document equates to the AIX variable `/opt/dcelocal`.

This guide uses the following keying conventions:

<Ctrl-x> or ^x  The notation `<Ctrl-x>` or ^x followed by the name of a key indicates a control character sequence. For example, `<Ctrl-C>` means that you hold down the control key while pressing `<C>`.

<Return>  The notation `<Return>` refers to the key on your terminal or workstation that is labeled with the word Return or Enter, or with a left arrow.
Problem Reporting

If you have any problems with the software or documentation, please contact your software vendor’s customer service department.
sams

Purpose

Builds DCE message system files

Synopsis


- **d** *dest_dir*
  Specifies the directory in which files are to be created. The default is the current directory.

- **f**
  Turns off text-field filtering for the `<a|b>` construct (described later in this reference page).

- **g** *gencat_command*
  Invokes the platform-specific `gencat` command specified by `gencat_command`. Enclose `gencat_command` command strings that contain spaces in single quotes. For example to invoke `gencat` when `sams` is invoked, use the `-g` option in the form: `-g 'gencat -m'`.

- **i** *interface*
  Names the Interface Definition Language (IDL) `interface` that is to contain `const` declarations for all message numbers.

- **m**
  Generates one documentation file for each message. Each filename is named by the symbolic message code.

- **n** *output_name*
  Specifies the base name of the output files.

- **o** *output_files*
  Specifies which files to generate. The default is to generate all files.

- **s** *style*
  Specifies the order in which documentation entries are to be generated. The order is indicated by one of the following letters:
  - **a**  Alphabetic by message name.
  - **n**  Numeric by message number.
  - **t**  Alphabetic by message text.

- **t** *table*
  Generates an in-core message table that includes only those messages that are in the specified `table`. The default is to include all messages.

- **x**
  Checks each message string that contains more than one `printf`-style argument specification to make sure that it follows the XPG4 convention of `%d$`, where `d` is a digit. Note that message text should normally not have to use the XPG4 conventions because `sams` automatically inserts them when generating the catalog.

- **input_file**
  Specifies the message input file.

The `sams` utility reads the specified input file and creates a number of output files. The name `sams` stands for symbols and message strings, which is what the
program manipulates. The input file consists of keywords, numbers, and text. Whitespace, except in quoted strings, is used only to separate tokens. If the text is a simple word, it can be entered unquoted. Text that is a keyword or that spans multiple lines must be enclosed within quotes. Within such quoted text, leading and trailing newlines are ignored, and the usual C escapes (for example, \t for a tab) are accepted. In addition, spaces and tabs after a newline are ignored. If you need leading whitespace, use the two-character sequence \n followed by the spaces.

An unquoted # (number sign) introduces a comment. Everything from the # (number sign) to the end of the line is ignored.

**Generated Output**

A DCE message identifier is a 32-bit number that is divided into three parts: a technology, a component, and the message code. The message code is assigned by sams or specified in the input file.

The technology and component determine the names of all files generated by sams, including the message catalog. The dce_msg_ *(3dce)* routines know how to parse a message identifier and reconstruct the message catalog name and retrieve the desired text by using the code field.

For DCE and Distributed File System (DFS) source code, the technology will be dce or dfs and the component will be a three-letter name. For application code, the technology is part of the component, which is a number specified by OSF, but the name dce is always used for the technology.

The application writer chooses the component names for the application. Because the component names are encoded in the message numbers and in the message catalog names, it is important to use unique component names. To avoid conflicts, the application must not use component names that are used by the DCE components or by other DCE applications with which it might be installed. See /usr/lib/nls/msg/$LANG/dce*.cat for the names of all the DCE components.

Component names of commercial or production applications can be registered with OSF by sending e-mail to dce-registry@osf.org. This ensures that the application component names will not subsequently be used by DCE or by other registered DCE applications.

The sams utility creates a number of output files, as specified by the -o flag. This flag takes a set of letters, picked from the following table. The component (and technology) headers determine part of the filenames. This can be overridden by using the -n flag to specify the base name. Note that this does not replace the name under which the message catalog must be installed. For example, given dce as the technology and XXX as the component name, the following files would be created:

<table>
<thead>
<tr>
<th>Letter</th>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>dceXXX.cat</td>
<td>Catalog created by gencat; the message file is assumed to have already been generated</td>
</tr>
<tr>
<td>d</td>
<td>dceXXXmsg.man</td>
<td>Subset of a UNIX style reference page</td>
</tr>
<tr>
<td>h</td>
<td>dceXXXmsg.h</td>
<td>Header file mapping codes to message numbers</td>
</tr>
<tr>
<td>Letter</td>
<td>File</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>i</td>
<td>interface.idl</td>
<td>IDL file defining message identifier constants</td>
</tr>
<tr>
<td>m</td>
<td>dceXXX.msg</td>
<td>Message file for gencat program</td>
</tr>
<tr>
<td>p</td>
<td>dceXXXmsg.sml</td>
<td>Problem determination guide</td>
</tr>
<tr>
<td>s</td>
<td>dceXXXsvc.c</td>
<td>Serviceability table</td>
</tr>
<tr>
<td>S</td>
<td>dceXXXsvc.h</td>
<td>Serviceability header file</td>
</tr>
<tr>
<td>t</td>
<td>dceXXXmsg.c</td>
<td>Table mapping message numbers to short text; this is the in-core table of default message texts</td>
</tr>
<tr>
<td>u</td>
<td>dceXXXmac.h</td>
<td>Serviceability-use convenience macros</td>
</tr>
<tr>
<td>x</td>
<td>dceXXX.idx</td>
<td>Index file for building a problem determination book</td>
</tr>
</tbody>
</table>

**Input format**

The input file is divided into three parts; the second part is optional.

The first part of the input file specifies a set of headers in the following format:

`header value`

They must be chosen from the following set:

- **collection size value**
  The number of messages in each collection. The default value is 100.

- **component comp**
  The name of the component for which the messages are being generated. Component names must be three characters long.

- **component code value**
  The numeric value of the component, for application code.

- **default flags**
  The default flags that should be assigned to all messages that do not specify their own flags. The flags should be chosen from the following set:

  - **incatalog**
    Put the message in the message catalog.

  - **intable**
    Put the message in the in-core text table.

  - **longtext**
    Message text is long, usually meaning it will not be returned as a status code, but instead used only as a message to be displayed to the user.

  - **undocumented**
    Do not put this message in any generated documentation files.

  - **obsolete**
    Reserve a number for this message but do not output any reference to it.

  - **reserved**
    Same action as `obsolete`. 
Each flag might be preceeded by the word not or an ! (exclamation point) to reverse its meaning. This header is optional; the default value is intable incatalog not undocumented not obsolete.

**technology tech**

The name of the technology for which the messages are being generated. This header can be omitted; the default value is dce. Technology names must be three characters long.

**value start**

The low-order bits of the status code to be assigned to messages. This header can be omitted; the default value is 1.

**table varname**

The name of the in-core message table that is created. This header can be omitted; the default value is XXX_msg_table where XXX is the component name or just msg_table for application code.

A typical header might look like the following:

```
technology dce
component dts
table dts_msg_table
```

The optional second part of the input file is used to specify the DCE serviceability table and handle. It should appear in the following format:

```
serviceability table name handle handle_name start
  subcomponent_list
end
```

The table name field specifies the name of the subcomponent table, as described in the service.idl interface. The handle handle_name field specifies the name of the serviceability handle to be used with this component. (For more information, see the dce_svc_register(3dce) reference page.)

The subcomponent_list argument is a series of lines in the following format:

```
sub-component table_index
  subcomp full_descr_id
```

where:

**table_index**

is the name of a #define (put in the serviceability header file) that will be used as the subscript into the table.

**subcomp**

is a single word (in quotes, if needed, so that it will not be mistaken for a sams keyword) that names the subcomponent and is used to group related messages.

**full_descr_id**

is the code for the message that contains the full description of the subcomponent.

For example:
This indicates that there are two subcomponents.

Note that each subcomponent must have an entry somewhere in the third part of the file (described in the following section) that is a standard message code that contains the full text describing the subcomponent. For example:

```plaintext
## Messages for serviceability table
start !intable undocumented
code dst_s_svc_general
text "General administrative actions"
end

start !intable undocumented
code dst_s_svc_provider
text "Interactions with time-providers"
end
```

The third part of the input file is usually the largest part. It consists of a series of records where each record specifies one message. Each record is of the following form:

```plaintext
start [flags]
field_list
end
```

The flags are optional and are as previously described for the default header. If specified, they are used instead of the default value. A common mistake is to believe that they act as additions to the default flags specified in the first part of the file.

The field_list is a set of key-value pairs from the following list:

- **action text**
  The text describes the action that should be taken when this error occurs. The text appears in the generated documentation. This field is optional and ignored if the message is undocumented.

- **attributes text**
  The text describes the attributes for this message. If this field and the sub-component field described later in this section are both present, then a convenience macro will be generated that provides all of the arguments to the serviceability messaging routine. This is described in more detail later in this section.

- **code name [= value]**
  This is the symbolic name of the message. It is used to create a header file that has #define statements that map all symbolic message names to their numeric value. It also appears in the generated documentation. An optional value can be given when needed to ensure compatibility with older message versions. By default, values are assigned by a simple counter in the order in which messages appear in the file.
**engineer text**
This is used to specify the software engineer responsible for the code where this message could occur. This field is optional and is never output.

**explanation text**
This is a verbose description of the message; it can be blank if the message is not for an error condition. It appears in the documentation files and should provide additional information that can be used for fault isolation. This field is optional if the message is undocumented.

**notes text**
Optional notes for translators. This text, if it exists, appears as comments in the message file.

**sub-component table_index**
This field is used in conjunction with the attributes field. It specifies which subcomponent the message is assigned to. The table_index must be one of the indices that is specified in the serviceability table portion of the file.

**tables ( name ...)**
If a single component is used for several executables, the message table can get unreasonably large, containing texts that will never be used. This keyword can be used to specify a space-separated list of tables for which the message is appropriate; the table to be generated is specified by the -t flag. If this keyword is not used or if the -t flag is not given, then the message will appear in the table. Otherwise, the message will appear in the table only if the table specified by the flag is also specified on this line.

**text text**
The message itself. It is stored in the in-core message table (unless the not intable flag is given) and in the message catalog. It is intended to be returned by dce_error_inq_text( ) and related routines (see the dce_msg_intro(3dce) reference page). Unless the longtext flag is given, the text must be shorter than the size of the dce_error_string_t typedef defined in dce/dce_error.h.

The text field is used as a printf-style format string and is generated in documentation. To support this dual-use, sams provides a <a|b> construct. When generating message strings to be used in a program, the a text is used; when generating documentation, the b text is used. The following is an example:

```
  text "Function <\%s|func> failed, status=<0x%8.8lx|code>"
```

If the text includes a space, you must enclose it in double quotes. Newlines are removed and whitespace is changed to one space. To write a single less-than sign, prefix it with a backslash.

Two typical message records might look like the following:

```
  start
  code dts_s_ok
  text "Successful completion"
  notes "Ok, yes, etc."
  explanation "Operation performed."
  action "None required."
  end

  start
  code dts_s_bad_timestring
  text "Invalid time string"
```
explanation "The given string could not be parsed as a valid time specification."
action "Correct input and try again."
end

In addition, the following constructs are accepted, but ignored. This is for compatibility with other systems that might need such fields.

Many messages can also be assigned to a single subcomponent and used with a single set of attributes. This is the largest part of the serviceability work. If a message has both the attributes and sub-component fields specified, then a convenience macro will be generated that specifies the initial parameters to the dce_svc_printf( ) call.

The following is a sample message definition:

```
start
code dts_s_out_of_range
attributes "svc_c_sev_fatal | svc_c_action_exit_bad"
subcomponent dts_s_provider
text "illegal value %ld from %s provider"
explanation "Received illegal value from local time-provider."
action "Fix provider and restart server."
end
```

The following is an example of using the definition to generate a serviceability message:

```
dce_svc_printf(DTS_S_OUT_OF_RANGE_MSG, 123, "Sundial");
```

**Allowing for Growth**

It is good practice to group related messages together, but you should leave space to allow additional messages to be added later. The sams utility provides two ways to do this.

First, the message numbering can be explicitly set by using the following construct:

```
set value = number
```

Note that the number used in this construct specifies the code field as in the value header, and not the full message identifier, as can be assigned by giving a value in the code field.

Second, messages can group into a collection, which is similar to an XPG4 message catalog set. To indicate the start of a collection, use the following construct:

```
start collection number
```

This is equivalent to using the first construct, except that the number is multiplied by the collection size. A common practice is to have at least one collection for each serviceability subcomponent.
Related Information

Functions: dce_error_inq_text(3dce), dce_svc_printf(3dce).
svcdumplog

Purpose

Prints contents of a binary serviceability log file

Synopsis

svcdumplog [-s num_of_entries] log_file

-s num_of_entries

Tells svcdumplog to skip the first num_of_entries log entries before printing, where num_of_entries is an integer.

Arguments

log_file

The log file to be printed.

Description

The svcdumplog program prints the contents of a binary log file.

DCE components log important information about their activities and state via the DCE serviceability interface. The log messages can be routed as desired via the dcecp log object. The messages can also be written in either binary or in text format. (Information about specifying message format can be found in the svcroute(5dce) reference page). When binary format has been specified, each log entry is written as a binary record of data defined (in dce/svclog.h) as the contents of the serviceability prolog structure. The svcdumplog utility prints the contents of such a binary log file as readable text.

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Related Information

Commands: log(8dce).

Functions: dce_svc_log_get(3dce), dce_svc_log_close(3dce), dce_svc_log_open(3dce), dce_svc_log_rewind(3dce).

Files: See svcroute(5dce) in the IBM DCE Version 3.1 for AIX and Solaris: Application Development Reference.

dce_intro

Purpose

Introduction to the cross-component DCE administration tools

Description

The *(8dce)* reference pages describe publicly accessible DCE administration commands that are general to DCE rather than specific to a particular component. These commands are as follows:

- **account**
  - Manages an account in the DCE Security Service

- **acl**
  - Manages DCE access control lists (ACLs)

- **attrlist**
  - Manipulates attribute lists

- **aud**
  - Manages the audit daemon on a DCE host

- **audevents**
  - Lists audit events on a DCE host

- **audfilter**
  - Manages event filters on a DCE host

- **audtrail**
  - Converts the audit trail into a readable format

- **cds**
  - Manages the CDS server daemon on any DCE host.

- **cdsalias**
  - Manages cellnames in the Cell Directory Service (CDS)

- **cdscache**
  - Manages a CDS cache

- **cdsclient**
  - Manages the CDS client daemon on any DCE host.

- **cell**
  - Operates on a DCE cell

- **cellalias**
  - Performs cell aliasing and connection tasks.

- **clearinghouse**
  - Manages a clearinghouse in CDS

- **clock**
  - Manages the clock on a local or remote host

- **csrc**
  - Builds a DCE character and code set registry on a host

- **dceagtd (AIX only)**
  - The DCE SNMP subagent

- **dcecp**
  - Administrative interface for DCE management tasks

- **dced**
  - The DCE host daemon

- **directory**
  - Manages a name service directory

- **dts**
  - Manages a Distributed Time Service (DTS) daemon
**ems** Manages the Event Management Services (EMS) daemon on a DCE host

**emsconsumer**
Manages EMS consumers and their event filter groups.

**emsevent**
Manages EMS event types and event type schemas.

**emsfilter**
Manages EMS event filters on a DCE host.

**emslog**
Manages the EMS log file on the current host.

**emsd** Starts the DCE EMS daemon.

**endpoint**
Manages endpoint information in local maps

**getcellname**
Gets the primary name of the cell

**getip**
Gets a host’s IP address

**group**
Manages a group in the DCE Security Service

**host**
Manages host information in a DCE cell

**hostdata**
Manages a DCE host’s principal name and cell affiliation information

**hostvar**
Manages host-specific variables on the local DCE host.

**keytab**
Manages server passwords on DCE hosts

**link**
Manages a softlink in CDS

**log**
Manages serviceability routing and debug routing

**name**
Compares and expands DCE names

**object**
Manages an object in the name service

**organization**
Manages an organization in the DCE Security Service

**principal**
Manages a principal in the DCE Security Service

**registry**
Manages a registry in the DCE Security Service

**rpcentry**
Manages a remote procedure call (RPC) name service entry

**rpcgroup**
Manages an RPC group entry in CDS

**rpcprofile**
Manages an RPC profile entry in CDS

**secval**
Manages the security validation service on a host

**server** Manages DCE application servers

**user** Manages user information in a DCE cell
utc Adds, compares, and converts Universal Time Coordinated (UTC) timestamps

uuid Generates and compares Universal Unique Identifiers (UUIDs)

xattrschema Manages schema information for extended registry attributes (ERAs)

See each command’s reference page for further information.

Errors

Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Related Information

Commands: account(8dce), acl(8dce), attrlist(8dce), aud(8dce), audevents(8dce), audfilter(8dce), audtrail(8dce), cds(8dce), cdfsalias(8dce), cdscache(8dce), cdscclient(8dce), cell(8dce), cellalias(8dce), clearinghouse(8dce), clock(8dce), csr(8dce), dce_config(8dce), dcecp(8dce), dced(8dce), directory(8dce), dts(8dce), endpoint(8dce), getcellname(8dce), getip(8dce), group(8dce), host(8dce), hostdata(8dce), hostvar(8dce), keytab(8dce), link(8dce), log(8dce), name(8dce), object(8dce), organization(8dce), principal(8dce), registry(8dce), rpcentry(8dce), rpcgroup(8dce), rpcprofile(8dce), secval(8dce), server(8dce), user(8dce), utc(8dce), uuid(8dce), xattrschema(8dce).

account

Purpose

A dcecp object that manages an account in the DCE Security Service

Synopsis

account catalog [cell_name] [-simplename]

account create account_name_list -mypwd password -password password
-group group_name -organization organization_name [-attribute attribute_list | -attribute value ]

account delete account_name_list

account generate account_name

account help [operation | -verbose ]

account modify account_name_list [-mypwd password] [-change attribute_list | -attribute value ]

account operations

account show account_name_list [-policies | -all ]

Arguments

account_name
A list of one or more names of accounts to act on. Note that accounts are identified by principal names, so when you create an account you supply a principal name for the account name.

Supply the names as follows:
1. Fully qualified account names in the form /.../ cell_name/ account_name or /.: account_name
2. Cell-relative account names in the form account_name. These names refer to an account in the cell identified in the _s(sec) convenience variable, or if the _s(sec) convenience variable is not set, in the local host's default cell.

Do not mix fully qualified names and cell-relative names in a list. In addition, do not use the names of registry database objects that contain account information; in other words, do not use account names that begin with /.:sec/account/.

account_name_list
The name of a single account to act on. See account_name_list for the name format.

cell_name
The name of a specific cell (or /.: for the local cell) in which to catalog accounts.
The name of the **account** operation for which to display help information.

**Description**

The **account** object represents registry accounts. Although an account is associated with one principal, one group, and one organization, it is identified by the principal's primary name. Alias names are differentiated for principals, so one principal can have multiple accounts under different alias names.

When this command executes, it attempts to bind to the registry server identified in the `_s(sec)` variable. If that server cannot process the request or if the `_s(sec)` variable is not set, the command binds to either an available slave server or the master registry server, depending on the operation. Upon completion, the command sets the `_b(sec)` convenience variable to the name of the registry server it bound to.

**Attributes**

The **account** object supports the following two kinds of attributes:

1. Account attributes might or might not have default values. They assume a default value or a value set by administrators.
2. Policy attributes regulate such things as account and password lifetimes for all accounts associated with a particular registry. Policy attributes have registry wide default values. They always assume the most restrictive value whether it is the registry wide default value or a value set for an individual account.
3. Public Key attributes regulate the creation and modification of public key pairs used for public key authentication.

**Account Attributes**

- **acctvalid {yes | no}**
  
  A flag set to determine account validity. Its value is either **yes** or **no**. An account with an `acctvalid` attribute set to **no** is invalid and cannot be logged in to. The default is **yes**.

- **client {yes | no}**
  
  A flag set to indicate whether the account is for a principal that can act as a client. Its value is either **yes** or **no**. If you set this flag to **yes**, the principal is able to log in to the account and acquire tickets for authentication. The default is **yes**.

- **created creators_name ISO_timestamp**
  
  A list of two items. The first is the principal name of the creator of the account, the second is an ISO timestamp showing the time of creation. This attribute is set by the system at the time of account creation and cannot be specified or modified.

- **description**
  
  A text string (limited to the Portable Character Set) typically used to describe the use of the account. The default is the empty string (""").

- **dupkey {yes | no}**
  
  A flag set to determine whether tickets issued to the account's principal can have duplicate keys. Its value is either **yes** or **no**. The default is **no**.
  
  In DCE this attribute is currently only advisory. However, Kerberos clients and servers make use of it when they interact with a DCE Security server.
expdate  *ISO_timestamp*

The date on which the account expires. To renew the account, change the
date in this field. To specify the time, use an ISO-compliant time format
such as  *CCYY-MM-DD-hh:mm:ss* or the string  *none*. The default is  *none*.

forwardabletkt {yes | no}

A flag set to determine whether a new ticket-granting ticket with a network
address that differs from the present ticket-granting ticket’s network address
can be issued to the account’s principal. The  *proxiabletkt* attribute
performs the same function for service tickets. Its value is either  *yes* or  *no*.
The default is  *yes*.

In DCE this attribute is currently only advisory. However, Kerberos clients
and servers make use of it when they interact with a DCE Security server.

goodsince  *ISO_timestamp*

The date and time the account was last known to be in an uncompromised
state. Any tickets granted before this date are invalid. The value is an ISO
timestamp. When the account is initially created, the  *goodsince* date is set
to the current date. Control over this date is especially useful if you know
that an account’s password was compromised. Changing the password can
prevent the unauthorized principal from accessing the system again using
that password, but the changed password does not prevent the principal
from accessing the system components for which tickets were obtained
fraudulently before the password was changed. To eliminate the principal’s
access to the system, the tickets must be canceled.

The default is the time the account was created.

If you set the  *goodsince* date to the date and time the compromised
password was changed, all tickets issued before that time are no longer
valid and unauthorized principal system access is eliminated.

group  *group_name*

The name of the group associated with the account. The value is a single
group name of an existing group in the registry. This attribute must be
specified for the  *account create* command; it does not have a default
value.

If a group is deleted from the registry, all accounts associated with the
group are also deleted.

home  *directory_name*

The file system directory in which the principal is placed at login. The
default is the  */* directory.

lastchange  *principal_name ISO_timestamp*

A list of two items. The first is the principal name of the last modifier of the
account; the second is an ISO timestamp showing the time of the last
modification. This attribute is set by the system whenever the account is
modified; it cannot be set or modified directly. The initial value consists of
the principal name of the creator of the account and the time the account
was created.

organization  *organization_name*

The name of the organization associated with the account. The value is a
single organization name of an existing organization in the registry. This
attribute must be specified for the  *account create* command; it does not
have a default value.
If an organization is deleted from the registry, all accounts associated with the organization are deleted also.

**password**

The password of the account. This attribute must be specified for the **account create** command; there is no default value. This attribute is not returned by an **account show** command.

**postdatedtkt {yes | no}**

A flag set to determine if tickets with a start time some time in the future can be issued to the account’s principal. Its value is either **yes** or **no**. The default is **no**.

In DCE, this attribute is currently only advisory. However, Kerberos clients and servers make use of it when they interact with a DCE Security server.

**proxiabletkt {yes | no}**

A flag set to determine whether a new ticket with a different network address than the present ticket can be issued to the account’s principal. The **forwardabletkt** attribute performs the same function for ticket-granting tickets. Its value is either **yes** or **no**. The default is **no**.

In DCE, this attribute is currently only advisory. However, Kerberos clients and servers make use of it when they interact with a DCE Security server.

**pwdvalid {yes | no}**

A flag set to determine whether the current password is valid. If this flag is set to **no**, the next time a principal logs in to the account, the system prompts the principal to change the password. (Note that this flag is separate from the **pwdexpdate** policy, which sets time limits on password validity.) Its value is either **yes** or **no**. The default is **yes**.

In DCE this attribute is currently only advisory. However, Kerberos clients and servers use it when they interact with a DCE Security server.

**renewabletkt {yes | no}**

A flag set to determine if the ticket-granting ticket issued to the account’s principal can be renewed. If this flag is set to **yes**, the authentication service renews the ticket-granting ticket if its lifetime is valid. Its value is either **yes** or **no**. The default is **yes**.

In DCE this attribute is currently only advisory. However, Kerberos clients and servers make use of it when they interact with a DCE Security server.

**server {yes | no}**

A flag set to indicate whether the account is for a principal that can act as a server. Its value is either **yes** or **no**. This flag should be **yes** for any server that engages in authenticated communications. The default is **yes**.

**shell**

The path of the shell that is executed when a principal logs in. The legal value is any shell supported by the home cell. The default value is the empty string (""").

**stdtgtauth {yes | no}**

A flag set to determine whether service tickets issued to the account’s principal use the standard DCE ticket-granting ticket authentication mechanism. Its value is either **yes** or **no**. The default is **yes**.

**usertouser {yes | no}**

For server principals, a flag set to determine if the server can use user-to-user RPC authentication. Its value is either **yes** (can use user-to-user or server-key-based RPC authentication) or **no** (can use only server-key-based RPC authentication). The default is **no**.
Policy Attributes

**maxtktlife relative_time**

The maximum amount of time that a ticket can be valid. To specify the time, use the Distributed Time Service (DTS) relative time format ([D]D- hh: mm: ss). When a client requests a ticket to a server, the lifetime granted to the ticket takes into account the `maxtktlife` set for both the server and the client. In other words, the lifetime cannot exceed the shorter of the server’s or client’s `maxtktlife`. If you do not specify a `maxtktlife` for an account, the `maxtktlife` defined as registry authorization policy is used.

**maxtktrenew relative_time**

The amount of time 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. To specify the time, use the DTS relative time format ([D]D- hh: mm: ss). The lifetime of the principal’s service tickets can never exceed the lifetime of the principal’s ticket-granting ticket. The shorter you make `maxtktrenew`, the greater the security of the system. However, since principals must log in again to renew their ticket-granting ticket, the time specified needs to balance user convenience against the level of security required. If you do not specify this for an account, the `maxtktrenew` lifetime defined as registry authorization policy is used.

This feature is not currently used by DCE; any use of this option is unsupported at the present time.

Public Key Attributes

**pkgenprivkey { integer | default}**

Updates the public key pair used by the security server for public key authentication. Used only with the `modify` operation and only for the principal named `krbtgt/cellname`. The `integer` argument defines the bit length of the key modulus. It can be a value of 0 or a number from 512 through 1024 inclusive. A 0 indicates that no key pair will be generated. The default for `integer` is 0.

The `default` argument indicates that the public key default for the key modulus should be used.

**pkkeycipherusage pk_attributes**

Generates or modifies information used to encrypt public key pairs. Used with the `create` and `modify` operations, this attribute allows you to generate new key pairs, update existing key pairs, and change the public key password. The `pk_attributes` listed below define the tasks to perform and supply the information needed to perform the tasks.

**generatekey { integer | default}**

Randomly generate a new public key pair to use for encryption. The randomly generated key pair will create a new pair if none exists for the account or update the existing pair. If you supply this attribute, you cannot supply the `publickeyfile` and `privatekeyfile` attributes. The `integer` argument defines the bit length of the key modulus. It can be a value of 0 or a number from 512 through 1024 inclusive. A 0 indicates that no key pair will be generated. The default for `integer` is 0.

The `default` argument indicates that the public key default for the key modulus should be used.

**oldpassphrase string**

The current public key password used to verify your identity when
creating or modifying public key attributes. To change only the password, supply this attribute and the **newpassphrase** attribute with no other public key attributes.

**newpassphrase** *string*
Use this attribute to supply a new password. To change the password, you must also supply the **oldpassphrase** attribute to verify your identity.

**privatekeyfile** *file_path*
Use the key stored in the file identified by the *file_path* option to create the private key part of a public key pair used for encryption. If you supply this attribute, you cannot supply the **generatekey** attribute.

**publickeyfile** *file_path*
Use the key stored in the file identified by *file_path* to create the public key part of a public key pair used for encryption. If you supply this attribute, you cannot supply the **generatekey** attribute.

**pksignatureusage** *pk_attributes*
Generates or modifies information used to generate digital signatures. Used with the **create** or **modify** operation, this attribute allows you to generate a new signed key pair, update an existing pair, and change the public key password. The *pk_attributes* define the tasks to perform and supply the information needed to perform the tasks. They are the same attributes as the ones described for the **pkkeycipherusage** attribute.

**pkmechanism** {file}
The public key mechanism to use for storing public key information.

The *file* argument indicates the public key information will be stored in a file that is given the account principal’s UUID as a filename in the directory `opt/dcelocal/var/security/pk_file/uuid`.

See the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide* for more information about account attributes.

### Errors

A representative list of errors that might be returned is not shown here. Refer to the *IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide* for complete descriptions of all error messages.

### Operations

#### account catalog

Returns a list of the names of all accounts in the registry. The syntax is as follows:

```
account catalog [cell_name] [-simplename]
```

**Options**

- **-simplename**
  Returns a list of account names in the registry without prepending the name of the cell.
The **catalog** operation returns a list of the names of all accounts in the local registry database. Use the **cell_name** argument to return a list of accounts in another cell's registry. By default, fully qualified names are returned in the form **cell_name/account_name**. Use the **-simplename** option to return the names without the cell name in the form **account_name**.

**Privileges Required**

You must have **r (read)** permission to the principal named in the account.

**Examples**

```
dcecp> account catalog -simplename
nobody
root
daemon
uucp
bin
dce-ptgt
dce-rgy
krbtgt/goodco.com
cell_admin
hosts/pmin17/self
hosts/pmin17/cds-server
hosts/pmin17/gda
ward
dcecp>
```

```
dcecp> account catalog
/.../goodco.com/nobody
/.../goodco.com/root
/.../goodco.com/daemon
/.../goodco.com/uucp
/.../goodco.com/bin
/.../goodco.com/dce-ptgt
/.../goodco.com/dce-rgy
/.../goodco.com/krbtgt/goodco.com
/.../goodco.com/cell_admin
/.../goodco.com/hosts/pmin17/self
/.../goodco.com/hosts/pmin17/cds-server
/.../goodco.com/hosts/pmin17/gda
/.../goodco.com/ward
dcecp>
```

**account create**

Creates a new account in the registry database. The syntax is as follows:

```
account create  account_name_list -mypwd password  -password password -group  group_name  -organization organization_name
[ -attribute attribute_list | -attribute  value]
```

**Note:** For security reasons, the **account create** command can only be implemented interactively in the **dcecp** shell or in a .tcl script. It cannot be implemented from the command line.

**Options**

- **attribute value**

  As an alternative to using the **-attribute** option with an attribute list, you can
specify individual attribute options by prepending a hyphen (-) to any attributes listed in the Attributes section of this reference page.

-attribute attribute_list
Allows you to specify attributes by using an attribute list rather than using the -attribute value option. The format of an attribute list is as follows:

{{attribute value}...{attributevalue}}

You must specify the group, organization, password, and mypwd attributes on the command line.

-group group_name
The name of the group to associate with the account. See Account Attributes for the format of a group name.

-mypwd password
Your DCE privileged password. You must enter your privileged password to create an account. This check prevents a malicious user from using an existing privileged session to create unauthorized accounts. You must specify this option on the command line; it cannot be supplied in a script.

-organization organization_name
The name of the organization to associate with the account. See Account Attributes for the format of an organization name.

-password password
The DCE account password. See Account Attributes for the format of a password.

The create operation creates a new account. The account name_list argument is a list of names of principals for which the accounts are to be created. This operation returns an empty string on success.

You must specify the group, organization, password, and mypwd attributes on the command line (either in an attribute list or with attribute options). The attributes specified are applied to all of the accounts created.

To protect the account password being entered, the account create command can be entered only from within dcecp. You cannot enter this command from the operating system prompt by using dcecp with the -c option.

Before you can create a new account, you must create a principal by using the principal create command. Then you must add the principal to an existing group and organization using the group add and organization add commands.

Privileges Required

You must have the following permissions:
1. gmau (groups, mgmt_info, auth_info, and user_info) permissions to the principal named in the account
2. rtM (read, test, Member_list) permissions to the organization named in the account
3. tM (test, Member_list) permissions to the group named in the account
4. r (read) permission on the registry policy object.

Examples
account(8dce)

dcecp> principal create John_Hunter
dcecp> group add users -member John_Hunter
dcecp> organization add users -member John_Hunter
dcecp> account create John_Hunter -group users -organization users -mypwd my.secret.password -password change.me
dcecp>

account delete

Deletes existing accounts from the registry. The syntax is as follows:

account delete account_name_list

The delete operation deletes existing accounts from the registry. The argument is a list of names of accounts to be deleted. If the accounts do not exist, an error is generated. This operation returns an empty string on success.

Privileges Required

You must have rmau (read, mgmt_info, auth_info, user_info) permissions for the principal named in the account.

Examples

dcecp> account delete john_hunter
dcecp>

account generate

Randomly generates a password for a named account. The syntax is as follows:

account generate account_name

To run the account generate command, the pwd_strength server must be running, the principal identified by account_name must exist, and the pwd_mgmt_binding and pwd_val_type Extended Registry Attributes must be attached to that principal. Otherwise, an error is generated. The command returns a randomly generated password on success.

See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information about the pwd_strength server.

After the password is generated, run the account create command to establish the account. Supply the randomly generated password as the account's password (using the -password option).

Privileges Required

You must have the gmau (groups, mgmt_info, auth_info, and user_info) permissions for the principal named in the account.

Examples
account help

Returns help information about the account object and its operations. The syntax is as follows:

account help  [operation | -verbose]

Options

-verbose
Displays information about the account object.

Used without an argument or option, the account help command returns brief information about each account operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the -verbose option for more detailed information about the account object itself.

Privileges Required

No special privileges are needed to use the account help command.

Examples

dcecp> account help
catalog       Returns the names of all accounts in the registry.
create       Creates an account in the registry.
delete       Deletes an account from the registry.
generate     Generates a random password for an account in the registry.
modify       Modifies an account in the registry.
show         Returns the attributes of an account.
help         Prints a summary of command-line options.
operations   Returns a list of the valid operations for this command.

account modify

Changes attributes and policies of existing accounts. The syntax is as follows:

account modify  account_name_list
                [ -mypwd password] [-change attribute_list | -attribute value]

Options

-attribute value
As an alternative to using the -change option with an attribute list, you can change individual attribute options by prepending a hyphen (-) to any attributes listed in the ATTRIBUTES section of this reference page.

-change attribute_list
Allows you to modify attributes by using an attribute list rather than individual attribute options. The format of an attribute list is as follows:

{{attribute value}...{attribute value}}
**account(8dce)**

```
-mypwd password
```

Lets you supply your privileged password when changing policy or administration data. You must enter your privileged password to change an account password; otherwise, the **-mypwd** option is optional. This check prevents a malicious user from using an existing privileged session to modify passwords of existing accounts.

The **modify** operation modifies account attributes. The **-add** and **-remove** options are not supported because the attributes created when the account is created cannot be deleted, nor can additional attributes be added. To change an account attribute, supply the new value in an attribute list or as an individual attribute option. The operation returns an empty string on success.

To protect any passwords being entered, you can execute the **account modify** command only from within the **dcecp** program; you cannot execute it from the operating system prompt using **dcecp** with the **-c** option.

**Privileges Required**

You must have **rm (read, mgmt_info)** permissions for the principal named in the account.

**Examples**

The following example changes the account’s expiration date and login shell by specifying the **expdate** and **shell** attributes as individual attribute options.

```
dcecp> account modify John_Hunter -expdate 1998 -shell /bin/csh
dcecp>
```

```
dcecp> account show John_Hunter
{acctvalid yes}
{client yes}
{created /.../my_cell.goodco.com/cell_admin 1994-06-15-18:31:08.000+00:00I-----}
{description {}}
{dupkey no}
{expdate 1995-06-16-00:00:00.000+00:00I-----}
{forwardabletkt yes}
{goodsince 1994-06-15-18:31:05.000+00:00I-----}
{group users}
{home /}
{lastchange /.../my_cell.goodco.com/cell_admin \
1994-06-16-12:21:07.000+00:00I-----}
{organization users}
{postdatedtkt no}
{proxiabletkt no}
{pwdvalid yes}
{renewabletkt yes}
{server yes}
{shell /bin/csh}
{stdtgtauth yes}
dcecp>
```

The following example generates a public key pair for John_Hunter.

```
dcecp> account modify John_Hunter -pkmechanism file \ > -generatekey 485 -newpassphrase pokey
dcecp>
```

---

24 IBM DCE Version 3.1 for AIX and Solaris: Administration Commands Reference
account operations

Returns a list of the operations supported by the account object. The syntax is as follows:

```
account operations
```

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the account operations command.

Examples

```
dcecp> account operations
catalog create delete generate modify show help operations
dcecp>
```

account show

Returns attribute information for the specified accounts. The syntax is as follows:

```
account show account_name_list [-policies | -all]
```

Options

- `-policies`
  Returns only account polices.

- `-all`
  Returns account attributes followed by account policies.

The show operation returns an attribute list describing the specified accounts. The argument is a list of names of accounts to be operated on. If more than one account is given, the attributes and policies are concatenated and a blank line inserted between accounts. The -policies option lets you return the policies of the account instead of the attributes. The -all option returns the attributes followed by the policies.

Attributes and policies are returned in lexical order. If the account has no policies, the operation displays the string nopolicy.

The policies that are actually in effect can be different from the account policies due to conflicts with registry wide policies. If this is the case, the show operation alters the attribute structure on output to include an effective tag and the effective value, much in the same way organization show does.

Privileges Required

You must have r (read) permission to the principal named in the account.

Examples

```
dcecp> account show John_Hunter
{acctvalid yes}
{client yes}
{created /.../my_cell.goodco.com/cell_admin 1994-06-15-18:31:08.000+00:00I-----}
```
account(8dce)

{description {}}
{dupkey no}
{expdate 1995-06-16-00:00:00.000+00:00I-----}
{forwardabletkt yes}
{goodsince 1994-06-15-18:31:05.000+00:00I-----}
{group users}
{home /}
{lastchange /.../my_cell.goodco.com/cell_admin \ 1994-06-16-12:21:07.000+00:00I-----}
{organization users}
{postdatedtkt no}
{proxiabletkt no}
{pwdvalid yes}
{renewabletkt yes}
{server yes}
{shell {}}
{stdtgtauth yes}
dcecp>

Related Information

Commands: dcecp(8dce), group(8dce), organization(8dce), principal(8dce), registry(8dce).
acl

Purpose

A dcecp object that manages DCE access control lists

Synopsis

acl check acl_name_list [-entry] [-type manager_type_name]

acl delete acl_name_list [-ic | -io | -entry] [-type manager_type_name] [-local]

acl help [operation | -verbose]

Arguments

acl_name_list
A list of one or more objects whose ACLs are to be acted on. You can identify objects by using the object's fully qualified names, for example, /./hosts/gumby.

Note: Use this syntax if CDS is available.

You can also use a list of string bindings with residual names appended. The residual name indicates whether the object is a principal, group, or organization by supplying its principal, group, or organization name. There are four possible formats you can use to specify a string binding.

Note: Use this syntax if CDS is not available.

In string syntax, you can use

{uuid@ prot_seq: net_addr residual_name}

Another allowable string syntax is

{uuid@ prot_seq: net_addr[ endpoint] residual_name}

In Tcl syntax, you can use

{uuid prot_seq net_addr residual_name}

Another allowable Tcl syntax is

{uuid prot_seq net_addr endpoint residual_name}

operation
The name of the acl operation for which to display help information.

Description

The acl object represents an access control list (ACL), which might exist on any object such as a server, name service entry, container (directory), or file.
ACLs consist of ACL entries. ACL entries are visible only as members of ACLs. There is no object that represents ACL entries, only the acl object representing an entire ACL. Most of the acl operations deal directly with the ACL. See Data Structures for a description of the syntax of ACLs and ACL entries. An ACL has one attribute, called cell, that represents the default cell of the ACL.

In most cases, the name of an object also specifies the name of the associated ACL to manipulate. However, some objects have more than one ACL, and some names can refer to more than one object. These ambiguities are resolved by using various options on the command line.

An object can have more than one ACL. For example, container objects—such as Cell Directory Service (CDS) directories and directories in the registry—have three ACLs: one ACL controls access to the container object itself, a second ACL specifies the default ACL on new objects added to the container (the Initial Object ACL), and a third ACL specifies the default ACL on new containers added to the container (the Initial Container ACL). By default, the acl commands operate on the ACL of the container object. Use the -ic option to operate on the Initial Container ACL. Use the -io option to operate on the Initial Object ACL. Simple objects (those that are not container objects) do not have Initial Container or Initial Object ACLs.

Some servers that have ACLs also store their network location information in a server entry in CDS. The server entry has the same name as the server itself and might also have an attached ACL. Use the -entry option to operate on the server entry ACL in CDS rather than the server’s ACL.

All dced objects have ACLs. When the dced on the local machine is in partial service mode, you must use the -local option to access dced object ACLs. To access dced object ACLs, specify only the residual portion of the object name to the acl command. For example, use hostdata, not /./hosts/gumby/config/hostdata.

Some DCE objects have more than one purpose. For instance, a registry object can represent a principal and it can also act as a directory (a container). An example is a principal name that identifies another cell (for instance, /.../comp.com) with which you want to establish authenticated operation. In this case, the cell maintains a principal name /./comp.com. The registry object for this principal name is as follows:

/./sec/principal/comp.com

Assume the cell also has a subordinate cell named /.../comp.com/test_cell. The cell maintains another principal name /./comp.com/test_cell. The registry object for this principal name is as follows:

/./sec/principal/comp.com/test_cell

Consequently, the registry object /./sec/principal/friendly.company.com also acts as a directory because it contains the hierarchical cell name /./sec/principal/friendly.company.com/test_cell. The ACL Manager that operates on registry objects differs from the ACL Manager that operates on registry directories. For instance, the latter ACL Manager has an i (insert) permission bit that controls who can add new objects to the directory. Consequently, most acl commands provide a -type option that lets you specify the appropriate ACL.
Manager when operating on registry objects that are also directories. You can list the ACL Managers available for registry objects by using the `acl show -managers` command.

**ACL Entry Syntax**

An ACL entry has the following syntax:

```
<type><key>:permissions
```

where:
- **type** Identifies the role of the ACL entry.
- **key** Identifies the specific principal or group to whom the entry applies. For an entry type of `extended`, key contains the ACL data.
- **permissions** The ACL permissions.

The syntax of an ACL entry is a list of two or three elements. The first element is the type, the optional second element is the key, and the last element is the set of permission bits. The permission bits are represented by a single character if the permission is granted and by a - (dash) if it is not. An ACL is a list of ACL entries. An example of an ACL is as follows:

```
{unauthenticated -r-----}
{user_obj crwx---}
{user britten crwx---}
{user mahler -rwx---}
{foreign_user /.../C=US/O=OSF/OU=dce/pro/bach crwxidt}
{group_obj -rwx---}
{group dds -rwx---}
{any_other -r-----},
{extended c417faf8-8340-11c9-ace3-08001e5559bb.a.b.c.a1.4.0a0b0c0d -rwx---}
```

On output the above syntax is used, with one addition. If masking produces ineffective bits in an ACL entry, the entry has two additional elements. The first is the identifier `effective`, and the second is the set of effective permissions. These elements are added only for those ACL entries that have ineffective bits, as seen in the following example:

```
{mask_obj -r-----}
{user_obj crwx---}
{user britten crwx--- effective -r-----}
```

On input, do not include the identifier `effective` or the effective permissions. You can enter permissions in any order, omitting the - (dash) for permissions not granted. For example, the above ACL could be entered as:

```
{mask_obj r}
{user_obj crwx}
{user britten wcrx}
```

**Defined ACL Entry Types**

- **user_obj** Permissions for the object’s real or effective owner.
**group_obj**
Permissions for the object’s real or effective owning group.

**other_obj**
Permissions for others authenticated in the local cell who are not otherwise named by a more specific entry type.

**user**
Permissions for a specific authenticated principal user in the ACL’s cell. This type of ACL entry must include a key that identifies the specific principal.

**group**
Permissions for a specific group in the ACL’s cell. This type of ACL entry must include a key that identifies the specific group.

**foreign_user**
Permissions for a specific, authenticated user in a foreign cell. This type of ACL entry must include a key that identifies the specific principal and the principal’s cell.

**foreign_group**
Permissions for a specific group in a foreign cell. This type of ACL entry must include a key that identifies the specific group and the group’s cell.

**foreign_other**
Permissions for all authenticated principals in a specific foreign cell, unless those principals are specifically named in an ACL entry of type **foreign_user** or are members in a group named in an entry of type **foreign_group**. This type of ACL entry must include a key that identifies the specific foreign cell.

**any_other**
Permissions for all authenticated principals unless those principals match a more specific entry in the ACL.

**mask_obj**
Permissions for the object mask that is applied to all entry types except **user_obj**, **other_obj**, and **unauthenticated**.

**unauthenticated**
Maximum permissions applied when the accessor does not pass authentication procedures. This entry is used for principals that have failed authentication due to bad keys, principals who are entirely outside of any authentication cell, and principals who choose not to use authenticated access. Permissions granted to an unauthenticated principal are masked with this entry, if it exists. If this entry does not exist, access to unauthenticated principals is always denied.

**extended**
A special entry that allows client applications running at earlier DCE versions to copy ACLs to and from ACL Managers running at the current DCE version without losing any data. The **extended** entry allows the application running at the lower version to obtain a printable form of the ACL. The **extended** ACL entry has the following form:

```
extended:uuid. ndr. ndr. ndr. ndr. number_of_bytes. data
```

where:

- **uuid** Identifies the type extended ACL entry. (This UUID can identify one of the ACL entry types described here or an as-yet-undefined ACL entry type.)
Up to three network data representation (NDR) format labels (in hexadecimal format and separated by periods) that identify the encoding of data.

A decimal number that specifies the total number of bytes in data. The ACL data in hexadecimal form. (Each byte of ACL data is two hexadecimal digits.) The ACL data includes all of the ACL entry specifications except the permissions (described later) that are entered separately. The data is not interpreted; it is assumed that the ACL Manager to which the data is being passed can understand that data.

Delegated permissions for the object's real or effective owner.

Delegated permissions for the object's real or effective group.

Delegated permissions for others in the local cell who are not otherwise named by a more specific entry type.

Delegated permissions for a specific principal user in the ACL’s cell. This type of ACL entry must include a key that identifies the specific principal.

Delegated permissions for a specific group in the ACL’s cell. This type of ACL entry must include a key that identifies the specific principal.

Delegated permissions for a specific, authenticated user in a foreign cell. This type of ACL entry must include a key that identifies the specific principal and the principal's cell.

Delegated permissions for a specific, authenticated group in a foreign cell. This type of ACL entry must include a key that identifies the specific group and the group’s cell.

Delegated permissions for all authenticated principals in a specific foreign cell, unless those principals are specifically named in an ACL entry of type foreign_user or foreign_user_delegate or are members in a group named in an entry of type foreign_group or foreign_group_delegate. This type of ACL entry must include a key that identifies the specific foreign cell.

Delegated permissions for all authenticated principals unless those principals match a more specific entry in the ACL.

The key identifier (principal, group name, or cell) specifies the principal or group to which the ACL entry applies. For entries of entry type extended, key is the data passed from one ACL Manager to another. A key is required for the following types of ACL entries:

- Requires a principal name only.
acl(8dce)

**group** Requires a group name only.

**foreign_user**
Requires a fully qualified cell name in addition to the principal name.

**foreign_group**
Requires a fully qualified cell name in addition to the group name.

**foreign_other**
Requires a fully qualified cell name.

**foreign_user_delegate**
Requires a fully qualified cell name, the principal name, and a key that identifies the principal and the principal’s cell.

**foreign_group_delegate**
Requires a fully qualified cell name, the group name, and a key that identifies the group and the group’s cell.

**Permissions**

The permissions argument specifies the set of permissions that defines the access rights conferred by the entry. Since each ACL Manager defines the permission tokens and meanings appropriate for the objects it controls, the actual tokens and their meanings vary. For example the Distributed File Service (DFS), the Directory Service, and the Security Service each implement a separate ACL Manager, and each can use a different set of tokens and permissions. Use the permissions operation to display the currently available tokens and their meanings. See the documentation for the DCE component you are using to obtain a more detailed description of its specific permissions.

**Attributes**

**cell default_cellname**
Represents the default cell of the ACL. Manipulation of this attribute is possible only through the modify and show operations.

See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information about ACL attributes.

**Errors**

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

**Operations**

**acl check**

Returns the permissions granted by the ACL to the principal entering the command. The syntax is as follows:

```
acl check acl_name_list [-entry]
    [-type manager_type_name]
```

**Options**
-entry Specifies that the command is to operate on the ACL of the namespace entry of the named object.

-type manager_type_name
   Specifies that the command uses a particular ACL Manager. This option is needed only for objects that have more than one purpose, such as for principal names that also act as directories.

The check operation returns the permissions granted in the specified object’s ACL to the principal that invoked the command. The argument is a list of names of object’s whose ACLs are to be operated on. If you specify no options, the permissions from the ACL for the object named by the operation are returned.

Privileges Required

The permissions required are defined by the object’s ACL Manager. Use the permissions operation to display the currently available tokens and their meanings. See the documentation for the DCE component you are using to obtain a more detailed description of its specific permissions.

Examples

dcecp> acl check {006f859c-ed3d-1d57-a383-0000c0239a70@ncacn_ip_tcp:130.105.5.45 \
> principal/aaa}
rwtdcia
dcecp>
dcecp> acl check ./:/hosts
rwtdcia
dcecp>

acl delete

Deletes all ACL entries from the object, except the user_obj entry, if it exists. The syntax is as follows:

acl delete acl_name_list [-ic | -io | -entry] [-type manager_type_name] [-local]

Options

-ic Specifies that the command is to operate on the Initial Container ACL of the named object.

-io Specifies that the command is to operate on the Initial Object ACL of the named object.

-entry Specifies that the command is to operate on the ACL of the namespace entry of the object.

-type manager_type_name
   Specifies that the command uses a particular ACL Manager. This option is needed only for objects that have more than one purpose, such as for principal names that also act as directories.

-local Specifies that the command is to operate on the ACL of a dced object while the dced on the local machine is in partial service mode.

The delete operation removes all ACL entries from the object, except the user_obj entry, if it exists. Note that if you use delete on an object whose ACL does not
acl(8dce)

contain a user_obj ACL entry (either because the object’s ACL Managers do not support user_obj entries or because the ACL is empty), the command displays a "bad syntax" error.

The argument is a list of names of objects whose ACLs are to be operated on. This operation returns an empty string on success.

Privileges Required

The permissions required are defined by the object's ACL Manager. Use the permissions operation to display the currently available tokens and their meanings. See the documentation for the DCE component you are using to obtain a more detailed description of its specific permissions.

Examples

dcecp> acl delete {/.:/hosts/oddball/gumby /.:/pokey}
dcecp>

acl help

Returns help information about the acl object and its operations. The syntax is as follows:

    acl help [operation | -verbose]

Options

-verbose

Displays information about the acl object.

Used without an argument or option, the acl help command returns brief information about each acl operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the -verbose option for more detailed information about the acl object itself.

Privileges Required

No special privileges are needed to use the acl help command.

Examples

dcecp> acl help
check Returns ACL permissions of invoker.
delete Deletes all ACL entries except 'user_obj' if it exists.
modify Adds, removes, or changes ACL entries and attributes.
permissions Returns permissions associated with an object.
replace Replaces entire ACL with new ACL entries and attributes.
show Returns ACL entries or attributes on an object.
help Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp>

acl modify

Changes attributes and entries of ACLs. The syntax is as follows:

    acl modify acl_name_list [-ic | -io | -entry] [-type manager_type_name]
acl(8dce)

[-cell new_cell_name] [-add acl_entry_list_with_permissions [-mask {calc | nocalc}] ]
[-change acl_entry_list_with_permissions [-mask {calc | nocalc}] ]
[-remove acl_entry_list_without_permissions] [-purge [-local]]

Options

-cell new_cell_name
Changes the value of the cell attribute by specifying the new default cell. It must be one value, not a list. The -cell option is always applied before the other options. Note that changing the default cell of an ACL that has user or group ACL entries, or their delegate counterparts, can be dangerous. The principal and groups mentioned in these ACL entries must be in the default cell. If the default cell changes, these ACL entries must change as well.

-add acl_entry_list_with_permissions
Adds the ACL entries to the ACL. The value of this option is a list of ACL entries with permissions filled in. You can use the -mask option to force or prevent mask recalculation.

-change acl_entry_list_with_permissions
Changes existing ACL entries in the ACL. The value of this option is a list of ACL entries with permissions filled in. The permissions are the new permissions placed on the specified ACL entries. The ACL entries must exist in the ACL or an error occurs. You can use the -mask option to force or prevent mask recalculation.

-remove acl_entry_list_without_permissions
Removes existing ACL entries from the ACL. The value of this option is a list of ACL entries with no permissions. The ACL entries must exist in the ACL or an error occurs.

-purge
Purges all masked permissions (before any other modifications are made), in all ACL entries except user_obj, other_obj, mask_obj, user_obj_delegate, other_obj_delegate, and unauthenticated if they exist. This option is useful only for ACLs that contain an entry of type mask_obj.

-mask {calc | nocalc}
If a modify operation causes a mask recalculation that unintentionally adds permissions to an existing ACL entry, the modify operation aborts with an error unless you specify the -mask option with a value of either calc or nocalc, or a unique abbreviation of one of these values.

Specifying calc creates or modifies the object’s mask_obj type entry with permissions equal to the union of all entries other than type user_obj, other_obj, mask_obj, and unauthenticated. This creation or modification is done after all other modifications to the ACL are performed. The new mask is set even if it grants permissions previously masked out. It is recommended that you use this option only if not specifying it results in an error. If you specify the calc option for an ACL Manager that does not support the mask_obj entry type, an error is returned.

Specifying nocalc means that a new mask should not be calculated.

The -mask option can be used only if the -add or -change option is also used and only if the object’s ACL Managers support the mask_obj ACL.
type. In addition, you cannot use the -mask option if you specify a mask_obj ACL entry in the command (by using the -add or -change options).

-ic Specifies that the operation act on the Initial Container ACL of the named object.

-io Specifies that the operation act on the Initial Object ACL of the named object.

-entry Specifies that the operation act on the ACL of the namespace entry of the named object.

-local Specifies that the operation act on the ACL of a dced object while the dced on the local machine is in partial service mode.

-type manager_type_name
   Specifies that the command uses a particular ACL Manager. This option is needed only for objects that have more than one purpose, such as for principal names that also act as directories. List available ACL managers using the acl show --manager command.

The modify operation changes one or more individual ACL entries. The argument is a list of names of ACLs to be modified. They are processed in the order they are entered. The specific operation to perform is described by using options.

Multiple actions can be specified on the command line; they are processed in a fixed order to guarantee proper processing of the ACLs. See [POSIX.6] for a description of this processing order. Either all the changes specified in the operation are made or none are. This operation returns an empty string on success.

Privileges Required

The permissions required are defined by the object’s ACL Manager. Use the permissions operation to display the currently available tokens and their meanings. See the documentation for the DCE component you are using to obtain a more detailed description of its specific permissions.

Examples

dcecp> acl modify /./hosts -add {user mahler rwcia}
dcecp> acl modify /./hosts -change {user mahler rwdtcia}
dcecp> acl modify /./hosts -add {group dce rwdtcia} -remove {user mahler}
dcecp>

call operations

Returns a list of the operations supported by the acl object. The syntax is as follows:

acl operations

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the acl operations command.
Examples

dcecp> acl operations
check delete modify permissions replace show help operations
dcecp>

acl permissions

Returns a list describing the permissions associated with an object. The syntax is as follows:

```
 acl permissions  acl_name_list [-ic | -io | -entry] [-type manager_type_name] [-local]
```

Options

- **-ic**  Specifies that the command is to operate on the Initial Container ACL of the named object.
- **-io**  Specifies that the command is to operate on the Initial Object ACL of the named object.
- **-entry**  Specifies that the command is to operate on the ACL of the namespace entry of the named object.
- **-type manager_type_name**  Specifies that the command uses a particular ACL Manager. This option is needed only for objects that have more than one purpose, such as principal names that also act as directories. List available ACL managers using the `acl show –manager` command.
- **-local**  Specifies that the command is to operate on the ACL of a dced object while the dced on the local machine is in partial service mode.

The permissions operation returns a list of the permissions associated with an object. For each permission, the operation shows the permission token and a description of the permission. The `manager_type_name` argument is a list of names of ACL Manager types whose permissions are to be returned. If more than one name is entered, the output is concatenated and a blank line inserted between each manager type.

Privileges Required

The permissions required are defined by the object's ACL Manager. Use the permissions operation to display the currently available tokens and their meanings. See the documentation for the DCE component you are using to obtain a more detailed description of its specific permissions.

Examples

```
dcecp> acl permissions .:/hosts
{r {read entry attributes}}
{w {update entry attributes}}
{d {delete entry}}
{t {test attribute values}}
{c {change ACL}}
{i {create new directory entries}}
{a {administer directory replication}}
dcecp>
```
acl replace

Replaces the entire ACL on the object specified by the argument with the supplied value. The syntax is as follows:

\texttt{acl \texttt{replace} \ acl\_name\_list \ [-ic | -io | -entry] \ [-type \ manager\_type\_name]}  
\texttt{-acl \ acl\_entry\_list \ [-cell \ new\_default\_cellname] \ [-local]}

Options

-\texttt{-ic} Specifies that the operation act on the Initial Container ACL of the named object.
-\texttt{-io} Specifies that the operation act on the Initial Object ACL of the named object.
-\texttt{-entry} Specifies that the operation act on the ACL of the namespace entry of the named object.
-\texttt{-type \ manager\_type\_name} Specifies that the command use a particular ACL Manager. This option is needed only for objects that have more than one purpose, such as for principal names that also act as directories.
-\texttt{-acl \ acl\_entry\_list} Specifies ACL entries and their new values.
-\texttt{-cell \ new\_default\_cellname} Specifies a new default cell for all of the ACLs named in \texttt{acl\_entry\_list}. The \texttt{-cell} option is always applied before the other options.
-\texttt{-local} Specifies that the operation act on the ACL of a dced object while the dced on the local machine is in partial service mode.

The \texttt{replace} operation replaces the entire ACL on the object specified by the argument with the supplied value. The argument is a list of names of ACLs to be operated on. The syntax of the value of the \texttt{-acl} option is a list of ACL entries. The \texttt{-cell} option specifies the new default cell of the ACL. Its value is the name of one cell only (it is not a list). This operation returns an empty string on success.

Privileges Required

The permissions required are defined by the object’s ACL Manager. Use the \texttt{permissions} operation to display the currently available tokens and their meanings. See the documentation for the DCE component you are using to obtain a more detailed description of its specific permissions.

Examples

\texttt{dcecp> acl \texttt{replace} \.:/\texttt{hosts} \ -acl \{\texttt{group dce rwdtcia}\}}

\texttt{dcecp>}

acl show

Returns a list of the ACL entries for the specified object. The syntax is as follows:

\texttt{acl \texttt{show} \ acl\_name\_list \ [-ic | -io | -entry] \ [-type \ manager\_type\_name] \ [-cell | -managers] \ [-local]}
acl show [-ic | -io] [-type] [-cell | -managers] [-local]

Options
-ic Specify that the command is to operate on the Initial Container ACL of the named object.
-io Specify that the command is to operate on the Initial Object ACL of the named object.
-entry Specify that the command is to operate on the ACL of the namespace entry of the named object.
-type manager_type_name Specify that the command uses a particular ACL Manager. This option is needed only for objects that have more than one purpose, such as for principal names that also act as directories.
-cell Returns the default cell name for the ACL.
-managers Returns a list of ACL Managers available for the named ACL.
-local Specify that the command is to operate on the ACL of a dced object while the dced on the local machine is in partial service mode.

The show operation returns a list of the ACL entries for the specified object. The argument is a list of names of objects whose ACLs are to be operated on. If more than one name is given, the output is concatenated and a blank line inserted between objects. If they exist, the mask_obj and unauthenticated ACL entries are displayed first.

Note that since UUIDs and not names are stored in ACLs, dcecp might not be able to determine the name associated with an ACL entry. In this case, the UUID is returned as the key instead of the name. The dcecp program might be unable to determine the name associated with an ACL entry if the default cell stored in the ACL is incorrect, or if the users and groups specified in the user and group entries are not registered in the default cell.

If a UUID replaces a name of a user and group, you can recover by adopting the orphaned UUID. To do this, create a new user or group using the UUID found in the ACL. The name of the new user or group is then available.

Privileges Required

The permissions required are defined by the object’s ACL Manager. Use the permissions operation to display the currently available tokens and their meanings. See the documentation for the DCE component you are using to obtain a more detailed description of its specific permissions.

Examples

dcecp> acl show ./:/hosts
{unauthenticated r--t---}
{user cell_admin rwdtcia}
{user hosts/absolut/cds-server rwdtcia}
{user hosts/absolut/self rwdtcia}
{user root rwdtcia}
acl(8dce)

Note: If you need to display the ACLs of a filename that is separated by a space character, ensure that you use the correct escape characters.

```dcecp
acl show "/./sec/principal/file\name
```

or

```dcecp
acl show "/./sec/principal/file\name"
```

Related Information

Commands: dcecp(8dce), account(8dce), group(8dce), organization(8dce), principal(8dce), registry(8dce), xattrschema(8dce).
attrlist

Purpose

A dcecp object that manipulates attribute lists

Synopsis

attrlist add  attrlist -member attrlist
attrlist getvalues attrlist -type typename
attrlist help [operation | -verbose ]
attrlist list  attrlist
attrlist modify attrlist [-add attribute_type attribute_values ] [-change attribute_type attribute_values ] [-remove attribute_type attribute_values ]
attrlist operations
attrlist remove attrlist -member attrlist

Arguments

attrlist  A list of one or more dcecp elements. An attrlist can be a single character, but usually consists of at least one attribute type and its value, as shown in the following:

{CDS_Convergence medium}

operation  The name of the attrlist operation for which to display help information.

Description

The attrlist object represents an attribute list as returned or accepted by many dcecp commands. Use this object to check or manipulate attribute lists so that they can be used by other commands, most commonly in scripts.

Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Operations

attrlist add

Appends one attribute list to another. The syntax is as follows:

attrlist add  attrlist -member attrlist
attrlist(8dce)

The `add` operation returns an attribute list with the attributes specified as the value of the required `-member` option appended.

**Privileges Required**

No special privileges are needed to use the `attrlist add` command.

**Examples**

```
dcecp> attrlist add {{a b} {c d}} -member {{e f} {g h}}
{a b} {c d} {e f} {g h}
dcecp>
```

**attrlist getvalues**

Returns the values of the attributes named in an attribute list. The syntax is as follows:

```
attrlist getvalues attrlist -type typename
```

The `getvalues` operation returns the values of all attributes that are both named in the attribute list and of the type specified by the required `-type` option. The value can be a single type, but if the attribute appears more than once in the attribute list, the value of each instance is returned on a separate line.

**Privileges Required**

No special privileges are needed to use the `attrlist getvalues` command.

**Examples**

```
dcecp> attrlist getvalues {{a w x} {c y} {a z}} -type a
{w x} z
```

This command can be used to filter the output of `show` operations. For example:

```
dcecp> attrlist getvalues [dir show /.:/hosts] -type CDS_UTS
1994-07-01-10:29:59.265-05:0010.000/00-00-c0-f7-de-56
```

With abbreviations, the above command could be entered as follows:

```
dcecp> at g [dir show /.:/hosts] -t CDS_UTS
1994-07-01-10:29:59.265-05:0010.000/00-00-c0-f7-de-56
```

**attrlist help**

Returns help information about the `attrlist` object and its operations. The syntax is as follows:

```
attrlist help [operation | -verbose]
```

**Options**
-verbose
Displays information about the attrlist object.

Used without an argument or option, the attrlist help command returns brief information about each attrlist operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the -verbose option for more detailed information about the attrlist object itself.

Privileges Required
No special privileges are needed to use the attrlist help command.

Examples

dcecp> attrlist help
add Adds attributes to a list.
getvalues Returns the values of specified attributes.
list Returns the attribute types present in a list.
modify Modifies an attribute list.
remove Removes attributes from a list.
help Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp>

attrlist list

Returns a list of attribute type names from an attribute list. The syntax is as follows:

attrlist list attrlist

The list operation returns a list of all the attribute type names in the attribute list in the order that they appear in the list.

Privileges Required
No special privileges are needed to use the attrlist list command.

Examples

dcecp> attrlist list {{a b} {c d}}
a c
dcecp>

attrlist modify

Removes and changes attributes and their values in an attribute list. The syntax is as follows:

attrlist modify attrlist [{-add
attribute_type attribute_values]
[-change attribute_type attribute_values]
[-remove attribute_type attribute_values}]

The modify operation returns an attribute list with attributes modified as specified by the -add, -remove and -change options. New attributes can be added, or new values added to existing attributes with -add. Entire attributes or attribute values
can be removed with `-remove`. The `-change` option removes all values from an existing attribute and replaces them with new values specified.

**Privileges Required**

No special privileges are needed to use the `attrlist modify` command.

**Examples**

```
dcecp> attrlist modify {{a b} {c d}} -add {{c e}}
{a b} {c d e}
dcecp>
dcecp> attrlist modify {{a b} {c d e}} -remove {{c e}}
{a b} {c d}
dcecp>
dcecp> attrlist modify {{a b} {c d e}} -change {{c f}}
{a b} {c f}
dcecp>
```

**attrlist operations**

Returns a list of the operations supported by the `attrlist` object. The syntax is as follows:

```
attrlist operations
```

The list of available operations is in alphabetical order except for `help` and `operations`, which are listed last.

**Privileges Required**

No special privileges are needed to use the `attrlist operations` command.

**Examples**

```
dcecp> attrlist operations
add getvalues list modify remove help operations
```

**attrlist remove**

Removes attributes and their values from an attribute list. The syntax is as follows:

```
attrlist remove  attrlist  -member  attrlist
```

The `remove` operation returns an attribute list after removing attribute types (and their values) specified as an argument to the required `-member` option.

This command removes entire attributes only; to remove specific values, use the `attrlist modify` command.

**Privileges Required**

No special privileges are needed to use the `attrlist remove` command.

**Examples**
dcecp> attrlist remove {{a b} {c d} {e f} {g h}} -member {e g} {a b} {c d}
dcecp>

Related Information

Commands: dcecp(8dce)
aud

Purpose

A dcecp object that manages the audit daemon on a DCE host

Synopsis

aud disable [remote_audit_daemon_name]
aud enable [remote_audit_daemon_name]
aud help [operation | -verbose ]
aud modify [remote_audit_daemon_name] {-changeattribute_list | -attributevalue }
aud operations
aud rewind [remote_audit_daemon_name]
aud show [remote_audit_daemon_name] [-attributes]
aud stop [remote_audit_daemon_name]

Arguments

operation
The name of the aud operation for which to display help information.

remote_audit_daemon_name
By default, operations pertain to the local audit daemon. The
remote_audit_daemon_name argument specifies the name or the binding of
the remote audit daemon on which to operate. The name syntax is as
follows:

/.../cellname/hosts/hostname/auditd

A remote audit daemon can also be specified with a string binding for the
remote host on which the audit daemon is running. Use a string binding
such as the following:

ncacn_ip_tcp:130.105.1.227[endpoint]

Alternatively, you can specify the binding by using dcecp string syntax such
as the following:

{ncacn_ip_tcp 130.105.1.227 1234}

Description

The aud object represents the audit daemon (called auditd in the reference
implementation) on a host. The daemon creates audit trails on a single host. Using
this command, you can enable or disable a daemon, change how a daemon acts
when the file system storage for its audit trails is full, or rewind an audit trail file.
This command operates on the audit daemon named in the optional `remote_audit_daemon_name` argument. If the argument is not supplied, the command operates on the audit daemon named by the `_s(aud)` convenience variable. If the variable is not set, the command operates on the audit daemon on the local host.

Attributes

`stostrategy {save | wrap}`
The audit trail storage strategy of the daemon. This attribute defines what the daemon does if the audit trail storage is full. Its possible values are:

- `save` If the specified trail size limit is reached (the default is 2 MB), `auditd` saves the current trail file to a new file (this file has the same name as the original trail file, with the date and time appended). Then, `auditd` deletes the contents of the original trail file and continues auditing from the beginning of this file. This is the default value for `stostrategy`.

- `wrap` The daemon overwrites the old audit trails.

`state {enabled | disabled}`
Specifies whether the audit daemon is accepting audit log requests. The values are `enabled` or `disabled`. The default is `enabled`.

See the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide* for more information about audit attributes.

Errors

A representative list of errors that might be returned is not shown here. Refer to the *IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide* for complete descriptions of all error messages.

Operations

`aud disable`
Disables an audit daemon. The syntax is as follows:

`aud disable [remote_audit_daemon_name]`

The `disable` operation disables the audit record logging service of an audit daemon and changes its `state` attribute to `disabled`. This operation returns an empty string on success.

Privileges Required

You must have `c (control)` permission on the audit daemon’s ACL, and you must be authenticated.

Examples

```
dcecp> aud disable
dcecp>
```
aud(8dce)

**aud enable**

Enables an audit daemon. The syntax is as follows:

```plaintext
aud enable [remote_audit_daemon_name]
```

The `enable` operation enables the audit record logging service of an audit daemon and changes its `state` attribute to `enabled`. This operation returns an empty string on success.

**Privileges Required**

You must have `c (control)` permission on the audit daemon’s ACL, and you must be authenticated.

**Examples**

```plaintext
dcecp> aud enable
dcecp>
```

**aud help**

Returns help information about the `aud` object and its operations. The syntax is as follows:

```plaintext
aud help [operation | -verbose]
```

**Options**

- `-verbose`
  
  Displays information about the `aud` object.

Used without an argument or option, the `aud help` command returns brief information about each `aud` operation. The optional `operation` argument is the name of an operation about which you want detailed information. Alternatively, you can use the `-verbose` option for more detailed information about the `aud` object itself.

**Privileges Required**

No special privileges are needed to use the `aud help` command.

**Examples**

```plaintext
dcecp> aud help
disable    Disables the audit daemon.
enable     Enables the audit daemon.
modify     Modifies the attributes of the audit daemon.
rewind     Rewinds the specified audit trail file to the beginning.
show       Returns the attributes of an audit daemon.
stop       Stops the audit daemon.
help       Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp>
```

**aud modify**

Changes the values of audit attributes. The syntax is as follows:
**aud modify**  
```plaintext
[remote_audit_daemon_name] [-change attribute_list ]
```

**Options**

- **-change attribute_list**  
  Allows you to specify attributes using an attribute list.

- **attribute value**  
  As an alternative to using the **-change** option with an attribute list, you can change individual attribute options by prepending a hyphen (-) to any attribute listed in the **Attributes** section of this reference page.

The **modify** operation allows modification of the audit daemon attributes. It accepts the **-change** option, which takes an attribute list as a value. The **aud modify** command also accepts the attribute options **-stostrategy** and **-state**. This operation returns an empty string upon success.

**Privileges Required**

You must have **c** *(control)* permission on the audit daemon’s ACL, and you must be authenticated.

**Examples**

```plaintext
dcecp> aud modify -change {{stostrategy wrap} {state enabled}}
dcecp> aud modify -stostrategy wrap -state enabled
dcecp>
```

**aud operations**

Returns a list of the operations supported by the **aud** object. The syntax is as follows:

**aud operations**

The list of available operations is in alphabetical order except for **help** and **operations**, which are listed last.

**Privileges Required**

No special privileges are needed to use the **aud operations** command.

**Examples**

```plaintext
dcecp> aud operations
disable enable modify rewind show stop help operations
dcecp>
```

**aud rewind**

Rewinds the central audit trail file to the beginning. The syntax is as follows:

**aud rewind**  
```plaintext
[remote_audit_daemon_name]
```

The **rewind** operation by default operates on the central trail file. This operation returns an empty string on success.
Privileges Required

You must have c (control) permission on the audit daemon's ACL, and you must be authenticated.

Examples

dcecp> aud rewind
dcecp>

aud show

Returns the attribute list for the audit daemon. The syntax is as follows:

aud show [remote_audit_daemon_name] [-attributes]

Options
-attributes
  Returns audit daemon attributes.

The show operation returns the attribute list for the audit daemon. The attributes are returned in lexical order. The -attributes option is provided for consistency with other dcecp commands. It does not change the function of the command.

Privileges Required

You must have r (read) permission on the audit daemon, and you must be authenticated.

Examples

dcecp> aud show
{ststrategy wrap}
{state enabled}
dcecp>

aud stop

Stops the audit daemon. The syntax is as follows:

aud stop [remote_audit_daemon_name]

The stop operation stops the audit daemon process. This operation returns an empty string on success.

Privileges Required

You must have c (control) permission on the audit daemon, and you must be authenticated.

Examples

dcecp> aud stop
dcecp>
Related Information

Commands: auditd(8sec), audevent(8dce), audfilter(8dce), audtrail(8dce), dcecp(8dce).

Files: aud_audit_events(5sec), dts_audit_events(5sec), sec_audit_events(5sec), event_class(5sec).
audevents

Purpose

A dcecp object that lists audit events on a DCE host

Synopsis

audevents catalog
audevents help [operation | -verbose ]
audevents operations
audevents query event_id
audevents show event_class_list | all
audevents catalog

Arguments

all All event classes.

event_class_list

The name of one or more recognized event classes. Legal event classes can be viewed with the catalog operation.

operation

The name of the audevents operation for which to display help information.

Description

The audevents object represents the event classes that are recognized by an audit daemon on a host. Each event class is defined in an event class configuration file, and the filename is the symbolic name of the event class.

This command operates only on the audit daemon on the local host.

Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Operations

audevents catalog

Returns a list of the names of all event classes. The syntax is as follows:

audevents catalog
The *catalog* operation returns a list of the names of all event classes. It takes no arguments. The order returned is arbitrary.

**Privileges Required**

You must have r (read) permission to the event class directory, `dcelocal/etc/audit/ec`.

**Examples**

```bash
dcecp> audevents catalog
dceAudit_admin_modify
dceAudit_admin_query
dceAudit_filter_modify
dceAudit_filter_query
dceDts_mgt_modify
dceDts_mgt_query
dceDts_synch
dceDts_time_provider
dceSec_authent
dceSec_control
dceSec_modify
dceSec_query
dceSec_server
dcecp>
```

**audevents help**

Returns help information about the *audevents* object and its operations. The syntax is as follows:

```bash
audevents help [operation | -verbose]
```

**Options**

- **-verbose**
  
  Displays information about the *audevents* object.

Used without an argument or option, the *audevents help* command returns brief information about each *audevent* operation. The optional *operation* argument is the name of an operation about which you want detailed information. Alternatively, you can use the `-verbose` option for more detailed information about the *audevents* object itself.

**Privileges Required**

No special privileges are needed to use the *audevents help* command.

**Examples**

```bash
dcecp> audevents help
catalog Returns the list of event classes for an audit daemon.
show Returns the contents of an event class.
help Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp>
```
audevents(8dce)

audevents operations

Returns a list of the operations supported by the audevents object. The syntax is as follows:

audevents operations

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the audevents operations command.

Examples

dcecp> audevents operations
       catalog show help operations
       dcecp>

audevents query

Returns the event name associated with the event number. The syntax is as follows:

audevents query event_id...

The query operation returns the event name associated with the event id. The argument is an event id written either as an integer or as a hexadecimal. If specified as a hexadecimal, the prefix 0x must be used.

Privileges Required

You must have r (read) permission to the event class directory, dcelocal/etc/audit/ec.

Examples

dcecp> audevents query 0x101
       event_name associated with 0x101 is AS_Request
       dcecp>

       dcecp> audevents query 257
       event_name associated with 257 is AS_Request
       dcecp>

audevents show

Returns the contents of an event class. The syntax is as follows:

audevents show event_class_list

The show operation returns the contents of an event class. The argument is a list of names of event classes. For each named event class, audevents show returns the event class name and the numbers of the events in the event class. (The numbers are 32-bit integers displayed in hexadecimal format.) If more than one event class is given in the argument, several lists of this format are output, concatenated as one list with a blank line between event classes.
Privileges Required

You must have r (read) permission to the event class directory, `dcelocal/etc/audit/ec`.

Examples

```bash
dcecp> audevents show dce_audit_filter_query
    {dce_audit_filter_query 0x101 0x102}
dcecp>

dcecp> audevents show {dce_audit_filter_query dce_dts_time_provider}
    {dce_audit_filter_query 0x101 0x102}
    {dce_dts_time_provider 0x211 0x212}
dcecp>
```

Related Information

Commands: `aud(8dce)`, `audfilter(8dce)`, `auditd(8sec)`, `audtrail(8dce)`, `dcecp(8dce)`.

Files: `audAudit_events(5sec)`, `dtsAudit_events(5sec)`, `event_class(5sec)`, `secAudit_events(5sec)`.
audfilter

Purpose

A dcep object that manages the event filters on a DCE host

Synopsis

audfilter catalog

audfilter create audit_filter_name_list -attribute guide_name_list

audfilter delete audit_filter_name_list

audfilter help [operation | -verbose ]

audfilter modify audit_filter_name_list [-add guide_name_list] [-remove guide_name_list]

audfilter operations

audfilter show audit_filter_name_list

Arguments

audit_filter_name_list

A list of one or more names of audit event filters. A filter name consists of a filter type and possibly a key, depending on the type.

The audit filter types are as follows:

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>principal</td>
<td>The key is a principal_name.</td>
</tr>
<tr>
<td>foreign_principal</td>
<td>The key is a /.../cellname/principal_name.</td>
</tr>
<tr>
<td>group</td>
<td>The key is a group_name.</td>
</tr>
<tr>
<td>foreign_group</td>
<td>The key is a /.../cellname/group_name.</td>
</tr>
<tr>
<td>cell</td>
<td>The key is a cellname.</td>
</tr>
<tr>
<td>cell_overridable</td>
<td>The key is a cellname.</td>
</tr>
<tr>
<td>world</td>
<td>This type has no key.</td>
</tr>
<tr>
<td>world_overridable</td>
<td>This type has no key.</td>
</tr>
</tbody>
</table>

Examples of audit filter names are principal admin, group dce, and world.

operation

The name of the audfilter operation for which to display help information.
Description

The `audfilter` object represents audit event filters, which consist of a list of guides. Audit event filters are kept by the audit daemon and used to determine whether an auditable event should be logged. An audit filter name consists of a filter type and possibly a key (dependent on the type).

This command operates on the audit daemon named by the `_s(aud)` convenience variable. If the variable is not set, the command operates on the audit daemon on the local host.

Several `audfilter` operations add and remove guide data that is stored in a filter. A guide specifies which action to take when a particular audit condition occurs. A single filter can contain multiple guides, each specifying various actions for different conditions. A guide is identified by a list of the three elements that make up the guide: audit conditions, audit actions, and event classes. Essentially, a guide specifies what (event classes) to audit, when (audit conditions), and how (audit actions). Note that event classes are definable by the administrator.

Audit Conditions

The possible audit conditions are as follows:

- **success** Audits only if the event succeeded.
- **denial** Audits only if the event failed due to access denials.
- **failure** Audits only if the event failed due to other reasons.
- **pending** Outcome not yet determined.

Audit Actions

The possible audit actions are as follows:

- **alarm** Sends the audit record to the system console.
- **all** Logs the event and signal the alarm. If all is set, the `show` operation returns the action `all`, not `{log alarm all}`.
- **ems** Sends the event to the Event Management Service
- **log** Logs the audit record either in the audit trail file of the Audit daemon or in a user-specified audit trail file.
- **none** Takes no audit action.

Errors

A representative list of errors that might be returned is not shown here. Refer to the *IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide* for complete descriptions of all error messages.
audfilter catalog

Returns a list of names of all filters in the audit daemon. The syntax is as follows:

audfilter catalog

The catalog operation returns a list of names of all filters maintained by the audit daemon. It takes no arguments. The names are a list of a type and, if necessary, a key. They are returned in an arbitrary order.

Privileges Required

No special permissions are required to use the catalog operation.

Examples

dcecp> audfilter catalog
{principal melman}
{foreign_principal /.../cell_X/kevins}
{group dce}
world
dcecp>

audfilter create

Creates a new audit filter. The syntax is as follows:

audfilter create  audit_filter_name_list -attribute  guide_name_list

Options

-attribute  guide_name_list
  Specifies a list of one or more guides to be added to the specified audit event filters that are created. A guide name consists of three elements: an event class, an audit condition, and an audit action.

  See Data Structures for more information about guide names.

The create operation creates a new audit filter. The argument is a list of names of audit filters to be created. Since a filter that has no guides is removed by the audit daemon during a clean-up (garbage collection) phase, this command requires an -attribute option whose value is a list of guides to be added to the specified audit filters on creation. All guides are added to all audit filters specified to be created. The operation returns an empty string on success.

Privileges Required

You must have w (write) permission on the audit daemon, and you must be authenticated.

Examples

dcecp> audfilter create {principal melman} -attribute {dce_sec_query denial log}
dcecp>
**audfilter delete**

Deletes the filter including all filter guides. The syntax is as follows:

```
audfilter delete audit_filter_name_list
```

The `delete` operation deletes the filter, including all filter guides. The argument is a list of names of audit filters to be deleted. The operation returns an empty string on success.

**Privileges Required**

You must have **w** (**write**) permission on the audit daemon, and you must be authenticated.

**Examples**

```
dcecp> audfilter delete {principal jones}
dcecp>
```

**audfilter help**

Returns help information about the **audfilter** object and its operations. The syntax is as follows:

```
audfilter help [operation | -verbose]
```

**Options**

- **-verbose**
  
  Displays information about the **audfilter** object.

Used without an argument or option, the **audfilter help** command returns brief information about each **audfilter** operation. The optional **operation** argument is the name of an operation about which you want detailed information. Alternatively, you can use the **-verbose** option for more detailed information about the **audfilter** object itself.

**Privileges Required**

No special privileges are needed to use the **audfilter help** command.

**Examples**

```
dcecp> audfilter help
catalog        Returns the list of filters for an audit daemon.
create         Creates a new filter with specified guides.
delete          Deletes a filter and its associated guides.
modify         Adds or removes one or more guides of a filter.
show           Returns a list of guides in a specified filter.
help           Prints a summary of command-line options.
operations     Returns a list of the valid operations for this command.
dcecp>
```

```
dcecp> audfilter help modify
-add            Adds guides to the specified filters.
-remove         Removes guides from the specified filters.
dcecp>
```
audfilter modify

Adds or removes one or more guides of a filter. The syntax is as follows:

```
audfilter modify  audit_filter_name_list
[-add  guide_name_list]
[-remove guide_name_list]
```

Options

- **-add guide_name_list**
  Specifies a list of one or more guides to be added to the specified audit event filters that are to be modified. A guide name consists of three elements: an audit condition, an audit action, and an event class.

  See Data Structures for more information about guide names.

- **-remove guide_name_list**
  Specifies a list of one or more guides to be removed from the specified audit event filters that are to be modified. A guide name consists of three elements: an audit condition, an audit action, and an event class.

  See Data Structures for more information about guide names.

The modify operation adds or removes one or more guides of a filter. The argument is a list of names of audit filters to be modified. In addition, the specific operation to perform is described with one or more of the following options: **-add** and **-remove**. The argument to both options is a list of guides. If more than one guide is specified, all guides are operated on, but not atomically. If the last guide is removed from a filter, the filter is deleted at some point by the audit daemon.

Atomicity of multiple actions is not guaranteed.

Similarly, the effect of adding a guide that partially exists in the specified filter is to change the existing guides. These changes guarantee that the semantics of the removal/addition are maintained. The operation returns an empty string on success.

Privileges Required

You must have **w** (write) permission on the audit daemon, and you must be authenticated.

Examples

```
dcecp> audfilter modify {principal jones} \
    -add {dce_dts_mgt_modify failure alarm} \ 
    -remove {dce_dts_mgt_query all log}
dcecp>
```

audfilter operations

Returns a list of the operations supported by the audfilter object. The syntax is as follows:

```
audfilter operations
```

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required
No special privileges are needed to use the `audfilter operations` command.

**Examples**

```
dcecp> audfilter operations
catalog create delete modify show help operations
```

**audfilter show**

Returns a list of guides in a specified filter. The syntax is as follows:

```
audfilter show  audit_filter_name_list
```

The `show` operation returns a list of guides in a specified filter. The argument is a list of filter names (a filter type, and if needed, a key) to be shown. If more than one is entered, the output is concatenated and a blank line inserted between filters.

**Privileges Required**

You must have `r` (read) permission on the audit daemon, and you must be authenticated.

**Examples**

```
dcecp> audfilter show {principal rousseau}
{dce_dts_mgt_modify failure alarm}
{dce_dts_mgt_query all log}
dcecp>
```

**Related Information**

Commands: `aud(8dce)`, `audevents(8dce)`, `auditd(8sec)`, `audtrail(8dce)`, `dcecp(8dce)`.

Files: `aud_audit_events(5sec)`, `dts_audit_events(5sec)`, `event_class(5sec)`, `sec_audit_events(5sec)`.
**Purpose**

A dcecp object that converts the audit trail into a readable format.

**Synopsis**

```bash
audtrail help [operation | -verbose ]
```

**Audtrail operations**

```bash
```

**Arguments**

- `audit_trail_file_name_list`:
  A list of one or more names of audit trail files. The names can be either full path names or path names relative to the current working directory.

- `operation`:
  The name of the `audtrail` operation for which to display help information.

**Description**

The `audtrail` object represents an audit trail file. This command currently supports only one operation, which converts the audit trail into a human readable format.

**Errors**

A representative list of errors that might be returned is not shown here. Refer to the *IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide* for complete descriptions of all error messages.

**Operations**

**Audtrail help**

Returns help information about the `audtrail` object and its operations. The syntax is as follows:

```bash
audtrail help [operation | -verbose ]
```

**Options**

- `-verbose`
  Displays information about the `audtrail` object.

Used without an argument or option, the `audtrail help` command returns brief information about each `audtrail` operation. The optional `operation` argument is the name of an operation about which you want detailed information. Alternatively, you can use the `-verbose` option for more detailed information about the `audtrail` object itself.
Privileges Required

No special privileges are needed to use the audtrail help command.

Examples

dcecp> audtrail help
show Returns or files the contents of an audit trail file.
help Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp>

audtrail operations

Returns a list of the operations supported by the audtrail object. The syntax is as follows:

audtrail operations

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the audtrail operations command.

Examples

dcecp> audtrail operations
show help operations
dcecp>

audtrail show

Returns information in the audit trail in a readable format. The syntax is as follows:


Options

-to filename
   Specifies the name of the file in which to store the audit trail output.

-before date
   Shows all of the event records recorded before the specified date.

-after date
   Shows all of the event records recorded after the specified date.

-event event_id
   Shows all of the event records that have the same event type. The event id can be specified either as an integer or as a hexadecimal. If specified as a hexadecimal, the prefix 0x must be used.

-reverse
   Shows all of the event records from the newest to the oldest date.

-first
   Extracts the first event record from the audit trail.

-last
   Extracts the last event record from the audit trail.
-notranslate
Do not translate the event-specific information into human understandable form. Event-specific information is translated by default.

-uuid
Display any UUIDs that are encountered along with any names associated with them (names are always displayed when available).

The **show** operation returns the audit trail in a readable format. This command takes as an argument a list of names of audit trail files. If more than one name is given, the output of each audit trail is concatenated and a blank line inserted between audit trails. The **-to** option specifies a destination filename for the trail. If this option is not present, the trail is returned from the command. If the option is present, this operation returns an empty string.

Because audit trail files can grow quite large, using the **-to** switch is strongly recommended to avoid reading the entire trail into memory.

Note that when **dcecp** processes output, it sends the entire set of returned information to an internal buffer before displaying it. Therefore, when the output is directed to the screen, it can take a long time to appear.

**Privileges Required**

You must have **r** (read) permission on the audit trail file on the local file system.

**Examples**

```
dcecp> audtrail show my_trail

--- Event Record number 1 ---
  o Event Information:
    - Event Number:       0x307 /* 775 */
    - Event Name:         EVT_REWIND
    - Event Outcome:      success
  o Server UUID:         00000000-0000-0000-0000-000000000000
  o Client UUID:         00000064-c914-21cf-a700-10005ab1418b
  o Cell UUID:           e4982482-c914-11cf-a7d0-10005ab1418b
  o Number of groups:    8
    o Group UUID:         0000000c-c914-21cf-a701-10005ab1418b
    o Group UUID:         0000000c-c914-21cf-a701-10005ab1418b
    o Group UUID:         00000066-c914-21cf-9e01-10005ab1418b
    o Group UUID:         00000066-c914-21cf-9e01-10005ab1418b
    o Group UUID:         00000067-c914-21cf-9e01-10005ab1418b
    o Group UUID:         00000068-c914-21cf-9e01-10005ab1418b
    o Group UUID:         00000069-c914-21cf-9e01-10005ab1418b
  o Authorization Status: Authorized with a pac
  o Date and Time recorded: 1996-08-15-17:21:49.659-05:00
--- End of Event record number 1 ---

--- Event Record number 2 ---
  o Event Information:
    - Event Number:       0x308 /* 776 */
    - Event Name:         EVT_STOP
    - Event Outcome:      success
  o Server UUID:         00000000-0000-0000-0000-000000000000
  o Client UUID:         00000066-c914-21cf-a700-10005ab1418b
  o Cell UUID:           e4982482-c914-11cf-a7d0-10005ab1418b
  o Number of groups:   Nil
  o Authorization Status: Authorized with a name
  o Date and Time recorded: 1996-08-16-09:10:37.427-05:00
--- End of Event record number 2 ---
```
Related Information

Commands: aud(8dce), audevents(8dce), auditd(8sec), audfilter(8dce), dcecp(8dce).

Files: aud_audit_events(5sec), dts_audit_events(5sec), event_class(5sec), sec_audit_events(5sec).
The `cds` object allows some low-level control over a CDS server in the local cell. Using it, you can disable a running server, which causes it to shut down gracefully. This command will also display a limited set of the attribute and counter information currently known to the specified server.

### Attributes

- **Child_Update_Failures**
  The number of times the server failed to update a child replica.

- **Creation_Time**
  The date-time stamp representing when the current server started.

- **Crucial_Replcas**
  The number of crucial replicas known to the server.

- **Future_Skew_Time**
  The skew time allowed the server.

- **Known_Clearinghouses**
  The list of clearinghouses known to the server.

- **Read_Operations**
  The number of read operations processed by the server.
Security failures
The number of times the CDS server had problems with the cell Security Service.

Skulks completed
The number of skulks completed by the server since it started.

Skulks initiated
The number of skulks initiated by the server since it started.

Times lookup paths broken
The number of times the lookup path was broken during a server operation.

Write operations
The number of write operations processed by the server since it started.

Errors
A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Operations

cds disable
Disables the specified CDS server. The syntax is as follows:

\texttt{cds disable server}\_name

The specified server must be running somewhere in the local cell, and you must have the privileges to access that machine. This operation returns an empty string on success.

Privileges required
You must have \texttt{dwc (delete, write, create)} permissions on the namespace entry of the server.

Example

\texttt{dcecp> cds disable .:/hosts/blech/cds-server}
\texttt{dcecp>}

\texttt{cds help}

Returns help information about the \texttt{cds} object and its operations. The syntax is as follows:

\texttt{cds help[operation | -verbose]}

Options
- \texttt{-verbose}

Displays information about the \texttt{cds} object.

Used without an argument or option, the \texttt{cds help} command returns brief information about each \texttt{cds} operation. The optional \texttt{operation} argument is the name
of an operation about which you want detailed information. Alternatively, you can use the `-verbose` option to display detailed information about the `cds` object itself.

**Privileges Required**

No special privileges are needed to use the `cds help` command.

**Examples**

```bash
dcecp> cds help
disable  Disables the specified CDS server.
show     Returns attribute information about the named CDS server.
help     Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp>
```

`cds operations`

Returns a list of the operations supported by the `cds` object. The syntax is as follows:

```bash
cds operations
```

The list of available operations is in alphabetical order except for `help` and `operations`, which are listed last.

**Privileges Required**

No special privileges are needed to use the `cds operations` command.

**Examples**

```bash
dcecp> cds operations
disable show help operations
dcecp>
```

`cds show`

Returns attribute information about the specified CDS server. The syntax is as follows:

```bash
cds show  server_name
```

The attributes returned mostly represent counter information, which can be used to help isolate a problems with a CDS server. The order in which the attributes are returned is fixed within CDS.

**Privileges Required**

You must have `r (read)` permissions on the namespace entry of the server.

**Examples**

```bash
dcecp> cds show /.:/hosts/blech/cds-server
{Creation_Time 1995-10-11-10:06:31.434-04:00I-----}
{Future_Skew_Time 0}
{Read_Operations 141384}
{Write_Operations 3589}
```
Related Information

Commands: \texttt{cdsd(8dce)}, \texttt{dcecp(8dce)}, \texttt{cdsclient(8dce)}. 
cdsalias

Purpose

A dcep object that lets you manipulate cell names in CDS.

Note: This command is not currently supported.

Synopsis

cdsalias catalog

cdsalias connect

cdsalias create cellalias_name

cdsalias delete cellalias_name

cdsalias help [operation | -verbose ]

cdsalias operations

Arguments

cellalias_name

A single, fully qualified alias name of the cell in the following form:

/.../cellalias_name

operation

The name of the cdsalias operation for which to display help information.

Description

The cdsalias object represents cell names as known by the Cell Directory Service (CDS). This object lets you manipulate alias and preferred names of DCE cells. Each cell has one preferred name. Cells can also have alias names. Currently this object affects only the CDS component. The security server and each host must also be informed of any new cell aliases.

This object can also be used to define a hierarchical relation between one cell and another by connecting the first cell’s root directory namespace into the the second cell’s namespace. When defining a hierarchical relationship, you must use the account command to establish a trust relationship between the cells.

To manipulate alias and preferred names, the CDS_DirectoryVersion attribute must be set to 4.0 or greater. See the Attributes section of the directory command for more information.

Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.
Operations

cdsalias catalog

Returns a list of all defined cell aliases in CDS. The syntax is as follows:

cdsalias catalog

The catalog operation returns a list of all defined cell aliases in CDS. Each alias name is labeled either alias or primary, depending on which name is the current preferred name.

Privileges Required

Requires r (read) permission on the root directory of the cell.

Examples

dcecp> cdsalias catalog
{CDS_CellAliases
{Alias /.../green.cell.osf.org}
{Primary /.../blue.cell.osf.org}}
dcecp>

cdsalias connect

Establishes a hierarchical relationship between two cells. The syntax is as follows:

cdsalias connect

The connect operation creates a hierarchical relationship between two cells. It takes no argument. The current preferred name of the cell is used and the last relative distinguished name (RDN) is removed to identify the parent cell. This operation returns an empty string on success.

Privileges Required

Requires a (auth_info) permission on the the local cell’s root directory. Also, the CDS server principal on the machine containing the master replica of the local cell’s root directory needs i (insert) permission on the parent cell’s root directory.

Examples

dcecp> cdsalias connect
dcecp>

cdsalias create

Creates a new alias cell name in CDS. The syntax is as follows:

cdsalias create cellalias_name

The create operation creates a new alias cell name in CDS. The required argument is a single fully qualified alias name of the cell. This operation returns an empty string on success.

Privileges Required
**cdsalias(8dce)**

Requires a (**auth_info**) permission on the root directory of the cell.

**Examples**

```bash
dcecp> cdsalias create /.../green.cell.osf.org
dcecp>
```

**cdsalias delete**

Deletes an existing alias cell name from CDS. The syntax is as follows:

```bash
cdsalias delete cellalias_name
```

The **delete** operation deletes an existing alias cell name from CDS. The required argument is a single fully qualified alias name of the cell. If the alias name does not exist, an error is returned. You cannot use this command to delete the preferred cell name. This operation returns an empty string on success.

**Privileges Required**

Requires a (**auth_info**) permission on the root directory of the cell.

**Examples**

```bash
dcecp> cdsalias delete /.../green.cell.osf.org
dcecp>
```

**cdsalias help**

Returns help information about the **cdsalias** object and its operations. The syntax is as follows:

```bash
cdsalias help [operation | -verbose]
```

**Options**

- **-verbose**
  Displays information about the **cdsalias** object.

Used without an argument or option, the **cdsalias help** command returns brief information about each **cdsalias** operation. The optional **operation** argument is the name of an operation about which you want detailed information. Alternatively, you can use the **-verbose** option for more detailed information about the **cdsalias** object itself.

**Privileges Required**

No special privileges are needed to use the **cdsalias help** command.

**Examples**

```bash
dcecp> cdsalias help
catalog          Returns the aliases known by CDS for the named cell.
connect          Establishes a hierarchical relationship between two cells.
create           Creates the named cdsalias for the local cell.
```
cdsalias(8dce)

delete Deletes the existing cdsalias from the local cell.
help Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.

cdsalias operations

Returns a list of the operations supported by the cdsalias object. The syntax is as follows:

cdsalias operations

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the cdsalias operations command.

Examples

dcecp> cdsalias operations
catalog connect create delete help operations
dcecp>

Related Information

dcecp(8dce), account(8dce), cellalias(8dce), directory(8dce), hostdata(8dce).
Purpose
A dcecp object that manages a local CDS cache

Synopsis

```
cdscache create server_name -binding server_binding

cdscache delete server_name

cdscache discard [server_name]

cdscache dump

cdscache help [operation -verbose]

cdscache operations

cdscache show server_name {-server -clearinghouse}
```

Arguments

```
opertaion

The name of the cdscache operation for which to display help information.

server_name

The simple name of the cached server machine. A simple name is not a cell-relative name or a fully qualified name (such as /:/hosts/pelican). Some examples of simple names are pelican and hosts/pelican.
```

Description

The cdscache object represents the Cell Directory Service (CDS) cache on the local node. The CDS cache contains information about servers and clearinghouses known to the local machine, and also contains user data about CDS entries that have been read. The create and delete operations apply only to the server information. The show and dump operations can display additional information.

Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Operations

```
cdscache create

Creates knowledge of a server in the local client’s cache. The syntax is as follows:

```
cdscache create server_name -binding server_binding

Options
-binding server_binding

The required -binding option lets you specify the binding information for a CDS server. This option takes a server_binding argument, which is the protocol sequence and network address of the server node. The string format is as follows:

protocol-sequence: network-address

The dcecp format is as follows:

{protocol-sequence network-address}

A protocol-sequence is a character string identifying the network protocols used to establish a relationship between a client and server. Protocol sequences have a specific format that depends on the network address that is supplied in the binding; for example ncacn_ip_tcp (for connection-based protocol) or ncadg_ip_udp (for datagram protocol) The network-address is a string representing the network address of the server node.

The create operation creates knowledge of a server in the local client’s cache. The server_name argument is the simple name of a cached server. (An example of a simple name would be pelican, as opposed to a cell-relative name like /./hosts/pelican.) This command is typically used to provide configuration information manually to a client that cannot configure itself automatically. Providing configuration information manually might be necessary, for instance, to provide the client with addressing information about a server across a WAN. Once the client knows about one server, it can find other servers through referrals. This operation returns an empty string on success.

Privileges Required

You must have w (write) permission to the CDS client system, /./hosts/hostname/cds-clerk.

Examples

The following command creates knowledge of the server pelican in the local client’s cache:

dcecp> cdscache create pelican -binding ncacn_ip_tcp:16.20.15.25

dcecp>

cdscache delete

Removes knowledge of a server that you had specifically created from the local client’s cache. The syntax is as follows:

cdscache delete server_name

The delete operation removes knowledge of a server that was specifically created from the local client’s cache. The required server_name argument is the simple name of a cached server. (An example of a simple name would be pelican, as opposed to a cell-relative name like /./hosts/pelican.) You can delete only servers that you have specifically created with the cdscache create command. This operation returns an empty string on success.
Privileges Required

You must have `w` (write) permission to the CDS client system, 
`/./hosts/dce_hostname/cds-clerk`.

Examples

The following command removes knowledge of the server `gumby` from the client cache:

```bash
dcecp> cdscache delete gumby
dcecp>
```

**cdscache discard**

Discards the contents of the client cache. The syntax is as follows:

```
   cdscache discard [server_name]
```

The `discard` operation discards information in the client cache on the host specified by `server_name`. If you do not specify `server_name`, the operation discards the information from the client cache on the local host. Only a single server name can be specified. This operation returns an empty string on success.

To perform the operation, `cdscache discard` does the following:

1. Brings down the client CDS.
2. Deletes a specific set of files.
3. Restarts the client CDS.

During the process, all cached information is discarded.

Privileges Required

You must have superuser (root) privileges on the CDS client system. No CDS permissions are required.

Examples

The following command discards the contents of the client cache on the local host:

```bash
dcecp> cdscache discard
dcecp>
```

**cdscache dump**

Displays the entire contents of the client cache. The syntax is as follows:

```
   cdscache dump
```

The `dump` operation displays the contents of the client cache on the screen.

Privileges Required

You must have superuser (root) privileges on the CDS client system. No CDS permissions are required.
Examples

The following command displays the contents of the client cache on the screen (the output is not shown in the example):

dcecp> cdscache dump
dcecp>

**cdscache help**

Returns help information about the *cdscache* object and its operations. The syntax is as follows:

```
cdscache help [operation | -verbose]
```

**Options**

- **-verbose**
  
  Displays information about the *cdscache* object.

Used without an argument or option, the *cdscache help* command returns brief information about each *cdscache* operation. The optional *operation* argument is the name of an operation about which you want detailed information. Alternatively, you can use the **-verbose** option for more detailed information about the *cdscache* object itself.

**Privileges Required**

No special privileges are needed to use the *cdscache help* command.

**Examples**

```
dcecp> cdscache help
create          Adds information about named server in local cds cache.
delete          Removes information about named server from local cds cache.
discard         Discards all cdsadv cache information on the specified host.
dump            Dumps all information from local cds cache.
show            Returns information stored in cds cache.
help            Prints a summary of command-line options.
operations      Returns a list of the valid operations for this command.
dcecp>
```

**cdscache operations**

Returns a list of the operations supported by the *cdscache* object. The syntax is as follows:

```
cdscache operations
```

The list of available operations is in alphabetical order except for *help* and *operations*, which are listed last.

**Privileges Required**

No special privileges are needed to use the *cdscache operations* command.

**Examples**
cdscache(8dce)

dcecp> cdscache operations
create delete discard dump show help operations
dcecp>

cdscache show

Returns information about clearinghouses or servers stored in the cache. The syntax is as follows:

\texttt{cdscache show server\_name \{-server \mid -clearinghouse\}}

Options

\textbf{-clearinghouse}

This option displays all the names and values of the attributes in the specified cached clearinghouse. The following are valid attributes:

\textbf{Creation Time}

Specifies the time at which this clearinghouse was added to the cache.

\textbf{Miscellaneous Operations}

Specifies the number of operations other than read and write (that is, skulks, new epochs, and so on) performed by this client on the cached clearinghouse.

\textbf{Read Operations}

Specifies the number of lookup operations of any sort performed by the client on the cached clearinghouse.

\textbf{Towers}

Specifies the protocol sequence and network address of the server that maintains the cached clearinghouse.

\textbf{Write Operations}

Specifies the number of write operations performed by this client on the cached clearinghouse.

\textbf{-server}

This option displays address information of a server in the local client’s cache. The following attributes are valid:

\textbf{Name}

The directory cell name.

\textbf{Towers}

The protocol sequence and network address of the server node.

The \texttt{show} operation displays information about clearinghouses or servers stored in the cache. The required \texttt{server\_name} argument is the simple name of a server or a CDS names of a clearinghouse for which you want to display information. You must use one of the \textbf{-clearinghouse} or \textbf{-server} options to select the information you want to display.

\textbf{Privileges Required}

You must have \texttt{r (read)} permission to the CDS client.

\textbf{Examples}
The following command displays all attributes of the cached clearinghouse 
././claire_ch:

dcecp> cdscache show ././claire_ch -clearinghouse
{CH_Name /.../blue.cell.osf.org/claire_ch}
{Created 1994-10-07-11:41:23.131}
{Others 458}
{Reads 150221}
{Tower {ncacn_ip_tcp 130.105.4.158}}
{Tower {ncadg_ip_udp 130.105.4.158}}
{Writes 162}
dcecp>

The following command displays all attributes of the cached server drkstr.

dcecp> cdscache show drkstr -server
{CH_Name /.../terrapin_cell.osf.org/drkstr_ch}
{Tower {ncacn_ip_tcp 130.105.5.16}}
{Tower {ncadg_ip_udp 130.105.5.16}}
dcecp>

Related Information

Commands: clearinghouse(8dce), dcecp(8dce), directory(8dce), link(8dce),
object(8dce).
cdsclient

Purpose

A dcecp object that represents a Cell Directory Service client.

Synopsis

    cdsclient disable  client_name
    cdsclient help  [operation | -verbose ]
    cdsclient operations
    cdsclient show  client_name

Description

The cdsclient object allows some low-level control over a CDS client in the local cell. Use it to disable a running client by shutting it down the client gracefully and to display a limited set of the attribute and counter information that is currently known to the client.

Arguments

    client_name
    The name of one CDS client running somewhere in the local cell. Specify the client name using one of the formats:

        /.../ cell_name/hosts/dce_hostname
        /cds-clerk

        /./hosts/dce_hostname
        /cds-clerk

    operation
    The name of the cdsclient operation for which to display help information.

Attributes

    Authentication_Failures
    The number of authentication failures encountered by the client since it started.

    Cache_Bypasses
    The number of times the client bypassed the cache when looking for information.

    Cache_Hits
    The number of times the client used the cache when looking for information.

    Creation_Time
    The date-time stamp representing when the current client started.
Miscellaneous_Operations

The number of non-read, non-write operations processed by the client since it started.

Protocol_Errors

The number of protocol errors encountered by the client since it started.

Read_Operations

The number of read operations processed by the client since it started.

Write_Operations

The number of write operations processed by the client since it started.

Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Operations

cdsclient disable

Disables the specified CDS client. The syntax is as follows:

cdsclient disable client_name

The specified client must be running somewhere in the local cell, and you must have the privileges to access that machine. This operation returns an empty string on success.

Privileges Required

You must have d (delete), w (write), and c (create) permissions on the namespace entry of the clerk.

Example

dcecp> cdsclient disable /.:hosts/blech/cds-clerk
dcecp>

cdsclient help

Returns help information about the cdsclient object and its operations. The syntax is as follows:

cdsclient help [operation | -verbose]

Options

-verbose

Displays information about the cdsclient object.

Used without an argument or option, cdsclient help returns brief information about each cdsclient operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the -verbose option to display detailed information about the cdsclient object itself.
cdsclient(8dce)

Privileges Required

No special privileges are needed to use the `cdsclient help` command.

Examples

dcecp> cdclient help
  disable Disables the specified CDS client.
  show Returns attribute information about the named CDS client.
  help Prints a summary of command-line options.
  operations Returns a list of the valid operations for this command.

cdsclient operations

Returns a list of the operations supported by the `cdsclient` object. The syntax is as follows:

cdsclient operations

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the `cdsclient operations` command.

Examples

dcecp> cdclient operations
disable show help operations
dcecp>

cdsclient show

Returns attribute information about the specified CDS client. The syntax is as follows:

cdsclient show client_name

The attributes returned mostly represent counter information, which can be used to help isolate a problems with a CDS client. The order the attributes are returned is fixed within CDS.

Privileges Required

You must have r (read) permissions on the namespace entry.

Example

dcecp> cdclient show /./hosts/blech/cds-clerk
  {Creation_Time 1995-10-11-15:09:45.187-04:001-----}
  {Protocol_Errors 0}
  {Authentication_Failures 0}
  {Read_Operations 78935}
  {Cache_Hits 55007}
Related Information

Commands: \texttt{cdsadv(8cds)}, \texttt{cds(8dce)}, \texttt{dcecp(8dce)}. 
cell

Purpose

A dcecp task object that operates on a DCE cell

Synopsis

- `cell backup [cell_name]`
- `cell catalog [cell_name]`
- `cell help [operation | -verbose ]`
- `cell operations`
- `cell ping [cell_name] [-clients] [-replicas]`
- `cell show [cell_name] [-simplename]`

Arguments

- `cell_name`
  - The name of a single cell to operate on. The name must be a fully qualified cell name such as either of the following:
    - `/../their_cell.goodco.com`  
    - `./`
- `operation`
  - The name of the `cell` operation for which to display help information.

Description

The `cell` task object represents a single DCE cell as a whole, including all machines, services, resources, principals, and so on. The optional `cell_name` argument is a single cell name (not a list of cell names). If omitted, the local cell (`./`) is the default.

Attributes

- `secservers`
  - Each value is the name of a security server in the cell.
- `cdsservers`
  - Each value is the name of a machine running a Cell Directory Service (CDS) server in the cell. The name is the simple name found under `./:/hosts`.
- `dtsservers`
  - Each value is the name of a Distributed Time Service (DTS) server in the cell.
- `hosts`
  - Each value is the name of a host in the cell, including machines mentioned previously as servers; for example, `hosts/machine1`.

See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information about attributes.
Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Operations

cell backup

Backs up the master security database and each clearinghouse with master replicas in the cell. The syntax is as follows:

cell backup [cell_name]

The cell backup command backs up the master security database and each clearinghouse with master replicas in the cell. It requires that dced be running on each of the server hosts. It takes no arguments or options.

Prepare a cell for regular backup operations by changing the access control lists (ACLs) on two of the dced objects on the local machine and setting up an extended registry attribute (ERA) that can specify a backup destination (typically a tape archive). Then add the new attribute to the principals for the master DCE Security Service registry database and all CDS clearinghouses with master replicas that you want to back up. To do this, follow these steps:

1. Modify ACLs on the local hostdata and srvrconf objects to allow the subsys/dce/dced-admin group access by using the following dcecp acl operations:

   dcecp> acl modify hostdata -add {group subsys/dce/dced-admin criI
   dcecp> acl modify srvrconf -add {group subsys/dce/dced-admin criI
   dcecp> acl modify srvrconf -add {group subsys/dce/dced-admin -d-rwx} -i0

2. Create an ERA as a string that specifies a backup destination. Name the ERA /./sec/xattrschema/bckp_dest and the type printstring. Select the ACL manager named principal and set its four permission bits to r (read), m (manage), r (read), and D (Delete) as shown in the following command:

   dcecp> xattrschema create /./sec/xattrschema/bckp_dest \
   > -encoding printstring -aclmgr {principalrmrD}

3. Add the new ERA (bckp_dest) to the principal dce-rgy (the DCE Security Service registry database). Set the value to be the tar filename or the device that is the backup destination, as follows:

   dcecp> principal modify dce-rgy -add {bckp_dest tarfilename_or_device}

4. Add the new ERA (bckp_dest) to the principal /./hosts/ hostname/cds-server (the CDS server). Set the value to be the tar filename or the device that is the backup destination, as follows:

   dcecp> principal modify /./hosts/ hostname/cds-server \
   > -add {bckp_dest tarfilename_or_device}
cell(8dce)

Now, whenever you want to back up your registry database or CDS database, you can simply invoke a `cell backup` command.

You can back up another cell by including the cell name as an argument to the `cell backup` command. Note that you need the necessary permissions in the remote cell. (Refer to the `registry` object reference page for the required privileges.) This command returns an empty string on success.

**Privileges Required**

The `cell backup` command requires that the administrator be logged in as the local superuser (root). It also requires the user to be authenticated to the security service as the cell administrator.

**Examples**

```bash
dcecp> cell backup
```

**cell catalog**

Lists the foreign cells that are known by the specified cell. The syntax is as follows:

```bash
cell catalog [cell_name]
```

The `catalog` operation returns a list of the names of all cells currently registered in the specified cell. The list includes the name of the specified cell itself and of any registered foreign cells. If no `cell_name` is provided, the operation returns cells registered in the local cell.

**Privileges Required**

You must have `r` (read) permission to the `./sec/principal` directory and `r` (read) permission to the specified cell principals.

**Examples**

```bash
dcecp> cell catalog /.: /.../gumby_cell /.../pokey_cell /.../barney_cell
dcecp>
```

**cell help**

Returns help information about the `cell` task object and its operations. The syntax is as follows:

```bash
cell help [operation | -verbose]
```

**Options**

- **-verbose**
  Displays information about the `cell` task object.

Used without an argument or option, the `cell help` command returns brief information about each `cell` operation. The optional `operation` argument is the name
of an operation about which you want detailed information. Alternatively, you can use the \texttt{-verbose} option for more detailed information about the \texttt{cell} task object itself.

\textbf{Privileges Required}

No special privileges are needed to use the \texttt{cell help} command.

\textbf{Examples}

\begin{verbatim}
dcecp> cell help
backup Backs up master security database and master clearinghouses.
catalog Returns the names of the cells known to a cell.
ping Shows the current server status of the cell.
show Shows attributes describing the configuration of a cell.
help Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp>
\end{verbatim}

\textbf{cell operations}

Returns a list of the operations supported by the \texttt{cell} task object. The syntax is as follows:

\begin{verbatim}
dcecp> cell operations
backup catalog ping show help operations
\end{verbatim}

\textbf{cell ping}

Performs quick checks to test whether a cell is running. The syntax is as follows:

\begin{verbatim}
dcecp> cell ping [cell_name] [-clients] [-replicas]
\end{verbatim}

\textbf{Options}

\textbf{-clients}

This option causes the command to ping every machine in the cell. It does this by looping through \texttt{/.:hosts} and doing a \texttt{host ping} on each host name. In case of failure, it generates an error and returns a list of hosts that could not be contacted. On success, it returns \texttt{DCE clients available}.

\textbf{-replicas}

This option causes the command to ping the master security server, each security replica in the cell, all the CDS servers in the cell, and all the DTS
servers in the cell. In case of failure, it generates an error and returns a list of servers that could not be contacted. On success, it returns DCE servers available.

The ping operation performs a quick check to test whether a cell is running.

If called with no option, it pings (using server ping) the master security server, the CDS server that currently holds the write copy of the cell root directory (/..), and all the DTS servers in the cell. In case of failure, it generates an error and returns a list of servers that could not be contacted. On success, it returns DCE services available.

The -replicas option causes the command to ping each security replica and CDS server as well as those mentioned above. In case of failure, it generates an error and returns a list of servers that could not be contacted. On success, it returns DCE servers available.

The -clients option causes the command to ping every machine in the cell. It does this by looping though /../hosts and doing a host ping on the host name. In case of failure, it generates an error and returns a list of hosts that could not be contacted. On success, it returns DCE clients available.

Privileges Required

You must have r (read) permission to the following directories: /../hosts, /../hosts/hostname, and /../subsys/dce/sec.

Examples

The following command tests whether the core services master servers are available:

dcecp> cell ping /.../blue.cell.osf.org
DCE services available
dcecp>

The following command tests whether the core services and their replicas are available:

dcecp> cell ping -replicas
DCE servers available
dcecp>

The following command tests the presence of all DCE hosts in a cell:

dcecp> cell ping -clients
DCE clients available
dcecp>

cell show

Returns attributes describing the configuration of the specified cell. The syntax is as follows:

    cell show  [cell_name] [-simplename]

Options
The `show` operation returns attributes describing the configuration of the specified cell. The returned attributes are as follows:

**secservers**
Each value is the name of a security server.

**cdsservers**
Each value is the name of a machine running a CDS server. The name is the simple name found under `./hosts`.

**dtsservers**
Each value is the name of a DTS server in the cell.

**hosts**
Each value is the name of a host in the cell, including machines mentioned previously as servers; for example, `hosts/machine1`.

See the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide* for more information about attributes.

**Privileges Required**

You must have `r` (read) permission to the following directories in the CDS namespace: `./hosts`, `./hosts/hostname`, and `./susbsys/dce/sec/master`.

**Examples**

```
dcecp> cell show /.../dcecp.cell.osf.org
(secservers
 /.../dcecp.cell.osf.org/subsys/dce/sec/ice
 /.../dcecp.cell.osf.org/subsys/dce/sec/fire)
(cdsservers
 /.../dcecp.cell.osf.org/hosts/frick}
(dtsservers
 /.../dcecp.cell.osf.org/hosts/frick
 /.../dcecp.cell.osf.org/hosts/ice
 /.../dcecp.cell.osf.org/hosts/ninja}
(hosts
 /.../dcecp.cell.osf.org/hosts/frick
 /.../dcecp.cell.osf.org/hosts/ice
 /.../dcecp.cell.osf.org/hosts/ninja}
dcecp>

dcecp> dcecp> cell show -simplename
(secservers
 subsys/dce/sec/ice)
(cdsservers
 hosts/frick}
(dtsservers
 hosts/frick
 hosts/ice
 hosts/ninja}
(hosts
 hosts/frick
 hosts/ice
 hosts/ninja}
dcecp>
```
cell(8dce)

Related Information

dcecp(8dec0, directory(8dce), host(8dce), server(8dce)).
cellalias

Purpose

A dcecp task object that manages cell name aliases.

Note: This command is not currently supported.

Synopsis

cellalias catalog

cellalias create cellalias_name [-force]

cellalias help [operation | -verbose ]

cellalias operations

Arguments

cellalias_name

A single fully qualified alias name for the cell alias in the form:

.../cellalias_name

operation

The name of the cellalias operation for which to display help information.

Description

The cellalias task object allows you to create and display alternative names for cells, known as cell aliases. You can create multiple aliases for a single cell, but only one per cellalias command.

When you create an alias, cellalias does the following:

1. Creates a new principal to represent the cell alias in the registry.
2. Performs a registry verify operation to ensure that all security replicas in the cell are up to date.
3. Creates the specified alias name in CDS using a cdsalias operation.
4. Performs a directory synchronize operation to ensure that all the CDS replicas are up to date.
5. Performs a hostdata operation on each host in the cell for which you are creating the alias.
6. Updates all dceld objects and the dcelocal/dce_cf.db and dcelocal/etc/security/pe_site files to reflect the new alias name. (This action can take a long time to complete in a cell with many hosts.)

Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.
cellalias catalog

Returns a list of all cell alias names for the local cell. The syntax is as follows:

cellalias catalog

In the list of cell alias names, the cell’s primary name (the name assigned when the cell principal was created) is listed first. The alias names are listed after the primary name.

Privilege Required

Requires r (read) permission on the root directory of the cell.

Examples

dcecp> cellalias catalog
/.../gumby
/.../pokey-alias
dcecp>

cellalias create

Creates a new alias for the local cell. The syntax is as follows:

cellalias create cell_alias_name [-force]

Options

-force Ignores errors encountered during execution.

The required cell_alias_name is a single fully qualified name. You must start dced in remote-update mode with the -r option before you use cellalias create. This operation returns an empty string on success.

Privilege Required

Requires a (auth_info) permission on the root directory of the cell.

Examples

dcecp> cellalias create /.../green.cell.rainbow.com
dcecp>

cellalias help

Returns help information about the cellalias task object and its operations. The syntax is as follows:

cellalias help [operation | -verbose]

Options

-verbose Displays information about the cellalias task object.
Used without an argument or option, the `cellalias help` command returns brief information about each cellalias operation. The optional `operation` argument is the name of an operation about which you want detailed information. Alternatively, you can use the `-verbose` option to display detailed information about the cellalias task object itself.

**Privilege Required**

No special privileges are required to use the `cellalias help` command.

**Examples**

```
dcecp> cellalias help
catalog       Returns the cell alias names currently in use.
create        Creates a new alias name for the local cell.
help          Prints a summary of command-line options.
operations    Returns a list of the valid operations for this command.
dcecp>
```

**cellalias operations**

Returns a list of the operations supported by the cellalias task object. The syntax is as follows:

```
cellalias operations
```

The list of available operations is in alphabetical order except for `help` and `operations`, which are listed last.

**Privileges Required**

No special privileges are needed to use the `cellalias operations` command.

**Examples**

```
dcecp> cellalias operations
catalog create help operations
dcecp>
```

**Related Information**

Commands: `account(8dce)`, `cdsalias(8dce)`, `dcecp(8dce)`, `directory(8dce)`, `hostdata(8dce)`, `registry(8dce)`.

---

**cellalias(8dce)**

Chapter 1. DCE Commands 93
clearinghouse(8dce)

clearinghouse

Purpose

A dcecp object that manages a clearinghouse in CDS

Synopsis

```
clearinghouse catalog [cell_name] [-simplename]
clearinghouse create clearinghouse_name_list
clearinghouse delete clearinghouse_name_list
clearinghouse disable clearinghouse_name_list
clearinghouse help [operation | -verbose ]
clearinghouse initiate clearinghouse_name_list -checkpoint
clearinghouse operations

clearinghouse repair clearinghouse_name_list -timestamps

clearinghouse show clearinghouse_name_list [-schema | -all | [-counters] | [-attributes] ]
clearinghouse verify clearinghouse_name_list
```

Arguments

```
cell_name
    The name of a single cell. The name must be a fully qualified cell name as shown in either of the following:

    ./:
    /.../their_cell.goodco.com

clearinghouse_name_list
    A list of one or more names of clearinghouses you want to operate on. A clearinghouse can be specified in either of the following forms:

    ./:/name_ch
    /.../cell_name/name_ch
```

operation

The name of the clearinghouse operation for which to display help information.

Description

The clearinghouse object represents Cell Directory Service (CDS) clearinghouses. Clearinghouses are databases located on CDS server machines that store data (directories, objects, and links) in CDS. The server machines hold files that contain
the actual clearinghouse data. Clearinghouses are also represented in the CDS namespace by an entry that contains information about the clearinghouse.

You must run the `create` operation on the server host where you want to create the new clearinghouse and the `delete`, `disable`, `initiate`, `repair`, and `verify` operations on the host where the clearinghouse to be operated on resides.

## Attributes

The following are the CDS-defined attributes that might be present in CDS clearinghouse objects:

- **CDS_AllUpTo**: Indicates the date and time the clearinghouse object has been updated to reflect the `CDS_CHDirectories` attribute.

- **CDS_CHDirectories**: Specifies the full name and Universal Unique Identifier (UUID) of every directory that has a replica in this clearinghouse.

- **CDS_CHLastAddress**: Specifies the current reported network address of the clearinghouse.

- **CDS_CHName**: Specifies the full name of the clearinghouse.

- **CDS_CHState**: Specifies the state of the clearinghouse. The state **on** indicates the clearinghouse is running and available.

- **CDS_CTS**: Specifies the creation timestamp (CTS) of the clearinghouse.

- **CDS_DirectoryVersion**: Specifies the current version of the directory in the clearinghouse in which the directory was created.

- **CDS_NSCellname**: Specifies the name of the cell in which the clearinghouse resides.

- **CDS_ObjectUUID**: Specifies the UUID of the clearinghouse. This read-only attribute is set by the system when the clearinghouse is created and cannot be modified by the user.

- **CDS_ReplicaVersion**: Specifies the current version of the replica in which the directory was created. The default is **3.0**. If an upgrade has taken place, the value will upgrade to **4.0**.

- **CDS_UpgradeTo**: A single-valued attribute used to control the upgrading of a clearinghouse from one version of CDS to another. By modifying this attribute, the process of upgrading a clearinghouse to a newer version of CDS can be initiated.

- **CDS_UTS**: Specifies the DTS-style, read-only timestamp of the most recent update to an attribute of the clearinghouse. The value is set by the system.
clearinghouse(8dce)

Counters

corruptions
Specifies the number of times that a clearinghouse generated a data
corruption event.

disables
Specifies the number of times that the clearinghouse was disabled since it
was last started.

enables
Specifies the number of times that a clearinghouse was enabled since it
was last started, not including the initial startup.

failedupgrades
Specifies the number of times that upgrades failed when using the
CDS-UpgradeTo attribute.

missingentries
Specifies the number of times the clearinghouse entry missing event was
generated.

reads
Specifies the number of read operations directed to this clearinghouse.

returnedrefs
Specifies the number of requests directed to this clearinghouse that
resulted in the return of a partial answer instead of satisfying the client’s
entire request.

rootunreachables
Specifies the number of times the root lost event was generated by the
clearinghouse.

skulkfailures
Specifies the number of times that a skulk of a directory, initiated from this
clearinghouse, failed to complete (usually because one of the replicas in the
replica set was unreachable).

writes
Specifies the number of write operations directed to this clearinghouse.

See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more
information about clearinghouse attributes and counters.

Errors

A representative list of errors that might be returned is not shown here. Refer to the
IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for
complete descriptions of all error messages.

Operations

clearinghouse catalog
Returns a list of the names of all clearinghouses in a cell. The syntax is as follows:

clearinghouse catalog [cell_name] [-simplename]

Option
-**simplename**  
  Returns a list of clearinghouse names in a cell without prepending the cell name.

The `catalog` operation returns a list of the names of all clearinghouses in a cell. If you do not specify the optional `cell_name` argument, the cell name defaults to the local cell.

**Privileges Required**

No special privileges are needed to use the `clearinghouse catalog` command.

**Examples**

```

dcecp> clearinghouse catalog  
/.../dcecp.cell.osf.org/frick_ch  
dcecp>

dcecp> clearinghouse catalog -simplename  
frick_ch  
dcecp>
```

**clearinghouse create**

Creates a new clearinghouse on the local machine. The syntax is as follows:

```
clearinghouse create clearinghouse_name_list
```

The `create` operation creates a new clearinghouse on the local machine. The `clearinghouse_name_list` argument is a list of one or more names of the clearinghouses you want to create. Clearinghouses should be named only in the root directory (that is, `/:`). This operation also stores a read-only replica of the root directory in the new clearinghouse. The process that creates the new clearinghouse initiates a skulk of the root directory, so all replicas of the root should be reachable when you enter the `clearinghouse create` command. To ensure this, perform an immediate skulk of `/:` before invoking the command, using the `directory synchronize` `/:` command. This operation returns an empty string on success.

**Privileges Required**

You need **w** (**write**) permission to the server on which you intend to create the clearinghouse, and **A** (**Admin**) permission to the cell root directory. The server principal (/,:/hosts/dce_hostname/cds-server) needs **r** (**read**), **w** (**write**), and **A** (**Admin**) permission to the cell root directory.

**Examples**

The following command creates a clearinghouse named `/:Boston_CH` on the local server system:

```
dcecp> clearinghouse create /:/Boston_CH  
dcecp>
```

**clearinghouse delete**

Deletes the specified clearinghouse from the local machine. The syntax is as follows:
clearinghouse delete  clearinghouse_name_list

The delete operation deletes the specified clearinghouse from the local server system. The clearinghouse_name_list argument is a list of one or more names of the clearinghouses you want to delete. Clearinghouses that contain master replicas of directories are not deleted (and also return errors). This command also automatically deletes all read-only replicas from the clearinghouse; however, you should delete all read-only replicas by hand (see directory delete -replica) before invoking this command since invoking many skulks causes the command to execute more slowly. This operation returns an empty string on success.

CDS does not permit you to delete a disabled (cleared) clearinghouse. Before you can do so, re-create the clearinghouse, using the clearinghouse create command.

Privileges Required

You must have w (write) and d (delete) permission to the clearinghouse and A (Admin) permission to all directories that store replicas in the clearinghouse. The server principal (/./hosts/dce_hostname/cds-server) must have d (delete) permission to the associated clearinghouse object entry and A (Admin) permission to all directories that store replicas in the clearinghouse.

Examples

The following command deletes a clearinghouse named /./Orion_CH from the local server system:

dcecp> clearinghouse delete /./Orion_CH
dcecp>

clearinghouse disable

Removes knowledge of the specified clearinghouse from the local server’s memory. The syntax is as follows:

clearinghouse disable  clearinghouse_name_list

The disable operation removes knowledge of the specified clearinghouse from the local server’s memory. The clearinghouse_name_list argument is a list of names of one or more clearinghouses you want to disable. Use this command when relocating a clearinghouse. This command removes the name of the prefix of the clearinghouse files from the /opt/dcelocal/var/directory/cds/cds_files file and notifies the local CDS server that the clearinghouse is disabled. The clearinghouse entry is not removed from the namespace, nor are the datafiles associated with the clearinghouse removed. This operation returns an empty string on success.

Privileges Required

You must have w (write) permission to the CDS server on which the clearinghouse resides.

Examples

The following command disables the clearinghouse /./Paris2_CH so that it can be moved to another server:
clearinghouse help

Returns help information about the clearinghouse object and its operations. The syntax is as follows:

`clearinghouse help [operation | -verbose]`

Options

- `-verbose`
  Displays information about the clearinghouse object.

Used without an argument or option, the `clearinghouse help` command returns brief information about each clearinghouse operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the `-verbose` option for more detailed information about the clearinghouse object itself.

Privileges Required

No special privileges are needed to use the `clearinghouse help` command.

Examples

dcecp> clearinghouse help
catalog    Returns the names of all clearinghouses in a cell.
create    Creates the named clearinghouse.
delete    Deletes the named clearinghouse.
disable    Disables the named clearinghouse.
initiate    Initiates an action on the named CDS clearinghouse.
repair    Repairs an aspect of the named CDS clearinghouse.
show    Returns the attributes of a clearinghouse.
verify    Verifies the consistency of the clearinghouse.
help    Prints a summary of command-line options.
operations    Returns a list of the valid operations for this command.
dcecp>

clearinghouse initiate

Initiates a defined action on the specified clearinghouse on the local machine.

**Note:** This command can be run only on a server machine.
The syntax is as follows:

`clearinghouse initiate clearinghouse_name_list -checkpoint`

Options

- `-checkpoint`
  Forces the clearinghouse to checkpoint to disk.

The initiate operation initiates a defined action on the specified clearinghouse. The required `clearinghouse_name_list` argument is a list of names of clearinghouses you want to initiate actions on. Currently, only a checkpoint action is available. This operation returns an empty string on success.
clearinghouse(8dce)

Privileges Required

You need `w` (write) permission on the clearinghouse server and `A` (admin) permission on the cell root directory. The server principal (/hosts/dce_hostname/cds-server) needs `r` (read), `w` (write), and `A` (Admin) permission on the cell root directory.

Examples

The following command initiates a checkpoint operation on the clearinghouse named `.:/oddball_ch` on the local system.

```
dcecp> clearinghouse initiate `.:/oddball_ch` -checkpoint
```

clearinghouse operations

Returns a list of the operations supported by the clearinghouse object. The syntax is as follows:

clearinghouse operations

The list of available operations is in alphabetical order except for `help` and `operations`, which are listed last.

Privileges Required

No special privileges are needed to use the `clearinghouse operations` command.

Examples

```
dcecp> clearinghouse operations
catalog create delete disable initiate repair show
verify help operations
```

clearinghouse repair

Repairs a specific problem on a specified clearinghouse on the local machine. The syntax is as follows:

clearinghouse repair  `clearinghouse_name_list` -timestamps

Options

- **-timestamps**
  Analyzes and repairs invalid timestamps found in a clearinghouse.

Use the `repair` operation to fix various problems that can occur in a clearinghouse. The required `clearinghouse_name_list` argument is a list of names of clearinghouses you want to initiate repair actions on. Currently, only invalid timestamps can be repaired. This operation returns an empty string on success.

Privileges Required
You need **w** (*write*) permission to the clearinghouse server and **A** (*Admin*) permission to the cell root directory. The server principal (/hosts/dce_hostname/cds-server) needs **r** (*read*), **w** (*write*), and **A** (*Admin*) permission to the cell root directory.

**Examples**

The following command repairs invalid timestamps in a clearinghouse named /.:/blech_ch on the local system.

```
dcecp> clearinghouse repair /.:/blech_ch -timestamps
dcecp>
```

**clearinghouse show**

Returns attribute and counter information associated with the specified clearinghouses on local or remote machines. The syntax is as follows:

```
clearinghouse show clearinghouse_name_list
[-schema | -all | [-counters] [-attributes]]
```

**Options**

- **-schema**
  Indicates whether attributes are single-valued or multivalued.

- **-all**
  Returns the attributes and counters for the clearinghouse.

- **-attributes**
  Returns the attributes for the clearinghouse.

- **-counters**
  Returns the counters for the clearinghouse.

The **show** operation displays attribute and counter information associated with the clearinghouses specified by **clearinghouse_name_list**, which is a list of one or more names of the clearinghouses. If more than one clearinghouse is specified, the attributes of all the clearinghouses are concatenated into one list. The order of the returned attributes is the lexical order of the object identifiers (OIDs) of each attribute for each clearinghouse.

If you supply no options, **clearinghouse show** returns the attributes associated with the specified clearinghouse.

**Privileges Required**

You must have **r** (*read*) permission to the clearinghouse.

**Examples**

```
dcecp> clearinghouse show /.:/drkstr_ch
{CDS_CTS 1994-06-18-20:16:22.150-05:0010.000/00-00-c0-f7-de-56}
{CDS_UTS 1994-06-19-17:17:43.911-05:0010.000/00-00-c0-f7-de-56}
{CDS_ObjectUUID 0066ccea-d978-1db3-8259-0000c0f7de56}
{CDS_AllUpTo 1994-07-01-21:30:18.948-05:0010.000/00-00-c0-f7-de-56}
{CDS_DirectoryVersion 3.0}
{CDS_CHName /.../terrapin/drkstr_ch}
{CDS_CHLastAddress
 {Tower ncacn_ip_tcp 130.105.5.16}
 {Tower ncadg_ip_udp 130.105.5.16}}
```
### clearinghouse(8dce)

```bash
dcecp> clearinghouse show ./:.//Chicago1_CH -counters
corruptions 0
disables 0
enables 1
failedupgrades 0
missing entries 0
reads 2336
returnedrefs 2
rootunreachables 0
skulkfailures 0
writes 68
dcecp>
```

### clearinghouse verify

Verifies the consistency of the specified clearinghouse on the local machine.

**Note:** This command can be run only on a server machine. The syntax is as follows:

```
clearinghouse verify clearinghouse_name_list
```

The **verify** operation verifies the consistency of the specified clearinghouse by checking internal attributes. The required `clearinghouse_name_list` argument is a list of one or more names of clearinghouses you want to verify.

The **clearinghouse verify** operation ensures that a clearinghouse is in a correct state by performing the following actions:

- Skulks directories if it is time to do so based on the value of the `CDS_Convergence` attribute.
- Upgrades directory replicas if a `CDS_UpgradeTo` attribute indicates to do so.
clearinghouse(8dce)

- Ensures that child and parent pointer information is correct. This involves ensuring that the set of replicas stored in the child pointer matches the set of replicas stored in the actual directory.
- Performs any needed IP address changes in the directories. Checks for primary cellname changes in the clearinghouse.
- Removes all old data (replica and attributes) based on the value of the `CDS_AllUpTo` attribute.

This command returns an empty string on success.

**Note:** You cannot use the `clearinghouse verify` command on a remote host. This command is valid for the local host only.

**Privileges Required**

You need `w` (write) permission to the clearinghouse server and `A` (Admin) permission to the cell root directory. The server principal (`/hosts/dce_hostname/cds-server`) needs `r` (read), `w` (write), and `A` (Admin) permission to the cell root directory.

**Examples**

The following command verifies the consistency of clearinghouses named `./:gumby_ch` and `./:pokey_ch`.

```
dcecp> clearinghouse verify { ./:gumby_ch ./:pokey_ch}
dcecp>
```

**Related Information**

Commands: `cdscache(8dce)`, `dcecp(8dce)`, `directory(8dce)`, `link(8dce)`, `object(8dce)`.
clock(8dce)

clock

Purpose

A dcecp object that manages the clock on a local or remote host

Synopsis

clock compare [dts_entity] [-server dts_entity]

clock help [operation | -verbose ]

clock operations

clock set [dts_entity] [-to DTS_timestamp [-abruptly | -epoch epoch_number | -bypass] -epoch epoch_number ]

clock show [dts_entity] [-dtsd | -inetd | -dced ]

clock synchronize [dts_entity] [[-abruptly] | [-dtsd] | [-inetd | -dced ]

Arguments

dts_entity
Identifies the dtsd server or clerk to act on.

With the -server option in the compare operation, dts_entity can identify a DTS time provider.

When used without the -dced or -inetd options, dts_entity can be either of the following:
1. The name of a dtsd server, which can be on a remote host, in the format:

    /.../cellname/hosts/hostname/dts-entity

2. As string binding for the remote host on which the dtsd is running, such as:

    ncacn_ip_tcp:130.105.1.227

    Alternatively you can specify the binding in dcecp string format, such as:

    {ncacn_ip_tcp 130.105.1.227}

    When used with the -dced or -inetd options, dts_entity identifies the server by a simple host name in the form hostname.

operation
The name of the clock operation for which to display help information.

Description

The clock object represents the clock on a system and the time that it tells. This object has commands to display and set the time. The time setting functionality is
provided by DTS, unless you specify either the -dced or -inetd option. The optional argument to the clock command is the name of a DCE Version 1.1 dtsd running on some machine. Without an argument, the _s(dts) convenience variable is checked. If this variable is not set, the command operates on the clock on the local machine.

Note: Time must never be set backwards in the DCE environment. Backwards movement in the clock causes a server to be unable to determine event ordering, resulting in inconsistency in the server’s database and corruption of the timestamps. For more information, see the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide—Core Components

Use the -epoch option to change only the epoch number of the dtsd.

Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Operations

clock compare

Returns the difference between the clocks on the local machine and a DTS server in the cell. The syntax is as follows:

clock compare [dts_entity] [-server dts_entity]

Options

-server dts_entity

Optionally names a specific DTS server against which to compare the host clock.

See Arguments for the format of the dts_entity argument.

The compare operation returns the difference between the clocks on the local machine and a DTS server in the cell. If a server is not specified, the command picks the last responding server returned by dts catalog. An optional argument compares a remote host’s clock against a DTS server. An optional -server option compares the clock against a specific DTS server.

The DTS server that responds to this operation might be communicating directly with an external time provider. If so, the provider attribute returned by this operation will be set to yes.

Privileges Required

You must have r (read) permission on /.:/hosts/ hostname/dts-entity to execute the command.

Examples
clock compare
{server /.:/gumby/hosts/oddball/dts_entity}
{provider no}
{skew -0-00:00:00.020I-----}
dcecp>

dcecp> clock compare -server /.:/hosts/santafe/dts-entity
{server /.:/hosts/santafe/dts-entity}
{provider yes}
{skew -0-00:00:00.292I1.431}
dcecp>

clock help

Returns help information about the clock object and its operations. The syntax is as follows:

clock help  [operation | -verbose]

Options
-verbose

Displays information about the clock object.

Used without an argument or option, the clock help command returns brief information about each clock operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the -verbose option for more detailed information about the clock object itself.

Privileges Required

No special privileges are needed to use the clock help command.

Examples

dcecp> clock help
compare  Returns the difference between the local clock and a server.
set     Sets the system clock to the specified time.
show    Returns the current time as a DTS style timestamp.
synchronize Synchronizes the local clock with the specified server.
help    Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp>

clock operations

Returns a list of the operations supported by the clock object. The syntax is as follows:

clock operations

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the clock operations command.
clock set

Sets the clock to the specified time. The syntax is as follows:

clock set [dts_entity] {-to DTS_timestamp
[-abruptly -epoch epoch_number | -bypass] | -epoch epoch_number}

Options

-to DTS_timestamp
This option specifies a DTS timestamp as the time to which to set the clock. You can specify the time in the ISO-compliant time format, as follows:

CCYY-MM-DD hh:mm:ss.fff

-abruptly
Specifies to set the clock abruptly rather than gradually adjust it to the computed time.

-bypass
Sets the system clock to the specified time without using DTS.

-epoch epoch_number
Specifies an epoch_number that matches the epochs of servers with which the local clock synchronizes.

The set operation sets the local clock to the specified time. An optional argument sets the clock on a remote host. The -to option specifies a DTS timestamp as the time to which to set the clock. If you do not specify the -abruptly option, DTS adjusts the clock gradually to the specified time. The -abruptly option changes to the specified time, without gradual adjustments. If you specify the -abruptly option, you must also specify the -epoch option to indicate a new epoch. You can also use the -epoch option without specifying a time to pull the specified dts_entity out of synchronization. The -bypass option causes DTS to be ignored and sets the system clock directly. This operation returns an empty string upon success.

Note: Time must never be set backwards in the DCE environment. Backwards movement in the clock causes a server to be unable to determine event ordering, resulting in inconsistency in the server’s database and corruption of the timestamps. For more information, see the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide—Core Components

Note that setting your system clock is a dangerous operation. If your machine is not synchronized with other machines in the cell, other DCE services, especially CDS, do not operate correctly.

Privileges Required

You must have w (write) permission on the clock object (/./hosts/hostame/dts-entity) if using DTS to set the time; otherwise no special privileges are required.

Examples
clock set -to 1994-07-15-16:28.000-04:00 -abruptly -epoch 1
dcecp>
dcecp> clock set -epoch 5
dcecp>

**clock show**

Returns a DTS-style timestamp including the time differential factor (TDF). The syntax is as follows:

```
clock show  [dts_entity] [-dtsd | -inetd | -dced]
```

**Options**

- **-dced** Use dced services instead of DTS to report the time.
- **-inetd** Use inetd socket connections instead of DTS to report the time.
- **-dtsd** Use DTS services to report the time.

The show operation returns a DTS-style timestamp with the TDF indicated. Use the dts_entity argument to specify a remote host on which to show the clock.

Two options let you specify that the time should be returned without using DTS services:
1. The **-dced** option specifies that dced services should be used instead of DTS services
2. The **-inetd** option specifies that inetd socket connections should be used instead of DTS

**Privileges Required**

You must have r (read) permission on the clock object (/:/hosts/hostname/dts-entity) if using DTS to show the time; otherwise no special privileges are required.

**Examples**

```
dcecp> clock show
1994-07-15-16:28:02.299+00:001------
dcecp>
```

```
dcecp> clock show oddball -dced
1994-07-16-17:29:05.321+00:001------
dcecp>
```

**clock synchronize**

Causes dtsd to synchronize gradually with a server. The syntax is as follows:

```
clock synchronize  [dts_entity] [[-abruptly]
[-dtsd] | -inetd | -dtsd]
```

**Options**

- **-abruptly**
  Causes the clock to be set abruptly rather than gradually adjusted to the computed time.
Use dced services instead of DTS as the time source.

Use inetd socket connections instead of DTS as the time source.

Use DTS services as the time source.

The synchronize operation causes the local dtsd to synchronize the local clock gradually with the cell time from DTS servers. The -abruptly option changes to the specified time immediately, without gradual adjustments.

By default, the time is retrieved from DTS. If the -dced option is specified, the time is retrieved from dced services. If the -inetd option is specified, the time is retrieved from inetd socket connections. The optional dts_entry argument synchronizes the clock on the named remote host. This operation returns an empty string on success.

Privileges Required

You must have w (write) permission on the clock object (/hosts/hostname/dts-entity) if using DTS to synchronize the time; otherwise no special privileges are required.

Examples

dcecp> clock synchronize

dcecp>

Related Information

Commands: dcecp(8dce), dts(8dce), dtsd(8dts), utc(8dce).
Purpose

Builds a DCE character and code set registry on a host

Synopsis

```
csrc [-i source_filename] [-o destination_filename]
```

Arguments

- `-i source_filename`
  Reads code set values from the source file you specify rather than from the default code set registry source file `/var/dce/nls/csr/code_set_registry.txt`.

- `-o destination_filename`
  Places the generated code set registry file in the location you specify rather than in the default location `/var/dce/nls/csr/code_set_registry.db`.

Description

The code set registry compiler `csrch` creates a character and code set registry file from the information supplied in a character and code set registry source file.

A code set registry source file is composed of a series of code set records. Each record describes, in human-readable form, the mapping between an OSF-registered or (optionally) a user-defined unique code set value and the character string that a given operating system uses when referring to that code set (called the local code set name).

A code set registry file is the binary version of the source file; the DCE RPC routines for character and code set interoperability use the file to obtain a client’s or a server’s supported code sets and to translate between operating system-dependent names for code sets and the unique identifiers assigned to them. A code set registry file must exist on each host in an internationalized DCE cell (a DCE cell that supports applications that use the DCE RPC character and code set interoperability features).

Creating the Source File

Code set registry source files are created for input to `csrch` in the following two instances:

1. By DCE licensees, when they are porting DCE to a specific operating system platform and plan for their DCE product to support internationalized DCE applications. In this instance, DCE licensees modify a template code set registry source file supplied on the DCE source tape to contain, for each code set that their platform supports, the local code set names for those supported code sets. Licensees can also add to this file any vendor-specific, nonOSF registered code set names and values that their platform supports.

2. Although IBM does not recommend it, cell administrators can modify the code set registry, when they are configuring machines that are part of an internationalized DCE cell.
internationalized DCE cell. In this instance, the cell administrator can add to each platform-specific source file any administrator-defined, non-OSF registered code set names and values.

Each code set record specifies one code set, and has the following form:

```plaintext
start
   field_list
end
```

where `field_list` consists of the following keyword-value or keyword-text pairs:

**description text**
A comment string that briefly describes the code set. The text field can contain multiple lines; use the backslash character ( \ ) to continue the line. Use this field to give a detailed description of the code set and character set(s).

**loc_name text**
A maximum 32-byte string (31 character data bytes plus a terminating NULL) that contains the operating system-specific name of a code set or the keyword NONE. Use this field to specify the name that your site uses to refer to this code set and the code set converters associated with it. For example, on UNIX platforms, code set converters are usually implemented under the `iconv` scheme. Check the `iconv` converter directory to determine the code set names.

**rgy_value value**
A 32-bit hexadecimal value that uniquely identifies this code set. A registry value can be one that OSF has assigned or one that a DCE licensee or cell administrator has assigned. Cell administrator-defined values must be in the range 0xf5000000 through 0xffffffff. The administrator must ensure that assignments in this range are unique.

**char_values value[: value]**
One or more 16-bit hexadecimal values that uniquely identify each character set that this code set encodes. A character value can be one that OSF has assigned or one that a DCE licensee or a cell administrator has assigned. Use the : (colon) to separate multiple character set values.

**max_bytes value**
A 16-bit value that specifies the maximum number of bytes this code set uses to encode one character. The count should include any single-shift control characters, if used.

In the source file, braces ({ }) can be used as synonyms for the `start` and `end` keywords. Use one or more spaces or tabs to separate field names and values. An unquoted # (number sign) introduces a comment; in this case, the `csrc` utility ignores everything between the comment character and the end of the line.

If cell administrators modify the code set registry, for each modified registry they must replace the NONE keyword with the local code set names for any site-specific supported code sets.

The following is an excerpt from the OSF-supplied code set registry source file:

```plaintext
start
description ISO 8859:1987; Latin Alphabet No. 1
loc_name NONE
```
Generating the Code Set Registry File

DCE licensees use `csrc` to create licensee-supplied code set registry files for their internationalized DCE product. Cell administrators of internationalized DCE cells can use the `csrc` utility to create site-specific code set registry files for each host in the cell. The cell administrator runs the `csrc` program on each host in the cell. In general, it should not be necessary for administrators to modify the registry.

When invoked with no options, `csrc` uses the default source file `/var/dce/nls/csr/code_set_registry.txt` and creates the default output file `/var/dce/nls/csr/code_set_registry.db`. Use the `-i` and `-o` options to redirect `csrc` to use a specific source file or generate a specific output file. The `csrc` utility also generates a log file named `CSRC_LOG` in the current directory.
Restrictions

You need **w** (write) permission to the **/var/dce/nls/csr** directory, which usually requires **root** privilege.

In the following example, the log file **CSRC_LOG** is created in the current working directory, **testi18n_app**:

```
csrc -i /test/i18n_app/code_set_registry.txt -o code_set_registry.db
```

Errors

A representative list of errors that might be returned is not shown here. Refer to the *IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide* for complete descriptions of all error messages.

/\texttt{/var/dce/nls/csr/code_set_registry.txt}

Default pathname for code set registry source file.

/\texttt{/var/dce/nls/csr/code_set_registry.db}

Default pathname for code set registry object file.

Related Information

Functions: **dce_cf_get_csrgy_filename(3dce)**, **dce_cs_loc_to_rgy(3rpc)**,**dce_cs_rgy_to_loc(3rpc)**, **rpc_rgy_get_codesets(3rpc)**.


---

csrc(8dce)

Chapter 1. DCE Commands 113
dceagtd

**dceagtd (AIX only)**

**Purpose**

The DCE SNMP subagent.

**Synopsis**

```
```

**Options**

- **-h hostname**
  This is the name of the host that sends requests. The default name is the name of the local host.

- **-c community**
  This option is used to specify the community name. The default community name is dcesnmp.

- **-d debuglevel**
  This option specifies the debug level. The valid values are 0–9. If you enter a value of 1–9, additional SNMPD debug information is displayed. If you enter a value of 0, the debug information is not displayed. However, the zero value does cause the time intervals to be shorten in order of magnitude. For example, `-d 0 -p 60`, changes the heartbeat poll interval to 60 seconds.

- **-p heartbeat_poll_interval**
  This option specifies the subagent heartbeat poll interval in minutes. The default time is 10 minutes. If you enter a value of 0 value, then all subagent heartbeat polling is disabled. However, you can change it by using the SNMP set function on the MIB variable, `aSubagtHeartbeatInterval`. For example, the following will change the DCE configured server status heartbeat poll interval to 2 hours:

  ```
  snmpinfo -c dcesnmp -m set 1.3.22.1.7.1.3.3.0=120
  ```

  This pollarization is used to determine how often to check the configured servers status for changes since the last check. Any changes in status will result in an SNMP trap being sent back to the SNMP manager.

- **-l bin_log_poll_interval**
  This option specifies the subagent BIN.LOG interval in minutes. The default is 10 minutes. A 0 value will disable all subagent BIN.LOG polling. However, this can be changed by using the SNMP set function on the MIB variable, `aSubagtBinLogInterval`. For example, the following will change the DCE subagent BIN.LOG poll interval to 24 hours.

  ```
  snmpinfo -c dcesnmp -m set 1.3.22.1.7.1.3.4.0=1440
  ```

- **-?**
  Shows the command syntax.

If you do not specify the parameters on the command line, then the DCE SNMP subagent uses the default values.
Description

This is the DCE SNMP subagent and is a daemon or server in the same sense as any of the other DCE daemon executables except that the subagent does not require DCE to be active or configured.

Note: Although DCE does not need to be configured, the subagent, itself, should be configured.
dceagtd

**dceagtd (Solaris only)**

**Purpose**

The DCE SNMP subagent.

**Synopsis**

```
```

**Options**

- **-h**  This option displays the command syntax as shown below:

  Usage: dceagtd [-h]

  [-k (don't read config file)]

  [-p port ]

  [-c config-file (default is <install_path>/opt/dcelocal/etc/dceagtd.reg)]

  [-a sec-config-file (default is <install_path>/opt/dcelocal/etc/dceagtd.acl)]

  [-i poll-interval (default is 30 seconds)]

  [-d trace-level (range 0..4, default is 0)]

  **Note**: The reference, `<install_path>`, is the install location (for example, `/opt/dcelocal`).

- **-k**  This option will cause the config-file to not be read. It is suggested that you not use this option. The subagent will not be fully functional unless the config-file is read.

- **-p**  This option is used to specify a specific port number. The subagent assigns a port number each time it is started. The port number might or might not remain the same. It is very likely that it will be different.

- **-c**  This option is used to change the config-file used by the subagent. The default is `<install_path>/etc/dceagtd.reg`, where `dceshared_path` is the install location (for example, `/opt/dcelocal`).

- **-a**  This option is used to change the security-config-file used by the subagent. The default is `<install_path>/etc/dceagtd.acl`, where `dceshared_path` is the install location (for example, `/opt/dcelocal`).

- **-i**  This option sets an internal poll interval (in microseconds) that is used by the Master Agent to determine how long to wait for a response to a request sent to this subagent. The default is 30 seconds.

  The `-p` and `-l` options that are available on the AIX version are now available through the `<install_path>/etc/snmptrap.tbl` file. The subagent reads this file when it is started. The “HEARTBEAT=” and “BINLOG=” entries are used to set the initial poll intervals in minutes, respectively. If “TRACE=1” is also set, then the poll intervals are measured in seconds and tracing information is written to `<install_path>/var/sysmgmt/adm/dceagtd/dceagtd.trace`.

- **-d**  This option will allow tracing of the SNMP information being sent between the subagent and Master Agent.
Description

This is the DCE SNMP subagent and is a daemon or server in the same sense as any other DCE daemon executable except that the subagent does not require DCE to be active or configured.

If you do not configure the subagent through config.dce and wish to start the subagent yourself, add the following line to the /etc/snmp/conf/enterprises.oid file:
"ibmDce" "1.3.6.1.4.1.2.11.20"

This is only needed if you want subagent SNMP traps to be sent out.
dcecp

Purpose

Administrative interface for DCE management tasks

Synopsis

```
dcecp [-s] [-local] [script_name | -c command ]
```

- **-c command**
  A list containing one or more valid dcecp commands. For a description of the dcecp command format, see Administration Objects.

- **-s**
  Turns off inheritance of the login context. The default is to inherit the current login context of the principal that invokes dcecp.

- **-local**
  The -local option specifies that the dcecp session should operate on the local dced object while the dced object is in a partial-service state.

Arguments

```
script_name
```

Filename of a user-defined script containing dcecp commands.

Description

The DCE control program, dcecp, is the primary DCE administration interface, providing local and remote access to routine DCE administrative functions from any DCE Version 1.1 and later platform.

The DCE control program is built on a portable command language called the tool command language (Tcl). Tcl allows the use of variables, if statements, list processing functions, loop functions and many other features commonly found in command languages. The control program extends these features, providing a set of commands for manipulating specific DCE objects. The control program also includes task scripts to help administrators perform some routine DCE management functions. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide—Core Components for information about the basic concepts and features of dcecp. All of Tcl is included in the dcecp language.

Invoking and Terminating dcecp

The DCE control program allows you to invoke dcecp commands in the following modes:

1. Interactive mode
2. Command-line mode

**Interactive Mode**

Activate interactive mode by entering the dcecp command without any arguments. At the dcecp prompt, enter a dcecp or Tcl command; dcecp executes the command, displays the result, and is ready to accept another command.
Command-Line Mode

Activate command-line mode from the system prompt by using one of the following methods:

1. Enter the `dcecp` command with a filename of a script containing `dcecp` commands, other valid Tcl commands, or both, as follows:

   `% dcecp myown.Tcl`

2. Enter the `dcecp` command with the `-c` option followed by a list containing one or more `dcecp` commands, as follows:

   `% dcecp -c directory create /.:/admin/printers`

Enter multiple `dcecp` commands by separating them with a `;` (semicolon) and enclosing the commands in `""` (quotation marks). Remember to escape shell metacharacters (for example by enclosing them in quotation marks). Multiple commands must be on a single line, as follows:

   `% dcecp -c "directory create /.:/admin/printers; \
    directory show /.:/admin/printers"`

When you use the `-c` option, operation results return to the interpreter, not to the shell. If you enter multiple operations, the output of only the last operation is returned to the shell. This problem can be overcome by using the following ugly, but serviceable workaround:

   `% dcecp -c "puts [dir help]; puts [principal help]"

Terminate an interactive `dcecp` session by using the `exit` and `quit` commands. Use the following command syntax:

   `exit n  quit  n`

Use the `n` argument to specify the exit value returned to the shell. The following example terminates a session and returns an exit value of 56 to the shell:

   `exit 56`

By default, `dcecp` returns 0 (zero) on success and 1 (one) if a command fails.

Startup Scripts

When you invoke `dcecp`, the following script files are executed in the order shown:

- `[info library]/init.tcl`
  Contains the standard Tcl initialization scripts with definitions for the `unknown` command and the `auto_load` facility.

- `$dcecp_library/init.dcecp`
  Contains the initialization scripts implementing the `dcecp` commands and tasks. The implementation sets the Tcl variable `dcecp_library` to `dceshared/dcecp` by default.
Administration Objects

A `dcecp` command has the following syntax:

```
object operation [argument] [-option [opt_arg]] ...
```

where:

- **object**: Specifies the name of a `dcecp` administration object. Examples of administration objects are Cell Directory Service (CDS) directories, access control lists (ACLs), Distributed Time Service (DTS) servers, server control objects, and so on. Each administration object is briefly described below.

- **operation**: Specifies the name of an action such as `create`, `show`, or `remove`, that is to be performed on an administration object. For complete descriptions of operations supported by each `dcecp` object, refer to individual object reference pages. Common operations are briefly described below.

- **argument**: Specifies the name of one or more specific objects to operate on. Most, but not all, `dcecp` objects take an argument. Refer to the individual reference pages for descriptions of the arguments supported by various objects.

- **-option**: Specifies a qualifier that controls the precise behavior of a `dcecp` command. Most, but not all, `dcecp` commands take options. Specify options by preceding the option name with a dash as in `-replica`. Some options take an argument, `opt_arg`, that can be a name or a value. The following command shows a `-clearinghouse` option and its argument, which is the name of a CDS clearinghouse:

  ```
directory create ././admin -clearinghouse ././boston_ch
  ```

The DCE control program supports the following `dcecp` administration objects. For complete descriptions of the administration objects, refer to the individual object reference pages.

- **account**: Manages an account in the DCE Security Service registry.

- **acl**: Manages DCE ACLs.

- **attrlist**: Manipulates attribute lists in scripts.

- **aud**: Manages the audit daemon on any DCE host.

- **audevents**: Displays the audit event classes on any DCE host.

- **audfilter**: Manages audit event filters on any DCE host.

- **audtrail**: Displays audit trail files on the local host.

- **cds**: Manages the CDS server daemon on any DCE host.
cdsalias
  Manages cell names known to CDS.

cdscache
  Manages the CDS clerk cache on any DCE host.

cdsclock
  Manages the CDS client daemon on any DCE host.

ccell
  Performs cellwide tasks.

cellalias
  Performs cell aliasing and connection tasks.

cclearinghouse
  Manages CDS clearinghouses on the local host.

clock
  Manages the clock on any DCE host.

directory
  Manages directory entries in the CDS namespace.

dts
  Manages DTS on any host.

ems
  Manages the Event Management Services (EMS) daemon on a DCE host.

emsconsumer
  Manages EMS consumers and their event filter groups.

emsevent
  Manages EMS event types and event type schemas.

emsfilter
  Manages EMS event filters on a DCE host.

emslog
  Manages the EMS log file on the current host.

epoint
  Displays remote endpoints, manages local endpoints.

group
  Manages DCE groups in the security service.

host
  Performs tasks involving a host in a DCE cell.

hostdata
  Manages host-specific information on any DCE host.

hostvar
  Manages host-specific variables on the local DCE host.

keytab
  Manages server key tables on any DCE host.

link
  Manages softlinks in CDS.

log
  Manages routing for DCE serviceability messages.

name
  Manages CDS name translation.

object
  Manages object entries in CDS.

organization
  Manages DCE organizations in the Security Service.

principal
  Manages DCE principals in the Security Service.
**registry**
Manages DCE security replicas and registry-wide information.

**rpcentry**
Manages a server entry in CDS.

**rpcgroup**
Manages a group entry in CDS.

**rpcprofile**
Manages a profile entry in CDS.

**secval**
Manages the security validation service on any DCE host.

**server**
Manages DCE servers on any DCE host.

**user**
Performs tasks involving individual user information.

**utc**
Manipulates Universal Time Coordinated (UTC) timestamps.

**uuid**
Manipulates (generates or compares) Universal Unique Identifiers (UUIDs).

**xattrschema**
Manages schemas for extended registry attributes (ERAs).

**Common Operations**

This section describes operations common to more than one object. Some operations presented here are implemented in all objects, some in only a few, and some only for specific types of objects such as containers (for instance, CDS directories).

**add**
Adds an object to a container. It is implemented for all objects that represent containers. The argument is a list of names of containers. The required `-member` option is used to specify the name of the member to be added to the containers. Its value is a list of members to be added. If lists are specified for both the `-member` option and as the argument, then each member name is added to each container. For example, it is used to add a member to a remote procedure call (RPC) group and is used to add an element to an RPC profile. This operation returns an empty string on success.

**catalog**
Returns the names of all instances of an object. It usually takes no argument. In some cases, though, an argument specifying a scope, such as a cell name, is optional. For example, the `principal catalog` command returns a list of all principals in the registry. By default, full names are returned. Some objects support a `-simplename` option, which returns names in a shorter form (either relative or not fully qualified). The order of the returned list depends on the object.

**create**
Creates a new instance of an object. It takes one argument, a list of names of instances to be created. This operation returns an empty string on success. Returns an error if the object already exists. For some objects this command takes a `-attribute` option or a set of attribute options to specify attributes on the new object.

**delete**
Destroys an instance of the object. It takes one argument, a list of names of instances to be deleted. This operation returns an empty string on success. If the object does not exist, an error is returned.

**help**
Returns help information on the object as described in the Help section. It
takes an argument, which can be an operation supported by the object or the -verbose option to return more information.

list

Returns a list of the names of all the members of a container. This operation returns names only and not any other information about the members. It is implemented on all objects that represent containers. The argument is a list of names of containers for which to return members. The order of the returned list depends on the object. If more than one container name is given, all member names are returned in one list.

modify

This operation is used to modify attributes, policies, counters, or any other information in an object. Therefore, all attributes, policies, counters, and so forth must have unique names. This operation is not available to all objects. The argument is a list of names of objects to modify.

The specific modification to be made to an object is described by one or more of the -add, -remove, or -change options. If more than one is used, the entire modify operation is treated atomically in that either it all will work or none of it will. The order of the options does not matter. Each option can be used only once per command invocation. This operation returns an empty string on success.

-add

Used to add an attribute to an object or merely to add values to an existing attribute. The value of this option is an attribute list.

-remove

Used to remove an entire attribute or merely some values from an attribute. The value of this option is an attribute list.

-change

Used to change one attribute value to another. The value of this option is an attribute list.

operations

Returns a list of the operations supported by the object. It takes no arguments, and always returns a Tcl list suitable for use in a foreach statement. The operations in the list are in alphabetical order with the exception of help and operations, which are listed last. To return the elements fully sorted, use the following command:

lsort [object operations]

remove

Removes an object from a container. It is implemented for all objects that represent containers. The argument is a list of names of containers. The remove operation requires one option, -member, which is used to specify the name of the member to be removed from the container. The value is a list of names of members of the containers. If the value of this option and the argument to the command are both lists, then each listed member is removed from each specified container. If the members do not exist an error is returned. This operation returns an empty string on success.

rename

This operation changes the name of a specified object. The argument is a single name of an object to be renamed, that is, it cannot be a list. Takes a required -to option with a value of the new name. The value can not be a list. This operation returns an empty string on success.

show

Returns information about an object instance. Objects can have various types of information such as attributes, counters, policies, and so on. The
show operation is used to return any of this information. Options are passed to the command to specify what information is to be returned. Most of the options used for this purpose are in the plural form such as -all, -attributes, -counters, and -members.

Unlike the list operation, which returns information about the members of a container, the show operation looks only at the named object instance. If the object is a container, the show operation does not return information about the members, only the container itself.

This operation takes one argument which is a list of names of instances to be shown.

synchronize
Tells the instance to synchronize with any replicas of itself. In CDS terminology, this operations performs a skulk on a directory; in DTS, it causes a server to synchronize. This operation is implemented for all objects that support replication. The argument is a list of instance names to synchronize. If more than one instance name is given, each instance synchronizes with all of its replicas. Pairwise synchronization is not supported. This operation returns an empty string on success.

Miscellaneous Commands

The DCE control program includes a set of commands for miscellaneous operations.

dcecp_initInterp
Initializes a base Tcl interpreter with all the dcecp commands.

echo
Displays the supplied string as output.

erertext
Takes a DCE status code as an argument and returns the text of the associated message as found in the message catalogs. The argument can be in decimal, octal (leading 0), or hexadecimal (leading 0x) notation.

login
Creates a new login context, which persists until the end of the dcecp session or until destroyed by logout. The login command also sets the _c convenience variable to the name of the cell logged in to and the _u convenience variable to the name of the principal that issued the login command. Convenience variables are discussed in a separate section of this reference page. Login contexts are stacked. Takes an account name as an argument. The password is prompted for and not echoed to the screen. Also takes the -password option to enter a password.

logout
Logs you out of the current login context as established with a previous login command. You can only log out of contexts that were created with the dcecp login. Trying to log out of an inherited context results in an error. Leaving dcecp logs out all contexts created in the session.

quit
Exits from dcecp. A synonym of the Tcl built-in command exit.

resolve
Takes a partial string binding and returns a fully bound string binding. Takes a required -interface option and an optional -object option with an interface identifier as an argument to provide enough information for the mapping to occur.

shell
Spawns a command shell for the user. The value of the SHELL environment variable is used to obtain the name of the shell to spawn.
When the command shell terminates, control is returned to **dcecp**. If the shell is called with arguments, they are passed to the shell and executed. Control is returned upon completion. Always returns an empty string, though an error exception is generated if the shell exits abnormally.

## Command Processing

The DCE control program supports the Tcl built-in commands as well as its own commands. If a command name is unknown to **dcecp**, it is passed to the `unknown` procedure and **dcecp** evaluates it using the following algorithm:

1. If the command is found in a **dcecp** script file, **dcecp** executes the command.
2. If the command exists as an executable UNIX program, **dcecp** executes the command. Therefore, you can invoke any UNIX command from the **dcecp** prompt (for example, `ls -l`). Because you do not leave **dcecp**, you do not lose any context you have established.
3. If you have invoked the command at the top level of the **dcecp** shell and the command requests C-shell-like history substitution (such as `!!`, `! number` or `old new`), **dcecp** emulates the C shell’s history substitution.
4. If you have invoked the command at the top level of the **dcecp** shell and the command is a unique abbreviation for another command, **dcecp** invokes the command.

## Abbreviations

The **dcecp** command makes use of two mechanisms to allow all object names, operation names, and options to be abbreviated to the shortest unique string in interactive commands.

The first mechanism relies on the `unknown` command whose behavior is described in the **Command Processing** section of this reference page.

The second mechanism is built in to the individual **dcecp** commands themselves. This mechanism allows the operation name to be abbreviated to the shortest unique operation string supported by the object, and the option names to be abbreviated to the shortest unique string representing an option supported by an object and operation.

For example, consider the following `directory create` command:

```
directory create /.:admin/printers/ascii -replica -clearinghouse /./SFO_CH
```

In the abbreviated form, the same command can be entered as follows:

```
dir cre /./admin/printers/ascii -r -c /./SFO_CH
```

Although abbreviating commands is a good way to save keystrokes in typing interactive commands, abbreviations are not recommended for use in scripts. New procedures in scripts can cause abbreviations to become ambiguous. Furthermore, abbreviations are not always portable. When scripts move to other machines, some definitions might be left behind so PAM scripts will not work correctly. Always spell out complete names in scripts.
Syntax

The `dcecp` commands have a default word order, which is *object operation*. This order facilitates adding new objects because new objects can simply be added along with their operations.

You can configure `dcecp` to accept commands ordered as *operation object* by loading a script called `verb-object.dcecp`. Users who have access to the *operation order* continue to have access to the *object operation* order. You can load the script for all users on a host by including the following line in the system’s `init.dcecp` file:

```
source verb-object.dcecp
```

You can configure *operation object* for individual users by including the line in that user’s `.dcecpprc` file.

Attribute Lists

Many commands need to specify attributes to operate upon. For example, the `modify` operation allows attributes to be changed and the `create` operation often allows attributes to be created along with the object. In all cases, you can use an attribute list to specify the attributes and their values. Doing so makes passing information from one command to another very easy. For example, an ACL copy operation could be written as follows:

```bash
# copy acl name1 to acl name2
# no error checking
proc acl_copy {name1 name2} {
    acl replace $name2 -acl [acl show $name1]
}
```

Attribute Options

While attribute lists are useful for writing scripts, they are often not user friendly. For those objects that have a fixed list of attributes (for instance, *principal* and *dts*, but not *object*), wherever an attribute list is allowed, options for each attribute that have the same name as the attribute are allowed followed by their values. For example, the following are equivalent:

```
principal create smith -attribute {{quota 5} {uid 123}}

principal create melman -quota 5 -uid 123
```

Lists of Lists

The DCE control program interpreter relies on list structures to parse command input and return command output. For instance, the following sample command removes the *user* ACL entry for the principal `melman` from an object called `/.:/foo`.

```
acl modify /.:/foo -remove {user melman}
```

Because the `-remove` option uses a list structure to group attributes and values in the option argument, it can take a list of ACL entries as in the following example, which removes the *user* ACL entry for the principals `melman` and `salamone`:

```
ac1 modify /.:/foo -remove {{user melman} {user salamone}}
```
Lists of one value that do not contain spaces do not require braces. The string syntax of an ACL entry allows the type and key to be separated by a : (colon), so the following are valid:

```
ACL modify /.:foo -remove user:melman
ACL modify /.:foo -remove {user:melman user:salamone}
```

If only one ACL entry given, that is, the -remove option's value has only one element (and that element does not contain spaces), then braces are not needed to delimit the list. The following are all valid, but all are examples with unnecessary braces:

```
ACL modify /.:foo -remove {{user melman}}
ACL modify /.:foo -remove {{{user melman}}}
ACL modify /.:foo -remove {user:melman}
ACL modify /.:foo -remove {{user:melman} {user:salamone}}
```

### Convenience Variables

All `dcecp` commands set several variables on execution. The variables contain the name of the object operated on, the return value of the last command, the cell name of the last object operated on, and so on. To avoid unnecessary typing, you can substitute the value of these variables into the next command.

Convenience variables behave just like other variables in `dcecp`. Thus, you can trigger variable substitution by prepending a `$` (dollar sign) before the name of the variable. Alternatively, you can trigger substitution by using `set`. The convenience variables can be set only by using the DCE control program.

The following variables are defined by `dcecp`:

- **_b**: Holds the name of the server bound to by the last command. This variable is actually a Tcl array where the indexes are used to identify the service. Currently there is only one index is defined: `sec`. Refer to the variable as `_b(sec)`. The value specifies the name of a server in whatever manner the service finds useful. This value could be the name of an RPC server entry in the namespace, a string binding, or the name of a cell. This variable cannot be set by the user.

- **_c**: Holds the cell name of the current principal. The `login` command sets the cell name (_c) and principal name (_u) convenience variables at login (see the `login` command). This variable cannot be set by the user.

- **_conf**: This variable alters the behavior of most commands that operate on a CDS object. It indicates the confidence you have in the local CDS daemon to fulfill requests. The legal values are `low`, `medium`, and `high`.

- **_e**: Holds the last DCE error code encountered. This variable has meaning only if `dcecp` is able to determine what the error code is. The value `-1` (negative one) is used when an actual error code is unavailable. This variable cannot be set by the user.

- **_h**: Holds the hostname the current user is operating on. This variable cannot be set by the user.
dcecp(8dce)

_local  Holds a flag that indicates the mode in which the dcecp session is operating. This variable is set to true if the dcecp session was started with the -local option.

_n  Holds a list of the names entered in the last command. These names are the names that the command operated on, typically entered as the third argument.

For example, the following command lists the simplenames of the directories in the .:/ directory:

dcecp> dir list ./ -simplename
hosts subsys absolut_ch cell-profile fs lan-profile
sec sec-v1

dcecp>

The _n variable then contains the following name:

dcecp> echo $_n
./:

dcecp>

The following command creates the .:/x and .:/y directories:

dcecp> dir create {./:./x ./:./y}

dcecp>

The _n variable then contains the following names:

dcecp> echo $_n
./:./x ./:./y

dcecp>

_o  Holds the object used in the last operation. For example, if the last command was dir show ./:, then _o is directory. This variable cannot be set by the user.

_p  Holds the parent of the object named in the _n variable. If the _n variable is a list, the *.p variable is a list of the same length, where each element is the parent of the corresponding element in _n. If an object in _n has no parent, the value of _p is the empty string. This variable cannot be set by the user.

The following example creates the directories named ./:/gumby and ./:/pokey. When the command completes the _n variable contains the names gumby and pokey.

dcecp> dir create {./:/gumby ./:/pokey}

dcecp>

The _p variable contains the names of the parents of the gumby and pokey directories.

dcecp> echo $_p
./: /:

dcecp>

_r  Holds the return value of the last executed command. This variable cannot be set by the user.

_s  Holds the name of the server bound to by the last command. This variable
dcecp(8dce)

is actually a Tcl array where the indexes are used to identify the service. The currently defined indexes are sec, cds, dts, and aud.

The value specifies the name of a server in whatever manner the service finds useful. This value could be the name of an RPC server entry in the namespace, a string binding, or the name of a cell. Users can set this variable by issuing the set command to select the server to use.

Each service treats the values of this variable (array) differently. For example, the Security Service uses this variable to select the registry to bind to for the next command, and as a default for the next registry operation. If bound to a read-only replica and an update is requested, dcecp tries to bind to the master registry to perform the change. CDS attempts to communicate only with the CDS server named by the variable. If the named CDS server cannot satisfy a request for any reason, the request fails. The auditing service and DTS uses its variable in a manner similar to the CDS server. To contact an audit daemon or DTS server on another host, set this variable to identify that server.

For information about an object’s use of this variable, see the object’s reference page or use the object’s help -verbose operation.

_u

Holds the current principal name. The login command sets the cell name (_c) and principal name (_u) convenience variables at login (see the login command). This variable cannot be set by the user.

Error Handling

All dcecp operations return either a list of some information or an empty string on success. If an error occurs, dcecp returns an error message. The DCE control program also provides a catch command to help scripts catch errors and invoke error handlers.

The DCE control program provides two global variables that store error information returned from commands. The errorInfo variable contains the stack-trace of the error messages. When errors occur, dcecp commands return one line error messages by default. If the variable dcecp_verbose_errors is set to 1, then a stack trace as it would appear in errorInfo is output as well.

When a dcecp command argument is a list of objects, the command operates on multiple objects. These operations are usually performed iteratively. If an error occurs, the command aborts at the time of error, producing an exception. Some operations will have finished and others will not have. These operations are always performed in the order listed, and the error message should make it clear on which object the command failed.

Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Help

The DCE control program provides several kinds of help. All returned help strings are obtained from appropriate message catalogs.

To see which operations an object supports, use the operations command. An example follows:
This command provides simple help similar to usage messages found on many systems. Users unsure of an operation name or of whether an operation is supported by an object can use this command to find the answer. The output is a \texttt{dcecp} list that can be used by other \texttt{dcecp} commands.

To see other information about an object, use an object’s \texttt{help} operation. All \texttt{dcecp} objects have a \texttt{help} operation that offers three kinds of information.

1. View brief information about an object’s operations by using \texttt{help} without arguments or options. Operations are listed in alphabetical order with the \texttt{operations} and \texttt{help} operations listed last because all objects support these operations. An example is as follows:

\begin{verbatim}
dcecp> principal help
Catalog create delete modify rename show help operations
\end{verbatim}

\begin{itemize}
\item \texttt{catalog} Returns all the names of principals in the registry.
\item \texttt{create} Creates a DCE principal.
\item \texttt{delete} Deletes a principal from the registry.
\item \texttt{modify} Changes the information about a principal.
\item \texttt{rename} Renames the specified principal.
\item \texttt{show} Returns the attributes of a principal.
\item \texttt{help} Prints a summary of command-line options.
\item \texttt{operations} Returns a list of the valid operations for this command.
\end{itemize}

2. View brief information about the options an operation supports by using \texttt{help} with one argument—the name of the operation. This operation returns attribute options in alphabetical order. If no options are supported, an empty string is returned. An example follows:

\begin{verbatim}
dcecp> principal help create
-alias Add principal named as an alias of specified uid.
-attribute Attribute list to be assigned to the new principal.
-fullname Fullname of the new principal.
-quota Quota of the new principal.
-uid User Identifier of the new principal.
-uuid Orphaned UUID to be adopted by the specified principal.
\end{verbatim}

3. View a short description of a \texttt{dcecp} object by using the \texttt{help} operation with the \texttt{-verbose} option. This operation returns text explaining what the object represents and how to use it. An example follows:

\begin{verbatim}
dcecp> principal help -verbose
This object allows manipulation of principal information stored in the DCE registry. The argument is a list of either relative or fully-qualified principal names. Specify fixed attributes using attribute options or an attribute list. Specify any extended attributes using an attribute list. Principal operations connect to a registry that can service the request. Specify a particular registry by setting the _s(sec) convenience variable to be a cell-relative or global replica name, or the binding of the host where the replica exists. The completed operation sets the _b(sec) convenience variable to the name of the registry contacted.
\end{verbatim}
Utility Library

The file opt/dcelocal/dcecp/utility.dcp contains Tcl functions useful for DCE administration. The functions, which can vary from release to release, are fully commented to document their use.

Reference Pages

Users can use the dceman command on [POSIX.2] systems to view the reference page for any dcecp object without exiting dcecp. This capability helps users avoid losing any context that has been established in the current dcecp session. For example, the user can get detailed help on the principal command by entering the following:

dcecp>

Command-Line Editing

You can edit a line before it is sent to dcecp by typing certain control characters and escape sequences. To enter a control character, hold down the <Control> key and press the appropriate character key. (Control characters are indicated in DCE documentation by the notation <Ctrl- x>, where x is the second key.) To enter an escape sequence, press <Escape> then press one or more character keys. (Escape sequences are indicated in DCE documentation by the notation <ESC   x>, where x is the second key.) Escape sequences are case-sensitive; control characters are not.

You can enter an editing command anywhere on a line. In addition, you can enter <Return> anywhere on the line.

You can specify a number [n] as a repeat count. To enter a repeat count, press <Escape>, a number, and the command you want to execute.

For example, <ESC 4><Ctrl-D> deletes the next four characters on a line.

Use the following control characters and escape sequences for line editing:

<table>
<thead>
<tr>
<th>Control Sequence</th>
<th>Action Performed</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Ctrl-A&gt;</td>
<td>Move to the beginning of the line</td>
</tr>
<tr>
<td>&lt;Ctrl-B&gt;</td>
<td>Move left (backward) [n]</td>
</tr>
<tr>
<td>&lt;Ctrl-D&gt;</td>
<td>Delete the next character [n]</td>
</tr>
<tr>
<td>&lt;Ctrl-E&gt;</td>
<td>Move to the end of the line</td>
</tr>
<tr>
<td>&lt;Ctrl-F&gt;</td>
<td>Move right (forward) [n]</td>
</tr>
<tr>
<td>&lt;Ctrl-G&gt;</td>
<td>Ring the bell</td>
</tr>
<tr>
<td>&lt;Ctrl-H&gt;</td>
<td>Delete the character before the cursor [n]</td>
</tr>
</tbody>
</table>
dcecp(8dce)

<Ctrl-I>
  Complete the filename (<Tab>)

<Ctrl-J>
  Done with the line (<Return>)

<Ctrl-K>
  Kill to the end of the line (or column [n])

<Ctrl-L>
  Redisplay the line

<Ctrl-M>
  Done with the line (alternate <Return>)

<Ctrl-N>
  Get the next line from history [n]

<Ctrl-P>
  Get the previous line from history [n]

<Ctrl-R>
  Search backward (or forward if [n]) through history for the text; start the line if the text begins with an up arrow

<Ctrl-T>
  Transpose the characters

<Ctrl-V>
  Insert the next character even if it is an edit command

<Ctrl-W>
  Wipe to the mark

<Ctrl-X><Ctrl-X>
  Exchange the current location and mark

<Ctrl-X><Ctrl-X>
  Get the next line from history [n]

<Ctrl-Y>
  Yank back the last killed text

<Ctrl-[]>
  Start an escape sequence (<Escape>)

<Ctrl-[]>
  Move forward to the next character c

<Ctrl-?>
  Delete the character before the cursor [n]

**Escape Sequence**

**Action Performed**

<ESC><Ctrl-H>
  Delete the previous word (<Backspace>) [n]

<ESC><Delete>
  Delete the previous word (<Delete>) [n]

<ESC><Space>
  Set the mark (<Space>); refer to the <Ctrl-X><Ctrl-X> and <Ctrl-Y> control characters

<ESC.->
  Get the last (or [n]th) word from the previous line
The DCE control program also supports filename completion. For example, suppose the root directory has the following files in it: `vmunix`, `core`, `vmunix.old`.

If you type `rm /v` and then press `<Tab>`, `dcecp` finishes off as much of the name as possible by adding `munix`. If the name is not unique, the terminal alarm sounds. If you enter `<ESC>?>`, `dcecp` displays the two possible complete filenames: `vmunix` and `vmunix.old`. If you respond by entering a . (period) and by entering `<Tab>`, `dcecp` completes the filename for you.

## Command History and Command-Line Recall

The DCE control program includes a history facility that stores previously entered commands. View the stored commands using the `history` command.

By default, the history facility stores the 20 most recent commands, but you can use a `history keep` command to change this as follows:

```
dcecp> history keep 50
```

Each stored command is numbered so you can recall it by using a ! (exclamation point) followed by the event number, as follows:

```
dcecp> !7
[execution of event 7]
dcecp>
```

Recall a specific command using an ! (exclamation point) followed by the first unique characters of a previously entered command, as follows:
You can also recall and revise the most recent command using the `old new` syntax familiar to UNIX users, as follows:

dcecp> directory create /.:/admin/printers
[error message]
dcecp>

Invocations

The following examples show some ways to issue `dcecp` commands:

1. Invoke `dcecp` for interactive use:

   % dcecp
   dcecp>

2. Invoke `dcecp` for a single command:

   % dcecp -c clock show
   1994-04-21-19:12:42.203+00:00:00----
   %

3. Invoke `dcecp` and run a script:

   % dcecp get_users.Tcl
   %

Simple Object Commands

```
dcecp> acl show -ic /.:
{unauthenticated r--t---}
{group subsys/dce/cds-admin rwdtica}
{group subsys/dce/cds-server rwdtica}
{any_other r--t---}
dcecp>

% dcecp -c directory show /.:/subsys
{RPC_ClassVersion {01 00}}
{CDS_CTS 1995-10-11-14:06:47.884826100/08-00-09-85-b5-a6}
{CDS_UTS 1995-10-23-03:06:43.209673100/08-00-09-85-b5-a6}
{CDS_ObjectUUID 0c27c0ac-03d6-11cf-ad88-08000985b5a6}
{CDS_Repli
```
The foreach Loop

dcecp> foreach i [group list temps] {
      account modify $i temps research -expdate 6/30/95}

Related Information

Commands: cds_intro(8cds), dce_intro(8dce), dts_intro(8dts), sec_intro(8sec).
The DCE host daemon

dced

Purpose

Synopsis

dced [-h | -i | -cfr] [-w route] [-b | -p | -s] [-e | prot_seq ] [-t n]

Options

- \texttt{h} \quad \text{Prints} \ \texttt{dced} \ \text{usage and exits.}
- \texttt{i} \quad \text{Initializes} \ \texttt{dced} \ \text{databases and ACLs and exits. If the databases exist, this option displays an error. See the list of databases in the FILES section of this reference page.}
- \texttt{c} \quad \text{Starts} \ \texttt{dced} \ \text{so it does not require DCE privacy encryption for remote key table management. The default is to use DCE privacy encryption.}
- \texttt{f} \quad \text{Starts the} \ \texttt{dced} \ \text{process in the foreground. The default is for} \ \texttt{dced} \ \text{to run in the background.}
- \texttt{r} \quad \text{Starts} \ \texttt{dced} \ \text{in remote-update mode. This mode allows DCE cell administration tasks to be performed by an administrator on a remote machine.}
- \texttt{-w route} \quad \text{Establishes the serviceability routing for} \ \texttt{dced} 's \ \text{messages.}
- \texttt{-b} \quad \text{Starts} \ \texttt{dced} \ \text{in bootstrap mode with the endpoint mapper service and access control lists (ACLs). This mode means it might need to wait for other daemons such as} \ \texttt{secd} \ \text{and} \ \texttt{cdsd} \ \text{before it can perform its own initialization.}
- \texttt{-p} \quad \text{Purges the existing machine context and removes the bindings file before starting.}
- \texttt{-s} \quad \text{Starts} \ \texttt{dced} \ \text{without the security validation service.}
- \texttt{-e} \quad \text{Starts} \ \texttt{dced} \ \text{without the endpoint mapper service. No protocol sequences are valid for this option.}
- \texttt{-t n} \quad \text{Specifies the number of minutes (where} \ n \ \text{is the number of minutes) to wait between updates to the pe_site file. The allowed values are from 10 minutes to 1440 minutes. The default is 1440 minutes (one day).}

\textit{Note:} \ The value of 0 is valid for the \texttt{-t} option. The 0 value keeps the current contents of the \texttt{/etc/dce/security/pe_site} file intact. This prevents \texttt{pe_site} files that have been set up to point to a preferred replica from being updated to point to another server.

Arguments

\texttt{prot_seq} \quad \text{Starts} \ \texttt{dced} \ \text{by using the specified remote procedure call (RPC) protocol sequence string or strings. Possible values include} \ \texttt{ncadg_ip_udp} \ \text{for a}
Description

The DCE host daemon is a process that provides services for the local host, and is also the server used by remote applications to access these host services. The DCE host daemon services include the following:

**endpoint mapper**

The endpoint mapper service maintains a database called the *local endpoint map* which allows DCE clients to find servers, individual services provided by servers, and objects managed by services on the host. The endpoint mapper service maps interfaces, object Universal Unique Identifiers (UUIDs), and protocol sequence registrations to server ports (endpoints). Servers register their bindings with the local endpoint mapper, and the endpoint mapper service on each host uses the local endpoint map to locate a compatible server for clients that do not already know the endpoint of a compatible server.

**Host data management**

The host data management service maintains local files of host data that include (among others) the *host_name*, *cell_name*, *cell_aliases*, and *post_processors* files. The *post_processors* file contains program names matched with the other host data items (such as UUIDs). The *dced* process runs the program if the corresponding host data item is changed. There might also be host-specific data files.

**Server management**

The server management service maintains data that describes the startup configuration (*srvrconf*) and execution state (*srvrexec*) for each server. It also has the functionality to start or stop particular servers, and enable or disable specific services of servers.

**Security validation**

The security validation service acts as the client side of the security server by assuring applications that the DCE security daemon (*secd*) that the host is using is legitimate. In addition, this service logs into the local machine when *dced* is invoked and automatically updates the local machine principal's keys.

**Key table management**

The key table management service allows for remote maintenance of server’s key tables (*keytab* files).

The DCE host daemon must be running before any other DCE-based servers are started. Each DCE host must run only a single *dced*, and it must run with root privileges since it typically listens on privileged or reserved network ports. Typically, *dced* starts each time a host boots. (A file called */etc/rc.dce* is responsible for configuration issues such as deleting the endpoint map database and starting *dced*.)

By default, the DCE host daemon listens on one well-known port for each RPC protocol sequence (that is, each combination of an RPC protocol and a transport protocol) supported by the host on which it is running. A *prot_seq* argument lets you limit the protocol sequences on which *dced* listens.
Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Options

The `dced` databases are as follows:

<table>
<thead>
<tr>
<th>dcelocal/var/dced/Ep.db</th>
<th>dcelocal/var/dced/cell_aliases</th>
</tr>
</thead>
<tbody>
<tr>
<td>dcelocal/var/dced/Hostdata.db</td>
<td>dcelocal/var/dced/cell_name</td>
</tr>
<tr>
<td>dcelocal/var/dced/Srvcfg.db</td>
<td>dcelocal/var/dced/host_name</td>
</tr>
<tr>
<td>dcelocal/var/dced/Srvexec.db</td>
<td>dcelocal/var/dced/post_processes</td>
</tr>
<tr>
<td>dcelocal/var/dced/Keytab.db</td>
<td>dcelocal/bin/dcecf_postproc</td>
</tr>
<tr>
<td>dcelocal/var/dced/Acl.db</td>
<td>/krb5/v5srvtab</td>
</tr>
<tr>
<td>dcelocal/var/dced/Xattrschema.db</td>
<td>/etc/rc.dce</td>
</tr>
<tr>
<td>dcelocal/dcecf.db</td>
<td></td>
</tr>
</tbody>
</table>

Related Information

Commands: `attribute(8dce)`, `endpoint(8dce)`, `hostdata(8dce)`, `secval(8dce)`, `keytab(8dce)`, `server(8dce)`

Library calls: `dce_server*(3dce)`, `dced_*(3dce)`, `rpc_mgmt_ep*(3rpc)`

directory

Purpose

A dcecp object that manages a name service directory

Synopsis

directory add directory_name_list -member child_pointer_list -clearinghouse clearinghouse_name

directory create directory_name_list [-attribute attribute_list [-single] ] [[-replica] -clearinghouse clearinghouse_name ]

directory delete directory_name_list [[-tree] | [-force] | -replica | -clearinghouse clearinghouse_name ]

directory help [operation | -verbose ]

directory list directory_name_list [[-directories] [-objects] [-links] [-simplename | -fullname ]

directory merge source_directory_name -into destination_directory_name [-clearinghouse clearinghouse_name] [-tree] [-nocheck]

directory modify directory_name_list [-add attribute_list | [-single] | -remove attribute_list | [-types] | -change attribute_list | -master clearinghouse_name | [-readonly clearinghouse_name_list] | [-exclude clearinghouse_name_list] ]

directory operations

directory remove directory_name_list -member child_pointer_list

directory show directory_name_list [[-schema] [-member child_pointer_list] [-replica] | -clearinghouse clearinghouse_name ]

directory synchronize directory_name_list

Arguments

directory_name_list
A list of one or more directory names to be operated on.

operation
The name of the directory operation for which to display help information.

source_directory_name
The name of a single directory whose contents are to be copied into a destination directory using the merge operation.

Description

The directory object represents Cell Directory Service (CDS) directories. CDS directories are containers for other objects, links, and other directories (as well as
clearinghouses). Any of these items that reside in a directory are called children of that directory. Directories also contain attributes that can be viewed or modified.

This object also represents CDS replicas. Replicas are read-only copies of directories stored in other clearinghouses. Several of the supported operations take options to indicate that the command is to operate on a specific replica.

If the \_s(cds) convenience variable is set, it is treated as the name of a clearinghouse to contact for this operation. This is the only clearinghouse that will be contacted in an attempt to complete the operation. These commands do not set the value of this variable after completion. If a \-clearinghouse option is used (as described in some commands below), it overrides the value of \_s(cds), but the command does not change the setting of \_s(cds).

Attributes

The following are the CDS-defined attributes for CDS directory objects:

**CDS\_AllUpTo**
Indicates the date and time of the last successful skulk on the directory. All replicas of the directory are guaranteed to receive all updates whose timestamps are less than the value of this attribute. The value of this attribute is a read-only DTS-style timestamp that is set by the system.

**CDS\_Convergence**
Specifies the degree of consistency among replicas. This attribute's value is defined as one of the following:

- **low**
  CDS does not immediately propagate an update. The next skulk distributes all updates that occurred since the previous skulk. Skulks occur at least once every 24 hours.

- **medium**
  CDS attempts to immediately propagate an update to all replicas. If the attempt fails, the next scheduled skulk makes the replicas consistent. Skulks occur at least once every 12 hours.

- **high**
  CDS attempts to immediately propagate an update to all replicas. If the attempt fails (for example, if one of the replicas is unavailable), a skulk is scheduled for within one hour. Skulks usually occur at least once every 12 hours. Use this setting temporarily and briefly, because it uses extensive system resources.

By default, every directory inherits the convergence setting of its parent at creation time. The default setting on the root directory is **medium**.

**CDS\_CTS**
Specifies the creation timestamp (CTS) of the directory. The value of this attribute is a read-only DTS-style timestamp that is set by the system.

**CDS\_DirectoryVersion**
Specifies the current version of the directory. The version is derived from the CDS\_DirectoryVersion attribute of the clearinghouse in which the directory was created. Multiple directory versions are supported in a cell. This read-only attribute is set by the system.

**CDS\_Epoch**
A Universal Unique Identifier (UUID) that identifies a particular instance of the directory. This read-only attribute is set by the system.
CDS_GDAPointers
A set-valued attribute that is present only in the root directory of a cell. This attribute contains location information about registered Global Directory Agents (GDAs) for that cell, similar to the CDS_Replicas attribute. It is created and only used by a GDA.

CDS_InCHName
Indicates whether a directory or any of its descendants can store clearinghouse names. If this value is true, the directory can store clearinghouse names. If it is false, the directory cannot store clearinghouse names. This read-only attribute is set by the system. As of DCE Release 1.1 and later, CDS creates this attribute on the cell root directory and gives it a value of true. The attribute will not appear in any other directory.

CDS_LastSkulk
Records the timestamp of the last skulk performed on this directory. This read-only attribute is set by the system.

CDS_LastUpdate
Records the timestamp of the most recent change to any attribute of a directory replica, or any change to an entry in the replica. This read-only attribute is set by the system.

CDS_ObjectUUID
Specifies the unique identifier of the directory. This read-only attribute is set by the system when the directory is created.

CDS_ParentPointer
Contains a pointer to this directory’s parent in the namespace. This read-only attribute is set by the system.

CDS_Replicas
Specifies the address, UUID, and name of every clearinghouse in which a copy of this directory is located. This attribute also specifies whether the replica in a particular clearinghouse is a master or read-only replica. This read-only attribute is set by the system.

CDS_ReplicaState
Specifies whether a directory replica can be accessed. The state on indicates that the directory replica can be accessed. This read-only attribute is set by the system.

CDS_ReplicaType
Indicates whether a directory replica is a master or read-only replica. Possible values are Master and ReadOnly. This read-only attribute is set by the system.

CDS_ReplicaVersion
Specifies the version of a replica of the directory. The default is 3.0. This read-only attribute is set by the system.

CDS_RingPointer
Specifies the UUID of a clearinghouse containing another replica of this directory. The CDS_RingPointer attribute appears on older directories, but not on DCE Release 1.1 and later directories. This read-only attribute is set by the system.

CDS_UpgradeTo
A single-valued attribute used to control the upgrading of a directory from one version of CDS to another. By modifying this attribute, the process of upgrading a directory to a newer version of CDS can be initiated. After this
attribute is set, the background process in CDS notices it and tries to contact each replica. If CDA can contact the replica, the 
**CDS_DirectoryVersion** attribute is changed to the value of this attribute.

**CDS_UTS**
Specifies the timestamp of the most recent update to an attribute of the directory. The value of this attribute is a read-only DTS-style timestamp that is set by the system.

See the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide* for more information about directory attributes.

### Errors

A representative list of errors that might be returned is not shown here. Refer to the *IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide* for complete descriptions of all error messages.

### Operations

**directory add**

Creates a child pointer in the parent directory. The syntax is as follows:

```
directory add  directory_name_list  -member  child_pointer_list  -clearinghouse  clearinghouse_name
```

**Options**

- **-member  child_pointer_list**
  This required option names the child pointers to be added to parent directories in the clearinghouse named by the required **-clearinghouse** option.

- **-clearinghouse  clearinghouse_name**
  This required option names the clearinghouse to which the child pointers are to be added.

The **add** operation creates a child pointer in the parent directory. The **directory_name_list** argument is a list of one or more names of parent directories to have child pointers added to them. The value of the required **-member** option is a list of names of child pointers to be added to each directory listed in the argument. Each child pointer name entered should contain only the last relative distinguished name (RDN) of the name. The child object must exist or the command returns an error. The full name of a clearinghouse that holds a replica of the child directory is given as the value to the required **-clearinghouse** option. This option can only have one value and is used for each value of the **-member** option. This operation returns an empty string on success. If a child pointer of the same name already exists, an error is returned.

This command is needed only to recreate a child pointer that was accidentally deleted, such as in a troubleshooting situation. Normally child pointers are created internally by CDS when creating directories with the **directory create** command.

**Privileges Required**

You must have **i (insert)** permission to the parent directory.
directory(8dce)

Examples

dcecp> directory add ./ -member foo -clearinghouse ./:oddball_ch
dcecp>

directory create

Creates a new directory of the specified name. The syntax is as follows:

directory create directory_name_list
[-attribute attribute_list [-single]]
[[[-replica] -clearinghouse clearinghouse_name]]

Options

-attribute attribute_list
  Allows you to specify the CDS_Convergence attribute or the CDS_UpgradeTo attribute in an attribute list. The format is as follows:

  {{attribute value}... {attribute value}}

  See Attributes for descriptions of CDS_Convergence and CDS_UpgradeTo.

-singel
  Valid only with the -attribute option, this option specifies that attribute values are single-valued. Otherwise, attributes are multivalued.

-replica
  This option specifies that the directory created is a replica of an existing directory. If you use the -replica option, you must specify a clearinghouse by using the -clearinghouse option.

-clearinghouse clearinghouse_name
  Required with the -replica option; optional when the -replica option is not present. The -clearinghouse option names the clearinghouse to which the child pointers are to be added.

The create operation creates a new directory of the specified name. The directory_name_list argument is a list of names of directories to be created.

An optional -attribute option specifies a list of attributes to be included in each created directory. The attribute values are multivalued unless the -single option is specified, in which case all attributes are single-valued. The -single option is valid only if the -attribute option is specified.

The -clearinghouse option specifies one clearinghouse to create all the directories in. If this option is not specified, the new directories are created in the master clearinghouse as the parent directory. The directory create command also takes a -replica option, which indicates that a directory replica is created; when this option is used, the -clearinghouse option is required. This operation returns an empty string on success.

Privileges Required

You must have the following permissions to create a directory: r (read) and i (insert) permission to the parent directory, and w (write) permission to the clearinghouse in which the master replica of the new directory is to be stored.
In addition, the server principal (hosts/hostname/cds-server) must have r (read) and i (insert) permission to the parent directory.

Examples

dcecp> directory create ./:/sales
dcecp>

directory delete

Deletes a directory. The syntax is as follows:

directory delete directory_name_list [[-tree] [-force] | -replica -clearinghouse clearinghouse_name]

Options
- -tree Removes the directory and everything (all directories, objects, links, and clearinghouses) beneath it.
- -replica Specifies that the directory to delete is a replica of an existing directory. The -clearinghouse option is required if you use this option.
- -force Allows the delete operation to proceed by deleting existing replicas.
- -clearinghouse clearinghouse_name
  Required with the -replica option, the -clearinghouse option names the single clearinghouse from which the replica is to be deleted.

The delete operation deletes a directory from the CDS name service. The directory_name_list argument is a list of names of directories to be deleted. If the directory is not empty, the command returns an error unless the -tree option is used. The -tree option, which takes no value, removes the directory and everything (all directories, objects, links, and clearinghouses) beneath it. The -force option also deletes replicas.

Used together, the -replica and -clearinghouse options let you delete a replica instead of a directory. The -clearinghouse option specifies the clearinghouse that contains the replica; only one value can be specified, not a list. This operation returns an empty string on success. If a specified directory does not exist, an error is generated.

The -replica and -clearinghouse options cannot be used with the -tree option.

Privileges Required

You must have d (delete) permission to the directory and w (write) permission to the clearinghouse that stores the master replica of the directory. The server principal (hosts/hostname/cds-server) needs A (Admin) permission to the parent directory or d (delete) permission to the child pointer that points to the directory you intend to delete.

Examples

dcecp> directory delete ./:/eng
dcecp>
The following command tries to delete a nonempty directory `/.:/depts/phrenology` and gets an error. The second attempt uses the `-tree` option to delete the directory and all the directories and objects beneath it.

```
dcecp> dir delete `.:/depts/phrenology
Error: Directory must be empty to be deleted
dcecp>
```

```
dcecp> dir delete `.:/depts/phrenology -tree
dcecp>
```

directory help

Returns help information about the directory object and its operations. The syntax is as follows:

```
directory help  [operation | -verbose]
```

Options

- `-verbose`
  
  Displays information about the directory object.
  
  Used without an argument or option, the directory help command returns brief information about each directory operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the `-verbose` option to display detailed information about the directory object itself.

Privileges Required

No special privileges are needed to use the directory help command.

Examples

```
dcecp> directory help
add       Creates a child pointer in the specified directory.
create    Creates the named directory.
delete     Deletes the named directory.
list       Lists the descendants of a directory.
merge      Merges the contents of one directory into another.
modify     Adds, removes or changes attributes in the named directory.
remove     Removes a child pointer in the specified directory.
show       Returns the attributes of a directory.
synchronize Skulks the named directory.
help       Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp>
```

directory list

Returns a list of the names of all the descendants of a directory. The syntax is as follows:

```
directory list directory_name_list [-directories] [-objects] [-links]
[-simplename  [-fullname]
```

Options
directory(8dce)

-directories
   Lists the names of all descendent directories.

-objects
   Lists the names of all descendent objects.

-links
   Lists the names of all descendent softlinks.

-simplename
   Returns just the RDN of the name.

-fullname
   Returns the entire name.

The list operation returns a list of the names of all the descendents of a directory. Descendants can include all directories, objects, links, and clearinghouses. The directory_name_list argument is a list of names of directories to be operated on. This command returns only the names of descendents, so there is no way to tell the class of each name unless by convention (for instance, most clearinghouses end with _ch). Use the following options to specify the types of descendents to return: directories, objects, links. The options take no values and can be used in combination. By default or if the -fullname option is specified, fullnames are returned. Use the -simplename option to return merely the last RDN of the name.

Privileges Required

You must have r (read) permission to the directory named in the argument.

Examples

```
dcecp> dir list /.:depts/administration -links
/.../ward_cell.osf.org/depts/administration/bump_server1
dcecp>
```

directory merge

Copies the contents of one directory into another directory. The syntax is as follows:

```
directory merge source_directory_name -into destination_directory_name
   [-clearinghouse clearinghouse_name] [-tree] [-nocheck]
```

Options

- `tree` Copies the contents of child directories (as well as the child directories themselves) into the destination directory.

- `into destination_directory_name`
   The argument to this required option specifies the name of the destination directory. The destination directory must exist.

- `clearinghouse clearinghouse_name`
   Places the new objects (the resulting merged directory) in a clearinghouse other than that of the newly created destination directory.

- `nocheck`
   Lets the merge operation proceed without first checking for object name collisions or access control list (ACL) problems. Use this option to save time when you are sure problems do not exist.

The merge operation copies the contents of one directory into another. The argument is the name of the source directory. This command takes a required -into
option to specify the destination directory, which must exist. For example, if ./a has two child objects ./a/b and ./a/c, then directory merge ./a -into ./x would result (assuming no errors) in the creation of the following objects: ./x/b and ./x/c.

Normally only the immediate contents of the directory are merged. These contents include all objects, links, and directories, but not the contents of child directories. To merge these as well, use the -tree option.

By default, the new objects are placed in the destination directory’s master clearinghouse, and all children (no matter how many levels down) are placed in the same clearinghouse. To place any newly created descendent directories in another clearinghouse, use the -clearinghouse option with a value. Only one clearinghouse can be specified for all directories involved in the merge operation. To specify more than one, use the -clearinghouse option after the merge has happened, or use separate commands.

This command first checks for any collisions or ACL problems before beginning to merge any objects. If problems are encountered, an error is generated after all objects are checked, and the names of all problem objects, links, or directories are returned in a list. The administrator should then address these problems and rerun the merge command. If the -nocheck option is specified, the check is not performed. This way time can be saved when trying a known nonproblematic merge. This is not an atomic operation and other changes to the involved objects can cause problems. This command should be issued when others are not modifying the involved directories. ACLs can be changed to ensure that no other principal has the modify permissions to the directories. If an error occurs during the actual merging process, it is generated and the operation aborts immediately.

The merge command actually re-creates the objects with the same writable attributes of the source objects. As a result, some read-only attributes will change between the source and destination. For example, the creation timestamp attribute (CDS_CTS) changes.

The resulting merged directory inherits its ACLs from the destination directory’s Initial Container or Initial Object ACLs. Consequently, the ACLs of the destination objects are likely to differ from the ACLs of the source objects. This operation returns an empty string on success.

Privileges Required

You must have r (read) to the source and destination directories and i (insert) permission to the destination directory.

Examples

The following command merges the directories but not the contents of the ./depts/phrenology directory into the ./depts/radiology directory:

dcecp> dir list ./depts/phrenology -simple
applications services staff users
dcecp>

dcecp> directory merge ./depts/phrenology -into ./depts/radiology
dcecp>
directory modify

Adds, removes, or changes a directory’s attributes and their values. The syntax is as follows:

directory modify directory_name_list
{-add attribute_list [-single] | -remove attribute_list [-types] |
-change attribute_list | -master clearinghouse_name
[-readonly clearinghouse_name_list] [-exclude clearinghouse_name_list]}

Options

-add attribute_list
This option adds a value to a modifiable, set-valued attribute (including application-defined attributes) of a directory. If you enter a byte data type, you must enter an even number of digits. You can only enter pairs of hexadecimal values for user-defined attributes.

-single
Used with the -add option, this option specifies that the attributes to be added are to be single-valued. Normally, all user defined attributes are defined to be multivalued, even if only one value is specified. This option is not legal without the -add option.

-remove attribute_list
This option removes a value from a multivalued or single-valued attribute (including application-defined attributes) of a directory. If you do not specify a value, the command removes the entire attribute. This command can delete attributes created with the -add and -change options.

-types
Used with the -remove option, this option specifies that the value of the -remove option is a list of attribute types. Use this option to remove the entire attribute, not just a value. This option is not legal without the -remove option.

-change attribute_list
This option changes the value of a modifiable, single-valued attribute of a directory. You can specify an application-defined attribute or the following attribute, which specifies the degree of consistency among replicas:

{CDS_Convergence value}

See Attributes for the format of CDS_Convergence.

-master clearinghouse_name
When changing the epoch of a directory, use the -master option to specify a new master clearinghouse for the directory.

-readonly clearinghouse_name_list
When changing the epoch of a directory, this option specifies which clearinghouses will hold a replica of the directory.

-exclude clearinghouse_name_list
When changing the epoch of a directory, the option specifies which clearinghouses will no longer be used as replicas for the directory.
The **modify** operation adds, removes, or changes a directory’s attributes and their values. The argument is a list of one or more names of directories to be operated on. Attribute options are not supported; use one or more of the **-add**, **-remove**, or **-change** options, each of which takes an attribute list as an argument.

Use the **-remove** option to remove a value from an attribute. You can use the **-types** option along with the **-remove** option to remove an entire attribute or list of attributes.

Some attributes in CDS are multivalued. For instance, the **CDS Replicas** attribute can specify the locations and names of several clearinghouses that maintain copies of a directory. The **-add** operation requires and indication of whether it will operate on single-valued or multivalued attributes. Multivalued attributes are the default case and are indicated by using no qualifying options. However, you can indicate the use of single-valued attributes by using the **-single** option.

To change the epoch of a directory, you must specify each clearinghouse that has a master or replica copy of the directory as either the new master (with the **-master** option), a readonly copy (with the **-readonly** option), or an excluded copy (with the **-exclude** option). Additional extra clearinghouses can also be specified.

Most attributes are usually managed by the client application. See the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide* for more information about attributes. All modifications are made to each directory listed in the argument. An error in any one causes the command to abort immediately and generate an error. This operation returns an empty string on success.

**Privileges Required**

You must have **w** (write) permission to the directory to add, remove, or change attributes.

**Examples**

The following command sets the **CDS Convergence** attribute on the **/.:/depts/radiology** directory to a value of low:

```
dcecp> directory modify /.:/depts/radiology -change {CDS_Convergence low}
dcecp>
```

To add the value **ontario** to the attribute **myname** of a directory named **/.:/sales**, read the **cds_attributes** file to verify that the attribute shown in the following display exists:

```
<table>
<thead>
<tr>
<th>OID</th>
<th>LABEL</th>
<th>SYNTAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.22.1.3.91</td>
<td>myname</td>
<td>char</td>
</tr>
</tbody>
</table>
```

Enter the following command to assign the value **ontario** to the attribute **myname**:

```
dcecp> directory modify /.:/sales -add {myname ontario}
dcecp>
```

To remove the value **1** from the user-defined, set-valued attribute **dirregion** of a directory named **/.:/sales**, follow these steps:

1. Read the **cds_attributes** file to verify that the attribute **dirregion** is listed, as shown in the following display:
2. Enter the following command to remove the value 1 from the attribute dirregion:

```bash
dcecp> directory modify /./sales -remove {dirregion 1}
dcecp>
```

3. To change the epoch of a directory with one master and two replicas, enter the following command:

```bash
dcecp> directory modify /./oddball -master /./gumby_ch \  
> -readonly /./pokey_ch -exclude /./goober_ch
dcecp>
```

directory operations

Returns a list of the operations supported by the directory object. The syntax is as follows:

```
directory operations
```

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the directory operations command.

Examples

```bash
dcecp> directory operations
add create delete list merge modify remove show synchronize help operations
dcecp>
```

directory remove

Deletes a child pointer from the directories specified. The syntax is as follows:

```
directory remove  directory_name_list -member child_pointer_list
```

Options

- `member child_pointer_list`
  This required option names the child pointers to be removed from each directory in the operation argument.

The remove operation deletes a child pointer from the directories specified. The directory_name_list argument is a list of names of one or more directories to be operated on. The required -member option allows you to list the child pointers to be removed from each specified directory.

The child_pointer_list argument value of the required -member option is a list of one or more child pointers (specified as only one RDN each) to be removed from each directory in the argument.
This command is needed only to delete a child pointer that remains after the child directory is deleted. Normally child pointers are removed internally by CDS when deleting directories with the directory delete command. This operation returns an empty string on success.

Privileges Required

You must have d (delete) permission to the child pointer or A (Admin) permission to the parent directory.

Examples

The following command deletes the child pointer that accidentally remains after the /./sales/east directory is deleted:

dcecp> directory remove /./sales -member east
dcecp>

directory show

Returns a list of attributes for the specified directories and, optionally, their specified contents. The syntax is as follows:

directory show directory_name_list [ -schema]
[ -member child_pointer_list | [ -replica] -clearinghouse clearinghouse_name]

Options

- member child_pointer_list
  The optional -member option takes one required value which is the last RDN of the child pointer in the directory specified by the optional argument. The returned list describes the child pointer information for the specified member stored in the specified directories. This option cannot be combined with the -replica or -clearinghouse option.

- replica clearinghouse_name
  Specifies that the directory shown is a replica of an existing directory. If you use the -replica option, you must specify a clearinghouse with the -clearinghouse option.

- clearinghouse clearinghouse_name
  Required with the -replica option, the -clearinghouse option names the clearinghouse in which the named replica exists.

- schema
  This option returns whether an attribute is single or multivalued. This attribute is specific to a directory, meaning that the same attribute can be single-valued on one directory and multivalued on another. This option can not be used with other options.

The show operation returns a list of attributes for the specified directories and, optionally, their specified contents. The directory_name_list argument is a list of names of directories to be operated on. When used without any options, this command returns the attributes associated with the named directories. If more than one directory is specified, then all the arguments are grouped together in one list. The order of the returned arguments is the lexical order of the object identifiers (OIDs) of each attribute for each directory.
You can request attributes of specific replicas in specific clearinghouses by using the `-replica` and `-clearinghouse` options. Alternatively, you can request attributes of child pointers by using the `-member` option.

**Privileges Required**

You must have `r` (*read*) permission to the directories named in the argument list.

**Examples**

dcecp> directory show /./depts/radiology
{RPC_ClassVersion
 {01 00}}
{CDS_CTS 1994-07-08-17:01:03.115+00:00:00010.000/00-00-c0-8a-df-56}
{CDS_UTS 1994-07-08-19:36:31.719+00:00:00010.000/00-00-c0-8a-df-56}
{CDS_ObjectUUID 2df03af4-9a76-11cd-8f2b-0000c08af56}
{CDS_RepliCcas
 {CH_UUID b32648c6-928d-11cd-b4b5-0000c08af56}
 {CH_Name /.../ward_cell.osf.org/pmin17_ch}
 {Replica_Type Master}
 {Tower ncacn_ip_tcp:130.105.1.227[]}
 {Tower ncadg_ip_udp:130.105.1.227[]}}}
{CDS_AllUpTo 1994-07-08-17:01:05.945+00:00:00010.000/00-00-c0-8a-df-56}
{CDS_Convergence medium}
{CDS_ParentPointer
 {{Parent_UUID 8eeb369a-9a4b-11cd-8f2b-0000c08af56}
 {TimeOut
 {expiration 1994-07-09-17:13:31.959}
 {extension +1-00:00:00:00010.000}}}
 {myname /.../ward_cell.osf.org/depts/radiology}}}
{CDS_DirectoryVersion 3.0}
{CDS_ReplicaState on}
{CDS_ReplicaType Master}
{CDS_LastSkulk 1994-07-08-17:01:05.945+00:00:00010.000/00-00-c0-8a-df-56}
{CDS_LastUpdate 1994-07-08-19:36:31.719+00:00:00010.000/00-00-c0-8a-df-56}
{CDS_RingPointer b32648c6-928d-11cd-b4b5-0000c08af56}
{CDS_Epoch 2f617aa6-9a76-11cd-8f2b-0000c08af56}
{CDS_ReplicaVersion 3.0}
{CDS_ReplicaVersion 3.0}
dcecp>

dcecp> directory show /./depts/radiology -schema
{RPC_ClassVersion multi}
{CDS_CTS single}
{CDS_UTS single}
{CDS_ObjectUUID single}
{CDS_RepliCcas multi}
{CDS_AllUpTo single}
{CDS_Convergence single}
{CDS_ParentPointer multi}
{CDS_DirectoryVersion single}
{CDS_ReplicaState single}
{CDS_ReplicaType single}
{CDS_LastSkulk single}
{CDS_LastUpdate single}
{CDS_RingPointer single}
{CDS_Epoch single}
{CDS_ReplicaVersion single}
dcecp>

directory synchronize

Initiates an immediate skulk of the directories specified. The syntax is as follows:

directory synchronize  directory_name_list
The `synchronize` operation initiates an immediate skulk of the directories specified. The `directory_name_list` argument is a list of names of one or more directories to be operated on. Skulks begin immediately in sequence. The command does not return until all skulks complete. This operation returns an empty string on success.

**Privileges Required**

You must have `A` (Admin), `w` (write), `i` (insert), and `d` (delete) permission to the directory. The server principal (`hosts/hostname/cds-server`) needs `A` (Admin), `r` (read), and `w` (write) permission to the directory.

**Examples**

The following command begins a skulk on the `./admin` directory:

```
dcecp> directory synchronize ./admin
```

**Related Information**

Commands: `clearinghouse(8dce)`, `dcecp(8dce)`, `link(8dce)`, `object(8dce)`. 
Purpose

A dcecp object that manages a dtsd process

Synopsis

- **dts activate**: [dts_server] [-abruptly]
- **dts catalog**: [cell_name] [-simplename] [-global]
- **dts configure**: [dts_server] [-global | -notglobal]
- **dts deactivate**: [dts_server]
- **dts help**: [operation | -verbose]
- **dts modify**: [dts_server] -change {attribute_list | -attribute value}
- **dts operations**
- **dts show**: [dts_server] [-all | [-attributes] | [-counters]]
- **dts stop**: [dts_server]
- **dts synchronize**: [dts_server] [-abruptly]

Arguments

- **cell_name**: The name of a single cell. This name allows access to DTS servers registered in a foreign cell. The name must be a fully qualified cell name as in either of the following:
  - `./`
  - `./../foreign_cellname`

- **dts_server**: Identifies the dtsd server to act on. Supply the name in one of the following forms:
  1. As a fully qualified name, for example:
     ```
     ./../cellname/hosts/dce_hostname/dts-entity
     ```
  2. As a string binding for the remote host on which dtsd is running in standard string-binding syntax or in dcecp string syntax, for example:
     ```
     ncacn_ip_tcp:130.105.1.227
     {ncacn_ip_tcp 130.105.1.227}
     ```

- **operation**: The name of the dts operation for which to display help information.
Description

The **dts** object represents the **dtstd** (DTS daemon) process running on a host. The DTS process does not maintain stored data as some other objects do. Consequently, the **dts** object represents the information in and about a process rather than stored data.

These commands all affect the local **dtstd** entity by default. Use the **dtsd** server argument to operate on a remote DCE **dtstd**. This argument is a single server entry or string binding representing a **dtstd** that will be contacted for the operation. If the **_s(dts)** convenience variable is set, it is treated as the name of a **dtstd** to contact for subsequent operations. If either method is used, the specified server is the only server contacted in an attempt to complete the operation. The argument on the command line takes precedence over the value of the **_s(dts)** convenience variable. These commands do not set the value of this variable after completion.

A number of attributes are associated with the **dts** object. All can be viewed with the **show** operation, and many can be changed with the **modify** operation. Attribute arguments can contain a maximum of 80 characters and are recalculated to a normalized date format. For example, if the input value is `0-0025:10:99.99999999`, the result is `1-01:11:39.990`.

Timestamps are specified in DTS and ISO formats. They can be specified in both absolute and relative time formats. See the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide* for more information.

Attributes

The **dts** object supports attributes and counters. Most attributes and counters pertain to **dtstd** processes in general. A subset of attributes and counters pertains only to **dtstd** processes that are enabled as DTS server entities. The format of all attributes of type **relative_time** is in DTS-style (`[-]DD-HH:MM:SS`).

**General Attributes**

- **autotdfchange** {yes | no}
  Specifies whether automatic changes to the time differential factor are enabled or disabled. The value is either **yes** or **no**. The value is determined by the operating system (that is, it cannot be changed with the **modify** operation).

- **clockadjrate**
  Specifies the rate at which the DTS server or clerk entity adjusts the node's clock during a synchronization. This attribute can not be set by a user, but is built in to **dtstd**.

- **clockresolution**
  Specifies the amount of time between system clock ticks. The value is determined by the operating system (that is, it cannot be changed with the **modify** operation).

- **globalservers** **relative-time**
  Specifies the set of global servers known by the node. The information returned for each server is as follows: the DCE name of the host followed by **/self**, the last time polled, the last observed time, the last observed skew, a binary value of whether the server was used in the last
synchronization, and the transport time. These subattributes are called respectively name, timelastpolled, lastobstime, lastobsskew, inlastsync, and transport.

globaltimeout relative-time
Specifies the amount of time the node waits for a response to a wide area network (WAN) synchronization request before sending another request or declaring a global server to be unavailable. The number of attempts made to reach the server is controlled by the queryattempts attribute. The default value is 0-00:00:15.000, and the range of possible values is 0-00:00:00.000 to 0-00:10:00.000.

localservers
Specifies the set of local servers known by the node. The information returned for each server is as follows: the principal name that the server is running as, the last time polled, the last observed time, the last observed skew, a binary value indicating whether the server was used in the last synchronization, and the transport time. These subattributes are called respectively name, timelastpolled, lastobstime, lastobsskew, inlastsync, and transport.

localtimeout relative-time
Specifies the amount of time the node waits for a response to a synchronization request before sending another request or declaring a server to be unavailable. The number of attempts made to reach the server is controlled by the queryattempts attribute. The default is 0-00:00:05.000, and the range of possible values is 0-00:00:00.000 to 0-00:01:00.000.

Note that this attribute controls only the initial contact with a time provider. During this initial contact, the time-provider itself determines the timeout value for actually reporting back times, allowing time providers attached to a slow source, like a modem, to request that dtst wait for a longer interval.

maxdriftate
Specifies the worst-case drift rate of the node’s clock, in nanoseconds per second, as determined by the manufacturer’s specifications (that is, it cannot be changed with the modify operation).

maxinaccuracy relative-time
Specifies the inaccuracy limit for the node. When the node exceeds the maximum inaccuracy setting, it attempts to synchronize. The default is 0-00:00:00.100, and the range of possible values is 0-00:00:00.0 to 10675199-02:48:05.478. The maximum number of hours is 24. A practical value is less than 60 seconds.

minservers integer
Specifies the minimum number of servers required for a synchronization. Settings of 1 or 2 for a DTS server might cause unreliable computed times. The default is 3 for a DTS server and 1 for a DTS clerk. The range of possible values is 1 to 10.

nexttdfchange
Specifies the future time at which the time differential factor is automatically changed. The value is determined by the operating system (that is, it cannot be changed with the modify operation).

queryattempts integer
Specifies the number of attempts a node makes to contact a server before the node considers the server unavailable. The default is 3, and the range of possible values is 1 to 10.
**status** Specifies the state of the DTS entity. This is a read-only attribute and its possible values are as follows:

**disabled**
- The DTS entity is disabled.

**enabled**
- The DTS entity is enabled.

**syncing**
- The DTS entity is synchronizing.

**updating**
- The DTS entity is updating the time.

**syncinterval** `relative-time`
- Specifies the interval a node must wait to synchronize. Also specifies synchronization frequency when a node reaches the value specified by the **maxinaccuracy** attribute. For clerks the default is `0-00:10:00.0`, and the range of possible values is `0-00:00:30.0` to `01-00:00:00.00`. For servers the default is `0-00:02.00.0`, and the range of possible values `0-00:00:30.0` to `01-00:00:00.00`.

**tdf** `relative-time`
- Specifies the time differential factor (TDF), which is the amount of time the server varies from Greenwich mean time (GMT) or Universal Time Coordinated (UTC). The default is based on time zone information, with the range of possible values being `-13-00:00:00` to `13-00:00:00`. This can not be set by a user, but rather is obtained from various time zone information repositories (such as the **TZ** environment variable, kernel structures, and so on).

**timerep**
- Specifies the internal timestamp format used by the node. This format is not related to the format used to display the current time to the user (see the **clock show** command). Currently DTS uses **V1.0.0** timestamps only. This attribute cannot be set by a user, but is built in to a **dtsd**.

**tolerance** `relative-time`
- Specifies the maximum separation allowed between the local clock and the computed time before synchronizations become abrupt rather than gradual (monotonic). The default is `0-00:05:00.000`, and the range of possible values is `0-00:00:00.500` to `10675199-02:48:05.478`.

**type**
- Specifies whether the node is a DTS **server** or **clerk**.

**version**
- Specifies the DTS software version installed on the node. This attribute cannot be changed with the **modify** operation.

### DTS Server Attributes

**actcourierrole**
- Specifies a server’s **acting** interaction with the set of global servers. The values are the same as for the **courierrole** attribute below. The difference between **actcourierrole** and **courierrole** is that even when the value of **courierrole** is **backup** there is no guarantee that the courier is acting as a courier unless **actcourierrole** also specifies **backup**. The **actcourierrole** attribute indicates the actual role of the server. The default is **courier**.

**checkinterval**
- Specifies the amount of time between checks for faulty servers. Applicable
dts(8dce)

only to servers that have external time-providers. The default is 0-01:30:00.00, and the range of the possible values is 0-00:00:30.000 to 10675199-02:48:05.478.

courierrole
Specifies a server’s interaction with the set of global servers. Possible values are as follows:

backup
The local server becomes a courier if none are available on the local area network (LAN). This is the default.

courier
The local server synchronizes with the global set of servers.

noncourier
The local server does not synchronize with the global set of servers.

epoch
Specifies the server’s epoch number. The default is 0, and the range of possible values is 0 to 255. This value can not be changed with the modify command; use the clock set command with the -epoch option to change its value.

provider
Specifies whether the entity used an external time-provider at the last successful synchronization. This attribute applies to servers only and can not be set by a user. The value is either yes or no.

serverentry
Specifies a server’s access control list (ACL) entry name. The default setting is the following recommended value: hosts/dce_hostname/dts-entity.

servergroup
Specifies the security group name for the time servers within the cell. The default is subsys/dce/dts-servers.

serverprincipal
Specifies a server’s principal name for authentication purposes. The default setting is the following recommended value: hosts/dce_hostname/self.

uuid uuid
Specifies the entity’s unique identifier, which is generated when the entity is created.

General Counters
abrupts
Specifies the number of times the node clock has been set non-monotonically (abruptly).

badlocalservers
Specifies the number of times a local server was contacted, but was not in the DTS security group.

badprotocols
Specifies the number of times the local node failed to process a received message containing an incompatible protocol version.

badtimeres
Specifies the number of times the local node failed to process a received message containing an incompatible timestamp format.
**creationtime**
Specifies the time at which the DTS entity was created and the counters were initialized.

**disables**
Specifies the number of times the DTS has been disabled.

**enables**
Specifies the number of times the DTS has been enabled.

**nolocalintersections**
Specifies the number of times the node’s time interval failed to intersect with the computed interval of the servers.

**nomemories**
Specifies the number of times the node has been unable to allocate virtual memory.

**providerfailures**
Specifies the number of times a server process initiated contact with a time-provider and did not receive the initial response within the interval specified by the `localtimeout` attribute.

**syncs**
Specifies the number of times the node synchronized successfully.

**syserrors**
Specifies the number of times a DTS process detected a system error.

**toofewservers**
Specifies the number of times a node failed to synchronize because it could not contact the required minimum number of servers.

**DTS Server Counters**

**badservers**
Specifies the number of times a non-local server was contacted, but was not in the DTS security group.

**diffepochs**
Specifies the number of times the node received time response messages from servers or clerks that had epoch numbers different from its own.

**epochchanges**
Specifies the number of times the server’s epoch has changed.

**noglobals**
Specifies the number of times the courier server could not contact any global servers.

**noresponses**
Specifies the number of times the courier server could not contact a specific global server.

**noserverintersections**
Specifies the number of times a server has detected faulty servers (other than itself).

**providerfailures**
Specifies the number of times the external time-provider signaled a failure, or the node was unable to access the time-provider.

**updates**
Specifies the number of times a server has attempted to synchronize its clock.
dts(8dce)

See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information about DTS attributes.

Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Operations

**dts activate**

Changes a DTS entity from an inactive state to an active state. The syntax is as follows:

```
dts activate  [dts_server] [-abruptly]
```

**Options**

- **-abruptly**
  Sets the clock abruptly rather than gradually adjust it to the computed time.

The activate operation changes a DTS entity from an inactive state to an active state. The status attribute is changed to enabled. This attribute tells the DTS entity to begin synchronizing. This operation takes an -abruptly option to determine whether the first clock adjustment due to synchronization is an abrupt or gradual one, and returns an empty string on success.

**Privileges Required**

You must have w (write) permission on the DTS entity to execute the command.

**Examples**

The following example activates a dtsd on the local host:

```
dcecp> dts activate
dcecp>
```

The following example activates a dtsd on a remote host named cyclops:

```
dcecp> dts activate /.:/hosts/cyclops/dts-entity
dcecp>
```

**dts catalog**

Returns a list of the names of all DTS servers registered in the local cell. The syntax is as follows:

```
dts catalog  [cell_name] [-simplename] [-global]
```

**Options**

- **-simplename**
  Returns a list of registered DTS servers without prepending the cell name.
Returns a list of registered global DTS servers.

The `catalog` operation returns a list of the names of all DTS servers registered in the default LAN profile (`/..:lan-profile`). Any DTS servers registered in the cell profile (`/..:cell-profile`) or in an additional LAN profile will also be returned. The additional LAN profile must exist at the root (`/..:`) level of the CDS namespace. The operation takes an optional `cell_name` argument that can return the names of DTS servers registered in a foreign cell. By default, fully qualified names are returned in the following form:

```
/.../cell_name/hosts/dce_hostname
/dts-entity
```

If the `-simplename` option is given, the cell name is not prepended to the DTS server names. The `-global` option returns only DTS servers that are operating as global servers. Names are returned in lexical order.

**Privileges Required**

You must have `r` (read) permission to the cell root (`/..:`) directory and to the LAN profile.

**Examples**

```
dcecp> dts catalog
/.../my_cell.goodcompany.com/hosts/frick/dts-entity
/.../my_cell.goodcompany.com/hosts/ice/dts-entity
/.../my_cell.goodcompany.com/hosts/ninja/dts-entity

dcecp>

dcecp> dts catalog -simplename
hosts/frick/dts-entity
hosts/ice/dts-entity
hosts/ninja/dts-entity

dcecp>
```

**dts configure**

Configure the local `dtsd` as a local or global server. The syntax is as follows:

```
dts configure [dts_server] {-global | -notglobal}
```

**Options**

- **global** Configures the system as a global server by adding the server’s entry to the cell profile
- **notglobal** Configures the system as a local server by removing the server’s entry from the cell profile

The `configure` operation sets the local `dtsd` to be a local or global server. You must specify either the `-global` or `-notglobal` option to indicate whether to configure the local `dtsd` as a global server. The difference is whether the server is listed in the `/..:cell-profile`. This command returns the string `global` or `notglobal` to indicate the current (new) state of the `dtsd`.

**Privileges Required**
You must have \texttt{w (write)} permission on the DTS entity in order to execute the command.

**Examples**

The following example sets the local \texttt{dtsd} to be a global DTS server:

```ini

dcecp> dts configure -global
global
dcecp>
```

\textbf{dts deactivate}

Changes a DTS entity from an active state to an inactive state. The syntax is as follows:

```ini

dts deactivate [dts_server]
```

The \texttt{deactivate} operation changes a DTS entity from an active state to an inactive state. The \texttt{status} attribute is changed to \texttt{disabled}, which tells the DTS entity to stop synchronizing. This operation returns an empty string on success.

**Privileges Required**

You must have \texttt{w (write)} permission on the DTS entity to execute the command.

**Examples**

```ini

dcecp> dts deactivate
dcecp>
```

\textbf{dts help}

Returns help information about the \texttt{dts} object and its operations. The syntax is as follows:

```ini

dts help [operation | -verbose]
```

**Options**

\texttt{-verbose}

Displays information about the \texttt{dts} object.

Used without an argument or option, the \texttt{dts help} command returns brief information about each \texttt{dts} operation. The optional \texttt{operation} argument is the name of an operation about which you want detailed information. Alternatively, you can use the \texttt{-verbose} option for more detailed information about the \texttt{dts} object itself.

**Privileges Required**

No special privileges are needed to use the \texttt{dts help} command.

**Examples**

```ini

dcecp> dts help activate Activates a DTS entity.
catalog Returns a list of DTS servers in the cell.
configure Configures current dtsd as 'global' or 'notglobal'.
```
dts(8dce)

deactivate Deactivates a DTS entity.
modify Modifies attributes of the DTS entity.
show Displays attributes or counter info of the named dtsd.
stop Stops the current dtsd process.
synchronize Synchronizes the local dtsd with DTS servers.
help Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.

dcecp> dts modify
Changes attributes of dtsd processes. The syntax is as follows:

dts modify [dts_server] {-change attribute_list | -attribute value}

Options
-change attribute_list
  Allows you to modify attributes by using an attribute list rather than using
  individual attribute options. The format of an attribute list is as follows:

  {{attribute value}...{attribute value}}

- attribute value
  As an alternative to using options with an attribute list, you can change
  individual attribute options by prepending a hyphen (-) to any attributes
  listed in the Attributes section of this reference page.

The modify operation changes attributes of dtsd processes. It allows attributes to
be changed with the -change option. Attribute options are also supported for all
modifiable attributes. This operation returns an empty string on success.

Privileges Required

You must have w (write) permission on the DTS entity to execute the command.

Examples

The following example sets the minimum number of servers needed for DTS
operation to 5 for a remote dtsd. (Generally a DCE cell should have a minimum of
three DTS servers.)

dcecp> dts modify ncacn_ip_tcp:130.105.1.227 -minservers 5
dcecp>

dcecp> dts modify ncacn_ip_tcp:130.105.1.227 -change {minservers 5}
dcecp>

dts operations

Returns a list of the operations supported by the dts object. The syntax is as
follows:


dts operations

The list of available operations is in alphabetical order except for help and
operations, which are listed last.
dts(8dce)

Privileges Required

No special privileges are needed to use the dts operations command.

Examples

dcecp> dts operations
activate catalog configure deactivate modify show stop
> synchronize help operations
dcecp>

dts show

Returns attribute information for the specified dtsd processes. The syntax is as follows:

\f[dts show \[dts_server\] \[-all | \[-attributes] \[-counters]]\]

Options

-attributes
Returns only the attributes for the local dtsd process.

-counters
Returns only the counters for the local dtsd process.

-all
Return the attributes and counters for the local dtsd process.

The show operation shows attribute information for the specified dtsd processes. When called with the -attributes option, dts show returns an attribute list giving the values of the attributes listed above. If called with the -counters option counter information is returned. If called with the -all or with both the -attributes and -counters options, both attribute and counter information is returned. The default behavior (invoked by using no options) is the same as if the -attributes option was used. Attributes and counters are listed in the order they are returned by the server.

Privileges Required

You must have r (read) permission on the DTS entity to execute the command.

Examples

dcecp> dts show
{checkinterval +0-01:30:00.000I-----}
{epoch 0}
{tolerance +0-00:10:00.000I-----}
{tdf -0-05:00:00.000I-----}
{maxinaccuracy +0-00:00:00.100I-----}
{minservers 2}
{queryattempts 3}
{localtimeout +0-00:00:05.000I-----}
{globaltimeout +0-00:00:15.000I-----}
{syncinterval +0-00:02:00.000I-----}
{type server}
{courierrole backup}
{actcourierrole courier}
{clockadjrate 10000000 nsec/sec}
{maxdriftrate 10000000 nsec/sec}
{clockresolution 10000000 nsec}
{version V1.0.1}
**dts stop**

Stops the **dtsd** process. The syntax is as follows:

```
dts stop [dts_server]
```

The **stop** operation stops the **dtsd** process. This operation returns an empty string on success.

**Privileges Required**

You must have **w (write)** permission on the DTS entity to execute the command.

**Examples**

The following example stops the **dtsd** process on remote host named **cyclops**:

```
dcecp> dts stop ./:/hosts/cyclops/dts-entity
```

**dts synchronize**

Causes **dtsd** to synchronize with DTS servers. The syntax is as follows:

```
dts synchronize [dts_server] [-abruptly]
```

**Options**

- **-abruptly**
  
  Sets the clock abruptly rather than gradually adjust it to the computed time.

The **synchronize** operation causes **dtsd** to synchronize with DTS servers. The machine’s clock is adjusted accordingly. By default, the clock is adjusted gradually. This command also takes the optional **-abruptly** option to set the clock abruptly. This operation returns an empty string on success.

**Privileges Required**

You must have **w (write)** permission on the DTS entity to execute the command.

**Examples**

The following example causes the local **dtsd** process to synchronize with other DTS servers in the cell:

```
dcecp> dts synchronize
```

---

*Chapter 1. DCE Commands 165*
The following example causes the `dtsd` process on a remote host named `cyclops` to synchronize immediately with other DTS servers in the cell:

```
dcecp> dts synchronize ./hosts/cyclops/dts-entity -abruptly
dcecp>
```

**Related Information**

Commands: `clock(8dce)`, `dcecp(8dce)`, `dtsd(8dts)`, `utc(8dce)`. 
Purpose

A DCE object that manages the EMS daemon on a DCE host.

Synopsis

```
ems catalog
ems help [operation | -verbose ]
ems operations
ems show [-host dce_hostname ]
```

Arguments

```
operation
    The name of one specific ems operation for which to display help information.
```

Description

The ems object represents the EMS daemon (called emsd) on a host.

This command operates on the EMS daemon on the local host, unless the —host option is specified. The format of the host name accepted is either an entire DCE name (/./hosts/jurassic.austin.ibm.com) or a host name with the domain name (jurassic.austin.ibm.com).

Attributes

```
eventlog_dir
    Specifies the directory name used where the EMS daemon puts the event log.
queue_size
    Specifies the queue size for the event queues.
```

Operations

```
ems catalog
```

Returns the list of all hosts that the EMS daemon is running on in the current cell. The syntax is as follows:

```
ems catalog
```

Options

```
none
```

Returns the list of all hosts that the EMS daemon is running on in the current cell.

Privileges Required
ems

No special privileges are needed to use the ems catalog command.

Examples

dcecp> ems catalog
./.:/hosts/eagle.austin.ibm.com
./.:/hosts/umesh.austin.ibm.com
dcecp>

ems help

Returns help information on the object. The syntax is as follows:

`ems help [operation | -verbose]`

Options

-verbose
Displays information about the ems object.

The ems help command returns help information on the object. The help operation takes an argument, which can be an operation supported by the object or the -verbose switch to return more information.

Privileges Required

No special privileges are needed to use the ems help command.

Examples

dcecp> ems help
catalog Returns a list of all hosts that the EMS daemon is running on.
help Prints a summary of command-line options.
operations Returns the valid operations for command.
show Returns the attributes for the EMS daemon.
dcecp>

ems operations

Returns a list of operations supported by the ems object. The syntax is as follows:

ems operations

The ems operations command returns a list of operations supported by the object. It takes no arguments, and always returns a TCL list suitable for use in a 'foreach' statement. The order of the elements is alphabetical with the exception that help and operations are listed last.

Privileges Required

No special privileges are needed to use the ems operations command.

Examples

dcecp> ems operations
catalog show help operations
dcecp>
**ems show**

Returns attribute information for the EMS daemon. The syntax is as follows:

```
ems show  [-host  dce_hostname]
```

**Options**

- **-host  dce_hostname**
  Specifies the host where the EMS daemon is running. The format of the host name is either an entire DCE name or a host name with a domain name.

  **Note:** The DCE host name is case-sensitive.

The `ems show` command returns the attribute list for the EMS daemon.

**Privileges Required**

You must have read (r) permission on

`/./hosts/dce_hostname/ems-server`

**Examples**

```
dcecp> ems show
  eventlog_dir  /opt/dcelocal/dce/var/ems}
  {queue_size  5000}
```

**Related Information**

Commands: `emsconsumer commands`, `emsevent commands`. 
**emsconsumer**

**Purpose**

A dcecp object that manages EMS consumers and their event filter groups.

**Synopsis**

```bash
emsconsumer catalog [-host dce_hostname]
emsconsumer delete consumer {-uuid uuid} [-host dce_hostname]
emsconsumer help [operation | -verbose ]
emsconsumer modify consumer {-uuid uuid} {-add | -delete} {filter filtername} [-host dce_hostname]
emsconsumer operations
emsconsumer show consumer {-uuid uuid} [-host dce_hostname ]
```

**Arguments**

- `consumer`
  A consumer name.
- `operation`
  The name of one specific `emsconsumer` operation (subcommand) that you want to see help information about.

**Description**

The `emsconsumer` object represents an EMS consumer. An EMS consumer registers with EMS to receive event data. It defines event filters to identify the events that should be forwarded to it.

This command operates on the EMS daemon on the local host, unless the `-host` option is specified. The format of the host name accepted is either an entire DCE name (`/./hosts/jurassic.austin.ibm.com`) or a host name with the domain name (`jurassic.austin.ibm.com`).

**Operations**

**emsconsumer catalog**

Returns the list of registered consumers with EMS on a host. The syntax is as follows:

```bash
emsconsumer catalog [-host dce_hostname]
```

**Options**
Specifies the host where the EMS Daemon is running. The format of the host name is either an entire DCE name or a host name with a domain name.

**Note:** The DCE host name is case-sensitive.

The `emsconsumer catalog` command returns the list of registered consumers with EMS on a host. The consumer names returned are in an arbitrary order.

**Privileges Required**

You must have read (r) permission on

`/./hosts/dce_hostname/ems-server/consumers`

**Examples**

```
dcecp> emsconsumer catalog
{consumer1 7e383761-f41f-11ce-9051-08005acd43c6 /./hosts/eagle.austin.ibm.com}
{consumer1 a4c7ff26-f449-11ce-a863-10005a4f3556 /./hosts/eagle.austin.ibm.com}
{consumer2 283cc40c-f447-11ce-9dd3-10005a4f3556 /./hosts/umesh.austin.ibm.com}
dcecp>
```

**emsconsumer delete**

Deletes a registered consumer from EMS on a host. The syntax is as follows:

```
emsconsumer delete consumer {-uuid uuid} [-host dce_hostname]
```

**Options**

- **-host dce_hostname**
  Specifies the host where the EMS Daemon is running. The format of the host name is either an entire DCE name or a host name with a domain name.

  **Note:** The DCE host name is case-sensitive.

- **-uuid uuid**
  Specifies the unique universal identifier (UUID) that is assigned to the consumer.

The `emsconsumer delete` command deletes a registered consumer from EMS on a host. The argument is the name of the consumer to be deleted or, in case of duplicate consumers, a consumer name with its assigned uuid. The command returns an empty string on success.

**Privileges Required**

You must have delete (d) permission on

`/./hosts/dce_hostname/ems-server/consumers`

**Examples**

```
dcecp> emsconsumer delete consumer2
```
```
dcecp> emsconsumer delete consumer1 -uuid 7e383761-f41f-11ce-9051-08005acd43c6
```
emsconsumer

emsconsumer help

Returns help information on the object. The syntax is as follows:

emsconsumer help  [operation | -verbose]

Options
-verbose
  Displays information about the emsconsumer object.

The emsconsumer help command returns help information on the object. The help operation takes an argument, which can be an operation supported by the object or the -verbose switch to return more information.

Privileges Required

No special privileges are needed to use the emsconsumer help command.

Examples

dcecp> emsconsumer help
catalog   Returns the list of registered consumers with EMS on a host.
delete    Deletes a registered consumer from EMS on a host.
modify    Modifies the event filter group associated with a consumer.
show      Returns the list of filter names in a consumer's filter group.
help      Prints a summary of command-line options.
operations Returns the valid operations for command.
dcecp>

emsconsumer modify

Modifies the event filter group associated with the given consumer. The syntax is as follows:

emsconsumer modify  consumer {-uuid uuid} {-add | delete} {filter filtername} {-host dce_hostname}

Options
-addr|delete
  Adds or deletes filternames from the consumer filter group.

filter  filtername
  Specifies the name of the consumer filter group.

-host  dce_hostname
  Specifies the host where the EMS daemon is running. The format of the host name is either an entire DCE name or a host name with a domain name.

  Note: The DCE host name is case-sensitive.

-uuid  uuid
  Specifies the unique universal identifier (UUID) that is assigned to the consumer.

The emsconsumer modify command modifies the event filter group associated with the given consumer. Filters can be added or deleted from a consumer event
filter group. Added filters are always placed at the end of the consumer event filter group. The command returns an empty string on success.

Privileges Required

You must have write (w) permission on

/l:/hosts/dce_hostname/ems-server/consumers

Examples

dcecp> emsconsumer modify consumer2 -add {filter foo}
dcecp>

emsconsumer operations

Returns a list of operations supported by the emsconsumer object. The syntax is as follows:

emsconsumer operations

The emsconsumer operations command returns a list of operations supported by the object. It takes no arguments and always returns a TCL list suitable for use in a ‘foreach’ statement. The order of the elements is alphabetical with the exception that help and operations are listed last.

Privileges Required

No special privileges are needed to use the emsconsumer operations command.

Examples

dcecp> emsconsumer operations
catalog delete modify show help operations
dcecp>

emsconsumer show

Returns the list of filter names in a consumer filter group. The syntax is as follows:

emsconsumer show consumer {-uuid uuid} [-host dce_hostname]

Options

-host dce_hostname
   Specifies the host where the EMS Daemon is running. The format of the host name is either an entire DCE name or a host name with a domain name.

   Note: The DCE host name is case-sensitive.

-uuid uuid
   Specifies the unique universal identifier (UUID) that is assigned to the consumer.

The emsconsumer show command returns the list of filter names in a consumers filter group. This command takes the consumer name as an argument.
emsconsumer

Privileges Required

You must have read (r) permission on

/./hosts/dce_hostname/ems-server

Examples

dcecp> emsconsumer show consumer2
    {foo2 foo3 foo4 foo5}

Related Information

Commands: emsfilter commands.
emsevent

Purpose

A dcecp object that manages EMS event types and event type schemas.

Synopsis

emsevent catalog [-host dce_hostname]
emsevent delete event_type [-host dce_hostname]
emsevent help [operation | -verbose ]
emsevent operations
emsevent show event_type [-host dce_hostname ]

Arguments

- event_type
  Name of the event type.
- operation
  The name of one specific emsevent operation (subcommand) that you want to see help information about.

Description

The emsevent object represents the EMS event type, which is a class of events with the same format. This format of the event types are defined by event type schemas. An event type schema consists of a list of attribute name/type pairs that specify the data format of an event.

This command allows for the list of available event types to be displayed, and the event type schema for a particular event type. It operates on the EMS daemon on the local host, unless the -host option is specified. The format of the host name accepted is either an entire DCE name (/./hosts/jurassic.austin.ibm.com) or a host name with the domain name (jurassic.austin.ibm.com).

Operations

emsevent catalog

Returns the list of available event types. The syntax is as follows:

emsevent catalog [-host dce_hostname]

Options

- -host dce_hostname
  Specifies the host where the EMS Daemon is running. The format of the host name is either an entire DCE name or a host name with a domain name.
emsevent

Note: The DCE host name is case-sensitive.

The emsevent catalog command displays a list of the available event types.

Privileges Required

You must have read (r) permission on

/./hosts/dce_hostname/ems-server/event-types

Examples

dcecp> emsevent catalog
SVC
audit

emsevent delete

Deletes an event type from EMS on a host. The syntax is as follows:

emsevent delete  event_type_name  [-host  dce_hostname]

Options

-host  dce_hostname
Specifications the host where the EMS Daemon is running. The format of the host name is either an entire DCE name or a host name with a domain name.

Note: The DCE host name is case-sensitive.

The emsevent delete command deletes an event type. The argument is the name of the event type to be deleted. The command returns an empty string if successful.

Privileges Required

You must have delete (d) permission on

/./hosts/dce_hostname/ems-server/event-types

or

/./hosts/dce_hostname/ems-server/event-types/event_types_name

Examples

dcecp> emsevent delete EventType

dcecp>

emsevent help

Returns help information on the object. The syntax is as follows:

emsevent help  [operation  |  -verbose]

Options
emsevent

-verbose
Displays information about the emsevent object.

The emsevent help command returns help information on the object. The help operation takes an argument, which can be an operation supported by the object or the -verbose switch to return more information.

Privileges Required

No special privileges are needed to use the emsevent help command.

Examples

dcecp> emsevent help
catalog Returns the list of available event types.
delete Deletes an event type.
help Prints a summary of command-line options.
operations Returns the valid operations for command.
show Returns the event type schema for a event type.
dcecp>

dcecp> emsevent operations

emsevent operations

Returns a list of operations supported by the emsevent object. The syntax is as follows:

emsevent operations

The emsevent operations command returns a list of operations supported by the object. It takes no arguments, and always returns a TCL list suitable for use in a ‘foreach’ statement. The order of the elements is alphabetical with the exception that help and operations are listed last.

Privileges Required

No special privileges are needed to use the emsevent operations command.

Examples

dcecp> emsevent operations
catalog delete show help operations
dcecp>

dcecp> emsevent show

emsevent show

Returns the event type schema for a event type. The syntax is as follows:

emsevent show  event_type [-host dce_hostname]

Options

-host dce_hostname
Specifies the host where the EMS daemon is running. The format of the host name is either an entire DCE name or a host name with a domain name.

Note: The DCE host name is case-sensitive.
emsevent

The emsevent show command returns the event type schema for a event type. A list of attribute name/type pairs is displayed.

Privileges Required

You must have read (r) permission on

/./hosts/dce_hostname/ems-server/event-types/event_type

Examples

dcecp> emsevent show SVC
   {version ems_c_attr_ulong_int}
   {t ems_c_attr_utc}
   {argtypes ems_c_attr_char_string}
   {table index ems_c_attr_ulong_int}
   {attributes ems_c_attr_ulong_int}
   {message index ems_c_attr_ulong_int}
   {format ems_c_attr_char_string}
   {file ems_c_attr_char_string}
   {prognname ems_c_attr_char_string}
   {line ems_c_attr_char_string}
   {threadid ems_c_attr_ulong_int}
   {component name ems_c_attr_char_string}
   {sc_name ems_c_attr_char_string}
   {attribute.debug ems_c_attr_ushort_int}
   {attribute.severity ems_c_attr_ushort_int}
   {attribute.actroute ems_c_attr_ulong_int}

Related Information

Commands: emslog commands.
emsfilter

Purpose

A dcecp object that manages EMS event filters on a DCE host.

Synopsis

emsfilter catalog [-host dce_hostname]
emsfilter delete filtername [-host dce_hostname]
emsfilter help [operation | -verbose ]
emsfilter operations
emsfilter show filtername [-host dce_hostname ]

Arguments

filtername
   A filter name.

operation
   The name of one specific emsfilter operation (subcommand) that you want
to see help information about.

The emsfilter object represents EMS event filters that are kept by the EMS daemon.
The EMS event filters are applied by EMS to events received from suppliers to
determine if the events are to be forwarded on to the consumers.

An EMS event filter is a collection of one or more filter expressions. Each filter
expression consists of an attribute name, an attribute operator, and an attribute
value.

This command operates on the EMS daemon on the local host, unless the -host
option is specified. The format of the host name accepted is either an entire DCE
name (/./hosts/jurassic.austin.ibm.com) or a host name with the domain name
(jurassic.austin.ibm.com).

Operations

emsfilter catalog

Returns a list of names of all filters from EMS on a host. The syntax is as follows:

emsfilter catalog [-host dce_hostname]

Options

-host dce_hostname
   Specifies the host where the EMS Daemon is running. The format of the
host name is either an entire DCE name or a host name with a domain
name.

   Note: The DCE host name is case-sensitive.
The `emsfilter catalog` command returns a list of names of all filters from EMS on a host. The filter names returned are in alphabetical order and not in the order received by EMS.

**Privileges Required**

You must have read (r) permission on

```
./hosts/dce_hostname/ems-server/filters
```

**Examples**

In the following example, there are two filters kept by the EMS daemon:

```
dcecp> emsfilter catalog
Filter1
Filter2
```

`emsfilter delete`

Deletes a filter and its associated filter expressions. The syntax is as follows:

```
emsfilter delete  filtername  [-host  dce_hostname]
```

**Options**

- `-host dce_hostname`
  
  Specifies the host where the EMS Daemon is running. The format of the host name is either an entire DCE name or a host name with a domain name.

  **Note:** The DCE host name is case-sensitive.

The `emsfilter delete` command deletes a filter and its associated filter expressions. The argument is a filtername to be deleted. If the filter to be deleted is currently being used by at least one consumer, it cannot be deleted and an error message is displayed. The command returns an empty string on success.

**Privileges Required**

You must have delete (d) permission on

```
./hosts/dce_hostname/ems-server/filters/filtername.
```

**Examples**

```
dcecp> emsfilter delete Filter1
```

`emsfilter help`

Returns help information on the object. The syntax is as follows:

```
emsfilter help  [operation  |  -verbose]
```

**Options**
Displays information about the emsfilter object.

The emsfilter help command returns help information on the object. The help operation takes an argument, which can be an operation supported by the object or the -verbose switch to return more information.

Privileges Required

No special privileges are needed to use the emsfilter help command.

Examples

dcecp> emsfilter help
  catalog Returns a list of names of all filters from EMS on a host.
  delete Deletes a filter and its associated filter expressions.
  help Prints a summary of command-line options.
  operations Returns the valid operations for command.
  show Returns a list of filter expressions in a specified filter.

dcecp>

emsfilter operations

Returns a list of operations supported by the emsfilter object. The syntax is as follows:

emsfilter operations

The emsfilter operations command returns a list of operations supported by the object. It takes no arguments, and always returns a TCL list suitable for use in a ‘foreach’ statement. The order of the elements is alphabetical with the exception that help and operations are listed last.

Privileges Required

No special privileges are needed to use the emsevent operations command.

Examples

dcecp> emsfilter operations
  catalog delete show help operations

dcecp>

emsfilter show

Returns a list of filter expressions in a specified filter. The syntax is as follows:

emsfilter show  fullname [-host dce_hostname]

Options

-host dce_hostname
  Specifies the host where the EMS daemon is running. The format of the host name is either an entire DCE name or a host name with a domain name.

  Note: The DCE host name is case-sensitive.

emsfilter

The **emsfilter show** command returns a list of filter expressions in a specified filter. The argument is a filter name to be shown.

**Privileges Required**

You must have read (r) permission on

`/.:/hosts/dce_hostname/ems-server/filters`

**Examples**

```
dcecp> emsfilter show Filter2
{event_type == SVC}
{file == file.c}
```

**Related Information**

Commands: **emsconsumer commands**.
emslog

Purpose

A dcecp object that manages the EMS log files on the current host.

Synopsis

`emslog help [operation | -verbose ]`

`emslog operations`

`emslog show [-dir directory ] [-to file]`

Arguments

`operation`
the name of one specific emslog operation (subcommand) you want to see help information about.

`directory`
the name of the directory where the log file is stored.

`file`
the name of the file where the log is stored.

Description

The emslog object represents the EMS event log, which is used to store events in case of failures of the EMS daemon. The EMS daemon writes all events to the event log and deletes the event record once the event has been transmitted to all the consumers that were supposed to get the event.

The event log is kept in a file on the machine where EMS daemon is running. This command operates on the EMS daemon on the local host.

Operations

`emslog help`

Returns help information on the object. The syntax is as follows:

`emslog help [operation | -verbose]`

Options

- `verbose`
  Displays information about the emslog object.

The `emslog help` command returns help information on the object. The help operation takes an argument, which can be an operation supported by the object or the `-verbose` switch to return more information.

Privileges Required

No special privileges are needed to use the `emslog help` command.
emslog

Examples

dcecp> emslog help
help  Prints a summary of command-line options.
operations Returns the valid operations for command.
show    Returns a list of events in the event log file.
dcecp>

emslog operations

Returns a list of operations supported by the emslog object. The syntax is as follows:

emslog operations

The emslog operations command returns a list of operations supported by the object. It takes no arguments, and always returns a TCL list suitable for use in a 'foreach' statement. The order of the elements is alphabetical with the exception that help and operations are listed last.

Privileges Required

No special privileges are needed to use the emslog operations command.

Examples

dcecp> emslog operations
show help operations
dcecp>

emslog show

Returns a list of events in the event log file. The syntax is as follows:

emslog show [-dir directory] [-to file]

Options

-dir directory  Specifies the directory where the log file is stored.
-to file       Specifies the filename that the output is captured into.

The emslog show command returns a list of events in the event log file. If the -dir option is not specified, the default event log directory, /opt/dcelocal/var/ems, is assumed.

Privileges Required

No special privileges are needed to use the emslog show command.

Examples

dcecp> emslog show
--- Start of an EMS event record ---
Type: SVC:Event Id: 8d1b0b00-e9e7-11ce-8af3-10005a890435
Name Service: DCE /.../eagle_dce/hosts/hidalgod.austin.ibm.com
Description Name: EMS_Test_Producer
PID: 565 UID: 0 GID: 0
Severity: NOTICE
Arrival Time: 1995-09-08-14:06:32.970+00:00
Printing 16 items
Item 1: [version] = ulong init 1
Item 2: [t] = 1995-09-08-14:06:32.970+00:00
Item 3: [argtypes] = char string
Item 4: [table_index] = ulong int 0
Item 5: [attributes] = ulong int 64
Item 6: [message_index] = ulong int 389738500
Item 7: [format] = char string Test Supplier starting
Item 8: [file] = char string supplier.c
Item 9: [progname] = char string EMS_Test_Producer
Item 10: [line] = ulong int 63
Item 11: [threadid] = ulong int 2
Item 12: [component_name] = char string sup
Item 13: [sc_name] = char string general
Item 14: [attribute.debug] = ushort int 0
Item 15: [attribute.severity] = ushort int 4
Item 16: [attribute.actroute] = ulong int 0
--- End of an EMS event record ---

Related Information

Commands: ems commands, emsevent commands.
Purpose

Starts the DCE Event Management Services Daemon.

Synopsis

```bash
emsd [-l log_directory] [-q queue_size] [-w svc_route...-w svc_route]
```

Options

- `-l log_directory`
  Specifies where the log file resides.

- `-q queue_size`
  Specifies the maximum number of events that are queued by EMS. The default size is 512. This value can also be set by setting the EMS_QUEUE_SIZE environment variable. Specifying the `-q` option overrides the environment variable setting.

- `-w svc_route`
  Specifies DCE serviceability routing instructions.

Description

The `emsd` command starts the Event Management Service (EMS) daemon. An EMS daemon must be running in the DCE cell before a consumer can receive events or a supplier can supply events. The EMS daemon runs under the local host machine principal identity (`host/dce_hostname/self`). A DCE Host daemon (`dced`) must be running on the local host when `emsd` is started. The `emsd` command also requires a CDS advertiser.

Privileges Required

No special privileges are needed to use the `emsd` command.

Examples

```bash
emsd -q 2048 -l /opt/dcelocal/var/ems emsd -w NOTICE:STDOUT:-
   -w NOTICE_VERBOSE:STDOUT-:-
```

Related Information

None.
**endpoint**

**Purpose**

A dcecp object that manages endpoint information in local RPC endpoint maps

**Synopsis**

```
endpoint create -interface interface_id -binding string_binding_list
[-object object_uuid_list] [-annotation annotation] [-noreplace]

endpoint delete -interface interface_id -binding string_binding_list
[-object object_uuid_list]

endpoint help [operation | -verbose]

endpoint operations

endpoint show [host_address] [-uuid | -interface interface_id | [-version versions] | [-object object_uuid_list]]
```

**Arguments**

- **host_address**
  A string binding identifying the host whose endpoint map is to be returned. See Data Structures for the format of host_address.

- **operation**
  The name of the endpoint operation for which to display help information.

**Description**

The endpoint object operates on remote procedure call (RPC) endpoint mappings on the local host. Endpoints contain an interface identifier and one or more string bindings; optionally, they contain object Universal Unique Identifiers (UUIDs) and an annotation.

Endpoint mappings are stored in the endpoint map maintained by the DCE daemon (dced) for DCE Version 1.1 hosts. DCE Version 1.0 uses the RPC daemon (rpcd) to maintain the endpoint map. The server object has some operations (for example, disable and enable) that affect endpoints maintained by dced. However, server object operations do not operate on endpoints maintained by DCE Version 1.0 hosts. The endpoint object affects all endpoint maps on the local host, whether maintained by rpcd or dced.

Since endpoints have no names, the arguments to these operations are not the name of an endpoint. Earlier versions of rpcd allowed remote access to endpoints, but this was a security problem. Only the endpoint show command allows access to endpoint maps on remote systems. The server object allows some remote operations on dced endpoint maps, which are free of the security problem, depending on how dced is configured.

Use the various endpoint operations to create, delete, and show RPC endpoint information in local host endpoint maps.
interface_id

The interface identifier of an RPC interface. The interface identifier takes the following form:

interface-uuid, major-version, minor-version

The version numbers are optional, but if you omit a version number, the value defaults to 0. The UUID is a hexadecimal string and the version numbers are decimal strings. For example:

-interface ecleeb60-5943-11c9-a309-08002b102989,3.11

Leading zeros in version numbers are ignored.

Alternatively, you can use dcecp string syntax in the following form:

{interface-UUID major-version.minor-version}

For example:

-interface {458ff2be-98c1-11cd-0000c08af56 1.0}

string_binding_list

An RPC string binding that describes a server’s location. The value has the form of an RPC string binding, without an object UUID. The binding information contains an RPC protocol, a network address, and (sometimes) an endpoint within [ ] (square brackets) as follows:

rpc-prot-seq: network-addr[ endpoint]

For a well-known endpoint, include the endpoint in the string binding surrounded by brackets. You might need to use the \ (backslash) to escape the brackets as shown in the following example. Without the backslash, dcecp interprets the brackets as enclosing another command.

-binding ncadg_ip_udp:63.0.2.17[5347]

For a dynamic endpoint, omit the endpoint from the string binding. For example:

-b ncacn_ip_tcp:16.20.15.25

Alternatively, you can use dcecp string syntax. For example:

-binding {ncacn_ip_tcp 130.105.1.227 1072}

object_uuid

The UUID of an object. The UUID is a hexadecimal string. For example:

-object 3c6b8f60-5945-11c9-a236-08002b102989

Alternatively, you can use dcecp string syntax. For example:

-object {3c6b8f60-5945-11c9-a236-08002b102989}

host_address

An RPC string binding that describes a host’s location. The binding
information contains an RPC protocol and the host’s network address. Any specific host’s network address can be obtained by using the `getip` command.

**annotation**
An informational text string that helps you to identify the purpose of the endpoint. Use single or double quotation marks around the annotation field of endpoints to include internal spaces in an annotation, for example:

```
-annotation "Bulletin Board Server, Version 1.3a"
```

Alternatively, you can use `dcecp` string syntax. For example:

```
-annotation {Bulletin Board Server, Version 1.3a}
```

**version**
Specifies which interface version numbers to be returned with a `show` operation. Specify versions by using one of the following values for the `-version` option:

- `all` The interface version is ignored.
- `exact` Both the major and minor versions must match the specified versions.
- `compatible` The major version must match the specified version, and the minor version must be greater than or equal to the specified version.
- `major_only` The major version must match the specified version; the minor version is ignored.
- `upto` The major version must be less than or equal to that specified. If the major versions are equal, the minor version must be less than or equal to that specified.

If the `-version` option is absent, the command shows `compatible` version numbers.

**Errors**
A representative list of errors that might be returned is not shown here. Refer to the *IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide* for complete descriptions of all error messages.

**Operations**

**endpoint create**
Creates new endpoints in the local endpoint map database. The syntax is as follows:

```
endpoint create -interface interface_id -binding string_binding_list
[-object object_uuid_list] [-annotation annotation] [-noreplace]
```

**Options**
The `create` operation creates new endpoints in the endpoint map database on the local host. This command takes no arguments. It requires the `-interface` and `-binding` options, and accepts the `-object` and `-annotation` options. The value of the `-binding` and `-object` options can be a list, but the others must be a single value. If the mapping already exists, it is replaced unless the `-noreplace` option is included.

This command creates a cross product from the `-interface`, `-binding`, and `-object` options and adds each element in the cross product as a separate registration in the local endpoint map. If you supply no object UUIDs, the corresponding elements in the cross product contain a nil object UUID. For example, suppose that you have an interface (if1), three bindings (b1, b2, and b3), and four object UUIDs (o1, o2, o3, and o4). The resulting 12 elements in the cross product are as follows:

```
{if1,b1,o1} {if1,b1,o2} {if1,b1,o3} {if1,b1,o4}
{if1,b2,o1} {if1,b2,o2} {if1,b2,o3} {if1,b2,o4}
{if1,b3,o1} {if1,b3,o2} {if1,b3,o3} {if1,b3,o4}
```

An annotation string is part of each of these 12 elements, but is not shown for clarity.

This operation returns an empty string on success.
Privileges Required

No special privileges are needed to use the `endpoint create` command.

Examples

The following command adds an endpoint to the local host’s endpoint map. This example uses the `\` (backslash) twice to escape the brackets. Without the two backslashes, `dcecp` interprets the brackets as enclosing another command.

```
dcecp> endpoint create -interface 458ffcbe-98c1-11cd-bd93-0000c08adf56,1.0 \\
    > -binding ncacn_ip_tcp:130.105.1.227[1067]\[1067]\n
dcecp>
```

The following example uses the `dcecp` string syntax to create an endpoint in the local host's endpoint map.

```
dcecp> endpoint create -interface {458ffcbe-98c1-11cd-bd93-0000c08adf56 1.0} \\
    > -binding {ncacn_ip_tcp 130.105.1.227 [1072]} \\
    > -object {76030c42-98d5-11cd-88bc-0000c08adf56} \\
    > -annotation {Bulletin Board Server, Version 1.3a}
dcecp>
```

endpoint delete

Deletes the specified endpoints from the local endpoint map database. The syntax is as follows:

```
endpoint delete -interface interface_id -binding string_binding_list \\
    [-object object_uuid_list]
```

Options

- **-interface interface_id**
  
  This required option declares the interface identifier of a single RPC interface.

  See Data Structures for the format of the interface identifier.

- **-binding string_binding_list**
  
  This required option declares a list of one or more string bindings.

  See Data Structures for the format of a protocol sequence.

- **-object object_uuid_list**
  
  Declares the UUID of an object. Each delete operation accepts a list of up to 32 object UUIDs. The UUID is a hexadecimal string.

  See Data Structures for the format of the object UUID.

The delete operation deletes the specified endpoints from the endpoint map database. This command takes no arguments. It requires the -interface and -binding options, and also accepts the -object option. The values of all but the -interface option can be lists. If the mappings do not exist, an error is generated. This command creates a cross product from the -interface, -binding, and -object options and removes each element in the cross product from the local endpoint map. See the `endpoint create` command above for more details.
This operation returns an empty string on success.

Privileges Required

No special privileges are needed to use the **endpoint delete** command.

Examples

The following command removes an endpoint object from the local host's endpoint map. This example uses the \ (backslash) twice to escape the brackets. Without the the two backslash, **dcecp** interprets the brackets as enclosing another command.

```
dcecp> endpoint delete -interface 458ffcbe-98c1-11cd-bd93-0000c08adf56,1.0 \
> -binding ncacn_ip_tcp:130.105.1.227[1072]
dcecp>
```

The following example uses the **dcecp** string syntax to delete an endpoint from the local host's endpoint map.

```
dcecp> endpoint delete -interface {458ffcbe-98c1-11cd-bd93-0000c08adf56 1.0} \
> -binding {ncacn_ip_tcp 130.105.1.227 1072}
dcecp>
```

**endpoint help**

Returns help information about the **endpoint** object and its operations. The syntax is as follows:

```
endpoint help  [operation  | -verbose]
```

Options

- **-verbose**
  Displays information about the **endpoint** object.

Used without an argument or option, the **endpoint help** command returns brief information about each **endpoint** operation. The optional **operation** argument is the name of an operation about which you want detailed information. Alternatively, you can use the **-verbose** option for more detailed information about the **endpoint** object itself.

Privileges Required

No special privileges are needed to use the **endpoint help** command.

Examples

```
dcecp> endpoint help
create  Creates RPC endpoints for the specified interface.
delete  Deletes a set of RPC endpoints.
show   Returns the RPC endpoints for a specified interface.
help    Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp>
```

**endpoint operations**

Returns a list of the operations supported by the **endpoint** object. The syntax is as follows:
endpoint operations

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the endpoint operations command.

Examples

```
dcecp> endpoint operations
create delete show help operations
```

endpoint show

Returns a list of information about endpoints for the local host or a remote host. The syntax is as follows:

```
endpoint show [host_address] [-uuid | -interface interface_id [-version versions] [-object object_uuid_list]]
```

Options

- `-uuid` Specifies that the UUID of the endpoint map is to be returned. It cannot be used with any other option.

- `-interface interface_id`
  This option specifies the interface identifier of a single RPC interface for which you want to see the endpoint mapping information.

  See Data Structures for the format of the interface identifier.

- `-version versions`
  Specifies interface version numbers to be returned with the show operation.

  See Data Structures for the exact behavior and format of the version values.

- `-object object_uuid_list`
  Declares the UUID of an object. Each show operation accepts a list of up to 32 object UUIDs.

  See Data Structures for the format of the object UUID.

The show operation returns a list of information about endpoints in the endpoint map of a local or remote host. With no options, it returns all the local endpoint mappings. The -interface, -version, and -object options can be used so that only those endpoint mappings matching the supplied values are returned. The -object option accepts a list as a value; the others do not. The optional host_address argument is the address of the remote host whose endpoint map is to be shown. If no argument is supplied, the local host’s endpoint map is used.

See Data Structures for the format of a host address.

If the -uuid option is specified, then the UUID of the specified host’s endpoint map is to be returned, rather than any information about the endpoints themselves. Each endpoint map is given a UUID on creation. If you know the current UUID of an
endpoint(8dce)

endpoint map, you can delete any other stale UUIDs that might be in the RPC entry. If you specify the -uuid option, you must not specify any other options.

Privileges Required

No special privileges are needed to use the endpoint show command.

Examples

The following example uses dcecp string syntax to specify an interface for which to return local endpoint map information:

```
dcecp> endpoint show -interface {458ffcbe-98c1-11cd-bd93-0000c08adf56 1.0}
{{object 76030c42-98d5-11cd-88bc-0000c08adf56}
 {interface {458ffcbe-98c1-11cd-bd93-0000c08adf56 1.0}}
 {binding {ncacn_ip_tcp 130.105.1.227 1072}}
 {annotation {Bulletin Board Server, Version 1.3a}}}
dcecp>
```

The following command returns the endpoint objects in the local endpoint map that contain the specified interface identifier. This interface supports two object UUIDs on two protocol sequences:

```
dcecp> endpoint show -interface 257df1c9-c6d3-11ca-8554-08002b1c8f1f,1.0
{{object a57104f4-dfd0-11ca-b428-08002b1c8a62}
 {interface {257df1c9-c6d3-11ca-8554-08002b1c8f1f 1.0}}
 {binding {ncacn_ip_tcp 130.105.1.227 1040}}
 {annotation {cdsd [910]}}}
{{object a57104f4-dfd0-11ca-b428-08002b1c8a62}
 {interface {257df1c9-c6d3-11ca-8554-08002b1c8f1f 1.0}}
 {binding {ncadg_ip_udp 130.105.1.227 1163}}
 {annotation {cdsd [910]}}}
{{object b32648c6-928d-11cd-b4b5-0000c08adf56}
 {interface {257df1c9-c6d3-11ca-8554-08002b1c8f1f 1.0}}
 {binding {ncacn_ip_tcp 130.105.1.227 1042}}
 {annotation cds_clerkserver}}
{{object b32648c6-928d-11cd-b4b5-0000c08adf56}
 {interface {257df1c9-c6d3-11ca-8554-08002b1c8f1f 1.0}}
 {binding {ncadg_ip_udp 130.105.1.227 1168}}
 {annotation cds_clerkserver}}
dcecp>
```

The following command returns the UUID of the endpoint map on the host with the specified network address:

```
dcecp> endpoint show ncadg_ip_udp:130.105.1.227 -uuid
7273c754-e51c-11cd-bc0e-0000c08de054
dcecp>
```

Related Information

Commands: dcecp(8dce), rpcentry(8dce), rpcgroup(8dce), rpcprofile(8dce), server(8dce),
getcellname

Purpose

Gets the primary name of the cell

Synopsis

getcellname

Description

The getcellname command prints the primary name of the local cell to standard output. If the command fails, it prints an error message to standard error.

Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

dcelocal/dce_cf.db
The local DCE configuration database.

Related Information

Functions: dce_cf_get_cell_name(3dce).
Purpose

Gets a host’s IP address

Synopsis

getip host

Arguments

host  The host argument indicates the name of the machine whose IP address you want to obtain.

Description

The getip command prints the IP address of the machine indicated in the host argument. A machine might have more than one IP address associated with it; if so, getip prints one of the addresses. If the command fails, it returns a status of 1.

Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Related Information

Functions: gethostbyname(3).
group

Purpose

A dcecp object that manages a group in the DCE Security Service

Synopsis

```plaintext
group add group_name_list -member member_name_list

group catalog [cell_name] [-simplename]

group create group_name_list {-attribute extended_rgy_attr_list | -attribute value}

group delete group_name_list

group help [operation | -verbose]

group list group_name_list [-simplename]

group modify group_name_list {-add extended_rgy_attr_list | -remove extended_rgy_attr_list | [-types] | -change extended_rgy_attr_list | -attribute value}

group operations

group remove group_name_list -member member_name_list

group rename group_name -to new_group_name

group show group_name_list [-all | -xattrs]
```

cell_name

The name of a cell to contact when processing the catalog operation. The name must be a fully qualified cell name, such as ./ or /.../ cell_name.

group_name

The name of a registry group to act on. See group_name_list for the name format.

group_name_list

A list of one or more names of groups to act on. Supply the names as either of the following:

1. Fully qualified names in the form /.../ cell_name group_name or ./ group_name.
2. Cell-relative names in the form group_name. These names refer to a group in the cell identified in the _s(sec) convenience variable, or if the _s(sec) convenience variable is not set, in the local host’s default cell.

Do not mix fully qualified names and cell-relative names in a list. In addition, do not use the names of registry database objects that contain group information; in other words, do not use names that begin with ./sec/group/.

operation

The name of the group operation for which to display help information.
The **group** object represents registry groups. Unless otherwise noted, all of the operations of this object take the names of the groups to act on as the argument. They must be group names, not the names of the database objects that contain registry information about groups (that is, the names must not begin with `/./sec/group/`).

When this command executes, it attempts to bind to the registry server identified in the `_s(sec)` variable. If that server cannot process the request or if the `_s(sec)` variable is not set, the command binds to either an available slave server or the master registry server, depending on the operation. Upon completion the command sets the `_b(sec)` convenience variable to the name of the registry server to which it bound.

### Attributes

**alias {yes | no}**

Used with the **create** and **modify** operations, the value of this attribute is either **yes** or **no**. Although each group can have only one primary name, it can have one or more alias names. All aliases refer to the same group, and therefore, carry the same Universal Unique Identifier (UUID) and group identifier (GID). While aliases refer to the same group, they are separate entries in the registry database. Therefore, the name supplied to the **group** command can refer to the group's primary name or alias name. The value of this attribute determines whether the name is a primary name (**alias no**) or an alias name (**alias yes**). The default is **no**.

**gid integer**

Used with the **create** operation to specify the Group Identifier. If this attribute is not present, then an identifier is assigned to the group automatically.

**uuid hexadecimal number**

Used with the **create** operation to adopt an orphaned UUID. Normally the UUID for a new group is generated by the registry. In cases where data exists tagged with the UUID of a group that has been deleted from the registry, this attribute can be used with the **create** operation to specify the old UUID for a new group. The UUID specified must be an orphan, that is, a UUID for which no name exists in the registry. An error occurs if you specify a name that is already defined in the registry. If this attribute is not present, a UUID is assigned to the group automatically.

**fullname string**

Used with the **create** and **modify** operations to specify the full name of the group to be added to the registry. The value is a string with spaces enclosed in quotation marks or braces. The **fullname** attribute defaults to a null string (that is, blank).

**inprojlist {yes | no}**

Used with the **create** and **modify** operations to include the group in the principal's project list. The value for this option is either **yes** or **no**. If it is **no**, then members of this group do not acquire the access rights of this group. The default is **yes**.

See the **IBM DCE Version 3.1 for AIX and Solaris: Administration Guide** for more information about group attributes.
Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Operations

**group add**

Adds members to a security group. The syntax is as follows:

```
  group add  group_name_list -member  member_name_list
```

**Options**

- `member member_name_list`
  
  A list of one or more names of principals to be added to each group in the argument. This option is required.

The **add** operation adds members to groups identified by `group_name_list`. The required `member_name_list` is a list of principal names to be added. The `member_name_list` can contain both local and fully qualified names. Use fully qualified names to add principals from foreign cells as members. If you are adding principals from a foreign cell, the Security Server (`secd`) must be running in the foreign cell.

If the principals named in `group_name_list` do not exist, the command returns an error. This operation returns an empty string on success.

**Privileges Required**

You must have `r` (read) and `M` (Member_list) permissions on the target group and `r` (read) and `g` (groups) permissions on the principal being added.

**Examples**

```
dcecp> principal create chopin

dcecp> group add users -member chopin

```

**group catalog**

Returns a list of the names of all groups in the registry. The syntax is as follows:

```
  group catalog  [cell_name] [-simplename]
```

**Options**

- `simplename`
  
  Returns a list of group names in the registry without prepending the cell name.

The **catalog** operation returns a list of the names of all groups in the local registry database. Use the `cell_name` argument to return a list of groups in another cell’s
registry. By default, fully qualified names are returned in the form cell_name/group_name. Use the -simplename option to return the names without the cell name in the form group_name.

Privileges Required
You must have r (read) permission to the /./sec/group directory.

Examples

dcecp> group cat
/.../my_cell.goodcompany.com/nogroup
/.../my_cell.goodcompany.com/system
/.../my_cell.goodcompany.com/daemon
/.../my_cell.goodcompany.com/uucp
/.../my_cell.goodcompany.com/bin
/.../my_cell.goodcompany.com/kmem
/.../my_cell.goodcompany.com/mail
/.../my_cell.goodcompany.com/tty
/.../my_cell.goodcompany.com/none
/.../my_cell.goodcompany.com/tcb
/.../my_cell.goodcompany.com/acct-admin
/.../my_cell.goodcompany.com/subsys/dce/sec-admin
/.../my_cell.goodcompany.com/subsys/dce/cds-admin
/.../my_cell.goodcompany.com/subsys/dce/dts-admin
/.../my_cell.goodcompany.com/subsys/dce/cds-server
/.../my_cell.goodcompany.com/subsys/dce/dts-servers
/.../my_cell.goodcompany.com/users
dcecp>

dcecp> group cat -simplename
nogroup
system
daemon
uucp
bin
kmem
mail
tty
none
tcb
acct-admin
subsys/dce/sec-admin
subsys/dce/cds-admin
subsys/dce/dts-admin
subsys/dce/cds-server
subsys/dce/dts-servers
subsys/dce/audit-admin
subsys/dce/dced-admin
dcecp>

group create

Creates a new group in the registry database. The syntax is as follows:

    group create  group_name_list {-attribute extended_rgy_attr_list | -attribute value}

Options

- attribute value

As an alternative to using the -attribute option with an attribute list, you can change individual attribute options by prepending a - (hyphen) to any
attributes listed in Attributes in this reference page. You cannot use this option to specify ERAs; it is only for the standard attributes described in Attributes.

-attribute extended_rgy_attr_list

Allows you to specify attributes, including ERAs, by using an attribute list rather than using the -attribute value option. The format of an attribute list is as follows:

```
{extended_rgy_attr_list value}...{extended_rgy_attr_list value}
```

See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information on ERAs.

The create operation creates a new group in the registry database. The argument is a list of names of groups to be created. Options are used to specify the attributes of the newly created group. All options are applied to all groups in the argument. This operation returns an empty string on success.

**Privileges Required**

You must have i (insert) permission to the directory in which the group is to be created.

**Examples**

```
dcecp> group create users4 -attribute {fullname "temporary users"}
dcecp>
```

**group delete**

Deletes groups from the registry. The syntax is as follows:

```
group delete group_name_list
```

The delete operation deletes groups from the registry. When a group is deleted, any accounts associated with the group are deleted as well. The argument is a list of names of groups to be deleted. If a named group does not exist, an error is generated. This operation returns an empty string on success.

This operation also deletes any accounts associated with groups that are deleted. To preserve accounts, add the desired principals to a different group by using the group add -member command. Modify the principals’ accounts to point to the new group by using the account modify command. Then you can delete the group by using the group delete command.

**Privileges Required**

You must have d (delete) permission to the directory in which the target group exists. You must have r (read) and D (Delete_object) permission on the group to be deleted.

**Examples**

```
dcecp> group delete users4
```
group help

Returns help information about the group object and its operations. The syntax is as follows:

\texttt{group help \ [operation | -verbose]}

Options

-**verbose**
  Displays information about the group object.

Used without an argument or option, the group help command returns brief information about each group operation. The optional \texttt{operation} argument is the name of an operation about which you want detailed information. Alternatively, you can use the \texttt{-verbose} option for more detailed information about the group object itself.

Privileges Required

No special privileges are needed to use the group help command.

Examples

\texttt{dcecp> group help}

add \hspace{1em} Adds a member to the named group.

\texttt{dcecp> group help catalog}

catalog \hspace{1em} Returns a list of all the names of groups in the registry.

create \hspace{1em} Creates a group.

delete \hspace{1em} Deletes a group.

\texttt{dcecp> group help list}

list \hspace{1em} Returns all of the members of a group.

\texttt{dcecp> group help modify}

modify \hspace{1em} Changes the information about a group.

\texttt{dcecp> group help remove}

remove \hspace{1em} Removes a specified member from the named group.

\texttt{dcecp> group help rename}

rename \hspace{1em} Renames the specified group.

\texttt{dcecp> group help show}

show \hspace{1em} Returns the attributes of a group.

\texttt{dcecp> group help operations}

operations \hspace{1em} Prints a summary of command-line options.

\texttt{group list}

Returns a list of the names of all members of a group. The syntax is as follows:

\texttt{group list \ group_name_list \ [-simplename]}

Options

\texttt{-simplename}
  Returns the list of group names in the registry without prepending the cell name.

The list operation returns a list of the names of all members of a group. The argument is a list of names of groups to be operated on. If more than one group is listed, the names are concatenated on output. By default, fully qualified names are returned in the form \texttt{cellname/memname}. Use the \texttt{-simplename} option to return them without prepending the cell name to the member name. The members of each group are listed in lexical order.

Privileges Required
You must have r (read) permission to the /.:sec/group directory.

Examples

dcecp> group list none
/.../my_cell.goodcompany.com/dce-rgy
/.../my_cell.goodcompany.com/krbtgt/my_cell.goodcompany.com
/.../my_cell.goodcompany.com/cell_admin
/.../my_cell.goodcompany.com/hosts/pmin17/self
dcecp> 

**group modify**

Changes attributes of groups. The syntax is as follows:

```
group modify  group_name_list
{-add extended_rgy_attr_list | -remove extended_rgy_attr_list [-types] |
-change extended_rgy_attr_list | -attribute value}
```

**Options**

- `-attribute value`  
  As an alternative to using options with an attribute list, you can change individual attribute options by prepending a - (hyphen) to any attributes listed in the Attributes section of this reference page. You cannot use this option to specify ERAs; it is only for standard group attributes described in Attributes.

- `-add extended_rgy_attr_list`  
  Allows you to modify attributes, including ERAs, by using an attribute list rather than using individual attribute options. The format of an attribute list is as follows:

  ```
  {{extended_rgy_attr_list value}...{extended_rgy_attr_list value}}
  ```

- `-change extended_rgy_attr_list`  
  Allows you to modify attributes, including ERAs, by using an attribute list rather than using individual attribute options. See the `-add` option for the attribute list format.

- `-remove extended_rgy_attr_list`  
  Allows you to modify attributes, including ERAs, by using an attribute list rather than using individual attribute options such as `-alias`, `-inprojlist`, and so on. See the `-add` option for the attribute list format.

  Without the `-types` option, `-remove` deletes individual attribute instances attached to the group. In this case, `extended_rgy_attr_list` is a list of attribute-value pairs. With the `-types` option, `-remove` deletes attribute types (and all instances of that type) attached to the group. In this case, `extended_rgy_attr_list` is a list of attribute types.

- `-types`  
  Used with the `-remove` option to remove attribute types (and all instances of that type) attached to the group.

See the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide* for more information about ERAs.

The modify operation changes attributes of groups. The argument is a list of names of groups to be operated on. All modifications are applied to all groups.
named in the argument. Groups are modified in the order they are listed, and all modifications to an individual group are atomic. Modifications to multiple groups are not atomic. A failure for any one group in a list generates an and aborts the rest of the operation. This operation returns an empty string on success.

The **-change** option can be used to modify the value of any standard attribute except for **gid** and **uuid**.

**Privileges Required**

You must have **r** (read) permission to the group to be modified and **f** (full_name) permission to modify the group's full name and/or **m** (mgmt_info) permission to modify the group's management information.

**Examples**

dcecp> group modify users3 -change {fullname "General Nursing Staff"}
dcecp>

dcecp> group modify users3 -add {test_era 101}

group operations

Returns a list of the operations supported by the **group** object. The syntax is as follows:

**group operations**

The list of available operations is in alphabetical order except for **help** and **operations**, which are listed last.

**Privileges Required**

No special privileges are needed to use the **group operations** command.

**Examples**

dcecp> group operations
add catalog create delete list modify remove rename show
> help operations

dcecp>
group remove

Removes a member from a group. The syntax is as follows:

```
group remove  group_name_list  -member  member_name_list
```

Options

- **-member member_name_list**
  A list of one or more names of principals to be removed from each group in the argument. This option is required.

The `remove` operation removes members from the groups identified by `group_name_list`. The required `member_name_list` is a list of principals to remove from the groups named in `group_name_list`. The `member_name_list` can contain both local and fully qualified names. Use fully qualified names to remove principals in foreign cells from the group.

When a member is removed from a group, any accounts associated with that principal and group are deleted. Remember that accounts are associated with a principal, a group, and an organization; therefore, any accounts whose principal name and group name match those given to this command are removed, but accounts for which only one name matches are untouched. This operation returns an empty string on success.

Privileges Required

You must have **r** (read) and **M** (Member_list) permissions on the target groups and **r** (read) permission on the member to be removed.

Examples

```
dcecp>  group remove users  -member  chopin

dcecp>
```

**group rename**

This operation changes the name of a specified group. The syntax is as follows:

```
group rename  group_name  -to  new_group_name
```

Options

- **-to new_group_name**
  Specifies the new name of the group. This option is required.

  See **Arguments** for a description of group names.

The `rename` operation changes the name of a specified group. The argument is a single name of a group to be renamed. The operation takes a required **-to** option with the value of the new name. The value can not be a list. This operation returns an empty string on success.

Privileges Required

You must have **r** (read) and **n** (name) permissions to the specified groups.

Examples
group(8dce)

dcecp> group rename users4 -to users_temporary
dcecp>

group show

Returns registry information for the specified groups. The syntax is as follows:

```bash
group show  group_name_list [-all | -xattrs]
```

Options

- **-xattrs**
  Returns ERAs instead of the default attributes.
- **-all**
  Returns ERAs in addition to the default attributes.

The `show` operation returns an attribute list for the specified groups. The argument is a list of names of groups to be operated on. If more than one group is given, the attributes are concatenated. Use the `-xattrs` option to return ERAs instead of the standard attributes. Use `-all` to return both types of attributes.

**Privileges Required**

You must have `r` (read) permission to the specified groups.

**Examples**

```bash
dcecp> group show users_temporary
{alias no}
{gid 5211}
{uuid 0000145b-9362-21cd-a601-0000c08adf56}
{inprojlist no}
{fullname {temporary users}}
dcecp>
```

**Related Information**

Commands: `dcecp(8dce)`, `account(8dce)`, `organization(8dce)`, `principal(8dce)`, `registry(8dce)`, `xattrshcema(8dce)`. 
host

Purpose

A dcecp task object that manages host information in a DCE cell

Synopsis

host catalog [cell_name] [-simulename]

host configure dce_hostname -cell cell_name
-secmaster master_security_server_name -cds cds_server_name -password
password [-admin admin_principal] {client | server}

host help [operation | -verbose ]

host operations

host ping [dce_hostname ]

host show [dce_hostname ]

host start [dce_hostname ]

host stop [dce_hostname ] [-force]

host unconfigure dce_hostname [-force]

Note: The host configure, host unconfigure, host show, host start, and host
stop commands are not supported at this time.

Arguments

cell_name
The name of a single cell to operate on. The name must be a fully qualified
cell name such as either of the following:

/day/their_cell.goodco.com

/d.: 

dce_hostname
The name of a single host to operate on. Some host commands accept
both fully qualified names (as in/day/hostname/hosts/dce_hostname ) and
cell relative names (as in hosts/dce_hostname ), while others will accept
only fully qualified names. See individual command descriptions in
Operations for details.

operation
The name of the host operation for which to display help information.

Description

The host task object represents DCE processes running on a machine in (or to be
added to) a DCE cell. The host task object allows administrators to configure and
start DCE on machines easily.
The host task object can configure and start the core DCE services on a client machine. The services include the DCE daemon (dced), the Cell Directory Service (CDS) client (cdsadv), the Distributed Time Service (DTS) daemon (dtsd), and the audit daemon (auditd). The argument to this command is the DCE name of a host to operate on. If an argument is omitted, the command operates on the local host, if possible. The behavior of commands operating locally might differ from the behavior of commands operating remotely, with more operations performed on the local host than might be possible remotely. See Operations for details.

Currently only a client can be configured using the host command.

Note: All operations of the host task object are not fully supported in this release.

Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Operations

host catalog

Returns a list of names of hosts in the cell. The syntax is as follows:

host catalog  [cell_name] [-simplename]

The catalog operation returns a list of names of hosts in the cell. By default, the names are fully qualified. Use the -simplename option to return cell-relative names. The optional cell_name argument specifies a cell to operate in.

Privileges Required

You must have r (read) permission to the .:/hosts directory in CDS.

Examples

The following example lists the full names of all the DCE hosts that have entries in the CDS .:/hosts directory in the local cell:

dcecp> host catalog
/.../my_cell.goodco.com/hosts/alpha
/.../my_cell.goodco.com/hosts/beta
/.../my_cell.goodco.com/hosts/gamma

dcecp>

The following example lists the simple names of all the DCE hosts that have entries in the CDS .:/hosts directory in the local cell:

dcecp> host catalog -simplename
hosts/alpha
hosts/beta
hosts/gamma

dcecp>
host configure

Configures a single machine named by the argument into an existing DCE cell.

Note: The host configure command is not supported at this time.

The syntax is as follows:

```
host configure dce_hostname
-cell cell_name -secmaster
master_security_server_name -cds cds_server_name -password password
[ -admin admin_principal ] { -client | -server }
```

Options

- **-cell cell_name**
  Specifies the name of the cell in which the host is to be configured. The format is /.../cellname.

- **-client**
  Configures the host as a DCE client machine. The machine will be configured to run dced (including the secval service), a DTS clerk (dtsd), cdsadv, and auditd.

- **-server**
  Configures the host as a DCE server machine. This option is currently not supported.

- **-secmaster master_security_server_name**
  Specifies the hostname of the security master server in the form hostname.

- **-cds cds_server_name**
  Specifies the hostname of any CDS server in the form hostname.

- **-password password**
  Specifies the password of the cell administrator.

- **-admin admin_principal**
  Optionally specifies the principal name of the cell administrator principal. It defaults to cell_admin.

The configure operation configures a single machine named by the dce_hostname argument into a DCE cell. The cell must already exist and must have a security and naming service operating. The DCE software must be installed on the machine. The dce_hostname argument is the name of the local host machine without the cell name prepended, as in the following:

```
hosts/ dce_hostname
```

This operation returns an empty string on success.

Privileges Required

You must have root authority.

Examples

The following example configures host hydra in the cell /.../my_cell.goodco.com:
host configure hosts/hydra -client \\
> -cell my_cell.goodco.com -password fstzk1 -secmaster scylla \\
> -cds charybdis

dcecp>

host help

Returns help information about the host task object and its operations. The syntax is as follows:

```
host help [operation | -verbose]
```

Options

- **-verbose**
  
  Displays information about the host task object.

Used without an argument or option, the host help command returns brief information about each host operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the -verbose option for more detailed information about the host task object itself.

Privileges Required

No special privileges are needed to use the host help command.

Examples

```
dcecp> host help

catalog         Returns a list of configured hosts in the cell.
configure       Configures a host into the cell as a client or server.
ing             Determines if DCE is responding on the specified host.
show            Returns all DCE processes configured on the specified host.
start           Starts DCE on the specified host.
stop            Stops DCE on the specified host.
unconfigure     Removes the host from the name and security databases.
help            Prints a summary of command-line options.
operations      Returns a list of the valid operations for this command.
dcecp>
```

host operations

Returns a list of the operations supported by the host task object. The syntax is as follows:

```
host operations
```

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the host operations command.

Examples
host ping

Tests whether DCE processes are accessible from the network. The syntax is as follows:

```
host ping dce_hostname
```

The ping operation tests whether DCE processes are accessible from the network. It contacts the endpoint mapper (either rpcd or dced, whichever listens on port 135) on the specified host. The dce_hostname argument is the fully qualified name of the host to ping as in the following:

```
./:/hosts/dce_hostname
```

The operation returns 1 if the host responds, 0 if it does not.

**Privileges Required**

No special privileges are required for the host ping command.

**Examples**

The following example pings host hydra:

```
dcecp> host ping ./:/hosts/hydra
1
```

host show

The show operation is not currently implemented.

Returns a list describing all processes that are configured to run on the specified host. The syntax is as follows:

```
host show [dce_hostname]
```

The show operation returns a list describing all processes that are configured to run on the specified host. The optional dce_hostname argument is the fully qualified or cell-relative name of a DCE host, such as hosts/name or ./:/hosts/name. If not given, the local host is assumed. The returned list contains the following:

1. The server name as output by the server catalog -simplename command.
2. One of the tokens running or notrunning.
3. An optional server-specific comment such as master or replica for a security server and clerk or server for a DTS server.

If the DCE daemons on the specified host were not started by the dcecp server command, the output of this command will not be as expected.
Privileges Required

You must have r (read) permission to the config/srvrconf container object on the specified host.

Examples

dcecp> host show hosts/hydra
{dced running}
{cdsd running}
{cdsadv running}
{secd running master}
{auditd notrunning}
{dtsd running clerk}
dcecp>

host start

Starts all DCE processes on the specified host.

Note: This command is not supported at this time.
The syntax is as follows:

**host start** [dce_hostname ]

The start operation starts all DCE processes on the specified host. This command depends on dced being running on the specified host; that is, it can not be used to start DCE on systems that use versions prior to Version 1.1. The processes that are started are all those listed in the server configuration data stored in the dced on the specified host with the boot or explicit values of the starton attribute. You can add servers to the server configuration data by using the server create command. The host configure command adds certain servers to the configuration data automatically.

The dce_hostname argument is the fully qualified or cell-relative name of the host to operate on, as in the following:

/./hosts/dce_hostname

hosts/dce_hostname

Without the dce_hostname argument, dced is started on the local host first, which requires the appropriate local permissions (usually root). If a host name is specified, dced must be running on that host. Before starting any host, make sure that a security server and a CDS server are both running somewhere in the cell. This operation returns an empty string on success.

Privileges Required

You must have x (execute) permission to the config/srvrconf container object on the specified host.

Examples

The following example starts all DCE processes on host hydra:
host stop

Stops all DCE processes on the specified host.

Note: This command is not supported at this time.
The syntax is as follows:

```
host stop [dce_hostname ] [-force]
```

Options

- **-force** Optionally, specifies that any servers that fail to stop normally should be stopped using a `server stop -method hard` command.

The `stop` operation stops running DCE processes on the specified host. This command depends on `dced` being on the specified host, that is, it can not be used to stop DCE on systems that use versions prior to Version 1.1. The `dce_hostname` argument is the fully qualified or cell-relative name of the host to operate on, as in

```
./:/hosts/dce_hostname
```

Processes are stopped as follows:

1. All servers listed in the server execution data are stopped. Servers implementing DCE core services are stopped last, in the appropriate order. If servers were not started as `srvrexe` objects, they will not be stopped.
2. If any servers fail to stop, and the `-force` option was specified, those servers are stopped by the command `server stop -method hard`.

This operation returns an empty string on success.

Privileges Required

You must have `s (stop)` permission on the `config/srvrexe` object for each server to be stopped.

Examples

The following example stops host `hydra`:

```
dcecp> host stop hosts/hydra
dcecp>
```

host unconfigure

Unconfigures a specified host from a cell.

Note: The `host unconfigure` command is not supported at this time.

The syntax is as follows:
host unconfigure  
dce_hostname  
[-force]

Options
-force  Optionally specifies that any errors that occur during an unconfigure operation are to be ignored and the unconfigure operation should continue.

The unconfigure operation unconfigures a specified host from a cell. To unconfigure a cell, the operation deletes the following:
1. All objects, directories and links from /.:/hosts/ dce_hostname including the directory itself
2. All principal names beginning with hosts/ dce_hostname
3. , but not accounts with the same names

The unconfigure operation takes the fully qualified name of a host to unconfigure as an argument, as in the following:

/hosts/dce_hostname

This operation returns an empty string on success.

Note: Before running the host unconfigure <host/dce_hostname> command, you must manually remove all replicas of <host/dce_hostname> in other clearinghouses.

Privileges Required

You must have the appropriate permission to delete CDS objects and directories. You must also have the appropriate permission to delete principals from the registry. Refer to the appropriate reference page on each object for more details.

Examples

The following example unconfigures host hydra from the cell:

dcecp> host unconfigure hosts/hydra
dcecp>

Related Information

Commands: dcecp(8dce), account(8dce), aud(8dce), directory(8dce), dts(8dce), registry(8dce), server(8dce).
hostdata

Purpose

A dcecp object that manages a DCE host’s cell affiliation information

Synopsis

hostdata catalog [host_name_list] [-simplename] [-local] [-unauth]

hostdata create hostdata_name_list {-attribute attribute_list | -attributevalue } [-binary] [-local][-entry ]

hostdata delete hostdata_name_list [-entry] [-local]

hostdata help [operation | -verbose ]

hostdata modify hostdata_name_list {-change attribute_list | -attributevalue } [-binary] [-local]

hostdata operations

hostdata show hostdata_name_list [-ifname residual_object_name | [-entry] | [-binary ]][-local] [-unauth]

Arguments

host_name_list
A list of one or more DCE host names specifying hosts for which to catalog servers. Host names can be in any of the following forms:

/./:hosts/hostname   /.../cell_name/hosts/hostname   hosts/hostname

For the catalog operation, the name can also be a single string binding representing the host with which to communicate. See hostdata_name_list for more information.

hostdata_name_list
A list of one or more names of host data items. Usually they are of the following form:

/./:hosts/hostname/config/hostdata/name

For the show operation, the name can also be a single string binding representing the host with which to communicate. For example:

{ncacn_ip_tcp 130.105.1.227}

A string binding is useful when the name service is not operating and cannot translate the other forms of host data item names. If you supply a single string binding, you must use the -ifname option to specify the host’s residual name. The host_data_name_list argument can be either a simple name or a fully qualified path.
**hostdata(8dce)**

**Note:** If the cell name is fully qualified and if the `+hostdata_name` is used, the `dce_hostname` field must be fully qualified, for example: `dce_hostname.austin.ibm.com`.

**operation**
The name of the `hostdata` operation for which to display help information.

**Description**
The `hostdata` object represents a `hostdata` entry stored by `dced` on a host that represents some data, usually a file. The data in the `hostdata` object is represented by the `hostdata/data` attribute of the `hostdata` entry. Remote manipulation of data in the `hostdata` object is accomplished by the `hostdata` command. The names of these `hostdata` objects are in the DCE namespace and are controlled by `dced`. Usually they are of the following form:

```
./hosts/dce_hostname/config/hostdata/name
```

However, a shorthand notation referring to the local machine consisting of just `name` can be used in some circumstances.

**Attributes**

- **uuid** *hexadecimal number*
  
  An internal identifier for the `hostdata` entry. Its value is a Universal Unique Identifier (UUID). If not specified on creation, one is generated by `dcecp`. This attribute cannot be modified after creation.

- **annotation** *string*
  
  A human-readable comment field limited to Portable Character Set (PCS) data. It cannot be modified after creation. This attribute defaults to a null string (that is, blank).

- **storage** *string*
  
  A PCS string that identifies the name of the data repository. In the current release of `dced`, it is a filename. It is required and cannot be modified after creation.

- **hostdata/data** *string*
  
  An attribute that represents the actual data. Its syntax is a list of strings. The data can be viewed and modified in two different modes, either as a string, or as binary data. By default the string mode is used, but some of the operations below accept a binary option to allow this attribute to be displayed or modified in binary form. When viewed as a string, each string in the list represents one line in the `hostdata` file.

See the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide* for more information about `hostdata` attributes.

**Errors**

A representative list of errors that might be returned is not shown here. Refer to the *IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide* for complete descriptions of all error messages.
Operations

hostdata catalog

Returns a list of the names of all hostdata objects on the specified host. The syntax is as follows:

```
hostdata catalog  [host_name_list]
[-simplename] [-local] [-unauth]
```

Options

- **-simplename**
  Returns a list of hostdata entries without prepending the cell name and name of the hostdata container.

- **-local**
  Specifies that the command should operate on the local dced hostdata object while the dced object is in a partial-service state.

- **-unauth**
  Specifies that the command should operate as if an unauthenticated user is running it. This option is useful for intercell access when the cell registries are not connected.

The catalog operation returns a list of the names of all hostdata objects on the specified host, in arbitrary order. The optional host_name_list argument specifies objects on a foreign host. By default, fully qualified names are returned. Use the -simplename option to return objects names without the prepended cell name and hostdata container names.

Privileges Required

You must have r (read) permission to the hostdata container on the host (/./hosts/dce_hostname/config/hostdata/hostdata_container).

Examples

```
dcecp> hostdata catalog
/./gumby1/hosts/fire/config/hostdata/dce_cf.db
/./gumby1/hosts/fire/config/hostdata/cell_name
/./gumby1/hosts/fire/config/hostdata/pe_site
/./gumby1/hosts/fire/config/hostdata/cds_attributes
/./gumby1/hosts/fire/config/hostdata/cds_globalnames
/./gumby1/hosts/fire/config/hostdata/host_name
/./gumby1/hosts/fire/config/hostdata/cell_aliases
/./gumby1/hosts/fire/config/hostdata/post_processors
/./gumby1/hosts/fire/config/hostdata/svc_routing
/./gumby1/hosts/fire/config/hostdata/krb.conf
/./gumby1/hosts/fire/config/hostdata/dfs-cache-info
/./gumby1/hosts/fire/config/hostdata/cds.conf
/./gumby1/hosts/fire/config/hostdata/passwd_override
/./gumby1/hosts/fire/config/hostdata/group_override

dcecp>
```

hostdata create

Creates a hostdata configuration object. The syntax is as follows:

```
hostdata create
hostdata_name_list[-attribute]
```
hostdata(8dce)

attribute_list | -attribute value} [-binary] [-local] [-entry ]

Options
-attribute attribute_list
  Allows you to specify attributes by using an attribute list rather than using the -attribute value option. The format of an attribute list is as follows:

  {{{attribute value}...{attribute value}}}

-attribute value
  As an alternative to using the -attribute option with an attribute list, you can specify individual attribute options by prepending a hyphen (-) to any attributes listed in Attributes.

-binary
  Specifies that the value of the data attribute is in binary form.

-local
  Specifies that the command should operate on the local dced hostdata object while the dced object is in a partial-service state.

-entry
  Creates only dced configuration information (UUID), not the actual hostdata commands object. If the -entry option is specified it must also be used when issuing the hostdata delete command, otherwise an error will be returned.

The hostdata_name_list argument is a list of names of hostdata entries to be created. The -attributes option specifies configuration information for dced. The contents of the hostdata file can be specified via the data attribute. The value of the option is applied to all elements of the argument list. This operation returns an empty string on success.

Privileges Required

You must have w (write) permission to the hostdata container on the host.

Examples

dcecp> hostdata create file1 -storage /tmp/file1 -data {{first line}}
dcecp>

dcecp> hostdata show file1
{uuid 8484188a-eb85-11cd-91b1-080009251352}
{annotation {}}
{storage /tmp/file1}
{hostdata/data {first line}}
dcecp>

dcecp> cat /tmp/file1
first line
dcecp>

hostdata delete

Deletes a hostdata entry and its data. The syntax is as follows:

hostdata delete  hostdata_name_list [-entry] [-local]

Options
Only the configuration information that dced keeps is deleted, not the actual hostdata.

Specifies that the command should operate on the local dced hostdata object while the dced object is in a partial-service state.

The hostdata_name_list argument is a list of names of hostdata entries to be deleted in the order specified. If the -entry option is present, only the configuration information that dced keeps is deleted, not the actual hostdata. This operation returns an empty string on success.

Privileges Required

You must have d (delete) permission to the hostdata container on the host.

Examples

dcecp> hostdata delete file1

dcecp>

hostdata help

Returns help information about the hostdata object and its operations. The syntax is as follows:

hostdata help  [operation | -verbose]

Options

-verbose

Displays information about the hostdata object.

Used without an argument or option, the hostdata help command returns brief information about each hostdata operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the -verbose option for more detailed information about the hostdata object itself.

Privileges Required

No special privileges are needed to use the hostdata help command.

Examples

dcecp> hostdata help
catalog Returns the list of hostdata object names.
create Creates a new hostdata configuration object.
delete Deletes a hostdata object and its associated data.
modify Modifies the data of a hostdata object.
show Returns the attributes of a hostdata object.
help Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp>

hostdata modify

This operation is used to change attributes of a hostdata entry, including the hostdata itself. The syntax is as follows:
hostdata modify  

Usage:  

hostdata modify  

options  

-attribute value  

As an alternative to using options with an attribute list, you can change individual attribute options by prepending a hyphen (-) to any attributes listed in the Attributes section of this reference page. In the current version of DCE, only the data attribute can be modified.  

-change attribute_list  

Allows you to modify attributes by using an attribute list rather than using individual attribute options. The format of an attribute list is as follows:  

{{attribute value}...{attribute value}}  

In the current version of DCE, only the data attribute can be modified.  

-binary  

Specifies that the value of the data attribute is in binary form.  

-local  

Specifies that the command should operate on the local dced hostdata object while the dced object is in a partial-service state.  

The argument is a list of names of hostdata entries to be modified. If more than one is specified, all modifications specified are made to each hostdata entry listed. In the current DCE Version, only the data attribute can be modified and only by completely replacing it. This operation returns an empty string on success.  

Privileges Required  

You must have w (write) permission to the hostdata container on the host.  

Examples  

dcecp> hostdata mod file1 -data {new first line}  
dcecp>  
dcecp> hostdata show file1  
{uuid cda3a184-eb85-11cd-91b1-0800009251352}  
{annotation {}}  
{storage /tmp/file1}  
{hostdata/data {new first line}}  
dcecp>  
dcecp> cat /tmp/file1  
nnew first line  
dcecp>  

hostdata operations  

Returns a list of the operations supported by the hostdata object. The syntax is as follows:  

hostdata operations  

The list of available operations is in alphabetical order except for help and operations, which are listed last.
Privileges Required

No special privileges are needed to use the hostdata operations command.

Examples

dcecp> hostdata operations
catalog create delete modify show help operations
dcecp>

hostdata show

Returns an attribute list of the hostdata entries specified in the argument. The syntax is as follows:

hostdata show  hostdata_name_list
[-ifname residual_object_name | [-entry] [-binary]] [-local] [-unauth]

Options

- ifname
  Specifies the dced object for which to return the value.

- entry
  Only the configuration information that dced keeps is returned, not the actual hostdata.

- binary
  Specifies to return the value of the data attribute in binary form.

- local
  Specifies that the command should operate on the local dced hostdata object while the dced object is in a partial-service state.

- unauth
  Specifies that the command should operate as if an unauthenticated user is running it. This option is useful for intercell access when the cell registries are not connected.

The hostdata_name_list argument is a list of names of hostdata entries. If called with the -entry option, the data attribute is not returned. The -binary option can be specified to indicate that the value of the data attribute should be returned in binary form. If the argument is a list of entries, the output is concatenated into a single list in the order specified. The -ifname option is used to identify the specific hostdata entry to show, but only when the argument is a string binding representing a host, not the fully qualified hostdata name.

Privileges Required

You must have r (read) permission to the hostdata container on the host (/hosts/host_name/config/hostdata/hostdata_container).

Examples

dcecp> hostdata show /hosts/mars/config/hostdata/cell_name
{uuid 00174f6c-6eca-1d6a-bf90-0000c09ce054}
{annotation {Name of cell}}
{storage cell_name}
{hostdata/data /.../my_cell}
dcecp>

dcecp> hostdata show ncacn_ip_tcp:15.122.24.148 -ifname cell_name
Related Information

Commands: `dcecp(8dce)`, `dced(8dce)`, `hostvar(8dce)`. 
keytab

Purpose

A dcecp object that manages server passwords on DCE hosts

Synopsis

```
keytab add keytab_name_list -member principal_name_list [-key plain_key | -version key_version | -registry | -random | -registry | -version key_version] [-ktname residual_keytab_name] [-noprivacy] [-local]
keytab catalog [dce_hostname] [-simplename] [-noprivacy] [-local]
keytab create keytab_name_list {attribute attribute_list | attribute value} [-ktname residual_keytab_name] [-entry] [-noprivacy] [-local]
keytab delete keytab_name_list [-entry] [-noprivacy] [-local]
keytab help [operation | -verbose]
keytab list keytab_name_list [-noprivacy] [-local]
keytab operations
keytab remove keytab_name_list -member principal_name_list [-version key_version] [-type key_type] [-noprivacy] [-local]
keytab show keytab_name_list [-entry | -members | -keys] [-ktname residual_keytab_name] [-noprivacy] [-local]
```

Arguments

```
dce_hostname
A list of one or more DCE host names specifying hosts for which to catalog key tables. Host names can be in any of the following forms:

/./hosts/dce_hostname
/.../cell_name/hosts/dce_hostname
hosts/dce_hostname

The name can also be a single string binding representing the host with which to communicate. For example:

{nacn_ip_tcp 130.105.1.227}

A string binding is useful when the name service is not operating and cannot translate the other forms of host names. If you supply a single string binding, you must use the -ktname option to specify the object’s residual name.

keytab_name_list
A list of one or more names of key tables to operate on. Key table names are similar to other dced objects with the following form:
```

Chapter 1. DCE Commands 223
keytab(8dce)

/path/to/keytab/

For the add, create, and show operations, the name can also be a single string binding representing the key table to operate on. See hostdata_name_list for more information on string bindings.

operation
The name of the keytab operation for which to display help information.

Description

The keytab object represents key tables (usually files) that store server keys (and key version numbers) on hosts. These key tables are manipulated remotely by using dced. The keys are considered members of the key table container. The keytab names are in the form

/path/to/keytab/

A key table has a set of keys. Each key contains a principal name, type, version, and value. The value can be created and changed, but is never shown on output. Removal of a key is based on the name, type, and version number. The dcecp syntax of a key is a list of principal_name, type (plain or des), version (a nonnegative integer), and value. The value of a des key is 64 bits long and can be represented in dcecp as Extended Registry Attributes (ERAs) of type byte (refer to the xattrschema object attributes for details). The value is valid on input, but is not displayed on output so that keys are not shown on the screen. For example:

```
melman des 1 key1
plain 3 key2
```

Multiple keys for the same principal are displayed as separate keys. See the example in the show operation below.

Attributes

- **uuid value**
  A Universal Unique Identifier (UUID) that is the internal identifier for the key table’s configuration information kept by dced. If the UUID is not specified when the key table is created, one is generated automatically. This attribute cannot be modified after it is created.

- **annotation string**
  A human-readable comment field in Portable Character Set (PCS) format. This attribute cannot be modified after creation. It defaults to a null string (that is, blank).

- **storage string**
  The name of the key table (usually a filename). It is required and cannot be modified after creation.

- **data key_list**
  The contents of the key table. Represented as a list of keys.

See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information about keytab attributes.
Errors

A representative list of errors that might be returned is not shown here. Refer to the *IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide* for complete descriptions of all error messages.

Operations

keytab add

Adds members to a key table. The syntax is as follows:

```
keytab add keytab_name_list
-member principal_name_list
{-key plain_key -version key_version [-registry] |
-random -registry [-version key_version]}
[-ktname residual_keytab_name] [-noprivacy] [-local]
```

Options

- **-member principal_name_list**
  List of principal names to be added to each key table in the argument.

- **-registry**
  Updates the principal’s key in the registry as well as on the host. This option can only be used if the key table already contains a key for the principal. Required if the -random option is used.

- **-random**
  Generates a random des key. Cannot be used with the -key option.

- **-key plain_key**
  Specifies a key explicitly. Cannot be used with the -random option.

- **-version key_version**
  Specifies a version number for the key. Required if the -registry option is not used.

- **-ktname residual_keytab_name**
  Specifies the keytab object to add members to. If you use this option, you must specify keytab_name_list as a string binding. See Arguments for more information about specifying a string binding for keytab_name_list.

- **-local**
  Specifies that the add operation operates on local files only.

- **-noprivacy**
  Specifies that keytabs are sent over the network unencrypted.

The add operation adds members to key tables. The argument is a list of names of key tables to which members should be added. The required -member option lists principal names to be added to each key table in the keytab_name_list argument. If the principals named do not exist, command will return an error. The operation adds each principal name and its key to the key table.

Use either the -random option to have dcecp generate a random des key or the -key option to specify a plain key explicitly. The same key (whether specified or randomly generated) is used for all principals being added to all key tables. The -registry option updates the principal’s key in the key table and in the registry. The -registry option is required if -random is used. The -version option specifies the version number of the key. You must specify either -registry or -version or both on
keytab(8dce)

any keytab add command. The -ktname option is used to identify the specific key table to operate on, but only when the argument is a string binding representing a key table, not the fully qualified key table name. This operation returns an empty string on success.

Privileges Required

You must have a (auth_info) permission to the keytab object.

Examples

dcecp> keytab add ./:/hosts/medusa/config/keytab/radiology
> -member melman -random -registry

dcecp>

dcecp> keytab add ./:/hosts/medusa/config/keytab/radiology
> -member melman -key yrrebnesor

dcecp>

keytab catalog

Returns a list of the names of all key tables on the specified host. The syntax is as follows:

keytab catalog [host_name_list]
[-simplename] [-noprivacy] [-local]

Options

-simplename
   Returns key table names without prepending the cell name.

-noprivacy
   Specifies the key tables sent over the network are not encrypted.

-local
   Specifies that the catalog operation operates on local files only.

The catalog operation returns a list of the names of all key tables on the host specified in the argument. The argument can be a list of one or more host names or a single string binding that identifies a host. If a host name is not specified, the current host is used. If the argument is a list, the output is concatenated. The return order is arbitrary.

Privileges Required

You must have r (read) permission to the keytab object on the host.

Examples

dcecp> keytab catalog
/.../pokey/hosts/jimbo/config/keytab/self

dcecp>

keytab create

Creates a key table. The syntax is as follows:
**keytab(8dce)**

**keytab create**  
`keytab_name_list`  
`{-attribute attribute_list | -attribute value}`  
`[-ktname residual_keytab_name] [-entry] [-noprivacy]`  
`[-local]`

**Options**

- **attribute attribute_list**
  Allows you to specify attributes by using an attribute list rather than using the `-attribute value` option. The format of an attribute list is as follows:
  
  ```
  {{attribute value}...{attribute value}}
  ```

- **attribute value**
  As an alternative to using the `-attribute` option with an attribute list, you can change individual attribute options by prepending a hyphen (`-`) to any attributes listed in `Attributes`.

- **ktname residual_keytab_name**
  Specifies the `keytab` object to create. If you use this option, you must specify `keytab_name_list` as a string binding. See `Arguments` for more information about specifying a string binding for `keytab_name_list`.

- **local**
  Specifies that the `create` operation operates on local files only.

- **noprivacy**
  Specifies that key tables are sent over the network unencrypted.

The `create` operation creates a key table. The argument is a list of names of key tables to be created. The command takes an `-attribute` option to specify configuration information for `dced`. The `-ktname` option identifies the specific key table to operate on, but only when the argument is a string binding representing a key table, not the fully qualified key table name. Use the `data` attribute to specify the contents of the key tables named in `keytab_name_list`. The `data` attribute is a list of keys with associated principal names, key types, versions, and key values in the form

```
principal_name key_type version {key_value}
```

where:

- **principal_name**
  Is the required name of the server principal for which the keytab file is being created.

- **key_type**
  Is a required code that specifies whether the key is stored in plain text or in DES encrypted format:
  1. `des` indicates DES encryption.
  2. `plain` indicates plain text.

- **version**
  Is the key’s required version number.

- **key_value**
  If the key type is `plain`, `key value` is required. If the key type is `des`, `key value` is optional; if one is not entered, a key value is randomly generated.
keytab(8dce)

This operation creates the key tables named in keytab_name_list and assigns all of them the values specified by the data attribute. This operation returns an empty string on success.

Privileges Required

You must have i (insert) permission to the keytab object on the host.

Examples

The following example creates two keys for user vmr and one key for pwang on host medusa. One of vmr 's keys is an automatically generated Data Encryption Standard (DES) key. Both vmr 's second key and pwang 's key are manually entered keys.

dcecp> keytab create /.:/hosts/medusa/config/keytab/radiology -attribute
> {{{storage /opt/dcelocal/keys/radiology} {data {{vmr des 2} 
> {{vmr plain 3 key2} {pwang des 2 key3}}}}}

dcecp>

keytab delete

Deletes a key table entry and its data. The syntax is as follows:

keytab delete keytab_name_list [-entry]
[-noprivacy] [-local]

Options

-entry Specifies that only the configuration information that dced keeps is deleted, not the actual key table.

-noprivacy Specifies that key tables are sent over the network unencrypted.

-local Specifies that the delete operation operates on local files only.

The delete operation deletes a key table entry and its data. The argument is a list of names of key table entries to be deleted in the order specified. If the -entry option is present, only the configuration information that dced keeps is deleted, not the actual key table. This operation returns an empty string on success.

Privileges Required

You must have d (delete) permission to the keytab object. If you are removing the key table, you must have D (Delete_object) permission to the keytab object as well.

Examples

dcecp> keytab delete /.:/hosts/medusa/config/keytab/radiology

dcecp>

keytab help

Returns help information about the keytab object and its operations. The syntax is as follows:

keytab help [operation | -verbose]
Options

-verbose
Displays information about the keytab object.

Used without an argument or option, the keytab help command returns brief information about each keytab operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the -verbose option for more detailed information about the keytab object itself.

Privileges Required

No special privileges are needed to use the keytab help command.

Examples

dcecp> keytab help
add         Adds keys into a key table.
catalog     Returns the list of key table names.
create      Creates a new key table entry and its keys.
delete      Deletes a key table and its associated data.
list        Lists all principals in a specified key table.
remove      Removes keys from a key table.
show        Returns the list of keys of a key table.
help        Prints a summary of command-line options.
operations  Returns a list of the valid operations for this command.

dcecp>

keytab list

Returns a list of all the principals in the specified key table. The syntax is as follows:

keytab list  keytab_name_list [-noprivacy]
[-local]

Options

-noprivacy  Specifies that key tables are sent over the network unencrypted.

-local     Specifies that the list operation operates on local files only.

The list operation returns a list of all the principals in the specified key table. If the argument is a list of key table names, the output is concatenated and a blank line inserted between key tables.

Privileges Required

You must have r (read) permission to the keytab object on the host.

Examples

dcecp> keytab list .:/hosts/medusa/config/keytab/self
/.../mycell/hosts/medusa/self
/.../mycell/hosts/medusa/cds-server
/.../mycell/hosts/medusa/cds-server
dcecp>
keytab(8dce)

**keytab operations**

Returns a list of the operations supported by the `keytab` object. The syntax is as follows:

```
keytab operations
```

The list of available operations is in alphabetical order except for `help` and `operations`, which are listed last.

**Privileges Required**

No special privileges are needed to use the `keytab operations` command.

**Examples**

```
dcecp> keytab operations
add catalog create delete list remove show help operations
dcecp>
```

**keytab remove**

Removes a member from a key table. The syntax is as follows:

```
keytab remove  keytab_name_list -member principal_name_list
```

```
[-version  key_version] [-type  key_type] [-noprivacy] [-local]
```

**Options**

- **-member principal_name_list**
  Specifies a list of one or more principal names of members to be removed from the key table.

- **-version key_version**
  Specifies a version number for the key.

- **-type key_type**
  Specifies whether the key is a `des` (data encryption standard) key or a `plain` key.

- **-noprivacy**
  Specifies that key tables are sent over the network unencrypted.

- **-local**
  Specifies that the `remove` operation operates on local files only.

The `remove` operation removes a member from a key table. The argument is a list of names of key tables from which to remove members. The value of the required `-member` option is a list of names of principals to be removed from the key tables listed in the argument. The two options `-version` and `-type` can be used to limit the keys removed. If either or both of these options is present, then only keys matching the values of these options are removed. The value of the `-version` option can be a list of version numbers. This operation returns an empty string on success.

**Privileges Required**

You must have `x` (execute) permission to the `keytab` object on the host.

**Examples**
keytab(8dce)

The following example removes all des keys for principal D_Britt:

dcecp> keytab remove ./:/hosts/jimbo/config/keytab/self -member D_Britt -type des
dcecp>

keytab show

Returns an attribute list of the key table entries specified in the argument. The syntax is as follows:

keytab show keytab_name_list
[-entry | -members]
[-keys] [-ktname residual_keytab_name] [-noprivacy] [-local]

Options

-entry Returns only the configuration information that dced keeps, not the actual key table data.

-members Specifies that only the data attribute of each entry be returned.

-keys Returns the actual values of keys.

-noprivacy Specifies that key tables are sent over the network unencrypted.

-ktname residual_keytab_name
Specifies the keytab object for which to list entries. If you use this option, you must specify keytab_name_list as a string binding. See Arguments for more information about specifying a string binding for keytab_name_list.

-local Specifies that the show operation operates on local files only.

The show operation returns an attribute list of the key tables specified in the argument. The argument is a list of names of key tables. If the operation is called without the -entry option, the data attribute is not returned. If the optional -members option is given, only the value of the data attribute is returned (a list of keys). Keys are not normally returned unless the -keys option is used. If the argument is a list, the output is concatenated and a blank line inserted between key tables. The -ktname option is used to identify the specific key table to operation on, but only when the argument is a string binding representing a key table, not the fully qualified key table name.

Privileges Required

You must have r (read) permission to the keytab object on the host.

Examples

dcecp> keytab show ./:/hosts/medusa/config/keytab/radiology -members
{melman des 1}
{melman plain 3}
{pwang des 2}
dcecp>

Related Information

Commands: dcecp(8dce), dced(8dce), xattrschema(8dce).
link(8dce)

link

Purpose

A dcecp object that manages a soft link in CDS

Synopsis

link create link_name_list 
[to_target_name | [timeout expiration_time 
extension_time ] | [attribute attribute_list ]}

link delete link_name_list

link help [operation | -verbose ]

link modify link_name_list [-add attribute_list] [-remove attribute_list] 
[-change attribute_list]

link operations

link show link_name_list [-schema]

Arguments

link_name_list
A list of one or more names of CDS soft links.

operation
The name of the link operation for which to display help information.

Description

The link object represents a Cell Directory Service (CDS) soft link. A soft link in
CDS contains an attribute that has a name that is the same as the name of the
object the soft link points to. The soft link contains several built-in attributes, but
users are free to add their own attributes. Softlinks can point to objects, directories,
and other soft links.

Attributes

The following CDS-defined attributes might be present in CDS link objects:

CDS_CTS
Specifies the creation timestamp (CTS) of the soft link. The is a read-only
DTS-style time stamp, which is set by the system.

CDS_LinkTarget
Specifies the full name of the directory, object entry, or other soft link to
which the soft link points.

CDS_LinkTimeout
Specifies a timeout value after which the soft link is either renewed or
deleted. Its value is a list of two elements enclosed in braces, as follows:

{expiration_time extension_time}
where:

**expiration_time**

Is a date and time after which CDS checks for the existence of the soft link’s target and either extends or deletes the soft link. The value is specified in the format `yyyy- mm- dd- hh: mm: ss`; portions of it can be defaulted.

**extension_time**

Is a period of time by which to extend the soft link’s expiration time (if the server has validated that the target still exists). The value is specified in the format `ddd- hh: mm: ss`; portions of it can be defaulted. The `extension_time` value is an optional value.

**CDS_UTS**

Specifies the timestamp of the most recent update to an attribute of the soft link. The value is a read-only DTS-style timestamp, which is set by the system.

See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information about link attributes.

### Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

### Operations

#### link create

Creates a new soft link entry in CDS. The syntax is as follows:

```plaintext
link create link_name_list {-to target-name [-timeout expiration_time extension_time] | -attribute attribute_list}
```

**Options**

- **-to target-name**
  Specifies a single name for the links to point to. If you do not use this option, you must specify the link target with the `-attribute` option.

- **-timeout expiration_time extension_time**
  Specifies the expiration time and extension period for all soft links named by the `link-name_list` argument. The option syntax is as follows:

  ```plaintext
  {expiration_time extension_time}
  ```

  See Attributes for more detailed information about link timeouts. If you omit the `-timeout` option, the link is permanent and must be explicitly deleted.

- **-attribute attribute_list**
  Allows you to specify attributes by using an attribute list. See Attributes for more detailed information about link attributes.
The create operation creates a new soft link entry in CDS. The required link_name_list argument is a list of one or more full CDS names of the soft links to be created. This operation returns an empty string on success.

Privileges Required

You must have i (insert) permission to the directory in which you intend to create the soft link.

Examples

The following command creates a permanent soft link named /./sales/tokyo/price-server that points to an object entry named /./sales/east/price-server. The expiration value indicates that CDS checks that the destination name /./sales/east/price-server still exists on June 25, 1995, at 12:00 p.m. If the destination name still exists, the soft link remains in effect another 90 days. Thereafter, CDS will check that the destination name exists every 90 days.

dcecp> link create /./sales/tokyo/price-server -to \
> /./sales/east/price-server -timeout {1995-06-25-12:00:0090-00:00:00}
dcecp>

You can enter the same information as the above example by using the -attributes option, as follows:

dcecp> link create /./sales/tokyo/price-server -attribute \
> {{CDS_LinkTarget /./sales/east/price-server} {CDS_LinkTimeout \
> {expiration 1995-06-25-12:00:00} {extension 90-00:00:00}}}
dcecp>

link delete

Removes a link entry from CDS. The syntax is as follows:

link delete link_name_list

The delete operation removes a link entry from CDS. This task is usually done through a client application. The required link_name_list argument is a list of one or more full CDS names of the link entry to be removed. This operation returns an empty string on success.

Privileges Required

You must have d (delete) permission to the link entry or A (Admin) permission to the directory that stores the link entry.

Examples

  dcecp> link delete /./sales/tokyo/price-server
  dcecp>

link help

Returns help information about the link object and its operations. The syntax is as follows:

  link help [operation | -verbose]
Options

-verbose
   Displays information about the link object.

Used without an argument or option, the link help command returns brief information about each link operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the -verbose option for more detailed information about the link object itself.

Privileges Required

No special privileges are needed to use the link help command.

Examples

dcecp> link help
create       Creates the named link.
delete       Deletes the named link.
modify       Adds, removes or changes an attribute in the named link.
show         Returns the attributes of a link.
help         Prints a summary of command-line options.
operations   Returns a list of the valid operations for this command.
dcecp>

link modify

Changes attributes in the specified soft links. The syntax is as follows:

link modify  link_name_list  [-add attribute_list] [-remove attribute_list] [-change attribute_list]

Options

-add attribute_list
   Adds one or more new attributes to a soft link or adds values to existing attributes when values are not already present. Add an attribute type with no value by specifying an attribute type with no value.

-remove attribute_list
   Removes an entire attribute or some attribute values from a soft link. If only the attribute type is specified after the option, the entire attribute is removed. If an attribute type and value are specified, only that value is removed. If an attribute or value is not present, an error is returned.

-change attribute_list
   Changes one attribute value to another for a soft link. Each attribute in the list has its existing value replaced by the new value given in the attribute list. For multivalued attributes, all existing values are replaced by all the values listed for the attribute in the attribute list. If an attribute or value is not present, an error is returned.

The modify operation can be used to change two attributes of a soft link: CDS_LinkTarget and CDS_LinkTimeout. The argument is a list of names of soft links to be operated on. The operation takes the -add, -remove, and -change options to specify an attribute list to describe the changes. All the changes are performed on each soft link named in the argument. This operation returns an empty string on success.
Privileges Required

You must have **w** (**write**) permission to the **link** object.

Examples

The following example sets the link expiration time to 1998 and the extension time to 10 days and 0 hours:

```
dcecp> link modify ../../depts/emergency -change { \\
> {CDS_LinkTimeout {expiration 1998-01-20-12:00:00} {extension +10-0:0:0}}} 
```

**link operations**

Returns a list of the operations supported by the **link** object. The syntax is as follows:

```
link operations
```

The list of available operations is in alphabetical order except for **help** and **operations**, which are listed last.

Privileges Required

No special privileges are needed to use the **link operations** command.

Examples

```
dcecp> link operations
create delete modify show help operations
```

**link show**

Returns attribute information associated with specified link entries. The syntax is as follows:

```
link show  link_name_list [-schema]
```

Options

- **-schema**
  This option returns whether an attribute is single or multivalued. The type of value is specific to a link, meaning that the same attribute can be single-valued on one link and multivalued on another.

The **show** operation displays attribute information associated with specified link entries. The required **link_name_list** argument is a list of one or more full CDS names of the soft links you want to show. If more than one link is shown, the attributes of all the soft links are concatenated into one list. The order of the returned attributes is the lexical order of the object identifiers (OIDs) of each attribute for each object.

Privileges Required

You must have **r** (**read**) permission to the link entry.
Examples

dcecp> link show ././depts/emergency
(CDS_CTS 1994-07-11-17:47:59.755+00:00/00-00-c0-8a-df-56)
(CDS_UTS 1994-07-11-17:52:44.698+00:00/00-00-c0-8a-df-56)
(CDS_LinkTarget /.../my_cell.acme_health.org/depts/radiology)
(CDS_LinkTimeout
{expiration 1995-07-11-00:00:00.000}
{extension +10-10:00:00.000I-----}}
dcecp>


dcecp> link show ././gumby -schema
(CDS-CTS single)
(CDS-UTS single)
(CDS-LinkTarget single)
dcecp>

Related Information

Commands: dcecp(8dce), clearinghouse(8dce), directory(8dce), object(8dce).
log

Purpose

A dcecp object that manages serviceability routing and debug routing

Synopsis

log help [operation | -verbose ]

log list {RPC_server_namespace_entry | string_binding_to_server }[-comp component_name_list]

log modify {RPC_server_namespace_entry | string_binding_to_server }-change {routing_specifications | debug_routing_specifications }

log operations

log show {RPC_server_namespace_entry | string_binding_to_server }[-debug]

Arguments

operation
   The name of the log operation for which to display help information.

RPC_server_namespace_entry
   Specifies the namespace entry of the target server. For example, ./hosts/dce_hostname/dts-entity is the name of the DTS server.

string_binding_to_server
   A remote procedure call (RPC) string binding that describes the target server’s network location. The value has the form of an RPC string binding, without an object Universal Unique Identifier (UUID). The binding information contains an RPC protocol, a network address, and an endpoint within [ ] (brackets), in one of the two following forms:

   rpc-prot-seq: network-addr[ endpoint]
   object_uuid@ rpc-prot-seq: network-addr[ endpoint]

Description

The log object represents the current state of message routing for a given server. It supports routing for both production serviceability and debug serviceability messages. Debug routing can be removed from production environment servers while still being used by application servers.

The log commands work on both the local and remote servers. You can identify the target server by supplying either the server’s entry in the namespace or a fully bound string binding. You can specify multiple target servers as a space-separated list. When specifying multiple servers, you can mix the namespace entry and string binding formats in the same list.

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.
**log help**

Returns help information about the log object and its operations. The syntax is as follows:

`log help  [operation | -verbose]`

**Options**

- **-verbose**
  Displays detailed information about the log object.

Used without an argument or option, the log help command returns brief information about each log operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the -verbose option for more detailed information about the log object itself.

**Privileges Required**

No special privileges are needed to use the log help command.

**Examples**

dcccp> log help
list Returns serviceability components registered by a server.
modify Changes serviceability routing specifications of a server.
show Returns serviceability routing settings for a server.
help Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcccp>

**log list**

Returns a list of serviceability components registered by the target servers. The syntax is as follows:

`log list  {RPC_server_namespace_entry | string_binding_to_server}  [-comp component_name_list]`

**Options**

- **-comp component_name_list**
  A list of one or more DCE serviceability component names for which associated subcomponents should be returned.

If you specify more than one server, the returned lists for the second and subsequent servers are concatenated to the returned list for the first server.

The -comp option specifies a space-separated list of DCE serviceability component names. For each named component, the command returns a list of the associated subcomponents. For each subcomponent in the list, the command displays its name, its level, and its description. The order of the component names is arbitrary. If you specify more than one component name, the resulting subcomponent lists are concatenated.

**Privileges Required**

No special privileges are needed to use the log list command.
log(8dce)

Examples

```
dcecp> log list /.:/hosts/goober/cds-server
svc cds dts rpc sec
dcecp>

dcecp> log list /.:/hosts/goober/cds-server -comp dts
general 0 "General server administration"
events 0 "Events received and acted upon"
arith 0 "Math operations"
ctlmsgs 0 "Control messages received"
msgs 0 "Messages received"
states 0 "Server state transitions"
threads 0 "Thread interactions"
config 0 "Server/cell configuration"
sync 0 "Server sync interactions"
dcecp>
```

**log modify**

Sets message routing specifications for one or more specified servers. The syntax is as follows:

```
log modify  {RPC_server_namespace_entry | string_binding_to_server}
-change {routing_specifications | debug_routing_specifications}
```

**Options**

- **-change**
  
  Specifies the routing specifications (production or debug) to change.

The **-change** option specifies the routing specifications you want to change. There is a fixed, well-known set of routing defaults. You can change these defaults, but you cannot add new routings or remove existing routings. Routing is always set on a per-server basis and is recorded in the log object for each server. This operation returns an empty string on success.

Serviceability, production, and debug messages can be written to any of the normal output destinations. You can specify routing for serviceability messages in any of the following ways:

1. Through the dcecp log object, if the server supports the remote serviceability interface
2. By the contents of the dce-local-path/svc/routing routing file
3. By the contents of an environment variable

For a complete discussion of the ways in which you can specify routings for serviceability and debug messages, refer to the svcroute(5dce) reference page.

**Privileges Required**

The privileges are determined by what the server allows for permissions.

Examples

```
dcecp> log modify /.:/tserver -change {{{FATAL TEXTFILE /dev/console} \ 
   {ERROR TEXTFILE /tmp/timop_errors.5.100} {NOTICE BINFILE /tmp/timop_log%ld}}}
dcecp>
```
log operations

Returns a list of the operations supported by the log object. The syntax is as follows:

log operations

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the log operations command.

Examples

dcecp> log operations
list modify show help operations

dcecp>

log show

Returns a list describing the current serviceability routing settings for a server. The syntax is as follows:

log show {RPC_server_namespace_entry | string_binding_to_server} [-debug]

Options

- debug
  Returns serviceability debug routing settings rather than serviceability production routing settings.

If you specify more than one server, the returned routings for the second and subsequent servers are concatenated to the returned routings for the first server. The order of the returned routing settings is arbitrary.

By default the operation returns production serviceability routing settings. Use the - debug option to return debug routing settings. Debug routing settings are not available on servers for which debugging has been turned off (production servers, for example).

Privileges Required

No special privileges are needed to use the log show command.

Examples

dcecp> log show /.../bigred/hosts/acme/cds-clerk
{ERROR STDERR -}
{FATAL FILE /dev/console}
{WARNING FILE /tmp/warnings.log}
dcecp>
log(8dce)

Related Information

See svcroute(5dce) in the IBM DCE Version 3.1 for AIX and Solaris: Application Development Reference

Files: svcroute(5dce).
name

Purpose

A dcecp object that compares and expands DCE names

Synopsis

name compare name name
name expand name
name get string_binding
name help [operation | -verbose ]
name operations
name parse name

Arguments

name The name of an object in the DCE namespace. Examples of names include principal names, names of security groups, names of Cell Directory Service (CDS) objects like directories, softlinks, child pointers and so on, names of remote procedure call (RPC) entries and RPC groups, and Distributed File Service (DFS) filenames.

operation The name of the name operation for which to display help information.

string_binding An RPC string binding (without the object UUID) that identifies the network location of the target name. It contains an RPC protocol and a network address in the form

rpc_prot_seq:network_addr

Description

The name object resolves, compares, and parses DCE names and string bindings.

Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Operations

name compare

Compares two names. The syntax is as follows:

name compare name name
name(8dce)

The **compare** operation compares two names given as arguments and returns 1 if both syntactically refer to the same name. Otherwise, it returns 0.

**Privileges Required**

No special privileges are needed to use the **name compare** command.

**Examples**

dcecp> name compare /.:/sales/east east
Error: Incomplete name
dcecp>

dcecp> name compare /.:/sales/east /.../org_cell/sales/east
1
dcecp>

**name expand**

Expands a simple DCE name to a global name. The syntax is as follows:

```
name expand name
```

The **expand** operation takes a single name as an argument and returns the canonical form of the name. This operation has the effect of converting /.: to /.../cellname.

**Privileges Required**

No special privileges are needed to use the **name expand** command.

**Examples**

```
dcecp> name expand /.:/sales
/.../org_cell/sales
dcecp>
```

**name get**

Returns a hostname given a full or partial string binding. The syntax is as follows:

```
name get string_binding
```

The **get** operation returns host name identified by a specified string binding. The **string_binding** argument is a single string binding; you cannot supply multiple bindings in one operation.

**Privileges Required**

No special privileges are needed to use the **name get** command.

**Examples**

```
dcecp> name get ncan_ip_tcp:15.21.248.170
hosts/goober
dcecp>
```
**name help**

Returns help information about the `name` object and its operations. The syntax is as follows:

```
name help [operation | -verbose]
```

**Options**

- `-verbose`
  
  Displays information about the `name` object.

Used without an argument or option, the `name help` operation returns brief information about each `name` operation. The optional `operation` argument is the name of the operation about which you want detailed information. Alternatively, you can use the `-verbose` option for more detailed information about the `name` object itself.

**Privileges Required**

No special privileges are needed to use the `name help` command.

**Examples**

```
dcecp> name help
compare  Compares two names syntactically.
expand   Returns the canonical form of a name.
get      Gets host name from a partial or full string binding.
parse    Parses name into cell name and residual name.
help     Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp>
```

**name operations**

Returns a list of the operations supported by the `name` object. The syntax is as follows:

```
name operations
```

The list of available operations is in alphabetical order except for `help` and `operations`, which are listed last.

**Privileges Required**

No special privileges are needed to use the `name operations` command.

**Examples**

```
dcecp> name operations
compare expand get parse help operations
```

**name parse**

Divides a name into a cell name and a residual name. The syntax is as follows:

```
name parse name
```
name(8dce)

The parse operation parses a name into a cell name and a residual name. The argument is a single DCE name. The operation returns a list of two elements: cell name and residual name. A name not beginning with a / (slash) is considered to be a name in the local cell.

Privileges Required

No special privileges are needed to use the name parse command.

Examples

dcecp> name parse hosts/goober
/.../pokey hosts/goober
dcecp>

Related Information

Commands: dcecp(8dce)
object

Purpose

A dcecp object that manages an object in the DCE Cell Directory Service (CDS)

Synopsis

object create object_name_list [-attribute attribute_list [-single] ]
object delete object_name_list
object help [operation | -verbose ]
object modify object_name_list {-add attribute_list | [-single] | -remove attribute_list | [-types] | -change attribute_list }
object operations
object show object_name_list [-schema]

Arguments

object_name_list
Examples of objects are remote procedure call (RPC) server entries, group entries, profile entries, and so on.

operation
The name of the object operation for which you want to see help information.

Description

An object object represents an entity in CDS that has a name and attributes. An object identifies a resource such as a host system, a printer, an application, or a file. Attributes consist of a type and one or more values. Every object is the child of a CDS directory.

Attributes

CDS_Class
Specifies the class to which an object belongs.

CDS_CTS
Specifies the creation timestamp of the CDS object. The value is a read-only DTS-style timestamp, which is set by the system.

CDS_ClassVersion
Contains the version number of the object’s class, which allows applications to build in compatibility with entries created by earlier versions.

CDS_ObjectUUID
Specifies the unique identifier of the object. The read-only identifier is set by the system at creation time.
CDS_UTS
Specifies the timestamp of the most recent update to an attribute of the object. The value is a read-only DTS-style timestamp, which is set by the system.

See the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide* for more information about object attributes.

Errors
A representative list of errors that might be returned is not shown here. Refer to the *IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide* for complete descriptions of all error messages.

Operations

object create
Creates a new object entry in CDS. The syntax is as follows:

```
object create  object_name_list [-attribute attribute_list [-single]]
```

Options

-attribute attribute_list
Allows you to specify attributes by using an attribute list. See Attributes for more information about object attributes.

-single
Specifies that attribute values are single-valued. Otherwise, attributes are multivalued. Valid only with the -attribute option.

The create operation creates a new object entry in CDS. This task is usually done through a client application. The required object_name_list argument is a list of the full CDS names of the object entries to be created.

Optionally, you can use the -attribute option to associate one or more attributes (see Attributes) with each object being created. The attribute values are multivalued unless the -single option is specified, in which case all attributes are single-valued. The -single option is valid only if the -attribute option is specified. This operation returns an empty string on success.

Privileges Required
You must have i (insert) permission to the parent directory.

Examples
The following command creates an object entry named /:/sales/east/floor1cp. The object entry describes a color printer on the first floor of a company’s eastern sales office.

```
dcecp> object create /:/sales/east/floor1cp -attribute 
> {{CDS_Class printer} {CDS_ClassVersion 1.0}}
dcecp>
```
object delete

Removes an object entry from CDS. The syntax is as follows:

**object delete**  **object_name_list**

The *delete* operation removes an object entry from CDS. The required *object_name_list* argument is a list of the full CDS names of the object entries to be deleted. This operation returns an empty string on success.

Privileges Required

You must have **d (delete)** permission to the object entry or **A (Admin)** permission to the directory that stores the object entry.

Examples

The following command deletes the object entry `./:/sales/east/floor1pr2`:

```
dcecp> object delete ./:/sales/east/floor1pr2
dcecp> 
```

**object help**

Returns help information about the object object and its operations. The syntax is as follows:

**object help**  **[operation | -verbose]**

Options

**-verbose**  
Displays information about the **object** object.

Used without an argument or option, the **object help** command returns brief information about each **object** operation. The optional **operation** argument is the name of an operation about which you want detailed information. Alternatively, you can use the **-verbose** option for more detailed information about the **object** object itself.

Privileges Required

No special privileges are needed to use the **object help** command.

Examples

```
dcecp> object help
create     Creates the named object.
delete     Deletes the named object.
modify     Adds, removes or changes an attribute in the named object.
show       Returns the attributes of an object.
help       Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp> 
```
object modify

Adds or removes attributes or changes attribute values for object entries in CDS. The syntax is as follows:

```
object modify object_name_list { -add attribute_list [-single] | -remove attribute_list [-types] | -change attribute_list }
```

Options

- **-add attribute_list**
  Adds one or more new attributes to an object entry.

- **-single**
  Can be used with the `-add` option to specify that attributes to be added are single-valued.

- **-remove attribute_list**
  Eliminates one or more attribute values from an attribute type of an object entry. For instance, removing a value from an attribute with three values leaves the attribute with two values. The argument is an attribute list of the following form:

  ```
  { {attribute value} ... {attribute value} }
  ```

  **Note:** A value cannot be removed from a single-valued attribute. To totally remove either a single-valued attribute or a set-valued attribute, the `-types` option must be used.

  To remove an attribute type as well as its values, use the `-types` option with the `-remove` option.

  If an attribute is not present, an error is returned. Fixed CDS attribute types, such as the CDS creation timestamp (CDS_CTS), cannot be removed.

- **-types**
  Can be used with the `-remove` option to remove the attribute type as well as its values. Invalid without the `-remove` option.

- **-change attribute_list**
  Changes one attribute value to another for an object entry. The existing value of each attribute in `attribute_list` is replaced by the new value given. For multivalued attributes, all existing values are replaced by all the values listed for the attribute in the attribute list. If an attribute or value is not present, an error is returned.

The `modify` operation adds or removes attributes, or changes attribute values for object entries in CDS. This task is usually done through a client application. The required `object_name_list` argument is a list of the full CDS names of the object entries to be modified. This operation returns an empty string on success.

**Privileges Required**

You must have **w** (write) permission to the object entry.

**Examples**
To add the `sales_record` attribute with a value of `region2` to an object entry named `/.:/Q1_records`, follow these steps:

1. Read the `cds_attributes` file to check that the attribute `sales_record` is listed, as shown in the following display:

<table>
<thead>
<tr>
<th>OID</th>
<th>LABEL</th>
<th>SYNTAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.22.1.3.66</td>
<td>sales_record</td>
<td>char</td>
</tr>
</tbody>
</table>

2. Enter the following command to assign the value `region2` to the attribute `sales_record` of an object entry named `/.:/Q1_records`.

```dcecp>
object modify /.:/Q1_records -add {sales_record region2}
dcecp>
```

To remove the `RPC_CLASS` and `RPC_CLASS_VERSION` attributes:

```dcecp>
object modify /.:/foo -remove {RPC_CLASS RPC_CLASS_VERSION} -types
```

### object operations

Returns a list of the operations supported by the `object` object. The syntax is as follows:

#### object operations

The list of available operations is in alphabetical order except for `help` and `operations`, which are listed last.

#### Privileges Required

No special privileges are needed to use the `object operations` command.

#### Examples

```dcecp>
object operations
create delete modify show help operations
dcecp>
```

### object show

Returns attribute information associated with specified object entries. The syntax is as follows:

#### object show  `object_name_list` [-schema]

#### Options

- `-schema`

  Indicates whether an attribute is single or multivalued. Note that the same attribute can be single-valued on one object and multivalued on another object.

The `show` operation displays attribute information associated with specified object entries. The required `object_name_list` argument is a list of the full CDS names of the object entries to be examined. If more than one object is shown, the attributes of all the objects are concatenated into one list. The order of the returned attributes is the lexical order of the object identifiers (OIDs) of each attribute for each object.
The -schema option indicates whether an attribute is single-valued or multivalued.

**Privileges Required**

You must have `r` (read) permission to the object entry. If you specify a wildcard object entry name, you also need `r` (read) permission to the directory that stores the object entry.

**Examples**

```bash
dcecp> object show /.:obj
{RPC_ClassVersion
 (0200)
 (0300))
{RPC_Group 1234}
{CDS_CTS 1994-07-01-22:06:54.990-05:0010.000/00-00-c0-f7-de-56}
{CDS_UTS 1994-07-01-22:07:37.248-05:0010.000/00-00-c0-f7-de-56}
{CDS_Class 0200}
dcecp>
dcecp> object show /.:obj -schema
{RPC_ClassVersion multi}
{RPC_Group multi}
{CDS_CTS single}
{CDS_UTS single}
{CDS_Class single}
dcecp>
```

**Related Information**

Commands: `dcecp(8dce)`, `clearinghouse(8dce)`, `directory(8dce)`, `link(8dce)`,
organization

Purpose

A dcecp object that manages an organization in the DCE Security Service

Synopsis

```
organization add organization_name_list -member member_name_list
organization catalog [cell_name] [-simplename]
organization create organization_name_list {-attribute extended_rgy_attr_list | -attribute value}
organization delete organization_name_list
organization help [operation | -verbose]
organization list organization_name_list [-simplename]
organization modify organization_name_list {-add extended_rgy_attr_list | -remove extended_rgy_attr_list | [-types] | -change extended_rgy_attr_list | -attribute value}
organization operations
organization remove organization_name_list -member member_name_list
organization rename organization_name -to new_organization_name
organization show organization_name_list [-all | [-policies] | [-xattrs]]
```

Arguments

```
cell_name
   The name of a cell to contact when processing the catalog operation. The name must be a fully qualified cell name, such as /.: or /.../ cell_name
operation
   The name of the organization operation for which to display help information.
organization_name
   The name of a single organization to act on. See organization_name_list for the name format.
organization_name_list
   A list of one or more names of organizations to act on. Supply the names as follows:
   1. Fully qualified names in the form: /.../ cell_name/organization_name or /.:/ organization_name
   2. Cell-relative names in the form organization_name. These names refer to an organization in the cell identified in the _s(sec) convenience variable, or if the _s(sec) convenience variable is not set, in the local host’s default cell.
```
organization(8dce)

Do not mix fully qualified names and cell-relative names in a list. In addition, do not use the names of registry database objects that contain organization information; in other words, do not use names that begin with "/:sec/org/".

Description

The organization object represents registry organizations. Unless otherwise noted, all organization operations take the names of the organizations to act on as an argument.

When this command executes, it attempts to bind to the registry server identified in the _s(sec) variable. If that server cannot process the request or if the _s(sec) variable is not set, the command binds to either an available slave server or the master registry server, depending on the operation. Upon completion the command sets the _b(sec) convenience variable to the name of the registry server to which it bound.

Attributes

The organization object supports two kinds of attributes: organization and policy.

1. Organization attributes consist of the organization’s name, Universal Unique Identifier (UUID), and organization identifier. Organization attributes might or might not have default values. They assume a default value or a value set by administrators.

2. Policy attributes regulate such things as account and password lifetimes for all accounts associated with a particular organization. If you do not set these attributes, they default to the value set for the registry as a whole with the registry modify command. Note that if a policy attribute value set for the registry as a whole is stricter than the value you set for an organization, the registry wide value is used.

Organization Attributes

orgid integer

Used with the create operation to specify the organization identifier for the organization. If this attribute is not set when an organization is created, an organization identifier is assigned automatically. Do not specify the -orgid attribute when creating two or more organizations with the same command. If you do, the second create operation will fail, since the organization identifier is already in use after the first is created. However, the alias and orgid attributes can be specified to create several aliases for an existing organization with one command.

uuid hexadecimal number

Used with the create operation to specify the organization’s UUID, a unique internal identifier. Use the UUID attribute only to adopt an orphaned UUID. Normally the UUID for a new organization is generated by the registry. In cases where data exists tagged with a UUID of an organization that has been deleted from the registry, use the create operation to specify the old UUID for a new organization. The UUID specified must be an orphan, that is, a UUID for which no name exists in the registry. An error occurs if you specify a name that is already defined in the registry.

fullname string

Used with the create and modify operations to specify the organization’s full name, a name used for information purposes only. The full name
typically describes or expands a primary name to allow easy recognition by users. For example, an organization could have a primary name of abc and a full name of Advanced Binary Corporation. The value is a string. If it contains spaces, it is displayed in quotation marks, on entry, must be enclosed in quotation marks or braces. The fullname attribute defaults to the null string (that is, blank).

Policy Attributes

Since organization policy attributes do not exist on an organization unless explicitly defined, they have no default values. The organization policy attributes are as follows:

acctlife { relative_time| unlimited}
Defines the lifespan of accounts. Specify the time by using the Distributed Time Service (DTS) relative time format ([/-]dd-hh:mm:ss) or the string unlimited.

pwdalpha {yes | no}
Defines whether passwords can consist entirely of alphanumeric characters. Its value is either yes or no.

pwdexpdate { ISO_timestamp | none}
Defines a date on which a password expires. Specify the date by using an ISO-compliant time format such as CC-MM-DD hh:mm:ss or the string none, which specifies that the password not expire.

pwdlife { relative_time | unlimited}
Defines the lifespan of passwords. Specify the time by using the DTS-relative time format ([/-] DD- hh: mm: ss) or the string unlimited.

pwdminlen integer
Defines the minimum number of characters in a password. Its value is a positive integer or the integer 0, which means there is no minimum length.

pwdspaces {yes | no}
Defines whether or not passwords can consist entirely of spaces. Its value is either yes or no.

See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information about organization and policy attributes.

Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Operations

organization add

Adds members to a security organization. The syntax is as follows:

organization add organization_name_list -member
member_name_list

Options
organization(8dce)

-member member_name_list
   Specifies a list of one or more names of principals to be added to each
   organization in the argument.

The add operation adds members to an organization. The organization_name_list
argument is a list of names of organizations to be added to. The member_name_list
argument of the required -member option is a list of names of principals to be
added to each organization in the argument. If the principals do not exist, the
command returns an error. This operation returns an empty string on success.

Privileges Required

You must have r (read) and M (Member_list) permissions on the target
organization and r (read ) permission on the principal being added.

Examples

dcecp> organization add managers -member W_White
dcecp>

organization catalog

Returns a list of the names of all organizations in the registry. The syntax is as
follows:

organization catalog [cell_name] [-simplename]

Options

-simplename
   Returns a list of organization names in the registry without prepending the
cell name.

The catalog operation returns a list of the names of all organizations in the local
registry in lexical order. Use the cell_name argument to return a list of organizations
in another cell's registry. By default, fully qualified names are returned in the form
cellnamel organization_name. Use the -simplename option to return them in the
form organization_name.

Privileges Required

You must have r (read) permission to the ./sec/org directory.

Examples

dcecp> organization catalog
/.../my_cell.goodcompany.com/none
/.../my_cell.goodcompany.com/users
/.../my_cell.goodcompany.com/managers
dcecp>

dcecp> organization catalog -simplename
none
users
managers
dcecp>
organization create

Creates a new organization in the registry database. The syntax is as follows:

organization create organization_name_list {-attribute extended_rgy_attr_list | -attribute value}

Options

- attribute value
  As an alternative to using the -attribute option with an attribute list, you can change individual attribute options by prepending a hyphen (-) to any attributes listed in the Attributes section of this reference page. You cannot use this option to specify ERAs; it is only for the standard attributes described in Attributes.

-attribute extended_rgy_attr_list
  Allows you to specify attributes, including ERAs, by using an attribute list rather than using the -attribute value option. The format of an attribute list is as follows:

  {{extended_rgy_attr_list value}...{extended_rgy_attr_list value}}

See the OSF DCE Administration Guide for more information on ERAs.

The create operation creates a new organization. The organization_name_list argument is a list of names of organizations to be created. Options specify the attributes of the newly created organization. All options are applied to all organizations in the argument list. This operation returns an empty string on success.

Privileges Required

You must have i (insert) permission to the directory in which the organization is to be created.

Examples

dcecp> organization create temps -fullname "Temporary Employees"
dcecp>
dcecp> organization create temps -attribute {fullname "Temporary Employees"}
dcecp>
dcecp> org create dce -fullname {Dist Comp Env} -orgid 101
dcecp>
dcecp> org create dce -fullname {Dist Comp Env} \
> -uuid c2aac790-dc6c-11cc-a6f8-080009251352
dcecp>

organization delete

Deletes organizations from the registry. The syntax is as follows:

organization delete organization_name_list

The delete operation deletes organizations from the registry. The organization_name_list argument is a list of names of organizations to be deleted. If a named organization does not exist, an error is generated. This operation returns an empty string on success.
This operation also deletes any accounts associated with organizations that are deleted. To preserve accounts, add desired principals to a different organization by using the `organization add -member` command. Modify the principals' accounts to point to the new organization by using the `account modify` command. Then you can delete the organization by using the `organization delete` command.

**Privileges Required**

You must have `d` (delete) permission to the directory in which the target organization exists. You must have `r` (read) and `D` (Delete_object) permissions on the organization to be deleted.

**Examples**

```
dcecp> organization delete temps
dcecp> 
```

**organization help**

Returns help information about the `organization` object and its operations. The syntax is as follows:

```
organization help [operation | -verbose]
```

**Options**

- `-verbose`
  Displays information about the `organization` object.

Used without an argument or option, the `organization help` command returns brief information about each `organization` operation. The optional `operation` argument is the name of an operation about which you want detailed information. Alternatively, you can use the `-verbose` option for more detailed information about the `organization` object itself.

**Privileges Required**

No special privileges are needed to use the `organization help` command.

**Examples**

```
dcecp> organization help
add Adds a member to the named organization.
catalog Returns a list of all the names of organizations.
create Creates an organization in the registry.
delete Deletes an organization from the registry.
list Returns a list of all the members of an organization.
modify Changes the information about an organization.
remove Removes a member from the named organization.
rename Renames the specified organization.
show Returns the attributes of an organization.
help Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp> 
```

**organization list**

Returns a list of the names of all members of an organization. The syntax is as follows:
organization list  organization_name_list [-simplename]

Options

-simplename
   Returns a list of member names in the organization without prepending the cell name.

The list operation returns a list of the names of all members of an organization. The organization_name_list argument is a list of names of organizations. By default, fully qualified names are returned in the form cellname/member_name. If the -simplename option is given, the cell name is not prepended to the member names. Names are returned in lexical order.

Privileges Required

You must have r (read) permission to the organization.

Examples

dcecp> organization list managers
/.../my_cell.goodcompany.com/W_Ward
/.../my_cell.goodcompany.com/L_Jones
/.../my_cell.goodcompany.com/S_Preska
/.../my_cell.goodcompany.com/S_Rohrer
/.../my_cell.goodcompany.com/J_Wanders
/.../my_cell.goodcompany.com/K_Parsons

dcecp>

dcecp> organization list {managers users}
/.../my_cell.goodcompany.com/W_Ward
/.../my_cell.goodcompany.com/L_Jones
/.../my_cell.goodcompany.com/S_Preska
/.../my_cell.goodcompany.com/S_Rohrer
/.../my_cell.goodcompany.com/J_Wanders
/.../my_cell.goodcompany.com/W_Ross
/.../my_cell.goodcompany.com/J_Severance
/.../my_cell.goodcompany.com/J_Hunter
/.../my_cell.goodcompany.com/B_Carr
/.../my_cell.goodcompany.com/E_Vliet
/.../my_cell.goodcompany.com/J_Egan
/.../my_cell.goodcompany.com/F_Willis

dcecp>

organization modify

Changes attributes and policies of organizations. The syntax is as follows:

organization modify  organization_name_list
{-add extended_rgy_attr_list  | -remove extended_rgy_attr_list  [-types]  | -change extended_rgy_attr_list  | -attribute value}

Options

- attribute value
   As an alternative to using options with an attribute list, you can change individual attribute options by prepending a hyphen (-) to any attributes listed in the Attributes section of this reference page. You cannot use this option to specify ERAs; it is only for standard group attributes described in Attributes.
organization

- **add extended_rgy_attr_list**
  Allows you to modify attributes, including ERAs, by using an attribute list rather than individual attribute options. The format of an attribute list is as follows:

  `{{extended_rgy_attr_list value}...{extended_rgy_attr_list value}}`

- **change extended_rgy_attr_list**
  Allows you to modify attributes, including ERAs, by using an attribute list rather than individual attribute options. See the **add** option for the attribute list format.

- **remove extended_rgy_attr_list**
  Allows you to modify attributes, including ERAs, by using an attribute list rather than using individual attribute options such as `-fullname`, `-acctlife`, and so on. See the **add** option for the attribute list format.

  Without the **-types** option, **remove** deletes individual attribute instances attached to the group. In this case, `extended_rgy_attr_list` is a list of attribute-value pairs. With the **-types** option, **remove** deletes attribute types (and all instances of that type) attached to the group. In this case, `extended_rgy_attr_list` is a list of attribute types.

- **-types**
  Used with the **-remove** option to remove attribute types (and all instances of that type) attached to the group.

See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information about ERAs.

The **modify** operation changes attributes and policies of organizations. (To change registrywide policies, use the **registry** command.)

The **organization_name_list** argument is a list of names of organizations to be operated on. All modifications are applied to all organizations named in the argument. Organizations are modified in the order they are listed and all modifications to an individual organization are atomic. Modifications to multiple organizations are not atomic. A failure for any one organization generates an error to be generated and aborts the rest of the operation. This operation returns an empty string on success.

The **-change** option can modify the value of any attribute except for **orgid** and **uuid**.

**Privileges Required**

You must have **r** (read) permission on the organization to be modified and **f** (**full_name**) permission to change the organization’s full name and/or **m** (**mgmt_info**) permission to change the organization’s managment information.

**Examples**

```
dcecp> organization modify temps -acctlife 180-00:00:00 \
> -pwdalphpa yes -pwdlife 30-00:00:00 \
> -pwdexpdate 1995-12-31-23:59:59 -pwdspaces yes
```

```
dcecp> organization modify temps -add {test_era 101}
dcecp>
```
organization operations

Returns a list of the operations supported by the organization object. The syntax is as follows:

organization operations

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the organization operations command.

Examples

dcecp> organization operations
add catalog create delete list modify remove
rename show help operations

dcecp>

organization remove

Removes a member from an organization. The syntax is as follows:

organization remove organization_name_list -member member_name_list

Options

- member member_name_list
  Specifies a list of one or more names of principals to be removed from each organization in the argument.

The remove operation removes members from an organization. The argument is a list of names of organizations from which to remove members. The value of the required - member option is a list of names of principals to remove from the organizations listed in the argument. When a member is removed from an organization, any accounts associated with that principal and group are deleted. Remember that accounts are associated with a principal, a group, and an organization; therefore, any accounts whose principal name and organization name match those given to this command are removed, but accounts for which only one name matches are untouched. This operation returns an empty string on success.

Privileges Required
organization(8dce)

You must have r (read) and M (Member_list) permissions on the target organizations and r (read) permission on the member to be removed.

Examples

dcecp> organization remove managers -member J_Wanders

dcecp>

dcecp> organization add rigel -member W_White

dcecp> account modify W_White -organization rigel

dcecp> organization add rigel -member W_Ross

dcecp> account modify W_Ross -organization rigel

dcecp> account show W_Ross

{created /.../my_cell.goodcompany.com/cell_admin 1994-06-30-12:39:48.000+00:00I-----}
{description {}}
{dupkey no}
{expdate none}
{forwardabletkt yes}
{goodsince 1994-06-30-12:39:48.000+00:00I-----}
{group users}
{home /}
{lastchange /.../my_cell.goodcompany.com/cell_admin 1994-06-30-12:39:48.000+00:00I-----}
{organization rigel}
{postdatedtkt no}
{proxiabletkt no}
{pwdvalid yes}
{renewabletkt yes}
{server yes}
{shell {}}
{stdtgtauth yes}
dcecp>

dcecp> organization remove gemini -member W_Ross

dcecp>

organization rename

This operation changes the name of a specified organization. The syntax is as follows:

organization rename  organization_name -to new_organization_name

Options

-to new_organization_name
  Specifies the new name of the organization.

  See Arguments for a description of organization names.

The rename operation changes the name of a specified organization. The organization_name argument is a single name of an organization to be renamed. The required -to option specifies the new name, which cannot be a list. This operation returns an empty string on success.

Privileges Required

You must have r (read) and n (name) permission to the specified organizations.

Examples

dcecp> organization list rigel

/.../my_cell.goodcompany.com/H_Lewis
organization rename rigel -to sirus

dcecp> organization list rigel

Error: Registry object not found

dcecp>

dcecp> organization list sirus

organization show

Returns registry information for the specified organizations. The syntax is as follows:

organization show  organization_name_list [-all | [-policies] | [-xattrs]]

Options
-policies
  Returns only the polices of the organization, with no other attributes.
-xattrs
  Returns only the ERAs of the organization, with no other attributes.
-all
  Return the attributes followed by the policies and ERAs.

The show operation returns an attribute list describing the specified organizations. The organization_name_list argument is a list of names of organizations to be operated on. If more than one organization is given, the attributes are concatenated together.

Attributes are returned in the following order: fullname, orgid, uuid. Policies are returned in the following order: acctlife, pwdalpha, pwdexpdate, pwdlife, pwdminlen, and pwdspaces. If the organization does not have any policies, then nopolicy is returned.

The policy set for an organization and the policy set for the registry as a whole might differ. If this is the case, show displays both policies and tags the registry policy with the label effective. The actual policy in effect is the stricter of the two displayed policies, regardless of the effective label.

Privileges Required

You must have r (read) permission on the specified organizations.

Examples
organization(8dce)

```bash
dcecp> organization show temps
{fullname {Temporary Employees}}
{orgid 103}
{uuid 00000067-9402-21cd-a602-0000c08adf56}
dcecp>

dcecp> organization show temps -policies
{acctlife +180-00:00:00.000I-----}
{pwdalpha yes}
{pwdexpdate 1995-12-31-23:59:59.000+00:00I-----}
{pwdlife +30-00:00:00.000I-----}
{pwdminlen 0}
{pwdspaces yes}
dcecp>

dcecp> organization show temps -policies
{acctlife 30 days}
{pwdalpha no}
{pwdexpdate none}
{pwdlife 4 effective 5 days}
{pwdminlen 6}
{pwdspaces no}
dcecp>

dcecp> organization show temps -all
{fullname {Temporary Employees}}
{orgid 103}
{uuid 00000067-9402-21cd-a602-0000c08adf56}
{acctlife +180-00:00:00.000I-----}
{pwdalpha yes}
{pwdexpdate 1995-12-31-23:59:59.000+00:00I-----}
{pwdlife +30-00:00:00.000I-----}
{pwdminlen 0}
{pwdspaces yes}
dcecp>
```

Related Information

Commands: account(8dce), dcecp(8dce), group(8dce), principal(8dce), registry(8dce), xattrschema(8dce).
principal

Purpose

A dcecp object that manages a principal in the DCE Security Service

Synopsis

 principal catalog [cell_name] [-simplename]
 principal create principal_name_list {-attribute extended_rgy_attr_list | -attribute value }
 principal delete principal_name_list
 principal help [operation | -verbose ]
 principal modify principal_name_list {-add extended_rgy_attr_list | -remove extended_rgy_attr_list | [-types] | -change extended_rgy_attr_list | -attribute value }

principal operations

 principal rename principal_name -to new_principal_name
 principal show principal_name_list [-all | -xattrs ]

Arguments

 cell_name
    The name of a cell to contact when processing the catalog operation. The name must be a fully qualified cell name, such as ./ or ./.../ cell_name

 operation
    The name of the principal operation for which to display help information.

 principal_name
    The name of a principal to act on. See principal_name_list for the name format.

 principal_name_list
    A list of one or more names of principals to act on. Supply the names as follows:
    1. Fully qualified principal names in the form

        /.../ cell_name/ principal_name or
        /.:/ principal_name

    2. Cell-relative principal names in the form

        principal_name

    These names refer to a principal in the cell identified in the _s(sec) convenience variable, or if the _s(sec) convenience variable is not set, in the local host’s default cell.
Do not mix fully qualified names and cell-relative names in a list. In addition, do not use the names of registry database objects that contain principal information; in other words, do not use names that begin with "/.:sec/principal".

Description

The principal object represents registry principals. Unless otherwise noted, all of the operations of this object take the names of principals to act on as an argument. These must be principal names, not the names of the database objects that contain registry information about principals (that is, the names must not begin with "/.:sec/principal").

When this command executes, it attempts to bind to the registry server identified in the _s(sec) variable. If that server cannot process the request or if the _s(sec) variable is not set, the command binds to either an available slave server or the master registry server, depending on the operation. Upon completion, the command sets the _b(sec) convenience variable to the name of the registry server it bound to.

Attributes

alias value

Used with the create and modify operations to specify whether the principal name is an alias. The value of this attribute is either yes (the name is an alias) or no (the name is not an alias). The default in no.

Each principal can have only one primary name, but might have one or more alias names. All of a principal's alias names refer to the same principal, and therefore share the same UUID and UNIX ID. While aliases refer to the same principal, they are separate entries in the registry database.

uid value

Used with the create operation only for cell principals, to specify the integer to use as user identifier, known as a Unix ID, for the cell principals. No two principals can have the same UNIX ID. However, aliases can share one.

If you do not enter this option for a cell principal, the next sequential UNIX number is supplied as a default by the registry. For all principals other than cell principals, the UNIX ID is extracted from information embedded in the principal's UUID and cannot be specified here. If this attribute is not supplied when a principal is created, one is supplied automatically.

uuid hexadecimal number

Used with the create operation to specify the internal identifier, known as a UUID, for the principal. No two principals can have the same UUID, so do not use this option when creating more than one principal with a single create command.

This option can also be used to adopt an orphaned UUID. Normally, the UUID for a new principal is generated by the registry. When data is tagged with a UUID of a principal that has been deleted from the registry, this option can be used on the create operation to specify the old UUID for a new principal. The UUID specified must be an orphan (a UUID for which no name exists in the registry). An error occurs if you specify a name or UUID that is already defined in the registry.
The `-alias` option cannot be used with this option. Both the `-fullname` and the `-quota` options can.

**fullname string**

Used with the `create` and `modify` operations, to specify the full name of the principal. This name is used for information purposes only. It typically describes or expands a primary name to allow easy recognition by users. For example, a principal could have a primary name of `jsbach` and a full name of `Johann S. Bach`. The value is a string. If the string contains spaces, you must surround them with quotation marks or braces for entry. This option defaults to a null string (that is, blank).

**quota { quota | unlimited}**

Used with the `create` and `modify` operations to specify the principal’s object creation quota, which is the total number of registry objects that can be created by the principal. It is either a nonnegative number or the string `unlimited`. A value of 0 prohibits the principal from creating any registry objects. Each time a principal creates a registry object, this value is decremented for that principal.

See the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide* for more information about principal attributes.

**Errors**

A representative list of errors that might be returned is not shown here. Refer to the *IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide* for complete descriptions of all error messages.

**Operations**

**principal catalog**

Returns a list of the names of all principals in the registry. The syntax is as follows:

```
principal catalog [cell_name] [-simplename]
```

**Options**

- `-simplename`
  
  Returns a list of principal names in the registry without prepending the cell name.

The `catalog` operation returns a list of the names of all principals in the local registry in lexical order. Use the `cell_name` argument to return a list of principals in another cell’s registry. By default, fully qualified names are returned in the form `cellname/principal_name`. Use the `-simplename` option to return them in the form `principal_name`.

**Privileges Required**

You must have `r` (read) permission to the `./sec/principal` directory.

**Examples**

```
dcecp> principal catalog
/.../small_cell.goodcompany.com/nobody
```
principal(8dce)

/pr.../small_cell.goodcompany.com/root
/pr.../small_cell.goodcompany.com/daemon
/pr.../small_cell.goodcompany.com/sys
/pr.../small_cell.goodcompany.com/bin
/pr.../small_cell.goodcompany.com/uucp
/pr.../small_cell.goodcompany.com/who
/pr.../small_cell.goodcompany.com/mail
/pr.../small_cell.goodcompany.com/tcb
/pr.../small_cell.goodcompany.com/dce-ptgt
/pr.../small_cell.goodcompany.com/dce-rgy
/pr.../small_cell.goodcompany.com/cell_admin
/pr.../small_cell.goodcompany.com/krbtgt/small_cell.goodcompany.com
/pr.../small_cell.goodcompany.com/hosts/pmin17/self
/pr.../small_cell.goodcompany.com/hosts/pmin17/cds-server
/pr.../small_cell.goodcompany.com/hosts/pmin17/gda
/pr.../small_cell.goodcompany.com/William_Ward
/pr.../small_cell.goodcompany.com/John_Hunter

dcecp>

principal create

Creates a new principal in the registry database. The syntax is as follows:

principal create principal_name_list
{-attribute extended_rgy_attr_list | -attribute value}

Options

- attribute value
As an alternative to using the -attribute option with an attribute list, you can change individual attribute options by prepending a hyphen (-) to any attributes listed in Attributes. You cannot use this option to specify ERAs; it is only for the standard attributes described in Attributes.

-attribute extended_rgy_attr_list
Allows you to specify attributes, including ERAs, by using an attribute list rather than using the - attribute value option. The format of an attribute list is as follows:

{{extended_rgy_attr_list value}...{extended_rgy_attr_list value}}

The create operation creates a new principal in the registry database. The principal_name_list argument is a list of names of principals to be created. Options are used to specify the attributes of the newly created principal. All options are applied to all principals in the argument. This operation returns an empty string on success.

Privileges Required

You must have i (insert) permission to the directory in which the principal is to be created.

Examples

The following command creates an alias postmaster for the principal with UNIX ID 1234:

dcecp> principal create postmaster -uid 1234 -alias yes
dcecp>
principal delete

Deletes principals from the registry. The syntax is as follows:

```
principal delete  principal_name_list
```

The delete operation deletes principals from the registry. When a principal is deleted, the principal's account is deleted as well. The principal_name_list argument is a list of names of principals to be deleted. Note that these names can be either a primary or alias names. In either case, an account associated with that name is deleted. If a named principal does not exist, an error is generated. This operation returns an empty string on success.

Privileges Required

You must have d (delete) permission to the directory in which the target principal exists. You must have r (read) and D (Delete_object) permissions on the principal to be deleted.

Examples

```
dcecp> principal delete ./:William_Smith
```

principal help

Returns help information about the principal object and its operations. The syntax is as follows:

```
principal help  [operation | -verbose]
```

Options

- **-verbose**
  Displays information about the principal object.

Used without an argument or option, the principal help command returns brief information about each principal operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the -verbose option for more detailed information about the principal object itself.

Privileges Required

No special privileges are needed to use the principal help command.

Examples

```
dcecp> principal help
catalog Returns all the names of principals in the registry.
create Creates a DCE principal.
delete Deletes a principal from the registry.
modify Changes the information about a principal.
rename Renames the specified principal.
show Returns the attributes of a principal.
help Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp>
```
principal modify

Changes attributes of principals. The syntax is as follows:

```
principal modify principal_name_list
{-add extended_rgy_attr_list | -remove extended_rgy_attr_list [-types] | -change extended_rgy_attr_list | -attribute value}
```

Options

- `attribute value`
  As an alternative to using options with an attribute list, you can change individual attribute options by prepending a hyphen (`-`) to any attributes listed in Attributes. You cannot use this option to specify ERAs; it is only for standard group attributes described in Attributes.

- `-add extended_rgy_attr_list`
  Allows you to modify attributes, including ERAs, by using an attribute list rather than using individual attribute options. The format of an attribute list is as follows:

  ```
  {{extended_rgy_attr_list value}...{extended_rgy_attr_list value}}
  ```

- `-change extended_rgy_attr_list`
  Allows you to modify attributes, including ERAs, by using an attribute list rather than using individual attribute options. See the `-add` option for the attribute list format.

- `-remove extended_rgy_attr_list`
  Allows you to modify attributes, including ERAs, by using an attribute list rather than using individual attribute options such as `-alias`, `-fullname`, and so on. See the `-add` option for the attribute list format.

  Without the `-types` option, `-remove` deletes individual attribute instances attached to the group. In this case, `extended_rgy_attr_list` is a list of attribute-value pairs. With the `-types` option, `-remove` deletes attribute types (and all instances of that type) attached to the group. In this case, `extended_rgy_attr_list` is a list of attribute types.

- `-types`
  Used with the `-remove` option to remove attribute types (and all instances of that type) attached to the group.

See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information about ERAs.

The `modify` operation changes attributes of principals. The `principal_name_list` argument is a list of names of principals to be operated on. All modifications are applied to all principals named in the argument. Principals are modified in the order they are listed, and all modifications to an individual principal are atomic. Modifications to multiple principals are not atomic. A failure for any one principal in a list generates an error and aborts the operation. This operation returns an empty string on success.

The `-change` option can be used to modify the value of any of the attributes except for `uid` and `uuid`. The value of the `-change` option is an attribute list describing the new values. The `-change` option also supports the following attribute options:

- `-alias`, `-quota`, and `-fullname`. Specify only ERA attributes by using the `-add` and `-remove` options.
Privileges Required

You must have r (read) permission to the principal to be modified and f (full name) permission to change the principal’s full name and/or m (mgmt_info) permission to change the principal’s management information.

Examples

dcecp> principal modify /.:/joe -fullname "Joe Long"
dcecp> principal show /.:/joe
{fullname {Joe Long}}
{uid 30014}
{uuid 0000753e-f51f-2e0e-b000-0000c08adf56}
{alias no}
{quota unlimited}
dcecp>

dcecp> principal modify joe -add {test_era 101}
dcecp>

dcecp> principal modify joe -add {test_era 101}
{fullname {Joe Long}}
{uid 30014}
{uuid 0000753e-f51f-2e0e-b000-0000c08adf56}
{alias no}
{quota unlimited}
{test_era 101}
dcecp>

principal operations

Returns a list of the operations supported by the principal object. The syntax is as follows:

principal operations

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the principal operations command.

Examples

dcecp> principal operations
catalog create delete modify rename show help operations
dcecp>

principal rename

This operation changes the name of a specified principal. The syntax is as follows:

principal rename  principal_name  -to  new_principal_name

Options

-to  new_principal_name
   Specifies the new name of the principal.
principal(8dce)

The **rename** operation changes the name of a specified principal. The argument is a single name of a principal to be renamed. The required `-to` option specifies the new name, which cannot be a list. This operation returns an empty string on success.

**Privileges Required**

You must have **r** (read) and **n** (name) permission to the registry object for the specified principal.

**Examples**

```
dcecp> principal rename K_Doe -to K_Smith
```

```
dcecp> principal list K_Doe
Error: Registry object not found
```

**principal show**

Shows registry information for the specified principals. The syntax is as follows:

```
principal show  principal_name_list [-all | -xattrs]
```

**Options**

- **-xattrs**
  
  Returns only the ERAs of the principal, with no other attributes.

- **-all**
  
  Return the attributes followed by the ERAs.

The **show** operation returns an attribute list describing the specified principals. The `principal_name_list` argument is a list of names of principals to be operated on. If more than one principal is given, the attributes are concatenated and a blank line inserted between principals. There is one attribute in addition to `fullname`, `uid`, `uuid`, `alias`, and `quota`. It is called **groups** and its value is a list of the group names that the principal is a member of. Attributes are returned in the following order: `fullname`, `uid`, `uuid`, `alias`, and `quota`, followed by **groups**.

If called with the **-xattrs** option, then ERAs are returned instead of the above attributes. If called with **-all**, both are returned.

**Privileges Required**

You must have **r** (read) permission to the specified principals.

**Examples**

```
dcecp> principal show ./:/joe
{fullname {Joe Long}}
{uid 30014}
{uuid 0000753e-f51f-2e0e-b000-0000c08adf56}
{alias no}
{quota unlimited}
{groups none gumby}
```

IBM DCE Version 3.1 for AIX and Solaris: Administration Commands Reference
Related Information

Commands: account(8dce), dcecp(8dce), group(8dce), organization(8dce), xattrschema(8dce), registry(8dce).

principal(8dce)
registry(8dce)

registry

Purpose

A dcecp object that manages a registry in the DCE Security Service

Synopsis

registry catalog [registry_replica_name] [-master]

registry checkpoint registry_replica_name [ -at hh:mm | -cpi { num | num | m | num | h }] [-now]

registry connect cell_name -group local_group_name -org local_org_name
-mypwd local_password -fgroup foreign_group_name -forg foreign_org_name
-facct foreign_account_name -facctpwd foreign_account_password [-expdate]
[-acctvalid] [-facctvalid]

registry delete registry_replica_name [-force]

registry designate registry_replica_name [-slave | -master | [-force]]

registry destroy registry_replica_name

registry disable [registry_replica_name]

registry dump [registry_replica_name]

registry enable [registry_replica_name]

registry help [operation | -verbose]

registry modify [registry_replica_name] { -change attribute_list | -attribute value | -key }

registry operations

registry replace registry_replica_name -address new_string_binding

registry show [registry_replica_name] [-attributes | -policies | -master | -replica | [-verbose]]

registry stop registry_replica_name

registry synchronize registry_replica_name

registry verify [registry_replica_name]

Arguments

cell_name

The name of a cell to contact when processing the connect operation. The name must be a fully qualified cell name, such as /.: or /.../cell_name.
**operation**
The name of the `registry` operation for which to display help information.

**registry_replica_name**
The name of one registry replica to act on. The replica can be a master or a slave replica. The argument, which overrides a value in the `_s(sec)` convenience variable, can be one of the following:

1. A specific cell name to bind to any replica in the named cell, such as `./:` or `./.../gumby1`.
2. The global name of a replica to bind to that specific replica in that specific cell, such as `./.../gumby1/subsys/dce/sec/oddball`.
3. The name of a replica as it appears on the replica list to bind to that replica in the local cell, such as `subsys/dce/sec/oddball`.
4. A string binding to a specific replica, such as `{ncadg_ip_udp 15.22.144.163}`.
   This form is used primarily for debugging or if the Cell Directory Service (CDS) is not available.

   For those operations for which `registry_replica_name` is optional, the value of `_s(sec)` is used if no argument is given. If the variable is not set, the default argument of `./:` is assumed.

**Description**

The `registry` object represents a DCE Security Service registry. The registry is a replicated database: each instance of a registry server, `secd`, maintains a working copy of the database in virtual memory and on disk. One server, called the master replica, accepts updates and handles the subsequent propagation of changes to all other replicas. All other replicas are slave replicas, which accept only queries. Each cell has one master replica and can have numerous slave replicas.

Note that the `registry` command cannot add, delete, or modify information in the registry database, such as names and accounts. Use the appropriate `account`, `principal`, `group`, or `organization` command to modify registry database entries.

Two access control lists (ACLs) control access to `registry` operations. For operations dealing with replication, the `replist` object’s ACL (usually `./sec/replist`) controls access. For those that deal with registry attributes and policies, the `policy` object’s ACL (usually `./sec/policy`) controls access.

When this command executes, it attempts to bind to the registry server identified in the `_s(sec)` variable. If that server cannot process the request or if the `_s(sec)` variable is not set, the command binds to either an available slave server or the master registry server, depending on the operation. Upon completion, the command sets the `_b(sec)` convenience variable to the name of the registry server to which it bound.

**Attributes**

The `registry` object supports the following kinds of attributes:

1. **Registry attributes**—These modifiable attributes apply to principals, groups, organizations, and accounts. The initial values for some of these attributes must be specified when the master Security Server is configured.
2. **Registry-wide policy attributes** — These modifiable attributes apply to organizations and accounts. The registry-wide organization and account policy overrides the policy set for individual accounts only if the registry-wide policy is more restrictive.

3. **Synchronization attributes** — These read-only attributes are maintained by each replica about itself. They cannot be directly modified. These attributes have no default value, but are computed when the replica is configured.

4. **Replica-specific attributes** — These read-only attributes are kept by the master replica for each slave replica. They cannot be modified directly. These attributes have no default value, but are computed or assigned when the replica is configured.

**Registry Attributes**

- **deftktlife** `relative_time`
  
  The default lifetime for tickets issued to principals in this cell's registry. Specify the time by using the Distributed Time Service (DTS) relative time format ([-] `DD- hh: mm: ss`). The default is 
  
  `+0-10:00:00.000`

- **hidepwd** `(yes | no)`
  
  Determines whether encrypted passwords are displayed. If this attribute is set to `yes`, an asterisk is displayed in place of the encrypted password in command output and files where passwords are displayed. The value is either `yes` or `no`. The default is `yes`.

- **maxuid** `integer`
  
  The highest number that can be supplied as a user identifier (`uid`) when principals are created. This maximum applies to both the system-generated and user-entered `uid` s. The value is an integer; the initial value depends on the configuration of your system.

- **mingid** `integer`
  
  The starting point for group identifiers (`gid`s) automatically generated when a group is created. You can explicitly enter a lower `gid` than this number; it applies only to automatically generated numbers. The value is an integer; the initial value depends on the configuration of your system.

- **minorgid** `integer`
  
  The starting point for organization identifiers (`orgid`s) automatically generated when an organization is created. This starting point applies only to automatically generated identifiers. You can manually specify an identifier lower than the `minorgid`. The value is an integer; the initial value depends on the configuration of your system.

- **mintktlife** `relative_time`
  
  The minimum amount of time before the principal's ticket must be renewed. The value is an integer. This renewal is performed automatically with no intervention on the part of the user. The shorter this time is, the greater the security of the system. However, extremely frequent renewal can degrade system performance. Both system performance and the level of security required by the cell should be taken into consideration when selecting the value of this attribute. This is a registry-wide value only; it cannot be set for individual accounts. The default is
  
  `+0-00:05:00.000`
**minuid** integer
The starting point for **uid** s automatically generated when a principal is created. This starting point applies only to automatically generated identifiers. You can manually specify an identifier lower than the **minuid**. The value is an integer; the initial value depends on the configuration of your system.

**version**
The version of the security server software. The initial value depends on the configuration of your system.

### Registry-wide Policy Attributes

**acctlife** { relative_time| unlimited}
This registry-wide organization policy defines the lifespan of accounts. Specify the time by using the DTS-relative time format ([-] DD- hh: mm: ss) or the string **unlimited** to define an unlimited lifespan for accounts. The default is **unlimited**.

**maxtktlife** relative_time
This registry-wide account policy defines the maximum amount of time that a ticket can be valid. Specify the time by using the DTS-relative time format ([-] DD- hh: mm: ss). When a client requests a ticket to a server, the lifetime granted to the ticket takes into account the **maxtktlife** set for both the server and the client. In other words, the lifetime cannot exceed the shorter of the server’s or client’s **maxtktlife**. If you do not specify a **maxtktlife** for an account, the **maxtktlife** defined as registry authorization policy is used. The default is

\[+1-00:00:00\.000\]

**maxtktrenew** relative_time
This registry-wide account policy defines the amount of time before a principal’s ticket-granting ticket expires and that principal must log in again to the system to reauthenticate and obtain another ticket-granting ticket. Specify the time by using the DTS-relative time format ([-] DD- hh: mm: ss). The lifetime of the principal’s service tickets can never exceed the lifetime of the principal’s ticket-granting ticket. The shorter you make ticket lifetimes, the greater the security of the system. However, since principals must log in again to renew their ticket-granting ticket, the time specified needs to balance user convenience against the level of security required. If you do not specify this attribute for an account, the **maxtktrenew** lifetime defined as registry authorization policy is used. The default is

\[+28-00:00:00\.000\]

This feature is not currently used by DCE; any use of this option is unsupported at the present time.

**pwdalpha** {yes | no}
This registry-wide organization policy defines whether passwords can consist entirely of alphanumeric characters. Its value is either **yes** or **no**. The default is **yes**.

**pwdexpdate** {ISO-timestamp | none}
This registry-wide organization policy defines a date on which a password expires. The date is entered as an internationalized date string or the string **none**, in which case there is no expiration date for the password. The default is **none**.
pwdlife \{relative_time\ | unlimited\}
This registry-wide organization policy defines the lifespan of passwords.
Specify the time by using the DTS-relative time format ([\-] DD- hh: mm: ss) or the string unlimited. The default is unlimited.

pwdminlen integer
This registry-wide organization policy defines the minimum number of characters in a password. Its value is a positive integer or the integer 0, which means there is no minimum length. The default is 0.

pwdspaces \{yes | no\}
This registry-wide organization policy defines whether passwords can consist entirely of spaces. Its value is either yes or no. The default is no.

Synchronization Attributes
name The name of the replica. It is in the form of a fully qualified CDS name.
type Indicates if the replica is a master or a slave.
cell The name of the cell that the replica is in. It is a fully qualified cell name.
uuid The Universal Unique Identifier (UUID) of the replica.
status The state of the replica. One of the following:
becomingmaster The replica is in the process of becoming a master.
becomingslave The replica is a master in the process of becoming a slave.
changingkey The replica is in the process of having its master key changed.
closed The replica is in the process of stopping.
copyingdb The replica is in the process of initializing (copying its database to) another replica.
deleted The replica is in the process of deleting itself.
disabled The replica is unavailable for updates, but will accept queries.
dupmaster Two masters have been found in the cell, and the replica is a duplicate of the real master.
enabled The replica is available for use.
initializing The replica is in the process of being initialized by the master replica or another up-to-date replica.
savingdb The replica is in the process of saving its database to disk.
unavailable The replica cannot be reached.
The database is a stub database that has not been initialized by the master replica or another up-to-date replica.

The replica is not known to the master.

The localized date and time that the master received the last replica’s last update.

The sequence number of the last update the replica received. A sequence number consists of two 32-bit integers separated by a dot (high.low). The high integer increments when the low integer wraps. An example of this attribute is

\{lastupdseq 0.178\}.

A list of the network addresses of the replica. There can be more than one for connectionless and connection-oriented protocols.

The network address of the master replica as determined by the replica. The address is not necessarily correct. More than one address can exist for connectionless and connection-oriented protocols for example.

The master sequence number, which is the sequence number of the event that made the replica the master as determined by the replica. The number is not necessarily correct. A sequence number consists of 32-bit integers separated by a dot (high.low). The high integer increments when the low integer wraps. An example of this attribute is

\{masterseqnum 0.100\}.

The UUID of the master replica as determined by the replica. This UUID is not necessarily correct. The value is a UUID.

DCE registry version supported by the security service. The possible values at DCE Version 3.1 are

\{secd.dce.1.0.2\} (DCE Version 1.3), at is

\{secd.dce.1.1\} (DCE Version 2.1), and \{secd.dce.1.2.2\} (DCE Version 2.2). All versions can be supported (that is by a DCE Version 3.1 security server running in a cell with DCE version 2.2, 1.3, or 1.1 replicas).

A list of two update sequence numbers that are still in the propagation queue and have yet to be propagated. The first number is the base propagation sequence number (the last number known to have been received by all replicas). The second number is the sequence number of the last update made on the master. This attribute is present only in the master replica. The sequence numbers consist of two 32-bit integers separated by a dot (high.low). The high integer increments when the low integer wraps. An example of this attribute is

\{updseqqueue \{0.100 0.178\}\}.

The name of the replica. It is in the form of a fully qualified CDS name.

The UUID of the replica.

Indicates if the replica is a master or a slave.
addresses
A list of the network addresses of the replica. More than one address can exist for connectionless and connection-oriented protocols.

propstatus
The status of the propagation. Possible values are as follows:

- delete The replica is marked for deletion.
- initmarked The replica is marked for initialization.
- initing The replica is in the process of initialization, that is, getting an up-to-date copy of the registry.
- update The replica is ready to receive propagation updates.

lastupdtime
The localized time of the last update sent to the replica. This information is meaningful only if propstatus is update.

lastupdseqsent
The sequence number of the last update sent to this replica. A sequence number consists of two 32-bit integers separated by a dot (high.low). The high integer increments when the low integer wraps. An example of this attribute is

\{lastupdseqsent 0.175\}

This information is meaningful only if propstatus is update.

numupdtogo
The number of outstanding updates. The value is an integer. This information is meaningful only if propstatus is update.

commstate
The state of the last communication with the replica.

lastcommstatus
The status message of the last communication with the replica.

See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information about attributes, policies, and synchronizations.

Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Operations

registry catalog
Returns a list of the names of the security servers running in the cell. The syntax is as follows:

registry catalog [registry_replica_name] [-master]
-master

Returns only the master security server name.

The catalog operation returns a list of the names of the security servers (that is, each copy of the registry) running in the cell. This is also known as the replica list. The order of elements returned is arbitrary. The optional registry_replica_name argument can specify the name of one other cell or a single string binding. If you specify the -master option, the operation returns only the name of the master.

This operation sets the _b(sec) variable to the name of the replica to which it binds.

Privileges Required

No special privileges are needed to use the registry catalog command.

Examples

dcecp> registry catalog
/.../dcecp.cell.osf.org/subsys/dce/sec/snow
/.../dcecp.cell.osf.org/subsys/dce/sec/ice
dcecp>

registry checkpoint

Specifies when registry checkpoints should be performed. The syntax is as follows:

registry checkpoint registry_replica_name [-at hh:mm | -cpi {num | numm | numh}] [-now]

Options

- at hh:mm
   Specifies the hours and minutes of the day (in UTC time) to perform the checkpoint.

- cpi { num | numm | numh }
   Specifies an interval at which to perform checkpoints.

- now
   Specifies an immediate checkpoint. This is the default.

The checkpoint operation lets you set the times when the registry database should be saved to disk (checkpointed). You must supply the name of a replica for the operation to bind to.

If you use the -at option, the checkpoint is performed at the specified time. The time is in UTC format. For example, to specify 3:30 p.m., the entry is 15:30. The checkpoint interval then reverts to the default or to the interval specified by the -cpi option.

If you use the -cpi option, the checkpoint is performed at the interval you specify until you specify another interval. This option takes an argument that specifies the interval time as seconds, minutes, or hours:

1. To specify seconds, supply only a number. For example, -cpi 101 specifies an interval of 101 seconds.
2. To specify minutes enter the number and m. For example, -cpi 101m specifies an interval of 101 minutes.
3. To specify hours, enter the number and h. For example, -cpi 101h specifies an interval of 101 hours.

If you use the -now option, a checkpoint is performed immediately. The checkpoint interval then reverts to the default or to the interval specified by the -cpi option. This operation returns an empty string on success and sets the _b(sec) variable to the replica to which it binds.

**Privileges Required**

You must have ad (auth_info, delete) permission to the replist object.

**Examples**

dcecp> registry checkpoint /.../gumby_cell/subsys/dce/sec/oddball -at 05:30
dcecp>

**registry connect**

Connects the local (that is, default) cell of the local host to the foreign cell specified by the argument. The syntax is as follows:

```
registry connect  cell_name  -group  local_group_name  -org  local_org_name
-mypwd  local_password  -fgroup  foreign_group_name  -forg  foreign_org_name  -facct
foreign_account_name  -facctpwd  foreign_account_password  [-expdate]  [-acctvalid]  [-facctvalid]
```

**Options**

- **-group local_group_name**
  Specifies the group for the local account.

- **-org local_org_name**
  Specifies the organization for the local account.

- **-mypwd local_password**
  Specifies the password for the administrator in the local cell.

- **-fgroup foreign_group_name**
  Specifies the group for the foreign account.

- **-forg foreign_org_name**
  Specifies the organization for the foreign account.

- **-facct foreign_account_name**
  Specifies the name for the foreign account.

- **-facctpwd foreign_account_password**
  Specifies the password for the administrator in the foreign cell.

- **-expdate account_expiration_date**
  Sets an expiration date for both local and foreign accounts.

- **-acctvalid**
  Marks the local account as a valid account. A valid local account allows users from the foreign cell to log in to nodes in the local cell. The default is invalid.
-facctvalid
Marks the foreign account as a valid account. A valid foreign account allows
users from the local cell to log in to nodes in the foreign cell. The default is
invalid.

The connect operation creates an account in the local cell for the specified foreign
cell (/./local_cell/sec/principal/krbtgt/foreign_account) and also creates an
account in the foreign cell for the local cell (/./
foreign_cell/sec/principal/krbtgt/local_account). Both accounts have the same key.
The argument must be the fully qualified name of a single cell. It cannot be a list or
a string binding.

The -group, -org, -mypwd, and -acctvalid options supply the account information
for the local cell. The -fgroup, -forg, -facct, -facctpwd, and -facctvalid options
supply the account information for the foreign cell.

This operation creates the group and organization, specified as the values of the
relevant options, if necessary, and puts the relevant principal in them, if necessary.

If the operation fails, it removes any organization, group, or both that it has created
and removes the relevant principals. To protect the password being entered, the
registry connect command can be entered only from within dcecp. You cannot
enter it from the operating system prompt by using dcecp with the -c option.

If you do not use the -acctvalid and -facctvalid options, you must mark the
accounts as valid (using the dcecp account command) before intercell access is
allowed. This operation returns an empty string on success.

Privileges Required
You must have a (auth_info) permission to the replist object and the permissions
required to create principals, groups, organizations, and accounts in the local and
foreign cells.

Examples

dcecp> getcellname
/.../my_cell.com
dcecp>

dcecp> registry connect /.../your_cell.com -group none -org none \ > -mypwd -dce- -fgroup none -forg none -facct cell_admin \ > -facctpwd -dce-
dcecp>

registry delete

Deletes a registry replica from the cell. The syntax is as follows:

registry delete  registry_replica_name
[-force]

Option

-force  Used when the target replica is not available, the -force option removes the
replica name from the master replica’s replica list and propagates the
deletion to other replicas that remain on the list.
The `registry delete` operation, when called with no options, performs an orderly deletion of a security replica specified as the `registry_replica_name` argument. To do so, the operation binds to the master replica. The master replica then performs the following tasks:

1. Marks the specified replica as deleted
2. Propagates this deletion to the other replicas on its replica list
3. Delivers the delete request to the specified replica
4. Removes the replica from its replica list

Note that the `dcecp` command returns before the deletion is complete because it simply tells the master to perform the delete procedure.

The `-force` option causes a more drastic deletion. It causes the master to first delete the specified replica from its replica list and then propagate the deletion to the replicas that remain on its list. Since this operation never communicates with the deleted replica, you should use `-force` only when the replica has died and cannot be restarted. If you use `-force` while the specified replica is still running, you should then use the `registry destroy` command to eliminate the deleted replica.

This operation returns an empty string on success and sets the `_b(sec)` variable to the master.

**Note:** Although the replica is deleted, its local configuration information remains in place on the host where the replica resided. You must clean up its configuration before you can configure another replica on that host. To do this, execute the `unconfig.dce -config_type local sec_rep` command on the host.

**Privileges Required**

You must have `d (delete)` permission to the `replist` object.

**Examples**

```
dcecp> registry delete /./subsys/dce/sec/ oddball
```

**registry designate**

Changes which replica is the master. The syntax is as follows:

```
registry designate registry_replica_name [-slave | -master [-force]]
```

**Options**

- **-slave** Makes the specified replica a slave. The `registry_replica_name` argument must identify the master replica.

- **-master** Makes the specified replica the master. The `registry_replica_name` argument must identify a slave replica.

- **-force** Forces `registry_replica_name` to become the master, even if other slave replicas are more up to date. Used only with the `-master` option.

The preferred method of creating a new master is to use this command with no options in this form:
**registry designate registry_replica_name**

This command changes the slave replica named in `registry_replica_name` to the master by performing an orderly transition. To do so, it binds to the current master and instructs the master to:

1. Apply all updates to the replica named in `registry_replica_name`
2. Become a slave
3. Tell the replica named in `registry_replica_name` to become the master

The `-slave` or `-master` options can also be used to change the master to a slave and a slave to a master. However, using these options is not recommended because updates can be lost. You should use them only if you must because the master replica is irrevocably damaged and is unable to perform the steps in the orderly transition. To use these options, enter the command as shown in the following list:

1. To make the master a slave:

   ```
   registry designate registry_replica_name -slave
   ```

   The `registry_replica_name` is the name of the replica to make a slave.

2. To make a slave the master:

   ```
   registry designate registry_replica_name -master
   ```

   The `registry_replica_name` is the name of a slave to make a master. If a master exists, the command fails. Also, if there are more up-to-date slaves than the one specified by `registry_replica_name`, the command fails unless you specify `-force` to override this default action.

This operation returns the empty string on success and sets the `_b(sec)` variable as follows:

1. If called with the `-force` or `-master` option, it sets `_b(sec)` to the replica to which it binds.
2. If called with no options, it sets `_b(sec)` to the master.

**Note:** After using the `dcecp registry designate` command, be sure to do the following:

- Update `pe_site` files
- Update `/krb5/krb.conf`
- Refresh cds cache on all machines
- Refresh `rpcccp` cache

```
rpcccp show group /.:/sec -u
rpcccp show group /.:/sec-v1 -u
``` 

**Privileges Required**

You must have a `auth_info` permission to the `replist` object.

**Examples**

```
dcecp> registry designate /.../my_cell/subsys/dce/sec/oddball
dcecp>
```
registry destroy

Deletes a registry replica. The syntax is as follows:

```
registry destroy registry_replica_name
```

The `destroy` operation causes the replica named in `registry_replica_name` to delete its copy of the registry database and to stop running.

The preferred way to delete replicas is to use the `delete` operation. However, the `destroy` operation can be used if `delete` is unusable because the master is unreachable or the replica is not on the master's replica list.

This operation returns an empty string on success and sets the `_b(sec)` variable to the replica to which it binds.

**Privileges Required**

You must have `d (delete)` permission to the `replist` object.

**Examples**

```
dcecp> registry destroy ././subsys/dce/sec/oddball
```

registry disable

Disables the master registry for updates. The syntax is as follows:

```
registry disable [registry_replica_name]
```

The `disable` operation disables the master registry for updates. Generally, use this mode for maintenance purposes. The `registry_replica_name` argument is a single name of a master registry to be disabled. If no argument is given, the operation uses the name in the `_s(sec)` convenience variable. If the `_s(sec)` variable is not set, the operation defaults to the master in the local cell.

This operation returns an empty string on success and sets `_b(sec)` to the name of the replica to which it binds.

**Privileges Required**

You must have `A (admin)` permission to the `replist` object.

**Examples**

```
dcecp> registry disable /.../my_cell.goodcompany.com/subsys/dce/sec/snow
```

registry dump

Returns the replica information for each replica in the cell. The syntax is as follows:

```
registry dump [registry_replica_name]
```

The `dump` operation returns the replica information for each replica in the cell. Replicas are displayed with a blank line between them.
The \textbf{registry dump} command is the same as the following script:


def registry_dump:
    r = []
    for i in registry_catalog:
        r += registry_show(i, -replica)
    return r

This operation sets the \texttt{\_b(sec)} variable to the last replica listed in the display.

\textbf{Privileges Required}

You must have \texttt{A (admin)} permission to the \texttt{replist} object.

\textbf{Examples}

```
dcecp> registry dump
{name \ldots/dcecp.cell.osf.org/subsys/dce/sec/snow}
{type master}
{cell \ldots/dcecp.cell.osf.org}
{uuid a1248a5e-e1e6-11cd-aa0c-0800092734a4}
{status enabled}
{lastupdatetime 1994-10-13-14:44:48.000-04:00-000}
{lastupdseq 0.271}
{addresses
  {ncacn\_ip\_tcp 130.105.5.121}
  {ncadg\_ip\_udp 130.105.5.121}}
{masteraddrs
  {ncacn\_ip\_tcp 130.105.5.121}
  {ncadg\_ip\_udp 130.105.5.121}}
{masterseqnum 0.100}
{masteruuid a1248a5e-e1e6-11cd-aa0c-0800092734a4}
{version secd.dce.1.1}
{updseqqueue {0.204 0.271}}
{name \ldots/dcecp.cell.osf.org/subsys/dce/sec/ice}
{type slave}
{cell \ldots/dcecp.cell.osf.org}
{uuid c77f46a-e1ec-11cd-9a16-0800092734a4}
{status enabled}
{lastupdatetime 1994-10-13-14:44:48.000-04:00-000}
{lastupdseq 0.271}
{addresses
  {ncacn\_ip\_tcp 130.105.5.45}
  {ncacn\_ip\_tcp 130.105.5.45}
  {ncadg\_ip\_udp 130.105.5.45}}
{masteraddrs
  {ncacn\_ip\_tcp 130.105.5.121}
  {ncadg\_ip\_udp 130.105.5.121}}
{masterseqnum 0.100}
{masteruuid a1248a5e-e1e6-11cd-aa0c-0800092734a4}
{version secd.dce.1.1}
dcecp>
```

\textbf{registry enable}

Enables the master registry for updates. The syntax is as follows:

\textbf{registry enable} \ [\texttt{registry\_replica\_name}]

The \texttt{enable} operation enables the master registry for updates. The \texttt{registry\_replica\_name} argument is a single name of a master registry to be
enabled. If no argument is given, the operation uses the name in the _s(sec) convenience variable. If the _s(sec) variable is not set, the operation defaults to the master in the local cell.

This operation returns an empty string on success and sets the _b(sec) variable to the replica to which it binds.

**Privileges Required**

You must have A (admin) permission to the replist object.

**Examples**

dcecp> registry enable /.../my_cell.goodcompany.com/subsys/dce/sec/snow
dcecp>

**registry help**

Returns help information about the registry object and its operations. The syntax is as follows:

```
registry help [operation | -verbose]
```

**Options**

- **-verbose**
  
  Displays information about the registry object.

Used without an argument or option, the registry help command returns brief information about each registry operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the -verbose option for more detailed information about the registry object itself.

**Privileges Required**

No special privileges are needed to use the registry help command.

**Examples**

dcecp> registry help
catalog Returns a list of all replicas running in the cell.
checkpoint Resets registry checkpoint interval dynamically.
connect Creates local and foreign cross-cell authenticated accounts.
delete Deletes a replica and removes from master replica list.
designate Changes which replica is the master.
destroy Destroys the specified replica and its registry database.
disable Disables the specified master registry for updates.
dump Returns replica information for each replica in the cell.
enable Enables the specified master registry for updates.
modify Modifies the master registry or replica.
replace Replaces replica information on master replica list.
show Returns attributes of the registry and its replicas.
stop Stops the specified security server process.
synchronize Reinitializes replica with up-to-date copy of the registry.
verify Returns a list of replicas not up-to-date with the master.
help Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
registry modify

Changes attributes of the registry. The syntax is as follows:

```
registry modify [registry_replica_name] {-change attribute_list | -attribute value | -key}
```

**Options**

* `attribute value`  
  As an alternative to using options with an attribute list, you can change individual attribute options by prepending a hyphen (-) to any attributes listed in ATTRIBUTES.

* `-change attribute_list`  
  Allows you to modify attributes by using an attribute list rather than using individual attribute options. The format of an attribute list is as follows:

  ```
  {{attribute value}...{attribute value}}
  ```

  The `-change` option cannot be used with the `-key` option.

* `-key`  
  Generates a new master key for the replicas listed as the argument. Cannot be used with the `-change` option.

The `modify` operation changes attributes of the registry. The `registry_replica_name` is required for the `-key` option but optional for all other options. If an argument is not supplied and the `_s(sec)` variable is not set, the operation defaults to master in the local cell. This operation returns an empty string on success.

Use the `-change` option to modify the value of any one of the attributes.

The operation also accepts the `-key` option to generate a new master key for a single replica named in the argument and to reencrypt that registry’s account keys using the new master key. The new master key is randomly generated. Each replica (master and slaves) maintains its own master key, which is used to access the data in its copy of the database. If you use the `-key` option, you must specify `registry_replica_name`.

The `-change` option and the `-key` option cannot be used together.

This operation sets the `_b(sec)` variable to the replica to which it binds.

**Privileges Required**

You must have A (admin) permission to the replist object.

**Examples**

```
dcecp> registry modify -version secd.dce.1.1
dcecp>
dcecp> registry modify -change {deftktlife +0-08:00:00.000I-----}
dcecp>
```
registry operations

Returns a list of the operations supported by the registry object. The syntax is as follows:

registry operations

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the registry operations command.

Examples

dcecp> registry operations
catalog checkpoint connect delete designate destroy disable dump enable modify replace show stop synchronize verify help operations
dcecp>

registry replace

Replaces the network address of a replica. The syntax is as follows:

registry replace  registry_replica_name -address new_string_binding

Options

-address
  The new address for the replica in RPC string-binding format (without the object UUID). The string binding contains an RPC protocol and a network address in the form:

  rpc_prot_seq:network_addr

The replace operation replaces the network address of the specified replica. The new address is used by the master and other replicas to contact the replica. This operation binds to the master, sets the _b(sec) variable to the master, and returns an empty string on success.

Privileges Required

You must have m (mgmt_info) permission to the replist object.

Examples

dcecp> registry replace /./susbs/dce/sec/maria -address ncadg_ip_udp:15.22.4.93
dcecp>

registry show

Returns information about the registry and its replicas. The syntax is as follows:

registry show  [registry_replica_name] [-attributes -policies -master -replica [-verbose]]
Options
- **attributes**
  Returns an attribute list of the registry-wide attributes.
- **policies**
  Returns only the registry-wide policies.
- **replica**
  Returns the synchronization information for the specified replica.
- **master**
  Returns the synchronization information kept by the master keeps for each slave.
- **verbose**
  Returns the synchronization information kept by the replica.

The `show` operation returns information about the registry and its replicas. An optional `registry_replica_name` argument specifies a single registry replica to contact. The operation returns a variety of different information based on the option given.

If called with no options or with the **attributes** option, the operation returns an attribute list of all the registry-wide attributes.

If called with the **policies** option, the operation returns an attribute list of all the registry-wide polices.

If called with the **replica** option, the operation returns the propagation information that is kept by the replica specified.

If called with the **master** option, the operation returns the propagation information that is kept by the master for each slave. Use the **verbose** option to return the propagation information that is kept by the replica. If you specify this option and the optional `registry_replica_name`, `registry_replica_name` must specify the name of the master or the local cell name.

This operation sets the `_b(sec)` variable to the replica to which it binds.

**Privileges Required**

You must have A (admin) permission to the `replist` object.

**Examples**

dcecp> registry show -attributes
  {mingid 31000}
  {minorgid 100}
  {minuid 30000}
  {maxuid 32767}
  {version secd.dce.1.0.2}
dcecp>

dcecp> registry show -policies
  {deftktlife +0-10:00:00.000I-----}
  {mintktlife +0-00:05:00.000I-----}
  {hidepwd yes}
dcecp>

dcecp> registry show /.../absolut_cell/subsys/dce/sec/ice -replica
registry(8dce)

{name /.../absolut_cell/subsys/dce/sec/ice}
{type slave}
{cell /.../absolut_cell}
{uuid 91259b6c-9415-11cd-a7b5-080009251352}
{status enabled}
{lastupdttime 1994-07-05-14:38:15.000-04:00I-----}
{lastupdseq 0.191}
{addresses
 {ncacn_ip_tcp 130.105.5.93}
 {ncadg_ip_udp 130.105.5.93}}
{masteraddrs
 {ncacn_ip_tcp 130.105.5.93}
 {ncadg_ip_udp 130.105.5.93}}
{masterseqnum 0.100}
{masteruuid 91259b6c-9415-11cd-a7b5-080009251352}
{supportedversions secd.dce.1.0.2}
{updseqqueue {0.187 0.191}}
dcep>
dcep> registry show /.../dcep.cell.osf.org/subsys/dce/sec/snow -master
{name /.../dcep.cell.osf.org/subsys/dce/sec/snow}
{uuid 91259b6c-9415-11cd-a7b5-080009251352}
{type master}
{addresses
 {ncacn_ip_tcp 130.105.5.93}
 {ncadg_ip_udp 130.105.5.93}}
{name /.../dcep.cell.osf.org/subsys/dce/sec/ice}
{uuid 91259b6c-9415-11cd-a7b5-080009251352}
{type slave}
{addresses
 {ncacn_ip_tcp 130.105.5.93}
 {ncadg_ip_udp 130.105.5.93}}
{propstatus update}
{lastupdttime 1994-10-13-14:58:28.000-04:00I-----}
{lastupdseqsent 0.528}
{numupdtogo 0}
{commstate ok}
{lastcommstatus {successful completion}}
dcep>

registry stop

Stops the specified security server process. The syntax is as follows:

registry stop registry_replica_name

The stop operation stops the security server specified in the argument. The registry_replica_name argument is required and must explicitly name one replica. (A cell name is not valid because more than one replica can operate in a cell.) This operation returns an empty string on success and sets the _b(sec) variable to the replica to which it binds.

Privileges Required

ou must have A (admin) permission to the replist object.

Examples

dcep> registry stop /..:/subsys/dce/sec/snow

dcep>
registry synchronize

Causes the specified replica to reinitialize itself with an up-to-date copy of the database. The syntax is as follows:

registry synchronize registry_replica_name

The synchronize operation reinitializes a slave replica with an up-to-date copy of the database. registry_replica_name is the name of the slave replica to operate on.

This operation binds to the master and tells the master to:
1. Mark the specified replica named in registry_replica_name for reinitialization.
2. Send a message to the replica informing it to reinitialize itself.
3. Gives the replica a list of other replicas with up-to-date copies of the registry.

The replica to be initialized then selects a replica from the list provided by the master and asks for a copy of the database. Note that the dcecp command returns before the synchronization is complete because it simply tells the master to perform the synchronize procedure.

Normally, you do not need to use the registry synchronize command because registries remain synchronized automatically. This operation returns an empty string on success.

This operation sets the _b(sec) variable to the master in the local cell.

Privileges Required
You must have A (admin) permission to the replist object.

Examples

dcecp> registry synchronize /.:/subsys/dce/sec/oddball

dcecp>

registry verify

Checks whether all registry replicas are up to date. The syntax is as follows:

registry verify [registry_replica_name]

Checks whether all registry replicas are up to date. If they are, it returns an empty string.

This operation sets the _b(sec) variable to the last replica to which it binds.

Privileges Required
You must have a (auth_info) permission to the replist object.

Examples

If the replicas are up to date, the command returns an empty string, as in the following:
registry(8dce)

dcecp> registry verify

dcecp>

If a replica is not up to date, the command returns the fully qualified replica name, as in the following:

dcecp> registry verify
/.../cell/subsys/dce/sec/oddball

dcecp>

Related Information

Commands: dcecp(8dce), group(8dce), organization(8dce), principal(8dce), secd(8sec).
rpcentry

Purpose

A dcecp object that manages an RPC entry in the DCE Cell Directory Service

Synopsis

rpcentry create entry_name_list

rpcentry delete entry_name_list

rpcentry export entry_name_list [-object object uuid_list] [-interface interface_id] [-binding string_binding_list]

rpcentry help [operation | -verbose]

rpcentry import entry_name_list -interface interface_id [-object object uuid] [-max integer] [-noupdate]

rpcentry operations

rpcentry show entry_name_list -interface interface_id_list [-object object uuid_list] [-noupdate]

rpcentry unexport entry_name_list [-object object uuid_list] [-interface interface_id] [-version versions]

Arguments

entry_name_list
Specifies a list of one or more names of the target name service entry. For an entry in the local cell, you can omit the cell name and specify only cell-relative names.

operation
The name of the rpcentry operation for which to display help information.

Description

The rpcentry object represents a remote procedure call (RPC) server entry in the cell name service. Use the rpcentry commands to create, modify, display, and delete name service entries.

interface_id
The interface identifier of an RPC interface. The interface identifier takes the following form:

interface-uuid, major-version. minor-version

The version numbers are optional, but if you omit a version number, the value defaults to 0. The UUID is a hexadecimal string and the version numbers are decimal strings. For example:

-interface ec1eeb60-5943-11c9-a309-08002b102989,3.11
Leading zeros in version numbers are ignored.

Alternatively, you can use `dcecp` string syntax in the following form:

```
{interface-UUID major-version.minor-version}
```

For example:

```
-interface {458ffcbe-98c1-11cd-bd93-0000c08ad56 1.0}
```

### string_binding_list

An RPC string binding that describes a server's location. The value has the form of an RPC string binding, without an object UUID. The binding information contains an RPC protocol, a network address, and (sometimes) an endpoint within `[ ]` (square brackets) as follows:

```
rpc-prot-seq: network-addr[ endpoint]
```

For a well-known endpoint, include the endpoint in the string binding surrounded by brackets. You might need to use the \ (backslash) to escape the brackets as shown in the following example. Otherwise `dcecp` interprets the brackets as enclosing another command.

```
-binding ncadg_ip_udp:63.0.2.17[5347]
```

For a dynamic endpoint, omit the endpoint from the string binding, for example:

```
-b ncacn_ip_tcp:16.20.15.25
```

Alternatively, you can use `dcecp` string syntax. For example:

```
-binding {ncacn_ip_tcp 130.105.1.227 1072}
```

### object_uuid

The UUID of an object. The UUID is a hexadecimal string, for example:

```
-object 3c6b8f60-5945-11c9-a236-08002b102989
```

Alternatively, you can use `dcecp` string syntax. For example:

```
-object {3c6b8f60-5945-11c9-a236-08002b102989}
```

### version

Specifies which interface version numbers should be returned by a `show` operation. Specify versions by using one of the following values for the `-version` option:

- **all** The interface version is ignored.
- **exact** Both the major and minor versions must match the specified versions.
- **compatible** The major version must match the specified version, and the minor version must be greater than or equal to the specified version.
**major_only**
The major version must match the specified version; the minor version is ignored.

**upto**
The major version must be less than or equal to that specified. If the major versions are equal, the minor version must be less than or equal to that specified.

If the -version option is absent, the command shows compatible version numbers.

**Errors**

A representative list of errors that might be returned is not shown here. Refer to the *IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide* for complete descriptions of all error messages.

**Operations**

**rpcentry create**

Creates an empty entry in the name service. The syntax is as follows:

```
rpcentry create entry_name_list
```

The create operation creates an empty entry in the name service. Since an empty entry is the same as an empty RPC group or RPC profile, calling rpcentry create is the same as calling rpcgroup create or rpcprofile create. The entry_name_list argument is a list of names of RPC entries to be created. If the RPC entry already exists, an error message is returned. This operation returns an empty string on success.

**Privileges Required**

To create an rpcentry, you need i (insert) permission to the parent directory and both r (read) permission and w (write) permission to the Cell Directory Service (CDS) object entry (the target name service entry).

**Examples**

The following command adds an unspecialized entry to the name service database:

```
dcecp> rpcentry create /.:/LandS/anthro/Cal_host_2
```

**rpcentry delete**

Removes the specified entry from the name service. The syntax is as follows:

```
rpcentry delete entry_name_list
```

The delete operation removes the specified entry from the name service. The entry_name_list argument is a list of one or more names of server entries to be deleted. This operation returns an empty string on success. If the entry does not exist, an error is returned.
Privileges Required

To delete an entry, you need r (read) permission to the CDS object entry (the target name service entry). You also need d (delete) permission to the CDS object entry or to the parent directory.

Examples

The following command removes the entry /.:LandS/anthro/Cal_host_2 from the local cell of the name service database:

dcecp> rpcentry delete /.:LandS/anthro/Cal_host_2
dcecp>

rpcentry export

Transfers information to the specified entry in the name service. The syntax is as follows:

```
rpcentry export entry_name_list {[-object object_uuid_list] [-interface interface_id -binding string_binding_list]}```

Options

- **-object object_uuid_list**
  Declares the UUID of an object. Accepts a list of up to 32 object UUIDs. The UUID is a hexadecimal string. See Data Structures for the format of the object UUID.

- **-interface interface_id**
  Declares the interface identifier of one RPC interface. If you specify an interface identifier, you must specify at least one -binding option.

  See Data Structures for the format of the interface identifier.

- **-binding string_binding_list**
  Declares a list of one or more protocol sequences (RPC bindings). To use this option, you must also use the -interface option to specify an interface identifier.

  See Data Structures for the format of a protocol sequence.

The export operation transfers information to the specified entry in the name service. The entry_name_list argument is a list of one or more names of server entries to be exported to. If an entry does not exist, it is created. Uses the -interface, -binding, and -object options to specify what to export. This operation returns an empty string on success.

Privileges Required

To export an entry, you need both r (read) permission and w (write) permission to the CDS object entry (the target name service entry). If the entry does not exist, you also need i (insert) permission to the parent directory.

Examples

The following example uses the dcecp string syntax to export an RPC entry to CDS:
rpcentry help

Returns help information about the rpcentry object and its operations. The syntax is as follows:

\texttt{rpcentry help [operation | -verbose]}

Options
- \texttt{-verbose}
  Displays information about the rpcentry object.

Used without an argument or option, the \texttt{rpcentry help} command returns brief information about each \texttt{rpcentry} operation. The optional \texttt{operation} argument is the name of an operation about which you want detailed information. Alternatively, you can use the \texttt{-verbose} option for more detailed information about the \texttt{rpcentry} object itself.

Privileges Required

No special privileges are needed to use the \texttt{rpcentry help} command.

Examples

\texttt{dcecp> rpcentry help}
create Creates a list of empty RPC entries.
delete Deletes a list of RPC entries.
export Stores bindings in a list of RPC entries.
import Returns the bindings from a list of RPC entries.
show Returns the attributes of a list of RPC entries.
unexport Deletes bindings from a list of RPC entries.
help Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.

\texttt{dcecp>}

rpcentry import

Returns a string binding from the specified RPC entry. The syntax is as follows:

\texttt{rpcentry import entry_name_list -interface interface_id [-object object_uuid] [-max integer] [-noupdate]}

Options
- \texttt{-interface interface_id}
  Declares the interface identifier of one RPC interface.
  See Data Structures for the format of the interface identifier.

- \texttt{-object object_uuid}
  Declares the UUID of one object. The UUID is a hexadecimal string.
  See Data Structures for the format of the object UUID.
**rpcentry(8dce)**

- **-max integer**
  Specifies the maximum number of string bindings to return. A value greater than one returns a list containing up to the number of bindings specified by the value.

- **-noupdate**
  Normally, name service data is cached locally on each machine in a cell. If a name service inquiry can be satisfied by data in the local CDS cache, this cached data is returned. However, locally cached copies of name service data might not include a recent CDS update. If the **-noupdate** option is not specified, **dcecp** goes to a CDS server to retrieve the required data, updating the local CDS cache. Use the **-noupdate** option to avoid taking the time to update the local cache when you have reason to believe that the local cache is up to date.

The **import** operation returns a string binding from the specified RPC entry. The **entry_name_list** argument is a list of names of RPC entries (not a list of RPC entries) to import from. The order of returned bindings is arbitrary.

**Privileges Required**

You need **r** (**read**) permission to the specified CDS object entry (the starting name service entry) and to any CDS object entry in the resulting search path.

**Examples**

The following command imports a binding:

```
dcecp> rpcentry import ./LandS/anthro/Cal_host_3 
> -interface {ec1eeb60-5943-11c9-a309-08002b102989 1.1} 
> -object 30dbeea0-fb6c-11c9-8eea-08002b0f4528 
{ncacn_ip_tcp 130.105.1.227}
```

**rpcentry operations**

Returns a list of the operations supported by the **rpcentry** object. The syntax is as follows:

```
rpcentry operations
```

The list of available operations is in alphabetical order except for **help** and **operations**, which are listed last.

**Privileges Required**

No special privileges are needed to use the **rpcentry operations** command.

**Examples**

```
dcecp> rpcentry operations
create delete export import show unexport help operations
```

**rpcentry show**

Returns a list containing the binding information in the specified RPC entries. The syntax is as follows:
rpcentry show  entry_name_list -interface
interface_id_list
[-object  object_uuid_list] [-noupdate]

Options

-interface interface_id_list
   Declares a list of one or more interface identifiers of RPC interfaces.
   See Data Structures for the format of the interface identifier.

-object object_uuid_list
   Declares the UUID of an object. Accepts a list of up to 32 object UUIDs.
   The UUID is a hexadecimal string.
   See Data Structures for the format of the object UUID.

-noupdate
   Normally, name service data is cached locally on each machine in a cell. If
   a name service inquiry can be satisfied by data in the local CDS cache, this
   cached data is returned. However, locally cached copies of name service
   data might not include a recent CDS update. If the -noupdate option is not
   specified, dcecp goes to a CDS server to retrieve the required data,
   updating the local CDS cache. Use the -noupdate option to avoid taking
   the time to update the local cache when you have reason to believe that
   the local cache is up to date.

The show operation returns a list containing the binding information in the specified
RPC entry. The entry_name_list argument is a list of one or more names of RPC
entries to return information about.

The returned list consists of two lists. Each item in the first list is also a list, the first
two elements of which are the interface identifier (the UUID and then the version),
and the remaining are string bindings in Tcl syntax. The second list is a list of object
UUIDs exported by the server. The order of the data returned is arbitrary.

Privileges Required

You need r (read) permission to the CDS object entry (the target name service
entry).

Examples

The following command uses the dcecp string syntax to show a name service
entry:

dcecp> rpcentry show /.:/subsys/applications/bbs_server
   {458fffcbe-98c1-11cd-bd93-0000c08adf56 1.0
   (ncacn_ip_tcp 130.105.1.227))
   {76030c42-98d5-11cd-88bc-0000c08adf56}
dcecp>

The following command operates from the system prompt to show a name service
entry:

% dcecp -c rpcentry show /.:/subsys/applications/bbs_server
   {458fffcbe-98c1-11cd-bd93-0000c08adf56 1.0
   (ncacn_ip_tcp 130.105.1.227))
   {76030c42-98d5-11cd-88bc-0000c08adf56}
%
**rpcentry unexport**

Removes binding information from an entry in the name service. The syntax is as follows:

```
rpcentry unexport entry_name_list {[-object object_uuid_list]
[-interface interface_id [-version versions]]}
```

**Options**

- **-object object_uuid_list**
  Declares the UUID of an object. Accepts a list of up to 32 object UUIDs. The UUID is a hexadecimal string.
  See Data Structures for the format of the object UUID.

- **-interface interface_id**
  Declares the interface identifier of an RPC interface. Only a single interface_id can be specified.
  See Data Structures for the format of the interface identifier.

- **-version versions**
  Specifies interface version numbers to be returned with the unexport operation.
  See Data Structures for the exact behavior and format of the version values.

The unexport operation removes binding information from an entry in the name service. The entry_name_list argument is a list of one or more entry names from which binding information is to be removed. This operation returns an empty string on success.

**Privileges Required**

You need d (delete) permission on the parent directory and r (read ) permission and w (write ) permission on the CDS object entry (the target name service entry).

**Examples**

The following example uses the dcecp syntax to unexport the binding information for an interface. The third command entered (rpcentry show) shows the RPC entry after the unexport operation; the object UUID remains in the entry.

```dcecp> rpcentry show /.:/subsys/applications/bbs_server
{458ffcb6-98c1-11cd-bd93-0000c08adf56 1.0
 {ncacn_ip_tcp 130.105.1.227}}
{76030c42-98d5-11cd-88bc-0000c08adf56}
```

```dcecp> rpcentry unexport /.:/subsys/applications/bbs_server \
> -interface {458ffcb6-98c1-11cd-bd93-0000c08adf56 1.0}
```

```dcecp> rpcentry show /.:/subsys/applications/bbs_server
{76030c42-98d5-11cd-88bc-0000c08adf56}
```
Related Information

Commands: dcecp(8dce), endpoint(8dce), rpcgroup(8dce), rpcprofile(8dce).
rpcgroup

Purpose

A dcecp object that manages an RPC group entry in CDS

Synopsis

rpcgroup add rpcgroup_name_list -member member_name_list
rpcgroup create rpcgroup_name_list
rpcgroup delete rpcgroup_name_list
rpcgroup help [operation | -verbose ]
rpcgroup import rpcgroup_name_list -interface interface_id [-object object_uuid] [-max integer] [-noupdate]
rpcgroup list rpcgroup_name_list [-member member_name_list] [-noupdate]
rpcgroup operations
rpcgroup remove rpcgroup_name_list -member member_name_list

Arguments

operation
The name of the rpcgroup operation for which to display help information.

rpcgroup_name_list
Specifies a list of one or more names of the RPC groups to be operated on.

Description

The rpcgroup object represents a remote procedure call (RPC) group entry in the Cell Directory Service (CDS). Each RPC group is named in the DCE namespace; therefore, each operation takes as an argument a list of names of group entries to manipulate. An RPC group is a container that contains only the names of RPC server entries or the names of other RPC groups; it contains no other data.

interface_id
The interface identifier of an RPC interface. The interface identifier takes the following form:

interface-uuid, major-version, minor-version

The version numbers are optional. If you omit a version number, the default is 0. The UUID is a hexadecimal string and the version numbers are decimal strings. For example:

-interfac ecleeb60-5943-11c9-a309-08002b102989,3.11

Leading zeros in version numbers are ignored.
Alternatively, you can use `dcecp` string syntax. For example:

```
-interframe {450ffbe-98c1-11cd-bd93-0000c08af65 1.0}
```

**object_uuid**
The UUID of an object. The UUID is a hexadecimal string, for example:

```
-object 3c6b8f60-5945-11c9-a236-08002b102989
```

Alternatively, you can use `dcecp` string syntax. For example:

```
-object {3c6b8f60-5945-11c9-a236-08002b102989}
```

**Errors**
A representative list of errors that might be returned is not shown here. Refer to the *IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide* for complete descriptions of all error messages.

**Operations**

### rpcgroup add

Adds a member to the specified group entry in CDS. The syntax is as follows:

```
rpcgroup add  rpcgroup_name_list -member member_name_list
```

**Options**

- **-member member_name_list**
  This required option declares the name of a member to be added to the specified group entry. The `member_name_list` argument is a list of names of one or more members to be added to all of the specified groups. (The names need not exist when they are added.) All members are added to all groups.

The `add` operation adds a member to the specified group entry in CDS. The required `rpcgroup_name_list` argument is a list of one or more full CDS names of the groups to which you want to add members. This operation returns an empty string on success. If `member_name_list` contains the names of duplicate or existing members, the duplicates are ignored and no errors are generated.

**Privileges Required**

You need `i` (insert) permission to the parent directory. You also need both `r` (read) permission and `w` (write) permission to the CDS object entry (the target group entry).

**Examples**

The following command adds the member `./LandS/anthro/Cal_host_3` to the group `./LandS/anthro/Calendar_group`:

```
dcecp> rpcgroup add ./LandS/anthro/Calendar_group  \
> -member ./LandS/anthro/Cal_host_3
dcecp>
```
**rpcgroup create**

Creates an empty RPC group entry in CDS. The syntax is as follows:

```
rpcgroup create  rpcgroup_name_list
```

The `create` operation creates a new (empty) RPC group entry in CDS. Since an empty group is the same as an empty RPC entry or RPC profile, calling `rpcgroup create` is the same as calling `rpcentry create` or `rpcprofile create`. The `rpcgroup_name_list` argument is a list of names of RPC groups to be created. The operation returns an empty string on success. If the RPC group already exists, an error is returned.

**Privileges Required**

You need `i` (insert) permission to the parent directory.

**Examples**

The following command creates a new group called `/./LandS/anthro/Calendar_group`:

```
dcecp> rpcgroup create  /./LandS/anthro/Calendar_group
```

**rpcgroup delete**

Removes the specified group from CDS. The syntax is as follows:

```
rpcgroup delete  rpcgroup_name_list
```

The `delete` operation removes the specified group entry from CDS. The `rpcgroup_name_list` argument is a list of names of RPC group entries to be deleted. This operation returns an empty string on success. If the RPC group entry does not exist, an error is generated.

**Privileges Required**

You need `w` (write) permission to the CDS object entry (the target group entry).

**Examples**

The following command removes the group `/./LandS/anthro/Calendar_group` from CDS.

```
dcecp> rpcgroup delete  /./LandS/anthro/Calendar_group
```

**rpcgroup help**

Returns help information about the `rpcgroup` object and its operations. The syntax is as follows:

```
rpcgroup help  [operation | -verbose]
```

**Options**
-verbose
Displays information about the rpcgroup object.

Used without an argument or option, the rpcgroup help command returns brief information about each rpcgroup operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the -verbose option for more detailed information about the rpcgroup object itself.

Privileges Required
No special privileges are needed to use the rpcgroup help command.

Examples

dcecp> rpcgroup help
add Adds members to a list of RPC groups.
create Creates a list of empty RPC groups.
delete Deletes a list of RPC groups.
import Returns the bindings from a list of RPC groups.
list Returns the members of a list of RPC groups.
remove Removes members from a list of RPC groups.
help Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp>

rpcgroup import

Returns a string binding from the specified RPC group. The syntax is as follows:

rpcgroup import  rpcgroup_name_list -interface interface_id
[-object object_uuid] [-max integer] [-noupdate]

Options

-interface interface_id
Declares the interface identifier of one RPC interface.
See Data Structures for the format of the interface identifier.

-object object_uuid
Declares the UUID of one object. The UUID is a hexadecimal string.
See Data Structures for the format of the object UUID.

-max integer
Specifies the maximum number of string bindings to return. A value greater than one returns a list containing up to the number of bindings specified by the value.

-noupdate
Normally, name service data is cached locally on each machine in a cell. If a name service inquiry can be satisfied by data in the local CDS cache, this cached data is returned. However, locally cached copies of name service data might not include a recent CDS update. If the -noupdate option is not specified, dcecp goes to one or more CDS servers to retrieve the required data, updating the local CDS cache. Use the -noupdate option to avoid taking the time to update the local cache when you have reason to believe that the local cache is up to date.
The import operation returns a string binding from the specified RPC group. The rpcgroup_name_list argument is a list of names of RPC groups to import from. The operation uses the -interface and -object options to specify matching bindings. The operation also accepts the -max option to specify a number of string bindings to return. The order of bindings returned is arbitrary.

Privileges Required

You need r (read) permission to the specified CDS object entry (the starting name service entry) and to any CDS object entry in the resulting search path.

Examples

The following command imports a binding:

```
dcecp> rpcgroup import /./ortho_group \
>   -interface {eceleeb60-5943-11c9-a309-08002b102989 1.1} \
>   -object 30dbeea0-fb6c-11c9-8eea-08002b0f4528 
{ncadg_ip_udp 15.22.48.25} 
{ncacn_ip_tcp 15.22.48.25} 
```

rpcgroup list

Returns a list of the names of all members of the specified group. The syntax is as follows:

```
rpcgroup list  rpcgroup_name_list [-member 
member_name_list] [-noupdate]
```

Options

- **-member member_name_list**
  Specifies a list of names of one or more members to be returned from all groups named in the rpcgroup_name_list argument. Use this option to check for specific member names. The member_name_list argument specifies a list of names of RPC entries, RPC groups, or RPC profiles; they are only references stored in the RPC group and do not have to exist outside of the group. All members specified are listed from all RPC groups specified in the argument.

- **-noupdate**
  Use -noupdate to avoid taking the time to update the local cache.

See rpcgroup import for more information.

The list operation returns a list of the names of all members of the specified group. The names returned are fully qualified and are returned in an arbitrary order. The rpcgroup_name_list argument is a list of names of RPC groups whose members’ names are to be returned.

Privileges Required

You need r (read) permission to the CDS object entry (the target group entry).

Examples
The following example lists all the members of the group
\texttt{/./subsys/applications/infobases}, in the order in which they were added to the
group:

\begin{verbatim}
dcecp> rpcgroup list ././subsys/applications/infobases
/.../my_cell.goodcompany.com/subsys/applications/video_server
/.../my_cell.goodcompany.com/subsys/applications/bbs_server
/.../my_cell.goodcompany.com/subsys/applications/audio_server1
/.../my_cell.goodcompany.com/subsys/applications/audio_server2
/.../my_cell.goodcompany.com/subsys/applications/clipart_server
/.../my_cell.goodcompany.com/subsys/applications/photo_server1
/.../my_cell.goodcompany.com/subsys/applications/photo_server2

dcecp>
\end{verbatim}

The following example uses the \texttt{-member} option to list a specific member of the
group \texttt{/./subsys/applications/infobases}:

\begin{verbatim}
dcecp> rpcgroup list ././subsys/applications/infobases \
> -member ././subsys/applications/bbs_server
/.../my_cell.goodcompany.com/subsys/applications/bbs_server

dcecp>
\end{verbatim}

\textbf{rpcgroup operations}

Returns a list of the operations supported by the \texttt{rpcgroup} object. The syntax is as
follows:

\texttt{rpcgroup operations}

The list of available operations is in alphabetical order except for \texttt{help} and
\texttt{operations}, which are listed last.

\textbf{Privileges Required}

No special privileges are needed to use the \texttt{rpcgroup operations} command.

\textbf{Examples}

\begin{verbatim}
dcecp> rpcgroup operations
add create delete import list remove help operations

dcecp>
\end{verbatim}

\textbf{rpcgroup remove}

Removes one or more members from the specified group. The syntax is as follows:

\texttt{rpcgroup remove \ rpcgroup\_name\_list \ -member \ member\_name\_list}

\textbf{Options}

\texttt{-member member\_name\_list}

This required option lets you specify a list of names of one or more
members to be removed from all groups named in the \texttt{rpcgroup\_name\_list}
argument. The \texttt{member\_name\_list} argument specifies a list of names of
RPC entries, RPC groups, or RPC profiles; these are only references
stored in the RPC group and need not exist outside of the group. All
members specified are removed from all RPC groups specified in the
argument.
The remove operation removes one or more members from the specified group. The rpcgroup_name_list argument is a list of names of RPC groups to have members removed from. The value of the required -member option is a list of names of RPC entries, RPC groups, or RPC profiles. If a specified member does not exist in an RPC group, an error is returned. This operation returns an empty string on success.

Privileges Required

You need r (read) permission and w (write) permission to the CDS object entry (the target group entry).

Examples

The following command removes the member ./subsys/applications/video_server from the RPC group ./subsys/applications/infobases:

dcecp> rpcgroup remove ./subsys/applications/infobases \
    > -member ./.../my_cell.goodcompany.com/subsys/applications/video_server

dcecp>

Related Information

Commands: dcecp(8dce), endpoint(8dce), rpcentry(8dce), rpcprofile(8dce).
**rpcprofile**

**Purpose**

A dcecp object that manages an RPC profile entry in CDS

**Synopsis**

```bash
rpcprofile add profile_name_list -member member_name_list
    {-interface interface_id | [-priority priority] | [-annotation annotation] | -default }
rpcprofile create profile_name_list
rpcprofile delete profile_name_list
rpcprofile help [operation | -verbose ]
rpcprofile import profile_name_list -interface interface_id [-object object_uuid]
    [-max integer] [-noupdate]
rpcprofile list profile_name_list [-member member_name_list] [-noupdate]
rpcprofile operations
rpcprofile remove profile_name_list {-default | -member member_name | -interface interface_id | -annotation annotation | -priority priority }
rpcprofile show profile_name_list {-default | -member member_name | -interface interface_id | [-version versions] | [-priority priority] | [-annotation annotation] | [-noupdate] }
```

**Arguments**

*operation*  
The name of the *rpcprofile* operation for which to display help information.

*profile_name_list*  
Specifies a list of one or more names of the RPC profile entries to be operated on.

**Description**

The *rpcprofile* object represents a remote procedure call (RPC) profile entry in the Cell Directory Service (CDS). Each operation described below, except *help* and *operation*, takes as an argument a list of one or more names of RPC profiles to be operate on. An RPC profile consists of members (also known as elements in other DCE documentation). A member can be either RPC server entries, RPC groups, or other RPC profiles; therefore each member of a profile has a name in the DCE namespace. Each profile can also have one default member (called the default profile element).

A profile entry contains no attributes, but does contain information about each member that is not contained in the member itself. The information stored for each member includes up to four fields of information consisting of interface and version, a member name, a priority (0 through 7), and an annotation. For example:
Various `rpcprofile` operations have options that correspond to the fields of information contained in profile members. Specifically, the options are `-interface`, `-member`, `-priority`, and `-annotation`.

**interface_id**
The interface identifier of an RPC interface. The interface identifier takes the following form:

```
interface-uuid, major-version, minor-version
```

The version numbers are optional, but if you omit a version number, the value defaults to 0. The UUID is a hexadecimal string and the version numbers are decimal strings. For example:

```
-inter |face ecleeb60-5943-11c9-a309-08002b102989,3.11
```

Leading zeros in version numbers are ignored.

Alternatively, you can use `dcecp` string syntax in the following form:

```
{interface-UUID major-version.minor-version}
```

For example:

```
-inter |face {458ffcbq-98c1-11cd-bd93-0000c08ad856 1.0}
```

**object_uuid**
The UUID of an object. The UUID is a hexadecimal string, for example:

```
-object 3c6b8f60-5945-11c9-a236-08002b102989
```

Alternatively, you can use `dcecp` string syntax. For example:

```
-object {3c6b8f60-5945-11c9-a236-08002b102989}
```

**host_address**
An RPC string binding that describes a host’s location. The binding information contains an RPC protocol and the host’s network address. Any specific host’s network address can be obtained by using the `getip` command.

**annotation**
An informational text string that helps you to identify the purpose of the endpoint. Use single or double quotation marks around the annotation field of endpoints to include internal spaces in an annotation, for example:

```
-annotation "Bulletin Board Server, Version 1.3a"
```

Alternatively, you can use `dcecp` string syntax. For example:

```
-annotation {Bulletin Board Server, Version 1.3a}
```
**version**

Specifies which interface version numbers to return with a *show* operation. Specify versions by using one of the following values for the **-version** option:

- **all** The interface version is ignored.
- **exact** Both the major and minor versions must match the specified versions.
- **compatible** The major version must match the specified version, and the minor version must be greater than or equal to the specified version.
- **major_only** The major version must match the specified version; the minor version is ignored.
- **upto** The major version must be less than or equal to that specified. If the major versions are equal, the minor version must be less than or equal to that specified.

If the **-version** option is absent, the command shows compatible version numbers.

**Errors**

A representative list of errors that might be returned is not shown here. Refer to the *IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide* for complete descriptions of all error messages.

**Operations**

**rpcprofile add**

Adds a member to the specified profile entry in CDS. The syntax is as follows:

```
rpcprofile add profile_name_list -member member_name_list
{-interface interface_id [-priority priority] [-annotation annotation] | -default}
```

**Options**

- **-member member_name_list**
  This required option declares the name of a member to be added to the specified profile entry. The *member_name_list* argument is a list of names of one or more members to be added to all of the specified profiles.

  See **Data Structures** for the format of the interface identifier.

- **-interface interface_id**
  Required when the **-default** option is not used, this option declares the interface identifier of an RPC interface. The add operation operates on only one *interface_id*.

- **-priority priority**
  Defines a search priority for the new profile element. The priority value is in the range 0 to 7 with zero having the highest priority. By default, a nondefault element is assigned a priority value of zero.
rpcprofile(8dceadd)

-annotation annotation
Defines an annotation string for the profile element. You can include internal spaces in an annotation by enclosing the string in quotation marks.

-default
Performs the operation on the default profile member. When you use the -default option, all of the other options except -member are illegal.

The add operation adds a member to the specified profile entry in CDS. The profile_name_list argument is a list of names of RPC profiles to have members added to. The value of the required -member option is a list of names which are references to an RPC entry, RPC group, or RPC profile (that is, they do not have to actually exist).

The operation accepts the -interface, -priority, and -annotation options with one value (not a list) each. All members are added to each profile identified in the argument list. It also accepts a -default option to indicate that the member being added is the default profile member. (If you specify the -default option, the only other option that can be supplied is -member.) This operation returns an empty string on success. If member_name_list contains the names of duplicate or existing members, the duplicates are ignored, and no errors are generated.

Privileges Required
You need i (insert) permission to the parent directory. You also need both r (read) permission and w (write) permission to the CDS object entry (the target profile entry).

Examples
The following command adds an element to the cell profile, /.:/cell-profile, in the local cell:

dcecp> rpcprofile add /.:/cell-profile
> -member /.:/Calendar_profile
> -interface ec1eeb60-5943-11c9-a309-08002b102989,1.1
> -annotation RefersToCalendarGroups

dcecp>

The following commands set up a user profile associated with the cell profile as its default element and add a user-specific element for the Calendar V1.1 interface:

dcecp> rpcprofile add /.:/LandS/anthro/molly_o_profile -default /.:/cell-profile
dcecp>

dcecp> rpcprofile add /.:/LandS/anthro/molly_o_profile
> -member {/.:/LandS/anthro/Calendar_group}
> -interface {ec1eeb60-5943-11c9-a309-08002b102989 1.1}
> -annotation {Calendar_Version 1.1_Interface}
dcecp>

The added profile element contains the global name of the member (specified by using its cell-relative name, /.:/LandS/anthro/Calendar_group) and the RPC interface identifier for the Calendar Version 1.1 interface.
rpcprofile create

Creates a new profile entry in CDS. The syntax is as follows:

```
rpcprofile create profile_name_list
```

The `create` operation creates a new (empty) profile entry in CDS. Since an empty profile is the same as an empty RPC entry or RPC group, calling `rpcprofile create` is the same as calling `rpcentry create` or `rpcgroup create`. The `profile_name_list` argument is a list of names of RPC profiles to be created. This operation returns an empty string on success. If the RPC profile already exists, an error is returned.

Privileges Required

You need `i` (insert) permission to the parent directory. You also need both `r` (read) permission and `w` (write) permission to the CDS object entry (the target profile entry).

Examples

```
dcecp> rpcprofile create /.://users/wards_profile
```

rpcprofile delete

Deletes the specified profile from CDS. The syntax is as follows:

```
rpcprofile delete profile_name_list
```

The `delete` operation deletes the specified profile from CDS. The `profile_name_list` argument is a list of names of RPC profiles to be deleted. This operation returns an empty string on success. If the RPC profile does not exist, an error is generated.

Privileges Required

You need `w` (write) permission to the CDS object entry (the target profile entry).

Examples

The following command deletes the profile named `./LandS/anthro/molly_o_profile`:

```
dcecp> rpcprofile delete /.://LandS/anthro/molly_o_profile
```

rpcprofile help

Returns help information about the `rpcprofile` object and its operations. The syntax is as follows:

```
rpcprofile help [operation | -verbose]
```

Options

- `verbose`
  Displays information about the `rpcprofile` object.
Used without an argument or option, the `rpcprofile help` command returns brief information about each `rpcprofile` operation. The optional `operation` argument is the name of an operation about which you want detailed information. Alternatively, you can use the `--verbose` option for more detailed information about the `rpcprofile` object itself.

**Privileges Required**

No special privileges are needed to use the `rpcprofile help` command.

**Examples**

```dcecp
rpcprofile help
```

**rpcprofile import**

Returns a string binding from the specified RPC profile. The syntax is as follows:

```
rpcprofile import profile_name_list -interface interface_id
([-object object_uuid] [-max integer] [-noupdate])
```

**Options**

- **-interface interface_id**
  Declares the interface identifier of an RPC interface. The `import` operation allows you to specify only one `interface_id`, not a list.

  See Data Structures for the format of the interface identifier.

- **-object object_uuid_list**
  Declares the UUID of an object. Each `import` operation accepts a list of up to 32 object UUIDs. The UUID is a hexadecimal string.

- **-max integer**
  Specifies the maximum number of string bindings to return. A value greater than 1 returns a list containing up to the number of bindings specified by the value.

- **-noupdate**
  Normally, name service data is cached locally on each machine in a cell. If a name service inquiry can be satisfied by data in the local CDS cache, this cached data is returned. However, locally cached copies of name service data might not include a recent CDS update. If the `-noupdate` option is not specified, `dcecp` goes to a CDS server to retrieve the required data, updating the local CDS cache. Use the `-noupdate` option to avoid taking the time to update the local cache when you have reason to believe that the local cache is up to date.

The `import` operation returns a string binding from the specified RPC profile. The `profile_name_list` argument is a list of names of RPC profiles to import from. Use
the -interface and -object options to specify matching bindings. Each of these options takes only one value, not a list of values. The import operation also accepts the -max option to specify a number of string bindings to return. If the value is greater than 1, a list of as many matching bindings less than or equal to the value is returned. The order of bindings returned is arbitrary.

Privileges Required

You need r (read) permission to the specified CDS object entry (the starting name service entry) and to any CDS object entry in the resulting search path.

Examples

The following example imports a binding:

```
dcecp> rpcprofile import /.:cell-profile \ > -interface {458ffcbe-98c1-11cd-bd93-0000c08adf56 1.0} 
(ncadg_ip_udp 15.22.48.25) 
(ncadg_ip_udp 15.22.50.213) 
(ncacn_ip_tcp 15.22.48.25) 
(ncacn_ip_tcp 15.22.50.213) 
dcecp>
```

**rpcprofile list**

Returns a list of the names of all members of the specified profile. The syntax is as follows:

```
rpcprofile list  profile_name_list [-member member_name_list] [-noupdate]
```

**Options**

- **-member** member_name_list
  Declares the names of members of the specified profile entry. The member_name_list argument is a list of names of one or more members to be listed.

- **-noupdate**
  Use this option to avoid taking the time to update the local cache. See rpcprofile import for more information.

The list operation returns a list of the names of all members of the specified profile. The names returned are fully qualified and are returned in an arbitrary order. The profile_name_list argument is a list of names of RPC profiles whose members’ names are to be returned. The members are concatenated on output into one list.

Privileges Required

You need r (read) permission to the CDS object entry (the target profile entry).

Examples

The following command lists entries in the cell profile /.:cell-profile in the local cell:

```
dcecp> rpcprofile list /.:cell-profile 
./../my_cell.goodcompany.com/sec 
./../my_cell.goodcompany.com/sec-v1 
./../my_cell.goodcompany.com/sec 
./../my_cell.goodcompany.com/sec
```
rpcprofile operations

Returns a list of the operations supported by the `rpcprofile` object. The syntax is as follows:

```bash
rpcprofile operations
```

The list of available operations is in alphabetical order except for `help` and `operations`, which are listed last.

Privileges Required

No special privileges are needed to use the `rpcprofile operations` command.

Examples

```bash
dcep> rpcprofile operations
add create delete import list remove show help operations

dcep>
```

rpcprofile remove

Removes one or more members from the specified profile. The syntax is as follows:

```bash
rpcprofile remove profile_name_list
{-default | -member member_name -interface interface_id | -annotation annotation | -priority priority}
```

Options

- **-default**
  Performs the `remove` operation on the default profile element. When you use the `-default` option, all of the other options are illegal.

- **-member member_name**
  Required when the `-default` option is not used, this option lets you specify the name a member to be removed from all profiles named in the `profile_name_list` argument. The value of the `-member` option is a single name of an RPC entry, RPC group, or RPC profile; the name is only a reference stored in the RPC profile and need not exist outside of the profile. The specified member is removed from all RPC profiles specified in the argument.

- **-interface interface_id**
  Declares the interface identifier of an RPC interface. The `remove` operation allows you to specify only one `interface_id`.

- **-annotation annotation**
  Defines an annotation string for the profile element to be removed. You can include internal spaces in an annotation by enclosing the string in quotation marks (or by using other `dcep` quoting mechanisms).

- **-priority priority**
  Defines a search priority for the profile element you want to see. The
priority value is in the range 0 to 7, with 0 having the highest priority. By
default, a nondefault element is assigned a priority value of 0.

See Data Structures for the format of the interface identifier.

The remove operation removes one member from the specified profiles. The
profile_name_list argument is a list of names of RPC profiles from which the
member is to be removed. The member to be removed must match the values
given in the following options: -member, -interface, and -annotation. These
options are all single-valued; they are not lists. The matching member is removed
from all RPC profiles specified in the argument. Also accepts a -default option, in
which case the above options are illegal and the default profile member is removed.
This operation returns an empty string on success. If the specified member does
not exist in an RPC, profile an error is returned.

Privileges Required

You need r (read) and w (write) permission to the CDS object entry (the target
profile entry).

Examples

The following example removes the member /.:subsys/applications/infobases
with interface {baf8c319-998f-11cd-ac7b-0000c08adf56 1.0} from the RPC profile
entry /.:users/admin_profile:

dcecp> rpcprofile remove /.:users/admin_profile \
> -member /.:subsysth/applications/infobases \
> -interface {baf8c319-998f-11cd-ac7b-0000c08adf56 1.0}

dcecp>

rpcprofile show

Returns a list of all members of one or more profiles. The syntax is as follows:

rpcprofile show profile_name_list
{-default [-member member_name] [-interface interface_id]
[-version versions] [-priority priority] [-annotation annotation]
[-noupdate]}

Options

-default
Performs the show operation on the default profile element. When you use
the -default option, all of the other options are illegal.

-member member_name
Specifies one member name for which to return profile information.

See Data Structures for the format of the interface identifier.

-interface interface_id
Declares the interface identifier of an RPC interface. The show operation
allows you to specify only one interface_id.

-version versions
Specifies interface version numbers to be returned. This option must be
used with the -interface option.

See Data Structures for the exact behavior of the version values.
rpcprofile(8dceadd)

-priority priority
  Defines a search priority for the profile element you want to see. The priority value is in the range 0 to 7, with 0 having the highest priority. By default, a nondefault element is assigned a priority value of 0.

-annotation annotation
  Defines an annotation string for the profile element. You can include internal spaces in an annotation by enclosing the string in quotation marks (or by using other dcecp quoting mechanisms).

-noupdate
  Use this option to avoid taking the time to update the local cache. See rpcprofile import for more information.

The show operation returns a list of all members of one or more profiles. The profile_name_list argument is a list of names of RPC profiles to have members of returned. An attribute list is returned for each member with all of the entered information. The list is in the following order: interface, member, priority, annotation. If any of the items is not given, they are not included in the output, that is, no place holder is included.

Only those members that match the values specified by the given options are returned. Each option can have only one value (that is, the value can not be a list). Also accepts a -default option, in which case the above options are ignored and the default profile member is returned.

Privileges Required

You need r (read) permission to the CDS object entry (the target profile entry).

Examples

The following example uses no options to show all the members of a profile:

dcecp> rpcprofile show /.:/users/temp_profile
  {{458ffcb8-98c1-11cd-bd93-0000c08ad5f6 1.0} /.../cell.co.com/subsys/appls/infobases 0}
  {{00000000-0000-0000-0000-000000000000 0.0} /.../cell.co.com/cell-profile 0}
  {{baf8c319-998f-11cd-ac7b-0000c08ad5f6 1.0} /.../cell.co.com/subsys/appls/infobases 0}
dcecp>

The following example uses the -interface option to show a single member of a profile.

dcecp> rpcprofile show /.:/users/temp_profile  
  > -interface {baf8c319-998f-11cd-ac7b-0000c08ad5f6 1.0}
  {{baf8c319-998f-11cd-ac7b-0000c08ad5f6 1.0} /.../cell.co.com/subsys/appls/infobases 0}

dcecp>

Related Information

Commands: endpoint(8dce), rpcentry(8dce), rpcgroup(8dce).
secval

Purpose
A dcecp object that manages the security validation service on a host

Synopsis
secval activate [host_name_list]
secval deactivate [host_name_list]
secval help [operation | -verbose ]
secval krb5update -cellname name_of_cell
secval operations
secval ping [host_name_list]
secval status [host_name_list]
secval update [host_name_list] [-pesite time_in_seconds]

Arguments
host_name_list
A list of one or more names of host systems whose security validation
systems you want to act on. If you do not specify this argument, the local
host is assumed. The argument is optional and takes either of the following
forms:

/./hosts/dce_hostname
/.../cell_name/hosts/dce_hostname

operation
The name of the secval operation for which to display help information.

Description
The secval object represents the security validation service running on a host, as
part of the dced server. This service is responsible for maintaining the security
credentials of the host machine.

Access to the commands is based on the access control list (ACL) of the security
validation object for a host. This takes the form of /.../
cell_name/hosts/dce_hostname/config/secval.

Errors
A representative list of errors that might be returned is not shown here. Refer to the
IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for
complete descriptions of all error messages.
secval activate

Activates a security validation service. The syntax is as follows:

```
secval activate [host_name_list]
```

The `activate` operation activates a security validation service. If the service is already activated, an error is returned. The optional `host_name_list` argument is a list of one or more names of host systems whose security validation systems you want to activate. This operation returns an empty string on success.

**Privileges Required**

You must have `x` (execute) permission to the security validation service object.

**Examples**

```
dcecp> secval activate
dcecp>
```

secval deactivate

Deactivates a security validation service. The syntax is as follows:

```
secval deactivate [host_name_list]
```

The `deactivate` operation deactivates a security validation service. If it is already deactivated, an error is returned. The optional `host_name_list` argument is a list of one or more names of host systems whose security validation systems you want to deactivate. This operation returns an empty string on success.

**Privileges Required**

You must have `s` (stop) permission to the security validation service object.

**Examples**

```
dcecp> secval deactivate
dcecp>
```

secval help

Returns help information about the `secval` object and its operations. The syntax is as follows:

```
secval help [operation | -verbose]
```

**Options**

- `-verbose`
  Displays information about the `secval` object.

Used without an argument or option, the `secval help` command returns brief information about each `secval` operation. The optional `operation` argument is the
name of an operation about which you want detailed information. Alternatively, you can use the -verbose option for more detailed information about the secval object itself.

Privileges Required

No special privileges are needed to use the secval help command.

Examples

```
dcecp> secval help
activate Enables the secval service.
deactivate Disables the secval service.
ping Contacts the dced secval to validate the security service.
status Returns 1 if secval is enabled, 0 if not.
update Updates a component of the secval.
help Prints a summary of command-line options.
operations Returns a list of the valid operations for this command.
dcecp>
```

**secval krb5update**

Builds a list of security replicas for the named cell in /etc/krb5.conf file. The syntax is as follows:

```
secval krb5update -cellname name_of_cell
```

Options

- **-cellname**

  The cellname used to query for security replica information.

The secval krb5update operation queries the cell indicated by the cellname option to retrieve the security replica information for that cell. It then resolves the hostname for the security replica and writes the hostnames of the replicas in the REALMS section of the krb5.conf file

Privileges Required

You must be root user on the machine where the command is issued.

Examples

```
dcecp> secval krb5update -cellname mycell.org.com
dcecp>
```

**secval operations**

Returns a list of the operations supported by the secval object. The syntax is as follows:

```
secval operations
```

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

```
No special privileges are needed to use the `secval operations` command.

**Examples**

```bash
dcecp> secval operations
activate deactivate ping status update help operations
dcecp>
```

### secval ping

Validates the credentials returned by a DCE security service. The syntax is as follows:

```
secval ping [host_name_list]
```

The `ping` operation validates the credentials returned by a security service. This operation is rarely invoked, but can be used to verify that `secd` is trusted. The operation returns 1 if the credentials are valid, 0 if they are not. The optional `host_name_list` argument is a list of one or more names of host systems whose security validation systems you want to validate. If the argument is a list of host names, a list is returned with a 1 or a 0 for each server.

**Privileges Required**

No special privileges are needed to use the `secval ping` command.

**Examples**

```bash
dcecp> secval ping
1
dcecp>
```

### secval status

Checks for an active secval. The syntax is as follows:

```
secval status [host_name_list]
```

The `status` operation returns 1 if the security validation service is activated, 0 if it is not. If the argument is a list, a list is returned, with a 0 or 1 for each server.

**Privileges Required**

No special privileges are needed to use the `secval status` command.

**Examples**

```bash
dcecp> secval status
1
dcecp>
```

### secval update

Updates a component of the secval service. The syntax is as follows:

```
secval update [host_name_list] [-pesite time_in_seconds]
```

**Options**
-pesite
  Sets the amount of time to wait between each pe_site Thread Maintenance update.

The update operation updates a component of the security validation service. Currently only updates to the pe_site Maintainer Thread are supported. Use the -pesite option to set the amount of time in seconds between each update. The update is performed after the time specified in time_in_seconds passes, if the -pesite option is not supplied, the update is performed immediately. This operation returns an empty string on success.

Privileges Required

You must have x (execute) permission to the security validation service object.

Examples

dcecp> secval update -pesite 300
dcecp>

Related Information

Commands: dcecp(8dce), dced(8dce)
server(8dce)

server

Purpose

A dcecp object that manages DCE application servers

Synopsis

server catalog [host_name_list] [-executing] [-simplename] [-local]
server create server_name_list {-attribute attribute_list | -attribute value} [-local]
server delete server_name_list [-local]
server disable server_name_list -interface interface_id_list
server enable server_name_list -interface interface_id_list
server help [operation | -verbose]
server modify server_name_list {-add extended_rgy_attr_list | -remove extended_rgy_attr_list | [-types] | -change attribute_list} [-local]
server operations
server ping server_name_list [-timeout timeout_method]
server show server_name_list [-executing] [-local]
server start server_name_list [-uuid uuid_list]
server stop server_name_list [-method method]

Arguments

host_name_list
A list of one or more DCE host names specifying hosts for which to catalog servers. Host names can be in any of the following forms:

/./hosts/dce_hostname
/../../cell_name/hosts/dce_hostname
hosts/dce_hostname

operation
The name of the server operation for which to display help information.

server_name_list
A list of one or more names of servers to act on. Server names have the form

/../../cell_name/hosts/dce_hostname
/config/service/name

where service is one of the following: srvrconf, srvrexec, or server. The first two replacements for service uniquely identify the correct service as
either the configuration service or the execution service. The third is a
simpler, but ambiguous term; however, the ambiguity can usually be
resolved by context. For example, the stop operation applies only to a
srvrexe object. In cases where it is still ambiguous, a srvrconf object is
assumed unless the -executing option is present.

Examples of server names are shown Operations.

**Description**

The server object refers to servers residing on a host. This one object can affect
both the running daemons and the configuration information used by dced to start
that daemon. The distinction is usually obvious by the definition of the operation or
by the name given as an argument. When this is not the case, the ambiguity is
resolved by a required option.

Almost all of these commands contact the dced on the target host to perform their
operations. Exceptions are noted below.

Some commands operate on a single server while other commands operate on
more than one server. See Arguments for a description of how to specify server
names.

Server configuration objects can contain application-specific extended registry
attributes (ERAs). Only the ERAs can be modified after creation; other attributes
cannot.

*interface_id*

The interface identifier of an RPC interface. The interface identifier takes
the following form:

```
interface-uuid, major-version. minor-version
```

The version numbers are optional, but if you omit one, the value defaults to
0. The UUID is a hexadecimal string, and the version numbers are decimal
strings. For example:

```
-interface ee6b0-5943-11c9-a309-a8002b102989,3.11
```

Leading zeros in version numbers are ignored.

Alternatively, you can use dcecp string syntax in the following form:

```
{interface-UUID major-version.minor-version}
```

For example:

```
-interface {458ffbec-98c1-11cd-bd93-0000c08ad56 1.0}
```

**Attributes**

*arguments string_list*

The command-line arguments passed to the program on startup. Its value is
a list of strings. Cannot be modified after creation.
directory directory_name
The working directory that the server is started with. Cannot be modified
after creation.

gid group_id
The POSIX group identifier (gid) that the server is started with. Can not be
modified after creation.

keytabs keytab_list
A list of UUIDs of related keytab objects in which the server stores its keys.
Cannot be modified after creation.

program program_name
The name of the server program to be run. Its value is a string. Cannot not
be modified after creation.

prerequisites uuid_list
A list of UUIDs of other server configuration objects that represents servers
that must be running before this one is started. In DCE Version 1.1, this
information is not used to start the other servers; it is merely a note to the
administrator. Future versions of dc ed will probably take action based on
this attribute. Cannot be modified after creation.

principals principal_name_list
A list of principal names that the server runs as. For example, secd runs as
three different principals. A fully qualified name is always returned on
output. On input a relative principal name represents a principal in the
default cell of the dced. Cannot be modified after creation.

services attribute_list
A list where each element is an attribute list of the following attributes:

  annotation string
  A human readable Portable Character Set (PCS) string describing
  the service. (This is not an internationalized string, for compatibility
  with DCE Version 1.0 endpoint map annotation strings.)

  bindings protocol_sequence_list
  A list of string bindings identifying the service.

  flags flag_name_list
  The value is a list of keywords to identify flags for the server.
  Currently only one is supported:

    disabled
    The mapping has been marked as disabled in the endpoint
    map.

  ifname interface_name
  The name of the interface of the service limited to PCS characters.

  interface interface_id
  The interface identifier (UUID and version) of the service.

  entryname service_name
  The name of the service (limited to PCS characters).

  objects object_uuid_list
  A list of object UUIDs the service supports.

executing { uuid pid}
A list of two elements, the UUID of the server instance and the pid (process
ID) of the running server. This attribute is only present if the server is running. This attribute is multivalued, one value for each instance of the server.

**starton starting_condition_list**

This attribute identifies when a server should be started. The value is a list of one or more of the following, none of which can be modified after creation.

- **auto** Start if `dced` receives a remote call that would be serviced by this server. Ignored for those servers that are repositories.
- **boot** Start at system startup.
- **explicit** Start if `dced` receives a command to start the server (such as the `server start` command in `dcecp`).
- **failure** Start if `dced` detects that the server exited with a nonsuccessful error code.

Specifying a null value to this attribute means the server will not be started. An example of a possible value is as follows:

```
{starton {boot explicit failure}}
```

**uid user_id**

The POSIX user identifier (uid) that the server is started with. Cannot not be modified after creation.

**uuid uuid**

The internal identifier of the object. It can be specified on creation, or automatically generated, but once created it cannot be modified.

Server configuration objects can also have ERAs attached to them. ERAs can be manipulated by the `modify` operation.

See the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide* for more information about server attributes.

**Errors**

A representative list of errors that might be returned is not shown here. Refer to the *IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide* for complete descriptions of all error messages.

**Operations**

**server catalog**

Returns a list of the names of all server configuration objects on a specified host. The syntax is as follows:

```
server catalog [host_name_list] [-executing] [-simplename] [-local]
```

**Options**
server(8dce)

-executing
Returns the name of all servers known by dced that are currently running on the specified host.

-simplename
Returns names but removes the /.../cellname/hosts/dce_hostname/config/service/portion of the name.

-local
Specifies that the command is to operate on the local dced object while the dced on the local machine is in partial service mode.

The catalog operation returns a list of the names of all server configuration objects on a specified host. If called with the -executing option, it returns the name of all server execution objects (running servers) known by dced that are currently executing on the specified host. If called with no arguments, it returns information about the servers on the local host. The optional host_name_list argument is a list of host names. If more than one is specified then the information returned is concatenated. The order of information returned is arbitrary. Fully qualified names are returned by default; use the -simplename option to return the names without prepending the cell name and the name of the server container.

Privileges Required
You must have r (read) permission to the applicable container (configuration or execution) object.

Examples

dcecp> server catalog /.:/hosts/foster
/.:/hosts/foster/config/srvrconf/try_tserver
dcecp>

server create

Creates a server configuration object. The syntax is as follows:

server create server_name_list
{-attribute attribute_list | -attribute value}
[-local]

Options
-attribute attribute_list
Allows you to specify attributes by using an attribute list rather than using the -attribute value option. The format of an attribute list is as follows:

{{attribute value}...{attribute value}}

- attribute value
As an alternative to using the -attribute option with an attribute list, you can change individual attribute options by prepending a hyphen (-) to any attributes listed in Attributes.

-local
Specifies that the command is to operate on the local dced object while the dced on the local machine is in partial service mode.

The create operation creates a server configuration object. The server_name_list argument is a list of names of server configuration objects to be created. An
-attribute option with an argument list as a value is required to define attributes for
the server to be created; the operation also accepts individual - attribute value. It
returns an empty string on success.

Privileges Required

You must have i (insert) permission to the configuration container object.

Examples

dcecp> server create ./:/hosts/foster/config/srvrconf/try_tserver 
> -arguments ./:/hosts/foster/test_server 
> -program tserver 
> -entryname ./:/hosts/foster/test_server 
> -services {{ifname {test server}}
>   {annotation {dcecp server test program}}
>   {interface {008bebed-c7c1-1ddc-9cb3-0000c0ba4944 1.0}}
>   {bindings {ncadg_ip_udp 130.105.5.50}}
>   {objects 0073f23a-2e1a-1ddd-b73a-0000c0ba4944}
>   {flags {}}
>   {entryname ./:/hosts/foster/test_server}}
> -principals tserver 
> -starton {boot auto explicit failure} 
> -directory {/opt/tserver}
dcecp>

server delete

Deletes a server configuration object. The syntax is as follows:

server delete server_name_list [-local]

The delete operation deletes a server configuration object. The server_name_list
argument is a list of names of server configuration objects to be deleted. This
operation returns an empty string on success. An error is returned if any of the
objects do not exist.

Options

-local Specifies that the command is to operate on the local dced object while the
dced on the local machine is in partial service mode.

Privileges Required

You must have d (delete) and r (read) permissions to the server configuration
object.

Examples

dcecp> server delete ./:/hosts/foster/config/srvrconf/try_tserver
dcecp>

server disable

Disables the specified server. The syntax is as follows:

server disable server_name_list -interface interface_id_list

Options
-interface interface_id_list
   Specifies a list of one or more RPC interfaces to be disabled. The interface
   identifier can be in string syntax or dcecp syntax.

   See Data Structures for a description of string and dcecp syntaxes.

The disable operation disables the specified server. It communicates with dced
and removes the endpoints for all interfaces registered by the server (except the
rpc_mgm interface) from the endpoint map. The server_name_list argument is a
list of names of server execution objects. The operation requires the -interface
option to specify a list of interfaces to be disabled. It returns an empty string on
success.

Privileges Required

You must have w (write) permission to the server execution object.

Examples

    dcecp> server disable ./:/hosts/foster/config/srvrexe/try_tserver -interface \{
{008bebed-c7c1-1ddc-9cb3-0000c0ba4944,1.0}
    dcecp>

server enable

Enables the specified server. The syntax is as follows:

    server enable  server_name_list -interface  interface_id_list

Options

-interface interface_id_list
   Specifies a list of one or more RPC interfaces to be enabled. The interface
   identifier can be in string syntax or dcecp syntax.

   See Data Structures for a description of string and dcecp syntaxes.

The enable operation enables the specified server. It communicates with dced
and enables any previously disabled endpoint mapping for all interfaces registered by
the server in the endpoint map. The argument server_name is a list of names of
server execution objects. This operation requires the -interface option to specify a
list of interfaces to be enabled and returns an empty string on success.

Privileges Required

You must have w (write) permission to the server execution object.

Examples

    dcecp> server enable ./:/hosts/foster/config/srvrexe/try_tserver
    dcecp>

server help

Returns help information about the server object and its operations. The syntax is
as follows:

    server help  [operation  | -verbose]
Options

-verbose
Displays information about the server object.

Used without an argument or option, the server help command returns brief information about each server operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the -verbose option for more detailed information about the server object itself.

Privileges Required

No special privileges are needed to use the server help command.

Examples

dcecp> server help
catalog  Returns the list of srvrconf or srvrexec object names.
create  Creates a new server configuration (srvrconf) object.
delete  Deletes a server configuration (srvrconf) object.
disable  Disables interfaces of server execution (srvrexec) object.
enable  Enables interfaces of server execution (srvrexec) object.
modify  Modifies the srvrconf object's variable attributes.
ping  Pings a server to see if it is receiving requests.
show  Returns the attributes of a srvrconf or srvrexec object.
start  Starts the specified server.
stop  Stops the specified running server.
help  Prints a summary of command-line options.
operations  Returns a list of the valid operations for this command.
dcecp>

server modify

Used to add or remove fixed attributes or ERAs and their values from the server configuration object. The syntax is as follows:

server modify  server_name_list
{-add  extended_ryg_attr_list  | -remove  extended_ryg_attr_list  [-types]  | -change  attribute_list}  [-local]

Options

-add extended_ryg_attr_list
Allows you to add ERAs that can be defined for your environment. You can specify the attributes to be added as a list. See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information about ERAs.

-remove extended_ryg_attr_list
Allows you to remove ERAs that can be defined for your environment. You can specify the attributes to be removed as a list. See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information about ERAs.

-types
Specifies that a list of attribute names instead of names and values was given as the value of the -remove option, indicating that the entire attribute should be removed and not just specified values.

-change attribute_list
Allows you to specify attributes by using an attribute list in the following format:
server(8dce)

{{attribute value}...{attribute value}}

See Attributes for more information about server attributes.

- **local** Specifies that the command is to operate on the local dced object while the dced on the local machine is in partial service mode.

The **modify** operation changes fixed attributes or adds or removes ERAs and their values from the server object. The server_name_list argument is a list of names of server objects to be modified. The operation accepts the **-change** option, which must have an attribute list as its value. Attribute options are not supported for this command. The name is always for a server configuration object; you can not modify a server execution object. This operation returns an empty string on success.

**Privileges Required**

You must have **w** (write) permission to the server configuration object.

**Examples**

```dcecp>
server modify /.:/hosts/foster/config/srvrconf/try_tserver
>  -add {data {second server list}}
dcecp>
```

**server operations**

Returns a list of the operations supported by the server object. The syntax is as follows:

```
server operations
```

The list of available operations is in alphabetical order except for **help** and **operations**, which are listed last.

**Privileges Required**

No special privileges are needed to use the **server operations** command.

**Examples**

```dcecp>
server operations
catalog create delete disable enable modify ping show start stop
>  help operations
dcecp>
```

**server ping**

Checks whether a server is receiving client requests. The syntax is as follows:

```
server ping  server_name_list [-timeout  timeout_method]
```

**Options**

- **-timeout timeout_method**
  Specifies the timeout method to use during communication with the server. Legal values are **min** (the default), **max** or **default**.
The **ping** operation queries a server to see whether it is receiving requests. This operation communicates directly with the server. The `server_name_list` argument is a list identifying the servers to ping.

The **-timeout** option controls the communication timeout used in contacting the server being pinged. Use **min** for speed, **max** for accuracy, and **default** for a compromise between speed and accuracy.

This operation returns a list of values, one for each server specified in the argument, in the same order. The values are **1** if the server is listening for RPC requests, **0** if it is not.

Each argument can be in one of the following formats:

1. The name of a server entry in the namespace to be imported from. For example:
   
   ```
   /.../brain_cell/hosts/wallis/srvreexec/event_server
   ```

2. A string binding with an object UUID specified. For example:
   
   ```
   {00337ea9-d979-1dd8-923f-0000c08adf56 ncacn_ip_tcp 15.121.12.72}
   ```

3. A string binding with an endpoint specified. For example:
   
   ```
   {ncacn_ip_tcp 15.121.12.72 1075}
   ```

4. An interface ID followed by a hostname, separated by commas. For example:
   
   ```
   {4885772c-c6d3-11ca-84c6-08002bic8fif,oddball}
   ```

5. An interface ID followed by an object UUID and a hostname, separated by commas. For example:
   
   ```
   {4885772c-c6d3-11ca-84c6-08002bic8fif,
   019ee420-682d-1109-a607-08002bodea7a,
   oddball}
   ```

**Privileges Required**

Often no special privileges are required, but this can vary depending on the individual server.

**Examples**

```
$ dcecp> server ping /.../brain_cell/hosts/wallis/srvreexec/event_server
1
$ dcecp>
```

**server show**

Returns information about servers. The syntax is as follows:

```
server show  server_name_list [-executing] [-local]
```

**Options**

- **-executing**
  
  Returns an attribute list for a running server rather than its associated configuration object.
Specifies that the command is to operate on the local `dced` object while the `dced` on the local machine is in partial service mode.

The `show` operation returns a list of both the fixed attributes and ERAs for the server entries specified in the argument. The argument `server_name_list` is a list of names of server object entries. If the names are ambiguous, server configuration objects are assumed unless the `-executing` option is present. If the argument is a list the output is concatenated into a single list in the order specified.

**Privileges Required**

You must have `r` (read) permission to the specified (configuration or execution) object.

**Examples**

```sh
dcecp> server show .:/hosts/foster/config/srvrconf/try_tserver
{uuid 003b24d2-a196-1df3-915f-0000c0ba4944}
{program tserver}
{arguments .:/hosts/foster/test_server}
{prerequisites {}}
{keytabs {}}
{entryname .:/hosts/foster/test_server}
{services
  {{ifname {test server}}}
  {annotation {dcecp server test program}}
  {interface {008bebed-c7c1-1ddc-9cb3-0000c0ba4944 1.0}}
  {bindings {ncadg_ip_udp 130.105.5.50}}
  {objects 0073f23a-2e1a-1ddd-b73a-0000c0ba4944}
  {flags {}}
  {entryname .:/hosts/foster/test_server}}}
{principals /.../foster_cell/tserver}
{starton boot auto explicit failure}
{uid 0}
{gid 0}
{dir /opt/tserver}
dcecp>
```

**server start**

Contacts a `dced` process to start a server based on a server configuration object.
The syntax is as follows:

```sh
server start server_name_list [-uuid uuid_list]
```

**Options**

- `-uuid uuid_list`
  
  A list of one or more UUIDs that identify the server to be started.

The `start` operation contacts a `dced` to start a server based on a server configuration object. The `server_name_list` argument is a list of names of server configuration objects. This operation returns the UUID of the started server on success. This is the UUID found in the `serverexec` object for the server.

**Privileges Required**

You must have `x` (execute) permission to the configuration object.

**Examples**

```sh
server(8dce)
```
server stop

Stops the specified running server processes. The syntax is as follows:

server stop server_name_list [-method method]

Options

-method method
Optionally specifies how dced should stop the server. The method must be one of the following:

- rpc: Use rpc_mgmt_server_stop_listening. This is the default.
- soft: Use a soft local mechanism, such as SIGTERM.
- hard: Use a hard local mechanism, such as SIGKILL.
- error: Use a state-preserving mechanism, such as SIGABRT.

The stop operation stops the specified running server processes. The server_name_list argument is a list of names of servers. This operation returns an empty string on success. It takes an optional -method option to specify how dced should stop the server.

The RPC runtime identifies servers not by name, but by interface, object UUID and endpoints. You should be aware that if you use the rpc method, the command cannot distinguish between two or more server instances binding without endpoints to the same interface and using the same object UUID. In this case, the command stops a randomly selected server, not necessarily the one named in server_name_list.

Privileges Required

You must have s (stop) permission on the execution object.

Examples

dcecp> server stop /./hosts/foster/config/srvrexec/try_tserver

dcecp>

Related Information

Commands: acl(8dce), account(8dce), dcecp(8dce), dced(8dce), hostdata(8dce), keytab(8dce).
user(8dce)

user

Purpose

A dcecp task object that manipulates user information in a DCE cell

Synopsis

user create user_name_list -mypwd password -password password -group group_name -organization organization_name [-force] [-attribute attribute_list | -attribute value ]

user delete user_name_list

user help [operation | -verbose ]

user operations

user show user_name_list

Arguments

operation
The name of the user operation for which to display help information.

user_name_list
A list of one or more names of principals to act on. Supply the names as follows:
1. Fully qualified principal names in the form

   /[...]/cell_name/principal_name or
   /:/principal_name

2. Cell-relative principal names in the form

   principal_name

These names refer to a principal in the cell identified in the _s(sec)
convenience variable, or if the _s(sec) convenience variable is not set,
in the local host's default cell.

Do not mix fully qualified names and cell-relative names in a list. In
addition, do not use the names of registry database objects that contain
principal information; in other words, do not use names that begin with
/:./sec/principal/.

Description

The user task object represents all of the data associated with a DCE user. This
consists of registry information and a Cell Directory Service (CDS) directory in the
default implementation. The user task object allows administrators to easily create
principals and accounts, delete principals and accounts, and view principal
information.

When it creates a principal and account, the user task object adds a CDS directory
named after the principal with the appropriate access control list (ACL). If necessary
the **user** task object also adds the principal to a group and an organization, creating the group and organization if necessary. Only the principal and account attributes are considered attributes of the **user** task object, and are the only ones displayed by the **show** operation.

This object is implemented as a script to allow it to be manipulated and extended on a per-site basis. For example, administrators might want to add Global Directory Service (GDS) and Distributed File Service (DFS) information to the object. Other possible modifications include the following:

1. Changing the location of the CDS directory created for users, or remove it completely.
2. Changing the default ACLs placed on the various objects.
3. Setting certain attributes or policies on all newly created principals and accounts to match the site’s policies.
4. Setting up site specific defaults for passwords (to be changed by the user later), groups, organizations, principal directories, and so on.
5. Supporting a **modify** operation.

### Attributes

**alias value**

Used with the **create** operation. The value of this attribute must be **yes** or **no**. Each principal can have only one name, but can have one or more alias names. All these names refer to the same principal and, therefore, the same Universal Unique Identifier (UUID) and UNIX ID (uid). While aliases refer to the same principal, they are separate entries in the registry database. Therefore the name supplied to a **user** command can refer to either the primary name or an alias name of a principal. The value of this attribute determines whether the name is a primary name (**alias no**) or an alias name (**alias yes**). The default is **no**.

**client {yes | no}**

A flag set to indicate whether the account is for a principal that can act as a client. The value of this attribute must be **yes** or **no**. If you set it to **yes**, the principal is able to log in to the account and acquire tickets for authentication. The default is **yes**.

**description**

A text string (limited to the Portable Character Set or PCS) typically used to describe the use of the account. The default is the empty string ("").

**dupkey {yes | no}**

A flag set to determine if tickets issued to the account’s principal can have duplicate keys. The value of this attribute must be **yes** or **no**. The default is **no**.

In DCE, this attribute is currently only advisory. However, Kerberos clients and servers will use it when they interact with a DCE Security server.

**expdate ISO_timestamp**

The date on which the account expires. To renew the account, change the date in this field. Specify the time by using an ISO compliant time format such as **CCYY-MM-DD hh:mm:ss** or the string **none**. The default is **none**.

**forwardabletkt {yes | no}**

A flag set to determine 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 proxiabletkt attribute performs the same function for service tickets. This attribute must have a value of yes or no. The default is yes.

In DCE, this attribute is currently only advisory. However, Kerberos clients and servers will use it when they interact with a DCE Security server.

fullname string
Used with the create operation, this attribute specifies the full name of the principal. It is for information purposes only. It typically describes or expands a primary name to allow easy recognition by users. For example, a principal could have a primary name of jsbach and a full name of Johann S. Bach. The value is a string. If it contains spaces, it is displayed in quotes, and on entry must be in quotations or braces (as per Tcl quoting rules). If not entered, the full name defaults to the null string (that is, blank).

force
Force creation of the specified group or organization if they do not exist.

group group_name
The name of the group associated with the account. The value is a single group name of an existing group in the registry. This attribute must be specified for the user create command; it does not have a default value.

If a group is deleted from the registry, all accounts associated with the group are also deleted.

home directory_name
The file system directory in which the principal is placed in at login. The default is the / directory.

organization organization_name
The name of the organization associated with the account. The value is a single organization name of an existing organization in the registry. This attribute must be specified for the account create command; it does not have a default value.

If an organization is deleted from the registry, all accounts associated with the organization are also deleted.

maxtktlife relative_time
The maximum amount of time that a ticket can be valid. Specify the time by using the Distributed Time Service (DTS) relative time format ([-] DD- hh: mm: ss). When a client requests a ticket to a server, the lifetime granted to the ticket takes into account the maxtktlife set for both the server and the client. In other words, the lifetime cannot exceed the shorter of the server's or client's maxtktlife. If you do not specify a maxtktlife for an account, the maxtktlife defined as registry authorization policy is used.

maxtktrenew relative_time
The amount of time 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. Specify the time by using the DTS-relative time format ([t] DD- hh: mm: ss). The lifetime of the principal's service tickets can never exceed the lifetime of the principal's ticket-granting ticket. The shorter you make maxtktrenew, the greater the security of the system. However, since principals must log in again to renew their ticket-granting ticket, the time needs to balance user convenience against level of security required. If you do not specify this attribute for an account, the maxtktrenew lifetime defined as registry authorization policy is used. This feature is not currently used by DCE; any use of this option is unsupported at the present time.
mypwd password
Let's you enter your password. You must enter your password to create an account. This check prevents a malicious user from using an existing privileged session to create unauthorized accounts.

password password
The password of the account. This attribute must be specified for the user create command; there is no default value. This attribute is not returned by a user show command.

postdatedtkt {yes | no}
A flag set to determine whether tickets with a start time some time in the future can be issued to the account's principal. This attribute must have a value of yes or no. The default is no.
In DCE, this attribute is currently only advisory. However, Kerberos clients and servers will use it when they interact with a DCE Security server.

proxiabletkt {yes | no}
A flag set to determine whether a new ticket with a different network address than the present ticket can be issued to the account's principal. The forwardabletkt attribute performs the same function for ticket-granting tickets. This attribute must have a value of yes or no. The default is no.
In DCE, this attribute is currently only advisory. However, Kerberos clients and servers will use it when they interact with a DCE Security server.

pwdvalid {yes | no}
A flag set to determine whether the current password is valid. If this flag is set to no, the next time a principal logs in to the account, the system prompts the principal to change the password. (Note that this flag is separate from the pwdexpdate policy, which sets time limits on password validity.) This attribute must have a value of yes or no. The default is yes.

renewabletkt {yes | no}
A flag set to determine if the ticket-granting ticket issued to the account's principal can be renewed. If this flag is set to yes, the authentication service renews the ticket-granting ticket if its lifetime is valid. This attribute must have a value of yes or no. The default is yes.
In DCE, this attribute is currently only advisory. However, Kerberos clients and servers will use it when they interact with a DCE Security server.

server {yes | no}
A flag set to indicate whether the account is for a principal that can act as a server. If the account is for a server that engages in authenticated communications, set this flag to yes. This attribute must have a value of yes or no. The default is yes.

shell path_to_shell
The path of the shell that is executed when a principal logs in. The legal value is any shell supported by the home cell. The default value is the empty string ("").

stdtgtauth {yes | no}
A flag set to determine whether service tickets issued to the account's principal use the standard DCE ticket-granting ticket authentication mechanism. This attribute must have a value of yes or no. The default is yes.

uid value
Used with the create operation, this specifies the UNIX ID (uid) for the
principal. No two principals can have the same uid. However, aliases can
share one uid. It is often called the Unix ID and is an integer. If this attribute
is not supplied, a UID is assigned to principal automatically.

See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more
information about principal and account attributes.

Errors

A representative list of errors that might be returned is not shown here. Refer to the
IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for
complete descriptions of all error messages.

Operations

user create

Creates a principal name, an account, and a directory in CDS for one or more DCE
users. The syntax is as follows:

user create  user_name_list  -mypwd  password  -password
password  -group  group_name  -organization  organization_name  [-force]
{-attribute  attribute_list  |  -attribute  value}

Options

- attribute value
As an alternative to using the -attribute option with an attribute list, you can
change individual attribute options by prepending a hyphen (-) to any
attributes listed in Attributes.

-attribute attribute_list
Allows you to specify attributes, including ERAs, by using an attribute list
rather than using the - attribute value option. The format of an attribute list
is as follows:

{{attribute_value}...{attribute_value}}

-force Forces creation of the specified group or organization if they do not exist.

-group group_name
The name of the group to associate with the account. See Attributes for
the format of a group name.

-mypwd password
Your privileged password. You must enter your privileged password to
create an account. This check prevents a malicious user from using an
existing privileged session to create unauthorized accounts. You must
specify this option on the command line; it cannot be supplied in a script.

-organization organization_name
The name of the organization to associate with the account. See Attributes
for the format of an organization name.

-password password
The account password. See Attributes for the format of a password.

The create operation creates a principal name, account, and a directory in CDS for
one or more DCE users. The user_name_list argument is the name of one or more
principals to be added to the registry. This operation returns an empty string on success. If the operation encounters an error, it attempts to undo any interim operations that have completed.

This command creates one or more principals and accounts for them. If a principal or account already exists, an error is generated. Each principal is then added to the specified group and organization (since the principal has just been created, it cannot have been a member of any group or organization). If the group or organization does not exist, an error is generated unless the \textbf{-force} option is used.

The operation creates a CDS directory called /./:users/ principal_name and adds an ACL entry to the default ACL so that the user has \texttt{rwtci} permissions on the directory. These permissions allow all access except for deleting the directory and administering replication on the directory.

Attributes and policies for the newly created principal and account can be specified with the \textbf{-attributes} option and specifying an attribute list as the value, or with attribute options. This command attempts to add any unknown attributes as ERAs on the created principal object. Policies of the organization cannot be specified, as they would probably affect more than the created user. The required group and organization names can be specified either as attributes in the \textbf{-attributes} option or via the \texttt{-group} and \texttt{-organization} options. The required \texttt{password} attribute can be provided as in the \texttt{account create} command, and the \texttt{-mypwd} option is also required.

Privileges Required

Because the \texttt{user create} command performs several operations, you need the permissions associated with each operation, as follows:

1. To create the principal name, you must have \texttt{i} (insert) permission to the directory in which the principal is to be created.

2. If the specified groups or organizations do not already exist and you use the \texttt{-force} option, you must have \texttt{i} (insert) permission to the directories in which the groups and organizations are to be created.

3. To create the account, you must have \texttt{m} (mgmt\_info), \texttt{a} (auth\_info), and \texttt{u} (user\_info) permission to the principal named in the account, \texttt{r} (read) permission to the organization named in the account, \texttt{r} (read) permission to the group named in the account, and \texttt{r} (read) permission on the registry policy object.

4. To create the directory in CDS you must have the following permissions:
   a. \texttt{r} (read) and \texttt{i} (insert) permission to the parent directory
   b. \texttt{w} (write) permission to the clearinghouse in which the master replica of the new directory is to be stored.

Examples

The following example creates a principal named \texttt{K\_Parsons} and adds him to a group named \texttt{users} and an organization named \texttt{users}:

```
dcecp> user create K\_Parsons -mypwd 3kl\_JL2 -password change.me \>
> -group users -organization users
dcecp> group list users
/.../my\_cell.goodco.com/W\_Ross
/.../my\_cell.goodco.com/J\_Severance
/.../my\_cell.goodco.com/J\_Hunter
/.../my\_cell.goodco.com/B\_Carr
/.../my\_cell.goodco.com/E\_Vliet
```
user(8dce)

.../my_cell.goodco.com/J_Egan
.../my_cell.goodco.com/F_Willis
.../my_cell.goodco.com/K_Parsons
dcecp>

dcecp> account show K_Parsons
{acctvalid yes}
{client yes}
{created /.../my_cell.goodco.com/cell_admin 1994-07-27-13:02:51.000+00:00I-----}
{description {}}
{dupkey no}
{expdate none}
{forwardabletkt yes}
{goodsince 1994-07-27-13:02:51.000+00:00I-----}
{group users}
{home /}
{lastchange /.../my_cell.goodco.com/cell_admin 1994-07-27-13:02:51.000+00:00I-----}
{organization users}
{postdatedtkt no}
{proxiabletkt no}
{pwdvalid yes}
{renewabletkt yes}
{server yes}
{shell {}}
{stdtgtauth yes}
dcecp>

user delete

Deletes DCE users. The syntax is as follows:

user delete user_name_list

The delete operation deletes the DCE users named in user_name_list. To delete a user, the operation proceeds as follows:
1. Deletes the principal from the registry, which also deletes the account and removes the principal from any groups and organizations.
2. Deletes the /.:/users/ principal_name directory and any contents.

This operation returns an empty string on success.

Privileges Required

Because the user delete command performs several operations, you need the permissions associated with each operation:
1. You must have d (delete) permission to the directory in which the target principal exists. You must have r (read) and D (Delete_object) permission on the principal to be deleted.
2. You must have r (read) and M (Member_list) permission on the target groups and organizations and r (read) permission on the member to be removed.
3. To delete the account, you must have r (read), m (mgmt_info), a (auth_info), and u (user_info) permissions for the principal named in the account.
4. To delete the directory in CDS, you must have d (delete) permission to the directory and w (write) permission to the clearinghouse that stores the master replica of the directory. The server principal needs a (auth_info) permission to the parent directory or d (delete) permission to the child pointer that points to the directory you intend to delete.

Examples
The following example deletes user K_Parsons from the cell:

```bash
dcecp> user delete K_Parsons
dcecp>
```

**user help**

Returns help information about the user task object and its operations. The syntax is as follows:

```
user help [operation | -verbose]
```

**Options**

- `-verbose`
  Displays information about the user task object.

Used without an argument or option, the `user help` command returns brief information about each user operation. The optional `operation` argument is the name of an operation about which you want detailed information. Alternatively, you can use the `-verbose` option for more detailed information about the user task object itself.

**Privileges Required**

No special privileges are needed to use the `user help` command.

**Examples**

```bash
dcecp> user help
create  Creates a DCE user.
delete  Deletes a DCE user.
show    Shows the attributes of a DCE user.
help    Prints a summary of command-line options.
operations  Returns a list of the valid operations for this command.
dcecp>
```

**user operations**

Returns a list of the operations supported by the user task object. The syntax is as follows:

```
user operations
```

The list of available operations is in alphabetical order except for help and operations, which are listed last.

**Privileges Required**

No special privileges are needed to use the `user operations` command.

**Examples**

```bash
dcecp> user operations
create delete show help operations
dcecp>
```
user show

Returns the attributes of one or more DCE users. The syntax is as follows:

user show user_name_list

The show operation returns the attributes of the users named in user_name_list. The information returned includes principal attributes, account attributes, and policies. The information is returned as if the following commands were run in the following order:

principal show
account show -all

Privileges Required

You must have r (read) permission to the principal named in the account.

Examples

dcecp> user show K_Parsons
{fullname {}}
{uid 5129}
{uuid 00001409-a943-21cd-be00-0000c08adf56}
{alias no}
{quota unlimited}
{groups users}
{acctvalid yes}
{client yes}
{created /.../my_cell.goodco.com/cell_admin 1994-07-27-13:02:51.000+00:00I-----}
{description {}%
{dupkey no}
{expdate none}
{forwardabletkt yes}
{goodsince 1994-07-27-13:02:51.000+00:00I-----}
{group users}
{home /}
{lastchange /.../my_cell.goodco.com/cell_admin 1994-07-27-13:02:51.000+00:00I-----}
{organization users}
{postdatedtkt no}
{proxiabletkt no}
{pwdvalid yes}
{renewabletkt yes}
{server yes}
{shell {}}
{stdtgtauth yes}
{nopolicy

dcecp>

Related Information

Commands: account(8dce), dcecp(8dce), directory(8dce), group(8dce), organization(8dce), principal(8dce), xattrschema(8dce).
UTC

Purpose

A dcecp object that manipulates UTC timestamps

Synopsis

utc add timestamp relative_timestamp

utc compare absolute_timestamp absolute_timestamp [-noinaccuracy]

utc convert absolute_timestamp [-gmt]

utc help [operation | -verbose]

utc multiply relative_timestamp { integer | floating_point_factor }

utc operations

utc subtract timestamp timestamp

Arguments

absolute_timestamp
An International Organization for Standardization (ISO) compliant time format of the following form:

CCYY-MMDD-hh:mm:ss.fff[+][-]hh:mmIsss.fff

The Time Differential Factor (TDF) component [+][-]hh:mm, if present, indicates the offset from Universal Time Coordinated (UTC) time and implies local system time. The inaccuracy component Isss.fff, if present, specifies the duration of the time interval that contains the absolute time.

floating_point_factor
A floating-point number such as 53.234.

integer
A whole number such as 79.

operation
The name of the utc operation for which to display help information.

relative_timestamp
A Distributed Time Service (DTS) timestamp of the following form:

[-]DD-hh:mm:ss.fffIsss.fff

Relative times often omit fractions of seconds (the leftmost .fff sequence) and generally lack an inaccuracy component (Isss.fff). For example, a relative time of 21 days, 8 hours, and 15 minutes is expressed as 21-08:15:00.

timestamp
A utc timestamp that can be a relative or absolute time. See the relative_timestamp and absolute_timestamp argument descriptions for the format of these timestamps.
utc(8dce)

Description

The utc object lets you add, compare, and convert timestamps in DTS and ISO formats.

Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Operations

utc add

Adds two timestamps. The syntax is as follows:

```
utc add  timestamp  relative_timestamp
```

The add operation returns the sum of two timestamps. The timestamps can be two relative times or an absolute time and a relative time.

Privileges Required

No special privileges are needed to use the utc add command.

Examples

```
dcecp> utc add 1994-10-18-13:21:50.419-04:00I----- +0-00:02:00.000I-----
1994-10-18-13:23:50.419-04:00I-----
dcecp>
```

utc compare

Compares two absolute timestamps indicating the temporal order. The syntax is as follows:

```
utc compare  absolute_timestamp  absolute_timestamp  [-noinaccuracy]
```

Options

- **-noinaccuracy**
  Specifies to ignore inaccuracies in comparisons.

The compare operation compares two timestamps and returns -1 if the first is earlier, 1 if the second is earlier, and 0 if the difference is indeterminate. Specify the -noinaccuracy option to ignore inaccuracies in comparisons; in this case a return of 0 indicates the times are the same.

Privileges Required

No special privileges are needed to use the utc compare command.

Examples
utc convert

Converts a timestamp from UTC to local time. The syntax is as follows:

`utc convert absolute_timestamp [-gmt]`

Options

- `gmt` Specifies to return a Greenwich mean time (GMT) formatted timestamp.

The `convert` operation accepts a timestamp and returns another timestamp that expresses the same time in the local time zone. If called with the `-gmt` option it returns a Greenwich mean time (GMT) formatted timestamp.

Privileges Required

No special privileges are needed to use the `utc convert` command.

Examples

```
dcecp> utc convert 1994-10-18-13:22:32.816-00:00I----- \
> 1994-10-18-09:22:32.816-04:00I----- -noinaccuracy
1
dcecp>
```

utc help

Returns help information about the `utc` object and its operations. The syntax is as follows:

`utc help [operation | -verbose]`

Options

- `-verbose` Displays information about the `utc` object.

Used without an argument or option, the `utc help` command returns brief information about each `utc` operation. The optional `operation` argument is the name of an operation about which you want detailed information. Alternatively, you can use the `-verbose` option for more detailed information about the `utc` object itself.

Privileges Required

No special privileges are needed to use the `utc help` command.

Examples

```
dcecp> utc help
add      Adds a relative and absolute, or two relative, timestamps.
compare  Compares two timestamps to determine which is earlier.
convert  Converts a timestamp into the local timezone or GMT.
multiply Multiplies a relative timestamp by a number.
```
utc multiply

Multiplying a relative time (a length of time) by an integer or floating-point factor. The syntax is as follows:

```
utc multiply relative_timestamp { integer | floating_point_factor }
```

The multiply operation accepts two arguments: a relative timestamp and an integer or floating-point factor. It multiplies the length of time (specified by the relative timestamp) by the integer or floating-point factor, returning the product as a relative timestamp.

Privileges Required

No special privileges are needed to use the utc multiply command.

Examples

dcecp> utc multiply +0-00:00:05.000I----- 3
+0-00:00:15.000I-----
dcecp>

utc operations

Returns a list of the operations supported by the utc object. The syntax is as follows:

```
utc operations
```

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the utc operations command.

Examples

dcecp> utc operations
add compare convert multiply subtract help operations
dcecp>

utc subtract

Subtracts one timestamp from another, returning the difference as a relative timestamp. The syntax is as follows:

```
utc subtract timestamp timestamp
```

The subtract operation returns the difference between two timestamps that express either an absolute time and a relative time, two relative times, or two absolute times. The return value is a relative timestamp.
Privileges Required

No special privileges are needed to use the utc subtract command.

Examples

dcecp> utc subtract 1994-10-18-13:22:32.816-00:00I----- +0-00:00:15.000I-----
dcecp>

Related Information

Commands: clock(8dce), dcecp(8dce), dts(8dce), dtsd(8dts).
uuid

Purpose
A dcecp object that generates and compares UUIDs

Synopsis

uuid compare uuid uuid
uuid create
uuid help [operation | -verbose ]
uuid operations

Arguments

uuid A UUID in the following form:
069d9fb6-943e-11cd-a35c-0000c08adf56

operation
The name of the uuid operation for which to display help information.

Description
The uuid object generates and compares Universal Unique Identifiers (UUIDs). UUIDs uniquely identify DCE entities such as principals, RPC entries, Cell Directory Service (CDS) replicas, and so on.

Errors
A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Operations

uuid compare

Compares two UUIDs. The syntax is as follows:

uuid compare uuid uuid

The compare operation compares two UUIDs, returning 1 if they are equal or 0 if they are not. Because the uuid compare command handles the comparison of UUIDs in current and previous DCE formats, you should use it rather than string compare.

Privileges Required
No special privileges are needed to use the uuid compare command.

Examples
uuid create

Returns a newly generated UUID. The syntax is as follows:

uuid create

The create operation returns a newly generated UUID. It takes no arguments.

Privileges Required

No special privileges are needed to use the uuid create command.

Examples

dcecp> uuid create 03bb2688-943e-11cd-8bfd-0000c08adf56
069d9fb6-943e-11cd-a35c-0000c08adf56

dcecp>

uuid help

Returns help information about the uuid object and its operations. The syntax is as follows:

uuid help  [operation | -verbose]

Options

-verbose
  Displays information about the uuid object.

Used without an argument or option, the uuid help command returns brief information about each uuid operation. The optional operation argument is the name of an operation about which you want detailed information. Alternatively, you can use the -verbose option for more detailed information about the uuid object itself.

Privileges Required

No special privileges are needed to use the uuid help command.

Examples

dcecp> uuid help
  compare  Compares two UUIDs for equality.
  create   Returns a newly generated UUID.
  help     Prints a summary of command-line options.
  operations Returns a list of the valid operations for this command.

dcecp>

uuid operations

Returns a list of the operations supported by the uuid object. The syntax is as follows:
uuid(8dce)

uuid operations

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the uuid operations command.

Examples

dcecp> uuid operations
compare create help operations
dcecp>

Related Information

Commands: dcecp(8dce), endpoint(8dce).
xattrschema

Purpose

A dcecp object that manages schema information for extended registry attributes (ERAs)

Synopsis

xattrschema catalog schema_name [- simplesname]

xattrschema create schema_entry_name_list [-attribute attribute_list | -attributevalue ]

xattrschema delete schema_entry_name_list

xattrschema help [operation | -verbose ]

xattrschema modify schema_entry_name_list [-change attribute_list | -attributevalue ]

xattrschema operations

xattrschema rename schema_entry_name -to new_schema_entry_name

xattrschema show schema_entry_name_list

operation
    The name of the xattrschema operation for which to display help information.

schema_entry_name
    The name of a single schema entry type.

schema_entry_name_list
    A list of one or more schema entry types to act on.

schema_name
    The name of the schema that defines the schema entry types named in
    schema_entry_name_list. Two schemas are currently supported:

    /.../cell_name/sec/xattrschema

    /.../cell_name/hosts/hostname/config/xattrschema

Description

The xattrschema object represents the schema information for an extended registry attribute (ERA). This command manipulates the schema type that defines ERAs. Schema types are identified by name. Other dcecp commands manipulate individual instances of ERAs. ERA instances are an attribute of a given schema type that has been attached to an object and assigned a value.

You can attach ERAs to principal, group, and organization objects and to server configuration and server execution objects supported by dced.
ERA entry types for principal, group, and organization objects have the following default name:

```
./:/sec/xattrschema/schema_entry_name
```

ERA types for dced server objects have the following name:

```
./:/hosts/hostname/config/xattrschema/schema_entry_name
```

ERA types are defined to be attached to only those objects supported by specified ACL managers.

### Attributes

- **acmgr description**
  A set that lists the ACL managers that support the object types on which ERAs of this type can be created. For each ACL manager type, the permissions required for attribute operations are also specified. Each ACL manager is described with a list, in the following format:

  ```
  {uuid queryset updateset testset deleteset}
  ```

  where the first element is the Universal Unique Identifier (UUID) of the ACL manager, and the rest are the sets of permissions (concatenated permission strings as found in an ACL) required to perform each type of operation. The value of this attribute is actually a list of these lists. For example:

  ```
  {8680f026-2642-11cd-9a43-080009251352 r w t D}
  {18dbdad2-23df-11cd-82d4-080009251352 r w t mD}
  ```

  This attribute is modifiable after creation, but only in a limited way. New ACL managers can be added, but existing ones cannot be removed or changed. This attribute must be specified on creation.

- **annotation string**
  A comment field used to store information about the schema entry. It is a Portable Character Set (PCS) string. The default is an empty string (that is, blank).

- **applydefs {yes | no}**
  Indicates that if this ERA does not exist for a given object on an attribute query, the system-defined default value (if any) for this attribute will be returned. If set to no, an attribute query returns an attribute instance only if it exists on the object named in the query. The value of this attribute must be yes or no. The default is no. When replacing an ERA schema, the applydefs attribute (sec_attr_sch_entry_use_defaults) is only advisory. Even if this attribute is set to yes for an ERA schema, it behaves as if it were set to no. The DCE code does not provide a default value for an ERA if the ERA is not explicitly attached to an object.

- **encoding type**
  The type of the ERA. This attribute cannot be modified after creation, and must be specified on creation. Legal values are one of the following:

  - **any** The value of the ERA can take on any encoding. This encoding type is only legal for the definition of an ERA in a schema entry. All instances of an ERA must have an encoding of some other value.
attrset

The value of the ERA is a list of attribute type UUIDs used to retrieve multiple related attributes by specifying a single attribute type on a query.

Handling of attribute set encoding for ERA schemas is not completely supported. Setting attr_set (sec_attr_enc_attr_set) encoding in an ERA schema allows for grouping a set of schema UUID under one unique UUID. The sec_rgy_attr_lookup_by_id() API currently behaves like sec_rgy_lookup_no_expand(). The seamless expansion of the attribute set into its components is not currently supported.

binding

The value of the ERA contains authentication, authorization, and binding information suitable for communicating with a DCE server. The syntax is a list of two elements.

The first element is a list of security information in which the first element is the authentication type, either none or dce, followed by information specific for each type. The type none has no further information. The type dce is followed by a principal name, a protection level (default, none, connect, call, pkt, pktinteg, or pktprivacy), an authentication service (default, none, or secret), and an authorization service (none, name, or dce). Examples of three security information lists are as follows:

{none}
{dce /:/melman default default dce}
{dce /:/melman pktprivacy secret dce}

The second element is a list of binding information, in which binding information can be string bindings or server entry names. Two examples of binding information are as follows:

{/.:/hosts/hostname/dce-entity
 /.:/subsys/dce/sec/master}
{ncadg_ip_udp:130.105.96.3
 ncadg_ip_udp:130.105.96.6}

byte

The value of the ERA is a string of bytes. The byte string is assumed to be pickle or is otherwise a self-describing type.

It is unlikely that attributes of this type will be entered manually. The format of output is hexadecimal bytes separated by spaces with 20 bytes per line. For example, the input attribute name bindata might produce the following output:

{bindata
 {00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13
  22 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 12 11 12 13}}

The braces indicate that bindata has one value. On input all whitespace is compressed so that users can enter the data as bytes or words or any combination, whichever is more convenient. Therefore, a user could enter the following:

{bindata
 {00010203 0405 06070809 0a0b 0c0d0e0f 10111213
  22212223 2425 26272829 2a2b 2c2d2e2f 12111213}}
i18ndata

The value of the ERA is a string of bytes with a tag identifying the (IBM registered) codeset used to encode the data.

Although it is unlikely that administrators will enter attributes of this type manually, the DCE control program does support entering binary data via the following notations: \ddd where \ddd can be one, two, or three octal digits, and \x hh where hh can be any number of hexadecimal digits.

integer

The value of the ERA is a signed 32-bit integer.

printstring

The value of the ERA is a printable Interface Definition Language (IDL) character string using Portable Character Set (PCS).

stringarray

An array of PCS strings; represented as a Tcl list of strings.

uuid

The value of the ERA is a UUID.

void

The ERA has no value. It is simply a marker that is either present or absent.

intercell

Specifies the action that should be taken by the privilege server when reading ERAs from a foreign cell. Possible values are as follows:

accept

Accepts ERAs from foreign cells. The only check applied is uniqueness if indicated by the unique attribute.

reject

Discards ERAs from foreign cells.

evaluate

Invokes a trigger function to a server that would decide whether the ERA should be kept, discarded, or mapped to another value.

The default is reject.

When creating an ERA schema, the intercell attribute is only advisory. Even if this attribute is set to accept (sec_attr_intercell_act_accept) or evaluate (sec_attr_intercell_act_evaluate), it behaves as if it were set to reject (sec_attr_intercell_act_reject). The DCE code will discard all ERA values for a principal, group, or organization when a principal’s EPAC is used for intercell access.

multivalued {yes | no}

Indicates that ERAs of this type can be multivalued (that is, multiple instances of the same attribute type can be attached to a single registry object). The value of this attribute must be yes or no. This attribute cannot be modified after creation. The default is no.

reserved {yes | no}

If set, this schema entry can not be deleted through any interface by any user. The value of this attribute must be yes or no. The default is no.

scope string

Indicates the name of a security directory or object in the registry. If it is an object, instances of this ERA can be attached only to this object. If it is a directory, instances of this ERA can be attached only to descendants of this
directory. The default is an empty string, which does not limit which objects ERAs can be attached to. For example, if this attribute is set to principal/org/dce only principals with a prefix of org/dce in the name can have this type of ERA. You cannot modify this attribute after it is created. The default is the empty string (that is, blank).

This attribute is only advisory in DCE Version 1.1. Future versions of DCE will support this functionality.

**trigtype type**
Identifies whether there is a trigger and if so what type it is. The possible values are: none, query, and update. If this attribute is anything other than none, then trigbind must be set. This attribute cannot be modified after creation. The default is none.

When creating an ERA schema, the update value (sec_attr_trig_type_update) for the trigtype attribute is not implemented. Update triggers for ERA schemas give the ability to check with a registered server before an ERA value is updated, but this function is not yet implemented.

**trigbind binding**
Contains binding information for the server that will support the trigger operations. This field must be set if trigtype is not none or if intercell is set to evaluate. The value of this attribute is of the format described by the binding encoding type. The default is the empty string (that is, blank).

**unique {yes | no}**
Indicates that each instance of the ERA must have a unique value within the cell for a particular object type (for instance, principal). The value of this attribute must be yes or no. This attribute cannot be modified after creation. The default is no.

When creating an ERA schema, the unique attribute (sec_attr_sch_entry_unique) is only advisory. Even if this attribute is set to yes for an ERA schema, it behaves as if it were set to no. The DCE code does not check or enforce the uniqueness of the ERA value attached to objects.

**uuid uuid**
The internal identifier of the ERA. The value is a UUID. This attribute cannot be modified after creation. If not specified on the create operation, a value is generated by the system.

See the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide* for more information about xattrschema attributes.

**Errors**

A representative list of errors that might be returned is not shown here. Refer to the *IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide* for complete descriptions of all error messages.

**Operations**

**xattrschema catalog**
Returns a list of all the schema entry types defined in the specified schema. The syntax is as follows:
xattrschema(8dce)

xattrschema catalog  

**Options**

- **-simplename**
  Returns only the residual part of the schema name.

The **catalog** operation returns a list of the names of all the schema entry types defined in the named schema. Use the **-simplename** option to return only the residual part of the names, instead of the fully qualified names.

**Privileges Required**

You must have r (read) permission to the schema container object (/./sec/xattrschema or ./hosts/hostname/config/xattrschema).

**Examples**

dcecp> xattrschema catalog ././sec/xattrschema
/.../my_cell/sec/xattrschema/pre_auth_req
/.../my_cell/sec/xattrschema/pwd_val_type
/.../my_cell/sec/xattrschema/pwd_mgmt_binding
/.../my_cell/sec/xattrschema/X500_DN
/.../my_cell/sec/xattrschema/X500_DSA_Admin
/.../my_cell/sec/xattrschema/disable_time_interval
/.../my_cell/sec/xattrschema/max_invalid_attempts
/.../my_cell/sec/xattrschema/passwd_override
/.../my_cell/sec/xattrschema/test_any
/.../my_cell/sec/xattrschema/test_void
/.../my_cell/sec/xattrschema/test_printstring
/.../my_cell/sec/xattrschema/test_printstring_array
/.../my_cell/sec/xattrschema/test_integer
/.../my_cell/sec/xattrschema/test_bytes
/.../my_cell/sec/xattrschema/test_i18n_data
/.../my_cell/sec/xattrschema/test_uuid
/.../my_cell/sec/xattrschema/test_attr_set
/.../my_cell/sec/xattrschema/test_binding

dcecp>

**xattrschema create**

Creates a new schema entry type. The syntax is as follows:

xattrschema create  

**Options**

- **-attribute value**
  As an alternative to using the **-attribute** option with an attribute list, you can change individual attribute options by prepending a hyphen (-) to any attributes listed in Attributes.

- **attribute_list**
  Allows you to specify attributes by using an attribute list rather than using the **-attribute value** option. The format of an attribute list is as follows:

  \{attribute value\}...

The **create** operation creates a new schema entry for an ERA. The argument is a list of one or more names of schema entry types to be created. Attributes for the
created schema entry types can be specified via attribute lists or attribute options. If
the command argument contains more than one schema name, you cannot specify
a UUID attribute. All attributes are applied to all entry types to be created. This
operation returns an empty string on success.

Privileges Required

You must have i (insert) permission to the container object (/./sec/xattrschema or
././hosts/ hostname/config/xattrschema).

Examples

dcecp> xattrschema create /./sec/xattrschema/test_integer \
> -encoding integer -aclmgr {group r r r}
dcecp>

xattrschema delete

Deletes a schema entry type. The syntax is as follows:

xattrschema delete schema_entry_name_list

The delete operation deletes a schema entry. The argument is a list of names of
schema entry types to be deleted. This command also deletes all ERA instances of
the schema entry. If the entry types do not exist, an error is generated. This
operation returns an empty string on success.

Privileges Required

You must have d (delete) permission to the container object (/./sec/xattrschema
or ././hosts/ hostname/config/xattrschema).

Examples

dcecp> xattrschema delete /./sec/xattrschema/test_integer
dcecp>

xattrschema help

Returns help information about the xattrschema object and its operations. The
syntax is as follows:

xattrschema help [operation | -verbose]

Options

-verbose

Displays information about the xattrschema object.

Used without an argument or option, the xattrschema help command returns brief
information about each xattrschema operation. The optional operation argument is
the name of an operation about which you want detailed information. Alternatively,
you can use the -verbose option for more detailed information about the
xattrschema object itself.

Privileges Required
xattrschema(8dce)

No special privileges are needed to use the xattrschema help command.

Examples

dcecp> xattrschema help
 catalog Returns a list of all entries in a schema.
 create Creates a schema entry.
 delete Deletes a schema entry.
 modify Modifies an existing schema entry.
 rename Renames an existing schema entry.
 show Returns the attributes of a schema entry.
 help Prints a summary of command-line options.
 operations Returns a list of the valid operations for this command.

dcecp>

xattrschema modify

This operation changes the attributes of a schema entry type. The syntax is as follows:

xattrschema modify  schema_entry_name_list
 {-change attribute_list | -attribute value} 

Options

- attribute value
  As an alternative to using options with an attribute list, you can change
  individual attribute options by prepending a hyphen (-) to any attributes
  listed in Attributes.

- change attribute_list
  Allows you to modify attributes by using an attribute list rather than using
  individual attribute options. The format of an attribute list is as follows:

  {{{attribute value}...{attribute value}}} 

  See Attributes for descriptions of the attributes.

The modify operation changes attributes of schema entry types in the security
service only. The argument is a list of names of schema entry types to be operated
on. All modifications are applied to all schema entry types named in the argument.
Schema entry types are modified in the order they are listed, and all modifications
to an individual schema entry are atomic. Modifications to multiple schema entry
types are not atomic. A failure for any one schema entry in a list generates an error
and aborts the operation. This operation returns an empty string on success.

The -change option modifies attributes. Its value is an attribute list describing the
new values for the specified attributes. The command supports attribute options as
well.

Privileges Required

You must have m (mgmt_info ) permission to the container object
/./sec/xattrschema.

Examples
xattrschema operations

Returns a list of the operations supported by the xattrschema object. The syntax is as follows:

xattrschema operations

The list of available operations is in alphabetical order except for help and operations, which are listed last.

Privileges Required

No special privileges are needed to use the xattrschema operations command.

Examples

dcecp> xattrschema operations
catalog create delete modify rename show help operations
dcecp>

xattrschema rename

Changes the name of a specified schema entry type. The syntax is as follows:

xattrschema rename schema_entry_name -to new_schema_entry_name

Options

-to new_schema_entry_name
    Specifies the new name. Specify the name in simple format, without the container-object portion (that is, without ./sec/xattrschema).

The rename operation changes the name of a specified ERA in the security service only. The argument is a single name of an ERA to be renamed. The new_schema_entry_name argument to the required -to option specifies the new name; this argument cannot be a list. This operation returns an empty string on success.

Privileges Required

You must have m (mgmt_info) permission to the container object ./sec/xattrschema.

Examples

dcecp> xattrschema rename ./sec/xattrschema/test_integer -to test_int
dcecp>

xattrschema show

Returns an attribute list describing the specified schema entry type. The syntax is as follows:
xattrschema(8dce)

xattrschema show  schema_entry_name_list

The show operation returns an attribute list describing the specified schema entry types. The argument is a list of names of schema entry types to be operated on. If more than one schema entry is given, the attributes are concatenated. Attributes are returned in arbitrary order.

Privileges Required

You must have r (read) permission to the container object (/.:/sec/xattrschema or /.:/hosts/ hostname/config/xattrschema).

Examples

dcecp> xattrschema show /.:/sec/xattrschema/test_integer
{aclmgr {principal {{query r} {update r} {test r} {delete r}}}}
{annotation {test_integer: encoding type integer}}
{applydefs yes}
{encoding integer}
{intercell reject}
{multivalued yes}
{reserved no}
{scope {}}
{trigbind {none {}}}
{trigtype none}
{unique no}
{uuid 5f439154-2af1-11cd-8ec3-080009353559}
dcecp>

Related Information

Commands: account(8dce), dcecp(8dce), group(8dce), organization(8dce), principal(8dce).
Purpose

Introduction to the DCE RPC programmer commands

Description

The DCE remote procedure call (RPC) component provides the following programmer commands:

**idl**
Invokes the Interface Definition Language (IDL) compiler to convert an interface definition, written in IDL, to output files. The output files include a header file, a server stub file, and a client stub file.

**idl –spmi**
Modifies the IDL compiler so that it automatically generates RPC instrumentation sensors in IDL stub files.

**rpcprotseq**
Determines the supported protocol sequences on a given host and prints them to standard output (stdout).

**rpcresolve**
Recursively resolves the elements of a namespace entry.

**uuidgen**
Creates a Universal Unique Identifier (UUID) string that you assign to an object to uniquely distinguish it from other objects.

See each command's reference page for further information.

**IDL Base Data Types and IDL-to-C**

The following table lists the IDL base data type specifiers. Where applicable, the table shows the size of the corresponding transmittable type and the type macro emitted by the IDL compiler for resulting declarations.

Note that you can use the `idl_` macros in the code you write for an application to ensure that your type declarations are consistent with those in the stubs, even when the application is ported to another platform. The `idl_` macros are especially useful when passing constant values to RPC calls. For maximum portability, all constants passed to RPC calls declared in your network interfaces should be cast to the appropriate type because the size of integer constants (like the size of the `int` data type) is unspecified in the C language.

The `idl_` macros are defined in `dce/idlbase.h`, which is included by header files that the IDL compiler generates.

<table>
<thead>
<tr>
<th>Specifier</th>
<th>Type Macro Emitted by idl</th>
</tr>
</thead>
<tbody>
<tr>
<td>small</td>
<td><code>idl_small_int</code></td>
</tr>
<tr>
<td>short</td>
<td><code>idl_short_int</code></td>
</tr>
<tr>
<td>long</td>
<td><code>idl_long_int</code></td>
</tr>
<tr>
<td>hyper</td>
<td><code>idl_hyper_int</code></td>
</tr>
</tbody>
</table>

Table 1. Base Data Type Specifiers—rpc_intro(1rpc)
Table 1. Base Data Type Specifiers—rpc_intro(1rpc) (continued)

<table>
<thead>
<tr>
<th>Specifier</th>
<th>(sign)</th>
<th>(size)</th>
<th>(type)</th>
<th>Size</th>
<th>Type Macro</th>
<th>Emitted by idl</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned small int</td>
<td>unsigned</td>
<td>small</td>
<td>int</td>
<td>8 bits</td>
<td>idl_usmall_int</td>
<td></td>
</tr>
<tr>
<td>unsigned short int</td>
<td>unsigned</td>
<td>short</td>
<td>int</td>
<td>16 bits</td>
<td>idl_ushort_int</td>
<td></td>
</tr>
<tr>
<td>unsigned long int</td>
<td>unsigned</td>
<td>long</td>
<td>int</td>
<td>32 bits</td>
<td>idl_ulong_int</td>
<td></td>
</tr>
<tr>
<td>unsigned hyper int</td>
<td>unsigned</td>
<td>hyper</td>
<td>int</td>
<td>64 bits</td>
<td>idl_uhyper_int</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td></td>
<td></td>
<td>float</td>
<td>32 bits</td>
<td>idl_short_float</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td></td>
<td></td>
<td>double</td>
<td>64 bits</td>
<td>idl_long_float</td>
<td></td>
</tr>
<tr>
<td>char</td>
<td></td>
<td></td>
<td>char</td>
<td>8 bits</td>
<td>idl_char</td>
<td></td>
</tr>
<tr>
<td>Boolean</td>
<td></td>
<td></td>
<td>Boolean</td>
<td>8 bits</td>
<td>idl_boolean</td>
<td></td>
</tr>
<tr>
<td>byte</td>
<td></td>
<td></td>
<td>byte</td>
<td>8 bits</td>
<td>idl_byte</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td></td>
<td></td>
<td>void</td>
<td>—</td>
<td>idl_void_p_t</td>
<td></td>
</tr>
<tr>
<td>handle_t</td>
<td></td>
<td></td>
<td>handle_t</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

Related Information

Commands: idl(1rpc), uuidgen(1rpc).

invokes the Interface Definition Language (IDL) compiler

Synopsis

`idl filename [options]`

Options

- `client file_type`
  Determines which client files to generate. If you do not specify this argument, the compiler generates all client files. The file types are as follows:
  
  none  Does not generate client files.
  stub  Generates only a client stub file.
  aux  Generates only a client auxiliary file. A client auxiliary file is generated only if the interface contains any out-of-line or self-pointing types. For more information, see the caution notes in the Description section of this reference page.
  all  Generates client stub and client auxiliary files. This is the default and is the same as not specifying the -client argument.

- `server file_type`
  Determines which server files to generate. If you do not specify this argument, the compiler generates all server files. The file types are as follows:
  
  none  Does not generate server files.
  stub  Generates only a server stub file.
  aux  Generates only a server auxiliary file. A server auxiliary file is generated only if the interface contains any out-of-line, self-pointing, or pipe types. For more information, see the caution notes in the Description section of this reference page.
  all  Generates server stub and server auxiliary files. This is the default and is the same as not specifying the -server argument.

- `lang language`
  Specifies which language to use to generate header and intermediate stub files. The valid languages are as follows:
  
  c  Generates C files. This is the default and is the same as not specifying the -lang argument.
  cxx  Generates C++ files.

- `no_cxxmgr`
  Causes the compiler to not overwrite the manager class header file. Use this argument if you implement application-specific C++ code in the manager class header file.
-cstub filename
 Specifies a pathname for the client stub file. When you give a filename, do not give a file extension; the idl compiler appends .c to the C source file and .o to the object file. If you do not use the -cstub argument, the idl compiler appends _cstub.c to the C source file and _cstub.o to the object file. If the -lang cxx option is used, the source file has a .Cxx extension.

-sstub filename
 Specifies a pathname for the server stub file. When you give a filename, do not give a file extension; the idl compiler appends .c to the C source file and .o to the object file. If you do not use the -sstub argument, the idl compiler appends _sstub.c to the C source file and _sstub.o to the object file. If the -lang cxx option is used, the source file has a .Cxx extension.

-caux filename
 Specifies a pathname for the client auxiliary file. When you give a filename, do not give a file extension; the idl compiler appends .c to the C source file and .o to the object file. If you do not use the -caux argument, the idl compiler appends _caux.c to the C source file and _caux.o to the object file.

This option allows makefile compatibility with OSF DCE Release 1.0.2 and earlier releases. For more information, see the caution notes in the Description section of this reference page.

-saux filename
 Specifies a pathname for the server auxiliary file. When you give a filename, do not give a file extension; the idl compiler appends .c to the C source file and .o to the object file. If you do not use the -saux argument, the idl compiler appends _saux.c to the C source file and _saux.o to the object file.

This option allows makefile compatibility with OSF DCE Release 1.0.2 and earlier releases. For more information, see the caution notes in the Description section of this reference page.

-header header_file
 Allows you to specify a name for the generated header file. By default the compiler takes the basename of the IDL file and appends the .h extension to it.

-out directory
 Places the output files in the directory you specify. By default the compiler places the output files in the current working directory.

-I directory
 Specifies a directory name that contains imported interface definition files. You can specify more than one directory by specifying additional -I directory arguments on the command line. The compiler searches the directories in the order you list them. If a file is present in more than one directory, the compiler takes the first occurrence of the file. The default behavior of the compiler is to first search the current directory, then all directories you specify, then the system IDL directory. The directory you specify is also passed to the language preprocessors and compilers.

-no_def_idir
 Specifies that the compiler search only the current directory for imported

Chapter 2. Remote Procedure Call Commands 369
files. When you use this with -I<directory>, the compiler searches only the
directories you list, not the current directory, and not the system IDL
directory.

-no_mepv
Causes the compiler to not generate a manager entry point vector (EPV) in
the server stub. Use this argument if the manager code and IDL file do not
use the same operation names. If you specify this argument you must
provide an EPV within the manager code that can be used when the
interface is registered with the remote procedure call (RPC) server runtime.
The name of the type that you construct an EPV with is if_name_v
major-version_minor-version_epv_t where if_name is the interface name. It
is not necessary to use this argument if the operation names in the
manager code and IDL file are the same. In this case, the compiler
generates a manager EPV in the server stub by using the names of the
operations in the IDL file. (For information on registering the server, see the
rpc_intro(3rpc) and rpc_server_register_if(3rpc) reference pages. See
also the IBM DCE Version 3.1 for AIX and Solaris: Application Development
Guide—Core Components.)

-cepv
Generates local routines in the client stub file (filename_cstub.c) and
defines a client entry point vector (CEPV) of the name if_name_v
major-version_minor-version_c_epv where if_name is the interface name.
The CEPV contains the addresses of the local routines. The client code
must call the routines indirectly by using the addresses in the CEPV;
otherwise, the stub routines in the client stub file must have the same
names as the operations in the IDL file. (For information on registering the
server, see the rpc_intro(3rpc) and rpc_server_register_if(3rpc)
reference pages.) See also the IBM DCE Version 3.1 for AIX and Solaris:
Application Development Guide—Core Components.)

-cpp_cmd 'c_preprocessor_command_line'
Allows you to specify a language preprocessor other than the default. The
compiler invokes the preprocessor found in that command line. The output
of the preprocessor is an expanded version of the input file(s) containing
replacement text for any preprocessor directives (for example, the #include
preprocessor directive).

-cpp_opt 'command_options'
Specifies additional options to be passed to the language preprocessor. You
can add options to the command line used to invoke the preprocessor
independent of the -cpp_cmd argument. The IDL compiler concatenates
the -cpp_cmd, -cpp_opt, -D, -U, and -I arguments and the source filename
into a command used to invoke the preprocessor.

The compiler repeats this process for each Attribute Configuration File
(ACF) and IDL file.

-no_cpp
Does not invoke the language preprocessor. Note that the preprocessor
must be run on files that contain preprocessor directives (such as #include)
in the interface definition.

-cc_cmd 'command_line'
Invokes the language compiler options you specify in the 'command_line'
argument rather than the default compiler and compiler options.
-cc_opt 'command_options'

Specifies additional options to be passed to the C or C++ compiler. You can add options to the command line used to invoke the compiler independent of the -cc_cmd argument. The IDL compiler concatenates the -cc_cmd, -cc_opt, and -I arguments and the source filename into a command that invokes the language compiler. This procedure is done for each generated stub or auxiliary file.

-Dname [= definition]

Defines a symbol name and an optional value to be passed to the language preprocessor. You can use this method of defining a symbol instead of using #define in the source code. You can use more than one -Dname argument on the command line. This argument has no effect if you use the -no_cpp argument.

-Uname

Removes (undefines) any initial definition of a symbol name as defined by -Dname. You can use this method to remove a symbol name instead of using #undef in the source code. You can use more than one -Uname argument on the command line. This argument has no effect if you use the -no_cpp argument. If you define and undefine a name on the same command line, undefining takes precedence.

-space_opt

Generates code for the marshalling and unmarshalling of data that is optimized for space, rather than speed.

-syntax_only

Checks only the syntax of the IDL file, but does not generate any output files.

-keep file_types

Specifies which files to retain. To produce the object modules, the IDL compiler first creates C or C++ source modules, then invokes the target compiler to produce object modules, and finally, deletes the source modules. If you do not use -keep, only the object modules are saved.

The file types are as follows:

none   Does not save the source or the object modules. Does not invoke the language compiler.

c_source   Saves only the source modules. Does not invoke the language compiler.

object   Saves only the object modules.

all   Saves both the source and the object modules.

-bug n, -no_bug n

Retains (-bug) or does not retain (-no_bug) a specified bug from earlier IDL compiler versions. (This is an NCS compatibility argument and is not supported in IBM DCE 3.1.)

-stdin

Takes the standard output of a previous utility as the input to the idl command. For example:

cat my_filename.idl | idl -stdin
The `idl` command invokes the Interface Definition Language (IDL) compiler to convert an interface definition, written in IDL, into output files. The possible output files include a header file, server stub file, client stub file, auxiliary files, and a manager class header file. The compiler constructs the names of the output files by keeping the basename of the interface definition source file but replacing the filename extension with the new extension (or suffix and extension) appropriate to the newly generated type of output file. For example, `math.idl` could produce `math_sstub.c` or `math_sstub.o` for the server stub.

The `idl` command accepts the following input:

1. An interface definition filename.
2. Arguments to indicate either special actions to be performed by the compiler, or special properties of the input or output files.

The IDL compiler searches through directories for any related Attribute Configuration File (ACF). For example, if you compile a file named `source.idl`, the compiler automatically searches for a file named `source.acf`. The compiler also searches for any imported IDL file (and its related ACF). The compiler searches for these files in the following order:

1. The current working directory. The compiler always searches this directory unless you specify the `-no_def_idir` and `-I` directory arguments together.
2. Any imported directory. The compiler searches each directory you are specifying in the `-I` directory argument.
3. The system IDL directory. The compiler automatically imports `nbase.idl`, which resides in the system IDL directory. The compiler always searches this directory unless you specify the `-no_def_idir` argument.
4. The directory specified in the source filename. If you explicitly specify a directory in the source IDL pathname, then that directory is searched for the corresponding ACF. For example, the following command causes the IDL compiler to look for `/path/pathname/my_source.acf` if `my_source.acf` is not found in the directories in 1 through 3 above:

```
idl /path/pathname/my_source.idl
```

Note that this directory is not searched for any imported IDL file or its corresponding ACF.
Restrictions

The following filenames are reserved by the IDL compiler. Naming an IDL file with one of these names might result in unexpected behavior.

<table>
<thead>
<tr>
<th>Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>ivector.idl</td>
</tr>
<tr>
<td>lbase.idl</td>
</tr>
<tr>
<td>nbase.idl</td>
</tr>
<tr>
<td>ncastat.idl</td>
</tr>
<tr>
<td>ndroid.idl</td>
</tr>
<tr>
<td>rpc.idl</td>
</tr>
<tr>
<td>rpcbase.idl</td>
</tr>
<tr>
<td>rpcpvt.idl</td>
</tr>
<tr>
<td>rpcsts.idl</td>
</tr>
<tr>
<td>rpctypes.idl</td>
</tr>
<tr>
<td>twr.idl</td>
</tr>
<tr>
<td>uuid.idl</td>
</tr>
</tbody>
</table>

CAUTION:

When the IDL compiler generates C code, it is ANSI C code. It also supports C compilers that are not fully ANSI compliant although a warning message might occur during compilation of the stubs by the C compiler. A C compiler that is not fully ANSI compliant might generate the following warning messages:

```
warning: & before array or function: ignored
warning: enumeration type clash, operator =
```

CAUTION:

Makefiles created before OSF DCE Release 1.0.3 can produce a compiler warning if they reference .caux.o or .saux.o (auxiliary) files. You can use these Makefiles without alteration and avoid warnings by forcing IDL to generate empty aux files. In the C shell, set the IDL_GEN_AUXFILES environment variable as follows:

```
setenv IDL_GEN_AUXFILES 1
```

Examples

1. Invoke the IDL compiler to compile the interface definition file `test.idl` and keep the generated C source modules. Only server files are generated. The server stub default filename is overridden by creating a file named `test_ss.c` for the server stub module.

   `idl test.idl -keep c_source -client none -sstub test_ss.c`

2. Invoke the IDL compiler to compile the interface definition file `test.idl`, but do not run the C preprocessor. The manager entry point vector is not defined in the generated server stub module. The IDL compiler searches the parent directory of the current directory for any IDL files that `test.idl` could import. The generated output files are located in the `output` subdirectory under the current directory.

   `idl test.idl -no_cpp -no_mepv -I.. -out./output`

Errors

A representative list of errors that might be returned is not shown here. Refer to the `IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide` for complete descriptions of all error messages.

Files

```
/lib/cpp
```

C preprocessor
idl(1rpc)

`dceshared/bin/idl`
Compiler

`dceshared/include/dce`
System IDL directory for imported files

`dceshared/include/dce/nbase.idl`
Predefined IDL types

`dceshared/nls/msg/LANG/idl.cat`
Compiler error messages

`dceshared/share/include/file.ext`
All .idl or .h files that are part of DCE RPC

Related Information

idl –spmi

Purpose

Modifies the Interface Definition Language (IDL) compiler so that it automatically generates RPC instrumentation sensors in IDL stub files.

Synopsis

idl –spmi[ option_string]

Options

–spmi Specifies that one or more System Performance Monitor Interfaces (SPMI) instrumentation options (c and s) are to be included on the IDL command line.

c Specifies that SPMI instrumentation code is added to the client stub and that an SPMI header file is generated for the client stub. One or more of the record options (e, m, or p) must be specified along with the c option.

s Specifies that SPMI instrumentation code is added to the server stub and that an SPMI header file is generated for the server stub. One or more of the record options (e, m, or p) must be specified along with the s option.

e Specifies to record the elapsed time of the entire remote procedure call. If you specify the e or m option, you can also use q.

m Specifies to record the elapsed time for marshalling and unmarshalling. If you specify the e or m option, you can also use q.

q Specifies to calculate the sums of squares of the elapsed times. The elapsed times are in nanoseconds.

p Specifies to record the RPC counts by protocol type (ncacn_ip_tcp and nacdg_ip_udp) and by the total of both types.

Description

The idl –spmi command modifies the IDL compiler so that it automatically generates RPC instrumentation sensors in IDL stub files. The spmi API from the Performance Toolkit/6000 is used for the initial collection mechanism. After applications are compiled with the instrumented stubs, they are Dynamic Data Supplier (DDS) applications.

The c or s option must be included in the option_string. Both can be included. One or more of the record options (e, m, or p) must be included in the option_string. The sum of squares calculation option, q, can be used only if an elapsed time option (e, m, or both) is also included in the option_string.

Privileges Required

No special privileges are needed to use the idl –spmi command.

Examples

None
idl –spmi

Related Information

Commands: idl

uuidgen

Purpose

Generates a Universal Unique Identifier (UUID)

Synopsis

uuidgen [options]

Options

-c Allows you to supply an existing UUID that uuidgen then outputs in the format you specify. This option is especially useful in combination with the -s option for converting an existing UUID into a C structure.

You must specify the -c option at the end of the uuidgen command line; all options that follow -c are ignored.

-i Produces an Interface Definition Language (IDL) file template and includes the generated UUID string in the template.

-o filename Redirects the generated UUID string to the file you specify.

-s Generates a UUID string as an initialized C structure.

-v Displays the version number of the UUID generator, but does not generate a UUID.

-h Displays information about the uuidgen command arguments. The arguments -h and -? can be used interchangeably.

-? Displays information about the uuidgen command arguments. The arguments -? and -h can be used interchangeably.

-n number_of_uuid_strings Generates a specified number of UUID strings.

-t old_uuid Translates an old-style UUID into a new-style UUID.

Description

The uuidgen command creates a Universal Unique Identifier (UUID) string that you assign to an object to uniquely identify it. One such use is in the UUID interface attribute of an IDL interface definition. The format for representing a UUID string consists of eight hexadecimal digits followed by a dash, followed by three groups of four hexadecimal digits separated by dashes, followed by a dash and twelve hexadecimal digits, as shown in the following sample:

01234567-89ab-cdef-0123-456789abcdef

Examples

1. Generate a UUID string with the following command:
uuidgen(1rpc)

This results in output like the following:

23c67e00-71b6-11c9-9dfc-08002b0ecef1

2. Generate a partial template, containing a generated UUID string, to be used to
develop an interface definition, with the following command:

```
uuidgen -i
```

This results in output like the following:

```
[ uuid(828bf780-71b6-11c9-b5a8-08002b0ecef1),
  version (1.0) ]
interface INTERFACENAME
{
}
```

3. Convert a UUID string from the old-style format to the new format with the
following command:

```
uuidgen -t 34DC23469EAF.AB.A2.01.7C.5F.2C.ED.A3
```

This results in output like the following:

34dc2346-9eaf-0000-aba2-017c5f2ceda3

4. Generate four UUID strings with the following command:

```
uuidgen -n 4
```

This results in output like the following:

612c0b00-71b8-11c9-973a-08002b0ecef1
612c0b01-71b8-11c9-973a-08002b0ecef1
612c0b02-71b8-11c9-973a-08002b0ecef1
612c0b03-71b8-11c9-973a-08002b0ecef1

5. Convert a UUID into a C structure with the following command:

```
uuidgen -s -c 1251ace6-93a1-11cd-95ad-0800097086e4
```

This results in output like the following:

```
= { /* 1251ace6-93a1-11cd-95ad-0800097086e4 */
  0x1251ace6,
  0x93a1,
  0x11cd,
  0x95,
  0xad
  {0x08, 0x00, 0x09, 0x70, 0x86, 0xe4}
};
```
Errors

A representative list of errors that might be returned is not shown here. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Problem Determination Guide for complete descriptions of all error messages.

Files

dceshared/bin/uuidgen
   Generator

dceshared/nls/msg/LANG/uuidgen.cat
   Generator error messages
Purpose

Introduction to RPC daemon and RPC control program commands

Description

The DCE remote procedure call (RPC) component provides two administrative facilities: the RPC daemon and the DCE RPC control program, `rpccp`.

Note:

These facilities are superseded by the DCE host daemon (`dced`) and the DCE control program (`dcecp`) in OSF DCE Version 1.1.

The RPC daemon is a process that provides the endpoint map service, which maintains the local endpoint map for local RPC servers and looks up endpoints for RPC clients. An endpoint is the address of a specific instance of a server executing in a particular address space on a given system (a server instance). Each endpoint can be used on a system by only one server at a time.

An endpoint map is a database where servers register their binding information, including endpoints, for each of their RPC interfaces and the associated RPC objects. Each combination of binding information, interface identifier, and object Universal Unique Identifier (UUID) uses a distinct element in the local endpoint map. The `rpcd` command starts the RPC daemon.

The DCE RPC control program, `rpccp`, provides a set of commands for accessing the operations of the RPC Name Service Interface (NSI). For managing endpoint maps, the control program supports showing endpoint map elements and removing any set of map elements from the local endpoint map or from any remote endpoint map. The `rpccp` command starts the RPC control program.

Exit Values

The RPC control program reports DCE error messages on the command line. If the command executes successfully, the internal value returned is `0` (zero); otherwise, the value is `-1` (negative one).

Related Information

Commands: `rpcd(8rpc)`, `rpccp(8rpc)`, `dced(8dce)`, `dcecp(8dce)`.  

**rpccp**

**Purpose**

Starts the DCE remote procedure call (RPC) control program

**Synopsis**

`rpccp [rpccp-command]`

**Arguments**

**rpccp-command**

Specifies one of the following control program commands:

- **add element**
  
  Adds an element to a profile in a name service entry; if the specified entry does not exist, creates the entry.

- **add entry**
  
  Adds an entry to the name service database.

- **add mapping**
  
  Adds or replaces server address information in the local endpoint map.

- **add member**
  
  Adds a member to a group in a name service entry; if the specified entry does not exist, creates the entry.

- **exit**
  
  Leaves the RPC control program.

- **export**
  
  Exports binding information for an interface identifier, object Universal Unique Identifiers (UUIDs), or both to a server entry; if the specified entry does not exist, creates the entry.

- **help**
  
  Displays a list of commands or the possible options of a specified command.

- **import**
  
  Imports binding information and an object UUID from a server entry.

- **quit**
  
  Leaves the RPC control program.

- **remove element**
  
  Removes selected elements from a profile.

- **remove entry**
  
  Removes an entry from the name service database.

- **remove group**
  
  Removes all group members and the group from the specified entry.

- **remove mapping**
  
  Removes specified elements from the local endpoint map or from the endpoint map of a specified remote host.

- **remove member**
  
  Removes a selected member from a group.
rpccp(8rpc)

remove profile
   Removes all profile elements and the profile from the specified entry.

show entry
   Shows the Name Service Interface (NSI) attributes of an entry.

show group
   Shows the members of a group.

show mapping
   Shows the elements of the local endpoint map.

show profile
   Shows the elements of a profile.

show server
   Shows the binding information, interface identifier, and object UUIDs in a server entry.

unexport
   Removes binding information, interface identifiers, and object UUIDs from a server entry.

Description

Note:

With the exception of the help subcommand, this facility was superseded by the DCE control program (dcecp) in OSF DCE Version 1.1. This command might be fully replaced by the dcecp command in a future release of DCE, and might no longer be supported at that time.

The RPC control program, rpccp, provides a set of commands for managing name service use for RPC applications and for managing the endpoint map.

You can use control program commands from within the control program or from the system prompt. To use the control program commands from inside the control program, Start and enter the control program by using the rpccp command alone, without any argument. The control program then displays the control program prompt, rpccp>, as follows:

    rpccp
    rpccp>

You can then enter any control program command, as in the following example:

    rpccp> show entry /.:LandS/anthro/pr_server_node3

Leave the control program and return to the system prompt by using the exit or quit command. If you enter invalid input, the control program displays the valid commands.

To use the control program commands from the system prompt, enter the rpccp command with an internal command of the control program as the first argument. You can do this either interactively or in a command procedure. For example, you can enter the show entry command as follows:

    rpccp show entry /.:LandS/anthro/pr_server_node3
Arguments and Options

Except for the exit and quit commands, rpccp commands have one or more options. Each option is identified by a - (dash) followed by a letter; for example, -s. Some options require arguments.

Commands that access NSI operations also require the name of a name service entry as an argument. The order of arguments and the entry-name option is arbitrary; for example, the following placements of arguments and options are equivalent:

```
rpccp> add element /./LandS/anthro/mis_node_2 \
     -i ec1eeb60-5943-11c9-a309-08002b102989,1.0
```

```
rpccp> add element -i ec1eeb60-5943-11c9-a309-08002b102989,1.0 \
    /./LandS/anthro/mis_node_2
```

Environmental Influences on Command Syntax

There are variations in the action of the control program, depending on whether commands are entered from the system prompt or from within the control program. For example, entering the annotation field of profile elements from the system prompt allows you to include internal spaces in an annotation.

<table>
<thead>
<tr>
<th>Function</th>
<th>At System Prompt</th>
<th>Inside rpccp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strings within quotation marks</td>
<td>Supported</td>
<td>Not required</td>
</tr>
<tr>
<td>Wildcard substitution</td>
<td>Supported</td>
<td>Unsupported</td>
</tr>
</tbody>
</table>

Note:

Some UNIX systems require that you place a \ (backslash) before string binding delimiters such as [ ] (brackets) or that you place the delimiters within ‘ ’ or “ ” (single or double quotation marks) at the system prompt.

The following table describes the scope of the RPC control program commands.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>All entries</td>
<td>add entry</td>
</tr>
<tr>
<td></td>
<td>remove entry</td>
</tr>
<tr>
<td></td>
<td>show entry</td>
</tr>
<tr>
<td>Server entry</td>
<td>export</td>
</tr>
<tr>
<td></td>
<td>import</td>
</tr>
<tr>
<td></td>
<td>show server</td>
</tr>
<tr>
<td></td>
<td>unexport</td>
</tr>
<tr>
<td>Group</td>
<td>add member</td>
</tr>
<tr>
<td></td>
<td>remove group</td>
</tr>
<tr>
<td></td>
<td>remove member</td>
</tr>
<tr>
<td></td>
<td>show group</td>
</tr>
<tr>
<td>Profile</td>
<td>add element</td>
</tr>
<tr>
<td></td>
<td>remove element</td>
</tr>
<tr>
<td></td>
<td>remove profile</td>
</tr>
<tr>
<td></td>
<td>show profile</td>
</tr>
<tr>
<td>Endpoint map</td>
<td>add mapping</td>
</tr>
<tr>
<td></td>
<td>remove mapping</td>
</tr>
<tr>
<td></td>
<td>show mapping</td>
</tr>
</tbody>
</table>
Environment Variables

The control program supports environment variables. Using environment variables facilitates interactive use of the control program.

To distinguish environment variables, `rpccp*(8rpc)` reference pages follow the convention of using all uppercase letters for examples of environment variables. Note that UNIX environment variables are case sensitive.

**User-defined environment variables**

You can set an environment variable to represent values to `rpccp`. Using an environment variable is helpful for specifying a long string such as the following:

1. A string representation of binding information *(binding string)*
2. A string representation of an object or interface UUID *(string UUID)*
3. An interface identifier *(the interface UUID and version numbers)*
4. The name of a name service entry

In the following example, the environment variable `JANE_CAL` represents an object UUID, while `./LandS/anthro/Cal_host_2`, the target name service entry, is in the local cell:

```
JANE_CAL=47f40d10-e2e0-11c9-bb29-08002b0f4528
export JANE_CAL
rpccp
rpccp> export -o JANE_CAL ./LandS/anthro/Cal_host_2
```

**DCE RPC environment variables**

**NLSPATH**

The environment variable `NLSPATH` must point to the location of `dcerpc.cat` and `dcedcs.cat`. Otherwise, any run-time status codes returned by the control program will be hexadecimal, rather than textual, form. The value of this variable must include both the pathname of the directory where the `.cat` files reside and the string `%N`.

**RPC_DEFAULT_ENTRY_SYNTAX**

The `dce` name syntax is the only syntax currently supported by the DCE Cell Directory Service (CDS). However, NSI is independent of any specific name service and, in the future, might support name services that use other name syntaxes. When alternative name syntaxes are supported, you can override the standard default with a process-specific default by setting the `RPC_DEFAULT_ENTRY_SYNTAX` environment variable. When this variable is set for a process, the control program uses it to find out the default syntax for the process. You can override this default in any NSI command of the control program by using the `-s` option to specify an alternative entry syntax. Setting `RPC_DEFAULT_ENTRY_SYNTAX` requires specifying the integer 3 to indicate the `dce` syntax. To set `RPC_DEFAULT_ENTRY_SYNTAX`, use the `name=value` command to define an environment variable. The following command specifies `dce` as the default name syntax in a login command file:
# .login command file
# setting dce as default name syntax,
RPC_DEFAULT_ENTRY_SYNTAX=3

RPC_DEFAULT_ENTRY
For the `import` command, you can use this environment variable to indicate the entry where the search operation starts. Usually, the starting entry is a profile.

The Name Service Interface

The remainder of this description contains information to help you use commands that call NSI to access name service entries.

DCE NSI is independent of any particular name service. CDS, however, is the only name service available for DCE Version 1.0 RPC applications. For more details on NSI, see the *IBM DCE Version 3.1 for AIX and Solaris: Application Development Guide—Core Components*. For a description of CDS, see the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide—Core Components*.

Name Service Entries

To store information about RPC servers, interfaces, and objects, NSI defines the following name service entries:

- **server entry**
  Stores binding information, interface identifiers, and object UUIDs for an RPC server

- **group**
  Corresponds to one or more RPC servers that offer a common RPC interface, type of RPC object, or both

- **profile**
  Defines search paths for looking in a name service database for a server that offers a particular RPC interface and object

Note that when NSI is used with CDS, the name service entries are CDS object entries.

Structure of Entry Names

Each entry in a name service database is identified by a unique global name made up of a cell name and a cell-relative name.

A **cell** is a group of users, systems, and resources that share common DCE services. A cell configuration includes at least one cell directory server, one security server, and one time server. A cell’s size can range from one system to thousands of systems. For information on cells, see the CDS portion of this book.

The following is an example of a global name:

```
/.../C=US/O=uw/OU=MadCity/LandS/anthro/Stats_host_2
```

The parts of a global name are as follows:

1. **cell name**
   The cell name must use X.500 name syntax. The symbol `/` begins a cell name. The letters before each `=` (equal sign) are abbreviations for country (C), organization (O), and organization unit (OU). For example:
For entries in the local cell, the cell name can be represented by a ./ prefix, in place of the actual cell name, as in the following example:

`./LandS/anthro/Stats_host_2`

For NSI operations on entries in the local cell you can omit the cell name.

2. cell-relative name

Each name service entry requires a cell-relative name, which contains a directory pathname and a leaf name.

a. directory pathname

Follows the cell name and indicates the hierarchical relationship of the entry to the cell root.

The directory pathname is the middle portion of the global name. The cell name precedes the directory pathname, and the leaf name follows it, as follows:

```
cell-name + directory-pathname + leaf-name
```

The directory pathname contains the names of any subdirectories in the path; each subdirectory name begins with a / (slash), as follows:

```
/sub-dir-a-name/ sub-dir-b-name/ sub-dir-c-name
```

Directory paths are created by name service administrators. If an appropriate directory path does not exist, ask your name service administrator to extend an existing path or create a new path. In a directory path, the name of a subdirectory should reflect its relationship to its parent directory (the directory that contains the subdirectory).

b. leaf name

Identifies the specific entry. The leaf name is the right-hand part of global name beginning with the rightmost slash.

In the following example, `./../C=US/O=uw/OU=MadCity` is the cell name, `/LandS/anthro` is the directory pathname, and `/Cal_host_4` is the leaf name:

```
/.../C=US/O=uw/OU=MadCity/LandS/anthro/Cal_host_4
```

If a name service entry is located at the cell root, the leaf name directly follows the cell name; for example, `./:cell-profile`.

Note that when NSI is used with CDS, the cell-relative name is a CDS name.

**Guidelines for Constructing Names of Name Service Entries**

A global name includes both a cell name and a cell-relative name composed of a directory pathname and a leaf name. The cell name is assigned to a cell root at its creation. When you specify only a cell-relative name to an NSI command, the NSI automatically expands the name into a global name by inserting the local cell name. When returning the name of a name service entry, a group member, or member in a profile element, NSI operations return global names.
The directory pathname and leaf name uniquely identify a name service entry. The leaf name should somehow describe the entry—by identifying its owner or its contents, for example. The remainder of this section contains guidelines for choosing leaf names. Note that directory pathnames and leaf names are case sensitive.

**Naming a Server Entry**

For a server entry that advertises an RPC interface or service offered by a server, the leaf name must distinguish the entry from the equivalent entries of other servers. When a single server instance runs on a host, you can ensure a unique name by combining the name of the service, interface (from the interface definition), or the system name for the server’s host system.

For example, consider two servers, one offering a calendar service on host JULES and one on host VERNE.

The server on JULES uses the following leaf name:

```
calendar_JULES
```

The server on VERNE uses the following leaf name:

```
calendar_VERNE
```

For servers that perform tasks on or for a specific system, an alternative approach is to create server entries in a system-specific host directory within the name service database. Each host directory takes the name of the host to which it corresponds.

Because the directory name identifies the system, the leaf name of the server entry name need not include the host name, as in the following example:

```
./LandS/host_1/Process_control
```

To construct names for the server entries used by distinctive server instances on a single host, you can construct unique server entry names by combining the following information: the name of the server’s service, interface, or object; the system name of the server’s host system, and a reusable instance identifier, such as an integer.

For example, the following leaf names distinguish two instances of a calendar service on the JULES system:

```
calendar_JULES_01

calendar_JULES_02
```

Avoid automatically generating entry names for the server entries of server instances—for example, by using unique data such as a time stamp (calendar_verne_15OCT91_21:25:32) or process identifier (calendar_jules_208004D6). When a server incorporates such unique data into its server entry names, each server instance creates a separate server entry, causing many server entries. When a server instance stops running, it leaves an obsolete server entry that is not reused. The creation of a new entry whenever a server instance starts might impair performance.
A server can use multiple server entries to advertise different combinations of interfaces and objects. For example, a server can create a separate server entry for a specific object (and the associated interfaces). The name of such a server entry should correspond to a well-known name for the object. For example, consider a server that offers a horticulture bulletin board known to users as `horticulture_bb`. The server exports the `horticulture_bb` object, binding information, and the associated bulletin-board interface to a server entry whose leaf name identifies the object, as follows:

`horticulture_bb`

Note that an RPC server that uses RPC authentication can choose identical names for its principal name and its server entry. Use of identical names permits a client that calls the `rpc_binding_set_auth_info` routine to automatically determine a server's principal name (the client will assume the principal name to be the same as the server's entry name). If a server uses different principal and server entry names, users must explicitly supply the principal name. For an explanation of principal names, see the IBM DCE Version 3.1 for AIX and Solaris: Application Development Guide.

Naming a Group

The leaf name of a group should indicate the interface, service, or object that determines membership in the group. For example, for a group whose members are selected because they advertise an interface named `Statistics`, the following is an effective leaf name:

`Statistics`

For a group whose members advertise laser-printer print queues as objects, the following is an effective leaf name:

`laser-printer`

Naming a Profile

The leaf name of a profile should indicate the profile users; for example, for a profile that serves the members of an accounting department, the following is an effective leaf name:

`accounting_profile`

Privileges Required

To use NSI commands to access entries in a CDS database, you need access control list (ACL) permissions. Depending on NSI operation, you need ACL permissions to the parent directory or the CDS object entry (the name service entry) or both. The ACL permissions are as follows:

1. To create an entry, you need `i` (insert) permission to the parent directory.
2. To read an entry, you need `r` (read) permission to the CDS object entry.
3. To write to an entry, you need `w` (write) permission to the CDS object entry.
4. To delete an entry, you need `d` (delete) permission either to the CDS object entry or to the parent directory.

Note that `write` permission does not imply `read` permission.
ACL permissions for NSI commands of the control program are described in the reference pages.

Notes

A server entry equates to an NSI binding attribute and, optionally, an object attribute; a group equates to an NSI group attribute; and a profile equates to an NSI profile attribute. Typically, each server's entries, groups, and profiles reside in distinct name service entries.

Examples

1. The following command starts the RPC control program:

   rpccp

2. The following command, entered at the system prompt rather than in rpccp, removes the entry /./LandS/anthro/Cal_host_2:

   rpccp remove entry
   /./LandS/anthro/Cal_host_2

Related Information

Commands: rpccp_add_element(8rpc), rpccp_add_entry(8rpc), rpccp_add_mapping(8rpc), rpccp_add_member(8rpc), rpccp_export(8rpc), rpccp_import(8rpc), rpccp_remove_element(8rpc), rpccp_remove_entry(8rpc), rpccp_remove_group(8rpc), rpccp_remove_mapping(8rpc), rpccp_remove_member(8rpc), rpccp_remove_profile(8rpc), rpccp_show_entry(8rpc), rpccp_show_group(8rpc), rpccp_show_mapping(8rpc), rpccp_show_profile(8rpc), rpccp_show_server(8rpc), rpccp_unexport(8rpc), dcecp(8dce).
add element

Purpose

Adds an element to a profile in a name service entry

Synopsis

rpccp add element profile-entry-name -m member {-d | -i if-id | -p priority} [-a annotation] [-s syntax]

Options

-m member
  Defines a member name for the profile element to be added (required).

-d
  Performs the add element operation on the default profile element. With the -d option, the -i and -p options are ignored.

-i if-id
  Defines an interface identifier for the profile element to be added. Only one interface can be added in a single operation. An interface identifier is required, unless the default profile element is being added. With the -d option, the -i option is ignored.
  
  The value has the following form:

  interface-uuid, major-version, minor-version

  The Universal Unique Identifier (UUID) is a hexadecimal string and the version numbers are a decimal string, for example:

  -i ec1eeb60-5943-11c9-a309-08002b102989,3.11

  Leading zeros in version numbers are ignored.

-p priority
  Defines a search priority for the new profile element. The priority value is in the range 0 to 7, with zero having the highest priority. When a default element is added (with the -d option), the -p option is ignored. By default, a nondefault element is assigned a priority value of zero.

-a annotation
  Defines an annotation string for the profile element.
  
  Note that the shell supports quotation marks around the annotation field of profile elements, which allows you to include internal spaces in an annotation; the control program does not. To specify or refer to annotations from within the control program, limit each annotation to an unbroken alphanumeric string; for example, CalendarGroup. To refer to annotations from the system prompt, do not incorporate quotation marks into any annotation.

-s syntax
  Indicates the name syntax of the entry name (optional). The only value for this option is the dce name syntax, which is the default name syntax. Until an alternative name syntax becomes available, specifying the -s option is unnecessary.
Arguments

profile-entry-name
Specifies the entry name of the target profile. For an entry in the local cell, you can omit the cell name and specify only the cell-relative name.

Description

The add element command adds an element to a profile in a name service entry. The name of the entry containing the profile and the entry name of the profile member in the new element are required. The entry of a profile might have been created previously (by either the add entry or add element command). But if the specified entry does not exist, the add element command tries to create the entry.

A profile element is a database record containing the following fields:

1. interface identifier
   This is the primary search key. The interface identifier consists of the interface UUID and the interface version numbers.

2. member name
   The entry name of one of the following kinds of name service entries:
   a. A server entry for a server offering the requested remote procedure call (RPC) interface and object
   b. A group corresponding to the requested RPC interface
   c. A profile

3. priority_value
   The priority value (0 (zero) is the highest priority; 7 is the lowest) is designated by the creator of a profile element to help determine the order for using the element. NSI search operations select among like priority elements at random. For the rpccp add element command, the default is 0.

4. annotation string
   The annotation string enables you to identify the purpose of the profile element. The annotation can be any textual information, for example, an interface name associated with the interface identifier or a description of a service or resource associated with a group. The annotation string is not a search key for the import or lookup operations.

Privileges Required

You need both r (read) permission and w (write) permission to the Cell Directory Service (CDS) object entry (the target profile entry). If the entry does not exist, you also need insert permission to the parent directory.

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

1. The following command adds an element to the cell profile, /cell-profile, in the local cell:

   rpccp> add element -i ec1eeb60-5943-11c9-a309-08002b102989,1.1
   -m ./Calendar_profile -a RefersToCalendarGroups ./cell-profile
2. The following commands start the control program, set up a user profile associated with the cell profile as its default element, and add a user-specific element for the Calendar Version 1.1 interface:

```
rpccp> add element ./LandS/anthro/molly_o_profile -d -m ./cell-profile
rpccp> add element ./LandS/anthro/molly_o_profile /
   -m ./LandS/anthro/Calendar_group \
   -i ec1eeb60-5943-11c9-a309-08002b102909,1.1 \
   -a Calendar_Version 1.1_Interface
```

The added profile element contains the global name of the member (specified by `./LandS/anthro/Calendar_group`, its cell-relative name) and the RPC interface identifier for the Calendar Version 1.1 interface.

**Related Information**

Commands: `rpccp_remove_element(8rpc)`, `rpccp_remove_profile(8rpc)`, `rpccp_show_profile(8rpc)`
add entry

Purpose

Adds a name service entry to the name service database

Synopsis

rpccp add entry entry-name [-s syntax]

Options

-s syntax
Indicates the name syntax of the entry name (optional). The only value for this option is the dce name syntax, which is the default name syntax. Until an alternative name syntax becomes available, specifying the -s option is unnecessary.

Arguments

entry-name
Specifies the name of the target name service entry. For an entry in the local cell, you can omit the cell name and specify only the cell-relative name.

Description

The add entry command adds an unspecialized entry to the name service database. The name of the entry is required.

The new entry initially contains no Name Service Interface (NSI) attributes. This command creates a general name service entry for an application or user. The application or user can later use the rpccp export, rpccp add element, and rpccp add member commands to make the generic entry into a server entry, a group, or a profile (or a combination), as follows:
1. For a server entry, specify the new entry as the target entry for the rpccp export command.
2. For a group, specify the new entry as the target group for the rpccp add member command.
3. For a profile, specify the new entry as the target profile for the rpccp add element command.

The add entry command enables administrators to add entries for users who lack the required permissions. If you have the permissions required by the rpccp add entry command, you can also add an entry using an rpccp export, rpccp add member, or rpccp add element command; if the entry you specify does not exist, the command creates the entry.

Privileges Required

To add an entry, you need i (insert) permission to the parent directory and both r (read) permission and w (write) permission to the Cell Directory Service (CDS) object entry (the target name service entry).
add entry(8rpc)

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

1. The following command adds an unspecialized entry to the name service database:

   rpccp> add entry ./LandS/anthro/Cal_host_2

2. The following command operates from the system prompt to add an unspecialized entry to the name service database:

   rpccp add entry ./LandS/anthro/Cal_host_3

Related Information

Commands: rpccp_remove_entry(8rpc), rpccp_show_entry(8rpc).
add mapping

Purpose

Adds or replaces server address information in the local endpoint map

Synopsis

rpccp add mapping -b \"string-binding\" -i \"interface-identifier\" [-a \"annotation-string\"] [-o \"object-uuid\"] [-N]

Options

-b \"string-binding\"
Specifies a string representation of a binding over which the server can receive remote procedure calls. At least one binding is required.

The value has the form of an remote procedure call (RPC) string binding, without an object Universal Unique Identifier (UUID), as in the following example:

-b ncadg_ip_udp:63.0.2.17[5347]

Note that depending on your system, string binding delimiters such as [ ] (brackets) might need to be preceded by a \ (backslash) or placed within ’ ’ or “” (single or double quotation marks). Requirements vary from system to system, and you must conform to the usage rules of a system.

-i \"interface-identifier\"
Specifies an interface identifier to register with the local endpoint map. An interface identifier is required. Only one interface can be added (that is, registered) in a single operation. The interface identifier has the following form:

interface-uuid, major-version, minor-version

The UUID is a hexadecimal string and the version numbers are decimal strings, as in the following example:

-i ecleeb60-5943-11c9-a309-08002b102989,1,1

Leading zeros in version numbers are ignored.

-a \"annotation-string\"
Specifies a character string comment to be applied to each cross product element that is added to the local endpoint map. The string can be up to 64 characters long, including the NULL terminating character.

The string is used by applications for informational purposes only. The RPC runtime does not use this string to determine which server instance a client communicates with, or for enumerating endpoint map elements.

-o \"object-uuid\"
Defines an object UUID that further determines the endpoint map elements that are removed (optional). Each add mapping command accepts up to 32 -o options.

The UUID is a hexadecimal string, as in the following example:
add mapping(8rpc)

-o 3c6b8f60-5945-11c9-a236-08002b102989

-N Specifies that existing elements in the local host’s endpoint map should not be replaced when the new information is added.

Description

The add mapping command adds to or replaces server address information in the local endpoint map.

Each element in the local endpoint map logically contains the following:
1. An interface ID, consisting of an interface UUID and versions (major and minor)
2. Binding information
3. An object UUID (optional)
4. An annotation string (optional)

This command should be used without the -N option when only a single instance of the server in question runs on the server’s host. Do not use the -N option if no more than one server instance on the host ever offers the same interface UUID, object UUID, and protocol sequence.

When local endpoint map elements are not replaced, obsolete elements accumulate each time a server instance stops running without explicitly unregistering its endpoint map information. Periodically, the RPC daemon rpcd will identify these obsolete elements and remove them. However, during the interval between these removals, the presence of the obsolete elements increases the chance that clients will receive endpoints to nonexistent servers. The clients will then waste time trying to communicate with these servers before giving up and obtaining another endpoint.

Allowing rpcd to replace any existing local endpoint map elements (by not specifying -N) reduces the chance of this happening.

For example, suppose an existing element in the local endpoint map matches the interface UUID, binding information exclusive of the endpoint, and object UUID of an element this routine provides. The routine changes the endpoint map according to the elements' interface major and minor version numbers.

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

The following command adds a map element to the local endpoint map. The command adds the map element that contains the specified interface identifier, server address (specified as a string binding), and object UUIDs.

```
rpccp add mapping -i ec1eeb60-5943-11c9-a309-08002b102989,1.1 \  -b ncadg_ip_udp:63.0.2.17[5347] -o 005077d8-8022-1acb-9375-10005a4f533a \  -o 001bc29a-8041-1acb-b377-10005a4f533a -a 'Calendar version 1.1'
```

This command adds the following elements:
Related Information

Commands: `rpccp_export(8rpc)`, `rpccpremove_mapping(8rpc)`, `rpccpshow_mapping(8rpc)`, `rpccpshow_server(8rpc)`

Subroutines: `rpc_ep_register(3rpc)`, `rpc_ep_register_no_replace(3rpc)`
add member

Purpose

Adds a member to a group in a name service entry

Synopsis

```
rpccp add member group-entry-name -m member [ -s syntax ]
```

Options

```
-m member
Declares the name of a member to be added to the specified group entry (required). You can add only one member at a time.

-s syntax
Indicates the name syntax of the entry name (optional). The only value for this option is the \texttt{dce} name syntax, which is the default name syntax. Until an alternative name syntax becomes available, specifying the \texttt{-s} option is unnecessary.
```

Arguments

```
group-entry-name
Specifies the name of the target group. For an entry in the local cell, you can omit the cell name and specify only the cell-relative name.
```

Description

The \texttt{add member} command adds a member to a group in a name service entry. The name of the entry containing the group and the name of the new group member are required. The entry of a group might have been created previously (by either the \texttt{add entry} or \texttt{add member} command). If the specified entry does not exist, the \texttt{add member} command tries to create the entry.

Privileges Required

You need both \texttt{r} (\texttt{read}) permission and \texttt{w} (\texttt{write}) permission to the Cell Directory Service (CDS) object entry (the target group entry). If the entry does not exist, you also need \texttt{i} (\texttt{insert}) permission to the parent directory.

Notes

This command was replaced at DCE Version 1.1 by the \texttt{dcecp} command and might not be provided in future releases of DCE.

Examples

The following command adds the member \texttt{/.:LandS/anthro/Cal_host_3} to the group \texttt{/.:LandS/anthro/Calendar_group}:

```
rpccp> add member -m /.:LandS/anthro/Cal_host_3 /./.:LandS/anthro/Calendar_group
```
Related Information

Commands: `rpccp_remove_group(8rpc)`, `rpccp_remove_member(8rpc)`, `rpccp_show_group(8rpc)`. 
exit

exit

Purpose

Causes the DCE Remote Procedure Call control program (rpccp) to complete running.

Synopsis

rpccp exit

Description

Causes the DCE Remote Procedure Call control program (rpccp) to complete running and returns operation to the parent process.

Privileges Required

No special privileges are needed to use the exit command.

Examples

The following example shows how to leave the RPC control program (rpccp) and return to the parent process:

rpccp> exit

Related Information

Command: quit
export

Purpose

Exports binding information for interface identifiers or object UUIDs to a server entry

Synopsis

rpccp export entry-name {[-i if-id] | [-o object-uuid]} -b string-binding [ssyntax]

Options

-i if-id  Declares the interface identifier of a remote procedure call (RPC) interface. The export command operates on only one -i option; if you enter more than one, the command ignores all but the last interface identifier. If you specify an interface identifier, you must specify at least one -b option. The -i and -o options can occur together or separately, but one of them is necessary.

The interface identifier takes the following form:

interface-uuid, major-version, minor-version

The version numbers are optional, but if you omit a version number, the value defaults to 0. The Universal Unique Identifier (UUID) is a hexadecimal string and the version numbers are decimal strings, as in the following example:

-i ecleeb60-5943-11c9-a309-08002b102989,3.11

Leading zeros in version numbers are ignored.

-o object-uuid  Declares the UUID of an object. Each export command accepts up to 32 -o options. The -i and -o options can occur together or separately, but one of them is necessary.

The UUID is a hexadecimal string, as in the following example:

-o 3c6b8f60-5945-11c9-a236-08002b102989

-b string-binding  Declares a string binding (optional). To use this option, you must also specify an interface identifier (using the -i option). Each command accepts up to 32 -b options.

The value has the form of an RPC string binding, without an object UUID. The binding information contains an RPC protocol sequence, a network address, and sometimes an endpoint within brackets, as follows:

rpc-prot-seq: network-addr[endpoint]

For a well-known endpoint, include the endpoint in the string binding, as in the following example:

-b ncadg_ip_udp:63.0.2.17[5347]
For a dynamic endpoint, omit the endpoint from the string binding, for example:

-b ncacn_ip_tcp:16.20.15.25

Note that depending on your system, string binding delimiters such as [ ] (brackets) might need to be preceded by a \ (backslash) or placed within ' ' or " " (single or double quotation marks). Requirements vary from system to system, and you must conform to the usage rules of a system.

-s syntax
Indicates the name syntax of the entry name (optional). The only value for this option is the dce name syntax, which is the default name syntax. Until an alternative name syntax becomes available, specifying the -s option is unnecessary.

Arguments

entry-name
Specifies the name of the target name service entry. Usually, the target is a server entry. However, objects also can be exported (without an interface identifier or any binding information) to a group or a profile.

For an entry in the local cell, you can omit the cell name and specify only the cell-relative name.

Description

The export command places binding information and an interface identifier, object UUIDs, or both into a server entry, or the command object UUIDs into a group’s entry. The export command searches the name service database for the entry with the specified entry name. If the entry exists, the command uses it; otherwise, the command tries to create a new name service entry using the specified entry name.

Minimally, the command requires the name of the entry and either an identifier and binding string or an object.

If the specified entry does not exist, the export command tries to create the entry.

Privileges Required

You need both r (read) and w (write) permission to the Cell Directory Service (CDS) object entry (the target name service entry). If the entry does not exist, you also need i (insert) permission to the parent directory.

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

1. The following example shows a control program export command that is stored in a file for later execution from the system prompt. The command exports two objects and an interface with two string bindings to the server entry /./LandS/anthro/Cal_host_3 in the local cell.
# file to export Calendar 1.1 at installation time
rpccp export \
-1 ec1eeb60-5943-11c9-a309-08002b102989,1.1 \
- b ncacn_ip_tcp:16.20.15.25 \
- b ncadg_ip_udp:63.0.2.17 \
- o 30dbeea0-fb6c-11c9-8ee0-08002b0f4528 \
- o 16977538-e257-11c9-8dc0-08002b0f4528 \
/./LandS/anthro/Cal_host_3

2. The following example shows the use of a user-defined environment variable as an interface identifier, to facilitate entering an export command interactively (in this case, from inside the control program). The two initial shell commands set up an environment variable Calendar_1_1, which represents the interface identifier of an RPC interface. The rpccp command then starts the control program, and the export command exports the Calendar interface and two string bindings to the server entry /./LandS/anthro/Cal_host_2 in the local cell.

Calendar_1_1=ec1eeb60-5943-11c9-a309-08002b102989,1.1
export Calendar_1_1
rpccp
rpccp> export -i Calendar_1_1 -b ncacn_ip_tcp:16.20.15.25 \\n- b ncadg_ip_udp:63.0.2.17 /./LandS/anthro/Cal_host_2

3. The following example shows the use of user-defined environment variables for object UUIDs to facilitate entering an export command interactively (in this case, from inside the control program). The initial shell commands set up the environment variables LUKE_CAL and JOSH_CAL, which represent personal calendars that are accessible as objects to an RPC server. The rpccp command then starts the control program, and the export command exports the two objects to the server's entry /./LandS/anthro/Cal_host_2 in the local cell.

LUKE_CAL=30dbeea0-fb6c-11c9-8ee0-08002b0f4528
export LUKE_CAL
JOSH_CAL=16977538-e257-11c9-8dc0-08002b0f4528
export JOSH_CAL
rpccp
rpccp> export -o LUKE_CAL -o JOSH_CAL /./LandS/anthro/Cal_host_2

Related Information

Commands: rpccp_import(8rpc), rpccp_show_server(8rpc), rpccp_unexport(8rpc).
help

Purpose

When issued without arguments, RPCCP HELP returns a list of commands for which help is available. When issued with an argument, it returns help for that command.

Synopsis

rpccp help [rpccp-command]

Arguments

rpccp-command

Optionally specifies one of the following control commands:

add element
add entry
add member
exit
export
import
quit
remove element
remove entry
remove group
remove mapping
remove member
remove profile
show entry
show group
show mapping
show profile
show server
unexport

Description

The help command displays information about the rpccp command set or the options and arguments associated with a specific command.

Notes

This command might be replaced in future releases by the dcecp command, and might no longer be supported at that time.
Examples

1. The following command is entered at the system prompt to display the internal commands of the control program:

   \texttt{rpccp help}

2. The following command displays the syntax of the \texttt{remove entry} command:

   \texttt{rpccp> help remove entry}

Related Information

Commands: \texttt{rpccp_add_element(8rpc)}, \texttt{rpccp_add_entry(8rpc)}, \texttt{rpccp_add_member(8rpc)}, \texttt{rpccp_export(8rpc)}, \texttt{rpccp_import(8rpc)}, \texttt{rpccp_remove_element(8rpc)}, \texttt{rpccp_remove_entry(8rpc)}, \texttt{rpccp_remove_group(8rpc)}, \texttt{rpccp_remove_mapping(8rpc)}, \texttt{rpccp_remove_member(8rpc)}, \texttt{rpccp_remove_profile(8rpc)}, \texttt{rpccp(8rpc)}, \texttt{rpccp_show_entry(8rpc)}, \texttt{rpccp_show_group(8rpc)}, \texttt{rpccp_show_mapping(8rpc)}, \texttt{rpccp_show_profile(8rpc)}, \texttt{rpccp_show_server(8rpc)}, \texttt{rpccp_unexport(8rpc)}
import

Purpose

Imports binding information and an object UUID from a server entry

Synopsis

\texttt{rpccp import \(\text{starting-entry-name} \ -i \text{id} \ [-v \text{versions}] \ [-e] \ [-n [integer]] \ [-o \text{object-uuid}] \ [-s \text{syntax}] \ [-u] \)}

Options

\textbf{-i \( i-f-id \)} Defines an interface identifier to be imported (required). You can import only one interface at a time.

The value has the following form:

\textit{interface-uuid, major-version, minor-version}

The Universal Unique Identifier (UUID) is a hexadecimal string and the version numbers are decimal strings, as in the following example:

\texttt{-i ec1eeb60-5943-11c9-a309-08002b102989,1.1}

Leading zeros in version numbers are ignored.

\textbf{-v \textit{versions}} Indicates how a specified interface version is used (optional). If it is used without the \textit{-i} option, the \textit{-v} option is ignored. The possible combinations of versions for the \textit{-v} option and their actions are as follows:

\begin{description}
\item[all] The interface version is ignored.
\item[exact] Both the major and minor versions must match the specified versions.
\item[compatible] The major version must match the specified version, and the minor version must be greater than or equal to the specified version.
\item[major_only] The major version must match the specified version; the minor version is ignored.
\item[upto] The major version must be less than or equal to that specified. If the major versions are equal, the minor version must be less than or equal to that specified.
\end{description}

If the \textit{-v} option is absent, the command shows compatible version numbers.

\textbf{-e} Shows the name of the entry where the binding is found (optional).

\textbf{-n [integer]} Declares that the import operation is to continue until no more potential bindings are found (optional). Providing a numeric value to this option restricts the number of imported bindings. If you omit the number, only one binding is imported. If repeated, this operation might return the same
import binding. For example, `-n` imports all available bindings, and `-n 5` imports up to five bindings. Note that the imported bindings are displayed as string bindings.

`-o object-uuid`
Declares the UUID of an object to be imported (optional). Only one UUID can occur in a single operation.

If an object is specified, the import operation limits its search to server entries that contain both the specified interface identifier and object UUID when searching for a potential binding. Without the `-o` option, the import operation ignores object UUIDs.

The UUID is a hexadecimal string, as in the following example:

```
-o 3c6b8f60-5945-11c9-a236-08002b102989
```

`-s syntax`
Indicates the name syntax of the entry name (optional). The only value for this option is the `dce` name syntax, which is the default name syntax. Until an alternative name syntax becomes available, specifying the `-s` option is unnecessary.

`-u` Updates the local Cell Directory Service (CDS) cache copy of name service data (optional).

Name service data is cached locally on each machine in a cell. If an `rpccp` inquiry can be satisfied by data in the local CDS cache, this cached data is returned. Locally cached copies of name service data might not include a recent CDS update, however. If the required data is not available in the local CDS cache, `rpccp` goes to a CDS server(s) to retrieve the required data. `rpccp` then updates the local CDS cache.

Using the `-u` option bypasses the local cache, allowing `rpccp` to go directly to a CDS server for the inquiry. The local CDS caches is then updated by `rpccp`.

**Arguments**

*starting-entry-name*
Indicates the name of the server entry where the import operation starts. For an entry in the local cell, you can omit the cell name and specify only the cell-relative name.

**Description**

The `import` command imports binding information and a remote procedure call (RPC) object UUID for a specific RPC interface from a server entry. The name of the entry and the interface identifier are required. The entry name can refer to a server entry, a group, or a profile.

**Privileges Required**

You need `r` (read) permission to the specified CDS object entry (the starting name service entry) and to any CDS object entry in the resulting search path.
import(8rpc)

Options

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

The following command imports an interface and object:

```
rpccp> import -i ec1eeb60-5943-11c9-a309-08002b102989,1.1 \
    -o 30dbee0-7b6c-11c9-8ee0-08002b0f4528 /./LandS/anthro/Cal_host_3
```

Related Information

Commands: rpccp_export(8rpc), rpccp_show_server(8rpc), unexport(8rpc).
quit

Purpose

Causes the DCE Remote Procedure Call control program (rpccp) to complete running.

Synopsis

rpccp quit

Description

Causes the DCE Remote Procedure Call control program (rpccp) to complete running and returns operation to the parent process.

Privileges Required

No special privileges are needed to use the quit command.

Examples

The following example shows how to leave the RPC control program (rpccp) and return to the parent process:

rpccp> quit

Related Information

Command: exit
remove element

Purpose

Removes selected elements from a profile

Synopsis

rpccp remove element profile-entry-name {-d | -i if-id | -m member | -a annotation} {-s syntax]

Options

- **d** Removes the default profile element. With the -d option, the -a, -i, and -m options are ignored.
- **i if-id** Defines an interface identifier for the profile element to be removed for a member specified with the -m option. Only one interface and member pair can be removed in a single operation. If you supply multiple instances of the -i option, the command uses the final instance. The -i and -m options take precedence over the -a option; if the default profile element is specified with the -d option, however, the -i and -m options are ignored.

The interface identifier value has the following form:

interface-uuid, major-version, minor-version

The Universal Unique Identifier (UUID) is a hexadecimal string and the version numbers are decimal strings, as in the following example:

- i eclee60-5943-11c9-a309-08002b102989,1.1

Leading zeros in version numbers are ignored.

- **m member** Defines a member name for the profile element to be removed. This option is required if the interface identifier is specified. Only one interface and member can be removed in a single operation. If you supply multiple instances of the -m option, the command uses the final instance. The -i and -m options take precedence over the -a option; if the default profile element is specified with the -d option, however, the -i and -m options are ignored.

- **a annotation** Removes all elements whose annotation fields match the specified annotation; in the presence of -d option or -i and -m options, the -a option is ignored.

Note that the shell supports the use of " " (quotation marks) around the annotation field of profile elements, which allows you to include internal spaces in an annotation; the control program does not. To specify or refer to annotations from within the control program, limit each annotation to an unbroken alphanumeric string; for example, CalendarGroup. To refer to annotations from the system prompt, do not incorporate quotation marks into any annotation.

- **s syntax** Indicates the name syntax of the entry name (optional). The only value for
this option is the \texttt{dce} name syntax, which is the default name syntax. Until an alternative name syntax becomes available, specifying the \texttt{-s} option is unnecessary.

\section*{Arguments}

\texttt{profile-entry-name}  
Indicates the name of the target profile. For an entry in the local cell, you can omit the cell name and specify only the cell-relative name.

\section*{Description}

The \texttt{remove element} command removes an element from a profile in the name service database. For a description of the fields in a profile element, see the \texttt{add element(8rpc)} reference page.

The \texttt{remove element} command requires the entry name of the profile. You must also specify either \texttt{-d}, or \texttt{-i} and \texttt{-m}, or \texttt{-a}.

\section*{Privileges Required}

You need \texttt{r} (\texttt{read}) and \texttt{w} (\texttt{write}) permission to the Cell Directory Service (CDS) object entry (the target profile entry).

\section*{Notes}

This command was replaced at DCE Version 1.1 by the \texttt{dcecp} command and might not be provided in future releases of DCE.

\section*{Examples}

The following sequence of commands sets up an environment variable \texttt{Calendar\_1\_1}, which represents the interface identifier of a remote procedure call (RPC) interface, exports it, and removes an element from a profile:

\begin{verbatim}
Calendar_1_1=ec1eeb60-5943-11c9-a309-08002b102989,1.1 export Calendar_1_1 rpccp rpccp> remove element -i Calendar_1_1 -m /.:LandS/anthro/Calendar_group /.:LandS/anthro/molly_o_profile
\end{verbatim}

\section*{Related Information}

Commands: \texttt{rpccp\_add\_element(8rpc)}, \texttt{rpccp\_remove\_profile(8rpc)}, \texttt{rpccp\_show\_profile(8rpc)}. 

\section*{Chapter 2. Remote Procedure Call Commands}
remove entry(8rpc)

remove entry

Purpose

Removes a name service entry from the name service database

Synopsis

rpccp remove entry entry-name [-s syntax]

Options

-s syntax
Indicates the name syntax of the entry name (optional). The only value for this option is the dce name syntax, which is the default name syntax. Until an alternative name syntax becomes available, specifying the -s option is unnecessary.

Arguments

entry-name
Indicates the name of the target name service entry. For an entry in the local cell, you can omit the cell name and specify only the cell-relative name.

Description

The remove entry command removes an entry from the name service database. The name of the entry is required.

Privileges Required

You need r (read) permission to the Cell Directory Service (CDS) object entry (the target name service entry). You also need d (delete) permission to the CDS object entry or to the parent directory.

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

The following command removes the entry ./:LandS/anthro/Cal_host_2 from the local cell of the name service database:

rpccp> remove entry ./:LandS/anthro/Cal_host_2

Related Information

Commands: rpccp_add_entry(8rpc), rpccp_show_entry(8rpc).
remove group

Purpose

Removes all group members and the group from the specified name service entry

Synopsis

```
rpccp remove group group-entry-name [-s syntax]
```

Options

- `-s syntax`
  Indicates the name syntax of the entry name (optional). The only value for this option is the `dce` name syntax, which is the default name syntax. Until an alternative name syntax becomes available, specifying the `-s` option is unnecessary.

Arguments

- `group-entry-name`
  Indicates the name of the target group. For an entry in the local cell, you can omit the cell name and specify only the cell-relative name.

Description

The `remove group` command removes a group from the name service database. The group need not be empty. The entry name of the group is required.

Privilege Required

You need `w` (write) permission to the Cell Directory Service (CDS) object entry (the target group entry).

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command removes the group from the name service entry `/.:/LandS/anthro/Calendar_group`:

```
rpccp> remove group /.:/LandS/anthro/Calendar_group
```

Related Information

Commands: `rpccp_add_member(8rpc)`, `rpccp_remove_member(8rpc)`, `rpccp_show_group(8rpc)`.
remove mapping

Purpose

Removes specified elements from the local endpoint map

Synopsis

rpccp remove mapping -b string-binding -i interface-identifier [-o object-uuid]

Options

-b string-binding
  Specifies a string representation of a binding over which the server can receive remote procedure calls. Each remove mapping command accepts up to 32 -b options. At least one binding is required.

  The value has the form of a remote procedure call (RPC) string binding, without an object UUID, as in the following example:

  -b ncadg_ip_udp:63.0.2.17[5347]

  Note that, depending on your system, string binding delimiters such as [ ] (brackets) might need to be preceded by a \ (backslash) or placed within " " or " " (single or double quotation marks). Requirements vary from system to system, and you must conform to the usage rules of a system.

-i interface-identifier
  Specifies an interface identifier to remove from the local endpoint map. An interface identifier is required. Only one interface can be removed in a single operation. The interface identifier has the following form:

  interface-uuid, major-version.minor-version

  The Universal Unique Identifier (UUID) is a hexadecimal string and the version numbers are decimal strings, as in the following example:

  -i eclee60-5943-11c9-a309-08002b102989,1.1

  Leading zeros in version numbers are ignored.

-o object-uuid
  Defines an object UUID that further determines the endpoint map elements that are removed (optional). Each remove mapping command accepts up to 32 -o options.

  The UUID is a hexadecimal string, as in the following example:

  -o 3c6b8f60-5945-11c9-a236-08002b102989

Description

The remove mapping command removes server address information from the local endpoint map. Each element in the local endpoint map logically contains the following:

1. interface ID, consisting of an interface UUID and versions (major and minor)
2. binding information
3. object UUID (optional)
4. annotation (optional)

This command requires one interface identifier (the -i option), at least one string binding (the -b option), and optionally, one or more object UUIDs (the -o option). Each instance of the command accepts from 1 to 32 -b options and from 0 to 32 -o options. The options work together to delimit the elements to be removed from the target endpoint map. The command removes any map element that contains the specified interface identifier, a specified string binding, and a specified object UUID (if any).

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

The following command operates from the system prompt to remove a map element from the local endpoint map. The command removes only the map element that contains the specified interface identifier, server address (specified as a string binding), and object UUID.

```
rpccp remove mapping -i ec1eeb60-5943-11c9-a309-08002b102989,1.1 \
   -b ncadg_ip_udp:16.20.16.64[3424] \
   -o 30dbeea0-fb6c-11c9-8eea-08002b0f4528
```

Related Information

Commands: `rpccp_add_mapping(8rpc)`, `rpccp_show_mapping(8rpc)`, `rpccp_show_server(8rpc)`.
remove member

Purpose

Removes a specified member from a group

Synopsis

rpccp remove member group-entry-name -m member [-s syntax]

Options

- **-m member**
  Declares the entry name of the group member to be removed (required).

- **-s syntax**
  Indicates the name syntax of the entry name (optional). The only value for this option is the **dce** name syntax, which is the default name syntax. Until an alternative name syntax becomes available, specifying the **-s** option is unnecessary.

Arguments

**group-entry-name**

Indicates the name of the target group. For an entry in the local cell, you can omit the cell name and specify only the cell-relative name.

Description

The **remove member** command removes a specified member from a specified group.

**Privileges Required**

You need **r** (**read**) permission and **w** (**write**) permission to the Cell Directory Service (CDS) object entry (the target group entry).

Notes

This command was replaced at DCE Version 1.1 by the **dcecp** command and might not be provided in future releases of DCE.

Examples

The following commands each remove a member from a group:

```bash
rpccp> remove member -m /./LandS/anthro/Cal_host_2 \\ /./LandS/anthro/Calendar_group
rpccp remove member -m /./LandS/anthro/Cal_host_3 \\ /./LandS/anthro/Calendar_group
```
Related Information

Commands: rpccp_add_member(8rpc), rpccp_remove_group(8rpc), rpccp_show_group(8rpc)
remove profile

Purpose

Removes all profile elements and the profile from the specified name service entry.

Synopsis

rpccp remove profile profile-entry-name [-s syntax]

Options

-s syntax

Indicates the name syntax of the entry name (optional). The only value for this option is the dce name syntax, which is the default name syntax. Until an alternative name syntax becomes available, specifying the -s option is unnecessary.

Arguments

profile-entry-name

Indicates the name of the target profile. For an entry in the local cell, you can omit the cell name and specify only the cell-relative name.

Description

The remove profile command removes a profile (and all of its elements) from the name service database. The entry name of the profile is required.

Privileges Required

You need w (write) permission to the Cell Directory Service (CDS) object entry (the target profile entry).

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

The following command removes the profile ./LandS/anthro/molly_o_profile:

rpccp> remove profile ./LandS/anthro/molly_o_profile

Related Information

Commands: rpccp_add_element(8rpc), rpccp_remove_element(8rpc), rpccp_show_profile(8rpc).
rpcprotseqs

Purpose

Determines the supported protocol sequences on a given host, and prints them to stdout.

Synopsis

rpcprotseqs

Description

The `rpcprotseqs` command determines the supported protocol sequences on a given host by calling the RPC runtime function: `rpc_network_inq_protseqs()`, and looping through the resultant protocol sequence vector to print out the protocol sequence strings.

Privileges Required

No special privileges are needed to use the `rpcprotseqs` command.

Examples

```bash
rpcprotseqs
nacn_ip_tcp
ncadg_ip_udp
```

Related Information

Subroutines: `rpc_network_inq_protseqs`
rpcresolve

Purpose

Recursively resolves the elements of a namespace entry.

Synopsis

rpcresolve –d | –n entry [–s] [–p] [–l #]

Options

–d Use the RPC_DEFAULT_ENTRY environment variable.
–l # Levels to recursively descend.
–n entry Namespace entry to resolve.
–p Profile entry information printed.
–s Server entry information printed.

Description

This is a utility program that recursively resolves the elements of a namespace entry. Profile and Group entries have their elements or members printed out, and Server entries have their object UUIDs, interface id, and bindings printed out. Each time an element needs further resolution, the printout of the next resolved piece of information is indented, to make it easier to tell which pieces of information go with which entry.

Examples

$ rpcresolve -n /./:NSTEST_DIR/profile
  -p

(P) /./NSTEST_DIR/profile
  (E) element : /.../cathywang_cell/NSTEST_DIR/profile1
    interface id: d5c89800-6dae-11c9-a1c1-08002b102989,0,0
    priority : 0
    annotation : IF1
  (P) /./cathywang_cell/NSTEST_DIR/profile1
    (E) element : /.../cathywang_cell/NSTEST_DIR:entry1
      interface id: d5c89800-6dae-11c9-a1c1-08002b102989,0,0
      priority : 0
      annotation : IF1
    (S) /.../cathywang_cell/NSTEST_DIR/entry1
      (E) element : /.../cathywang_cell/NSTEST_DIR/profile2
        interface id: d5c89800-6dae-11c9-a1c1-08002b102989,0,0
        priority : 1
        annotation : IF1
      (P) /.../cathywang_cell/NSTEST_DIR/profile2
        (E) element : /.../cathywang_cell/NSTEST_DIR/entry2
          interface id: d5c89800-6dae-11c9-a1c1-08002b102989,0,0
          priority : 0
          annotation : IF1
        (S) /.../cathywang_cell/NSTEST_DIR/entry2
          (E) element : /.../cathywang_cell/NSTEST_DIR/profile3

interface id: d5c89800-6dae-11c9-a1c1-08002b102989,1,1
priority : 0
annotation : IF11
(P) /.../cathywang_cell/NSTEST_DIR/profile3
  (E) element : /.../cathywang_cell/NSTEST_DIR/entry3
    interface id: d5c89800-6dae-11c9-a1c1-08002b102989,1,1
    priority : 0
    annotation : IF11
(S) /.../cathywang_cell/NSTEST_DIR/entry3
(E) element : /.../cathywang_cell/NSTEST_DIR/profile4
  interface id: 00000000-0000-0000-0000-000000000000,0,0
  priority : 0
  annotation : IF30
(P) /.../cathywang_cell/NSTEST_DIR/profile4
  (E) element : /.../cathywang_cell/NSTEST_DIR/entry4
    interface id: d5c89800-6dae-11c9-a1c1-08002b102989,3,0
    priority : 0
    annotation : IF30
(S) /.../cathywang_cell/NSTEST_DIR/entry4

Resolve server entry information:

$ rpcresolve -n/.:/NSTEST_DIR/group

(G) /.:/NSTEST_DIR/group
(G) /.:/cathywang_cell/NSTEST_DIR/group1
(S) /.:/cathywang_cell/NSTEST_DIR/entry1
(0) fbe696e0-6dae-11c9-b093-08002b102989
(0) 02d52fc0-6daf-11c9-b958-08002b102989
(1) d5c89800-6dae-11c9-a1c1-08002b102989,0,0
    (B) ncadg_ip_udp:127.0.0.1[1234]
    (B) ncadg_ip_udp:16.20.16.54 1249
(1) e9eb0340-6dae-11c9-b23d-08002b102989,0,0
    (B) ncadg_ip_udp:127.0.0.1[1234]
(S) /.:/cathywang_cell/NSTEST_DIR/entry2
(1) d5c89800-6dae-11c9-a1c1-08002b102989,0,0
    (B) ncadg_ip_udp:127.0.0.1[1234]
(G) /.:/cathywang_cell/NSTEST_DIR/group2
(S) /.:/cathywang_cell/NSTEST_DIR/entry1
(0) fbe696e0-6dae-11c9-b093-08002b102989
(0) 02d52fc0-6daf-11c9-b958-08002b102989
(1) d5c89800-6dae-11c9-a1c1-08002b102989,0,0
    (B) ncadg_ip_udp:127.0.0.1[1234]

Related Information

Commands: rpccp(8rpc), dcecp(8dce)

Purpose

Shows the NSI attributes of a name service entry

Synopsis

rpccp show entry entry-name [-i if-id] [-s syntax] [-u]

Options

-`i` *if-id*  Selects a specified interface identifier (optional). Only elements containing that identifier are shown. The interface identifier value has the following form:

```
interface-uuid, major-version, minor-version
```

The Universal Unique Identifier (UUID) is a hexadecimal string and the version numbers are decimal strings, for example:

```
-`i` ecleeb60-5943-11c9-a309-08002b102989,1.1
```

Leading zeros in version numbers are ignored.

-`s` *syntax*  Indicates the name syntax of the entry name (optional). The only value for this option is the `dce` name syntax, which is the default name syntax. Until an alternative name syntax becomes available, specifying the `-s` option is unnecessary.

-`u`  Updates the local Cell Directory Service (CDS) cache copy of name service data (optional). Name service data is cached locally on each machine in a cell. If an `rpccp` inquiry can be satisfied by data in the local CDS cache, this cached data is returned. Locally cached copies of name service data might not include a recent CDS update, however. If the required data is not available in the local CDS cache, `rpccp` goes to a CDS server(s) to retrieve the required data. `rpccp` then updates the local CDS cache.

Using the `-u` option bypasses the local cache, allowing `rpccp` to go directly to a CDS server for the inquiry. The local CDS cache is then updated by `rpccp`.

Arguments

*entry-name*  Indicates the name of the target name service entry. For an entry in the local cell, you can omit the cell name and specify only the cell-relative name.

Description

The `show entry` command shows the Name Service Interface (NSI) attributes of a name service entry. The name of the entry is required.

Note that this operation shows all of the compatible bindings for a given interface.
show entry(8rpc)

The **show entry** command shows the same list of string bindings as the import operation returns for the specified entry. This list includes all string bindings that refer to a major version that matches the specified version and a minor version that is equal to or greater than the specified version. The list might include string bindings exported for other versions of the interface that are upwardly compatible, rather than for this particular version of the interface.

**Privileges Required**

You need **r** *(read)* permission to the CDS object entry (the target name service entry).

**Notes**

This command was replaced at DCE Version 1.1 by the **dcecp** command and might not be provided in future releases of DCE.

**Examples**

The following commands show the NSI attributes of name service entries:

```plaintext
rpccp show entry ./LandS/anthro/Cal_host_3
rpccp> show entry ./LandS/anthro/Calendar_group
```

**Related Information**

Commands: **rpccp_add_entry(8rpc)**, **rpccp_remove_entry(8rpc)**.
Purpose

Shows the members of a group

Synopsis

```
rpccp show group group-entry-name [-m member] [-r [integer]] [-s syntax] [-u]
```

Options

- `-m member`
  Declares the name of a single group member.

- `-r [integer]`
  Indicates that the `show group` operation recurses. If any members of a group are also groups, their entries are shown. By default, the `-r` option causes the `show group` operation to recurse until all nested groups are expanded; for example, `-r` shows the members of the specified group and all nested groups.

  You can limit recursion to one or more levels by specifying a decimal integer as part of the `-r` option. For example, `-r 1` shows the members of the specified group and, for members that are groups, the command also shows their members; then recursion stops.

  Without the `-r` option, only the members of the specified group are shown.

- `-s syntax`
  Indicates the name syntax of the entry name (optional). The only value for this option is the `dce` name syntax, which is the default name syntax. Until an alternative name syntax becomes available, specifying the `-s` option is unnecessary.

- `-u`
  Updates the local Cell Directory Service (CDS) cache copy of name service data (optional).

  Name service data is cached locally on each machine in a cell. If an `rpccp` inquiry can be satisfied by data in the local CDS cache, this cached data is returned. Locally cached copies of name service data might not include a recent CDS update, however. If the required data is not available in the local CDS cache, `rpccp` goes to a CDS server(s) to retrieve the required data. `rpccp` then updates the local CDS cache.

  Using the `-u` option bypasses the local cache, allowing `rpccp` to go directly to a CDS server for the inquiry. `rpccp` then updates the local CDS cache.

Arguments

```
  group-entry-name
```

Indicates the name of the target group. For an entry in the local cell, you can omit the cell name and specify only the cell-relative name.

Description

The `show group` command shows the members of a group in the name service database. The entry name of the group is required. Unless it is limited to a specific
member (by the -m option), the `show group` command shows all members. The command shows only the members in the specified group; the -r option enables you to show members of nested groups.

**Privileges Required**

You need `r` (read) permission to the CDS object entry (the target group entry). If you use the -r option, you also need `r` (read) permission to any nested groups.

**Notes**

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

**Examples**

1. The following example shows all the members of a group, in the order in which they were added to the group:

   ```sh
   rpc> show group /./LandS/anthro/Calendar_group
   ```

2. The following command shows a specific member of a group:

   ```sh
   rpc> show group -m /./LandS/anthro/Cal_host_2 \
       /./LandS/anthro/Calendar_group
   ```

**Related Information**

Commands: `rpcp_add_member(8rpc)`, `rpcp_remove_group(8rpc)`, `rpcp_remove_member(8rpc)`
show mapping

Purpose

Shows the elements of the local or a remote endpoint map

Synopsis

```
rpccp show mapping [host-address] [-i if-id [ -v versions ] ] [-o object-uuid]
```

Options

- `-i if-id` Defines an interface identifier to be shown (optional). Only one interface can be shown in a single operation. If specified, only elements containing this interface identifier are shown. The `-i` option can be qualified by the `-v` option. The value has the following form:

```
interface-uuid, major-version, minor-version
```

The Universal Unique Identifier UUID is a hexadecimal string and the version numbers are decimal strings, as in the following example:

```
-i ecleeb60-5943-11c9-a309-08002b102989,1.1
```

Leading zeros in version numbers are ignored.

- `-v versions` Indicates how a specified interface version is used (optional). If it is used without the `-i` option, the `-v` option is ignored. The possible combinations of versions for the `-v` option and their actions are as follows:

```
all
exact
compatible
major_only
upto
```

If the `-v` option is absent, the command shows compatible version numbers.

- `-o object-uuid` Defines an object to be shown (optional). Each `show mapping` command accepts up to 32 `-o` options. The UUID is a hexadecimal string, as in the following example:

```
-o 3c6b8f60-5945-11c9-a236-08002b102989
```
Arguments

`host-address`

The `host-address` argument is a string binding that indicates where to find
the target endpoint map. When accessing the local endpoint map, you can
specify which protocol sequence to use (optional), as in the following
example:

```plaintext
ncadg_ip_udp:
```

When accessing a remote endpoint map, you must specify both a protocol
sequence and a network address for the remote system (required), as in
the following example:

```plaintext
ncadg_ip_udp:16.20.16.44
```

An endpoint is unnecessary in local or remote host addresses, and the
remove mapping command ignores any endpoint specified as part of a
host address.

Description

The `show mapping` command shows elements of an endpoint map. Each element
corresponds to an object UUID, interface identifier, annotation, and binding
information. The binding information contains a remote procedure call (RPC)
protocol sequence, a network address, and an endpoint within square brackets, as
follows:

```plaintext
rpc-prot-seq: network-addr[endpoint]
```

The endpoint map can be either the local endpoint map or the endpoint map of a
specified remote host. If entered without a remote host address, the command
accesses the local endpoint map. For the local endpoint map, a `show mapping`
command without any options displays all the map elements. For a remote endpoint
map, map elements are accessible only for protocol sequences that are supported
on both your system and the remote system.

The options list a selected subset of map elements. The `-i` option selects a specific
interface, and the `-v` option qualifies the `-i` option. The `-o` option selects a specific
object. You can use from 0 to 32 `-o` options per command. The options work
together to specify the subset of elements for the target protocol sequence(s).

Notes

Note that to ensure that you can remotely display all map elements from every
remote endpoint map, run the RPC control program on a system that supports all of
the protocol sequences available in your network environment.

This command was replaced at DCE Version 1.1 by the `dcecp` command and might
not be provided in future releases of DCE.

Examples

1. The following command shows the map elements in the local endpoint map that
   contain the specified interface identifier:

```plaintext
show mapping(8rpc)
```
show mapping(8rpc)

```bash
rpccp> show mapping -i ec1eeb60-5943-11c9-a309-08002b102989,1.1
```

2. The following command accesses the endpoint map of the remote host specified by the host address (`ncadg_ip_udp:16.20.16.44`) and displays the one map element that contains both the specified interface identifier and the specified object UUID:

```bash
rpccp show mapping -i ec1eeb60-5943-11c9-a309-08002b102989,1.1 \
-o 30dbeea0-fb6c-11c9-8eea-08002b0f4528 ncadg_ip_udp:16.20.16.44
```

Related Information

Commands: `rpccp_remove_mapping(8rpc)`, `rpccp_show_server(8rpc)`. 
show profile

Purpose

Shows the elements of a profile

Synopsis

rpccp show profile profile-entry-name [-d | -a annotation | -i if-id | [-v versions] | -m member ] [-r [integer]] [-s syntax] [-u]

Options

-d Selects the default profile element. With the -d option, the -a, -i, and -m options are ignored.

Although that the -a option does operate with the -d option, you should not use them together.

-a annotation

Declares a single annotation field (optional). The -a option selects only elements containing the specified annotation. The option is case sensitive.

The -a option works alone or in combination with the -i or -m options or both; only elements containing all the specified values are displayed.

Note that the shell supports the use of " " (quotation marks) around the annotation field of profile elements, allowing you to include internal spaces in an annotation; the control program does not. To specify or refer to annotations from within the control program, limit each annotation to an unbroken alphanumeric string; for example, CalendarGroup. To refer to annotations from the system prompt, do not incorporate quotation marks into any annotation.

-i if-id Selects a specified interface identifier (optional). Only elements containing that interface identifier are shown. The interface identifier value has the following form:

interface-uuid, major-version. minor-version

The Universal Unique Identifier UUID is a hexadecimal string and the version numbers are decimal strings, for example:

-i eclee60-5943-11c9-a309-08002b102989,1.1

Leading zeros in version numbers are ignored.

The -i option works alone or in combination with the -a or -m options or both; only elements containing all the specified values are displayed. When the -d option is specified, the -i option is ignored.

-v versions

Indicates how a specified interface version is used (optional). If it is used without the -i option, the -v option is ignored. The possible combinations of versions for the -v option and their actions are as follows:

all The interface version is ignored.
show profile(8rpc)

**exact**  Both the major and minor versions must match the specified versions.

**compatible**

The major version must match the specified version, and the minor version must be greater than or equal to the specified version.

**major_only**

The major version must match the specified version; the minor version is ignored.

**upto**

The major version must be less than or equal to that specified. If the major versions are equal, the minor version must be less than or equal to that specified.

If the -v option is absent, the command shows compatible version numbers.

**-m member**

Declares a single member name (optional). Only elements containing that member name are shown.

The -m option works alone or in combination with the -a or -i options or both; only elements containing all the specified values are displayed. When the -d option is specified, the -m option is ignored.

**-r [integer]**

Indicates that the show profile operation recurses. If the member of any element of a profile is also a profile, its elements are shown. By default, the -r option causes the show profile operation to recurse until all nested profiles are expanded; for example, -r shows the elements of the specified profile and of all nested profiles.

You can limit recursion to one or more levels by specifying a decimal integer as part of the -r option. For example, -r 1 shows the elements of the specified profile and, for element members that are profiles, the command also shows their elements; then recursion stops.

Without the -r option, only the profile elements in the specified entry are shown.

**-s syntax**

Indicates the name syntax of the entry name (optional). The only value for this option is the dce name syntax, which is the default name syntax. Until an alternative name syntax becomes available, specifying the -s option is unnecessary.

**-u**

Updates the local Cell Directory Service (CDS) cache copy of name service data (optional). Name service data is cached locally on each machine in a cell. If an rpccp inquiry can be satisfied by data in the local CDS cache, this cached data is returned. Locally cached copies of name service data might not include a recent CDS update, however. If the required data is not available in the local CDS cache, rpccp goes to a CDS server(s) to retrieve the required data. rpccp then updates the local CDS cache.

Using the -u option bypasses the local cache, allowing rpccp to go directly to a CDS server for the The local CDS cache is then updated by rpccp.

**Arguments**

**profile-entry-name**

Indicates the name of the target profile. For an entry in the local cell, you can omit the cell name and specify only the cell-relative name.
The `show profile` command shows the elements of a profile in the name service database. The entry name of the profile is required.

By default, all elements in the profile are shown. You can select a subset of the elements by specifying the `-a`, `-i`, or `-m` options. The `-r` option enables you to show nested profiles.

**Privileges Required**

You need `r` (read) permission to the CDS object entry (the target profile entry). If you use the `-r` option, you also need `r` (read) permission to any nested profiles.

**Notes**

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

**Examples**

1. The following command shows the cell profile `/./cell-profile` in the local cell:

   ```
   rpcgp show profile `/./cell-profile`
   ```

2. This sequence sets up an environment variable `MOLLY_O PROFILE`, which represents the user profile `/./LandS/anthro/molly_o_profile`, exports it, and show the user profile associated with the `MOLLY_O PROFILE` environment variable:

   ```
   MOLLY_O_PROFILE="/./LandS/anthro/molly_o_profile"
   export MOLLY_O_PROFILE
   rpcgp
   rpcgp> show profile MOLLY_O_PROFILE
   ```

**Related Information**

Commands: `rpcgp_add_element(8rpc)`, `rpcgp_remove_element(8rpc)`, `rpcgp_remove_profile(8rpc)`.
show server

Purpose

Shows binding information, interface identifiers, and object UUIDs in a server entry

Synopsis

```
rpccp show server server-entry-name [-i [if-id] ] [-o [object-uuid] ] [-s syntax] [-u]
```

Options

- **-i [if-id]**
  Shows interface identifiers from binding information found in the entry (optional). Without the -i option, the command displays all interface identifiers.

  To display a specific interface, supply its identifier as the value. The value has the following form:

  `interface-uuid, major-version.minor-version`

  The Universal Unique Identifier UUID is a hexadecimal string and the version numbers are decimal strings, for example:

  `-i ec1eeb60-5943-11c9-a309-08002b102989,1.1`

  Leading zeros in version numbers are ignored.

- **-o [object-uuid]**
  Shows object UUIDs found in the entry (optional). Without the -o option, the command displays all object UUIDs. To display a specific object UUID, supply its string representation as the value, as in the following example:

  `-o 3c6b8f60-5945-11c9-a236-08002b102989`

- **-s syntax**
  Indicates the name syntax of the entry name (optional). The only value for this option is the `dce` name syntax, which is the default name syntax. Until an alternative name syntax becomes available, specifying the -s option is unnecessary.

- **-u**
  Updates the local Cell Directory Service (CDS) cache copy of name service data (optional). Name service data is cached locally on each machine in a cell. If an `rpccp` inquiry can be satisfied by data in the local CDS cache, this cached data is returned. Locally cached copies of name service data might not include a recent CDS update, however. If the required data is not available in the local CDS cache, `rpccp` goes to a CDS server(s) to retrieve the required data. `rpccp` then updates the local CDS cache.

  Using the -u option bypasses the local cache, allowing `rpccp` to go directly to a CDS server for the inquiry. The local CDS cache is then updated by `rpccp`. 
Arguments

**server-entry-name**
Indicates the name of the target server. For an entry in the local cell, you can omit the cell name and specify only the cell-relative name.

Description

The **show server** command shows the remote procedure call (RPC) binding information, interface identifiers, and object UUIDs in a server entry. The entry name of the server entry is required.

This operation shows all of the potential bindings for an interface. By default, this command displays bindings for the specified version of the interface and for upwardly compatible versions of the interface.

Privileges Required

You need **r** (read) permission to the CDS object entry (the target server entry).

Notes

This command was replaced at DCE Version 1.1 by the **dcecp** command and might not be provided in future releases of DCE.

Examples

1. The following command shows a server entry in the local cell:

   ```
   rpccp> show server ./LandS/anthro/Cal_host_2
   ```

2. The following command displays a specific object and interface from a server entry in the local cell:

   ```
   rpccp show server ./LandS/anthro/Cal_host_2 \ 
   -o 16977538-e257-11c9-8dc0-08002b0f4528 \ 
   -i ec1eeb60-5943-11c9-a309-08002b102989,1.1
   ```

Related Information

Commands: **rpccp_export(8rpc), rpccp_import(8rpc), rpccp_unexport(8rpc).**
unexport(8rpc)

unexport

Purpose

Removes binding information, interface identifiers, and object UUIDs from a server entry

Synopsis

rpccp unexport entry-name {[-i if-id [-v versions] | [-o object-uuid] [-s syntax]}

Options

-\i if-id  Defines an interface identifier to be unexported (optional). Only one interface can be unexported in a single operation. If specified, binding information for this interface is removed from the entry. The -i option can be qualified by the -v option. The value has the following form:

interface-uuid, major-version.minor-version

The Universal Unique Identifier (UUID) is a hexadecimal string and the version numbers are decimal strings, for example:

-\i ecee60-5943-11c9-a309-08002b102989,1.1

Leading zeros in version numbers are ignored.

-\v versions  Indicates how a specified interface version is used (optional). If it is used without the -i option, the -v option is ignored. The possible combinations of versions for the -v option and their actions as follows:

all The interface version is ignored.

exact Both the major and minor versions must match the specified versions.

compatible The major version must match the specified version, and the minor version must be greater than or equal to the specified version.

major_only The major version must match the specified version; the minor version is ignored.

upto The major version must be less than or equal to that specified. If the major versions are equal, the minor version must be less than or equal to that specified.

If the -v option is absent, the command shows compatible version numbers.

-\o object-uuid  Defines an object to be unexported (optional). Each unexport command accepts up to 32 -o options. The UUID is a hexadecimal string, for example:

-\o 3c6b8f60-5945-11c9-a236-08002b102989
unexport(8rpc)

-s syntax
Indicates the name syntax of the entry name (optional). The only value for this option is the dce name syntax, which is the default name syntax. Until an alternative name syntax becomes available, specifying the -s option is unnecessary.

-u
Updates the local copy of name service data (optional). Name service data that is requested by applications is sometimes stored locally. If a local copy of name service data satisfies an NSI command, the RPC control program uses the local copy. Local copies of name service data are not automatically updated. Specify the -u option to display the current contents of an entry that has recently changed.

Arguments

entry-name
Indicates the name of the target name service entry. Usually, the target is a server entry. However, objects also can be exported (without an interface identifier or binding information) to a group or a profile.

For an entry in the local cell, you can omit the cell name and specify only the cell-relative name.

Description

The unexport command removes binding information and an interface identifier, object UUIDs, or both from a server entry, or it removes object UUIDs from a group’s entry. The command requires the entry name and either the interface identifier or one or more object UUIDs.

By default, the unexport operation removes compatible interface versions.

Privileges Required

You need both r (read) and w (write) permission to the Cell Directory Service (CDS) object entry (the target name service entry).

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

This sequence of commands sets up an environment variable Calendar_1_1, which represents the interface identifier of a remote procedure call (RPC) interface, exports it, and removes (unexports) the Calendar Version 1.1 interface from the server entry /./LandS/anthro/Cal_host_2 in the local cell:

```
Calendar_1_1=ec1eb60-5943-11c9-a309-08002b102989,1.1
export Calendar_1_1
rpccp
rpccp> unexport -i Calendar_1_1 /./LandS/anthro/Cal_host_2
```

Related Information

Commands: rpcp_export(8rpc), rpcp_import(8rpc), rpcp_show_server(8rpc)
unexport(8rpc)
Purpose

Introduction to CDS commands

Description

The DCE Cell Directory Service (CDS) provides the following management commands:

**cdsbrowser**

Starts the CDS browser utility. This utility is based on the OSF/Motif® graphical user interface. The browser can display an overall directory structure as well as show the contents of directories.

**cdscp**

Starts the CDS control program. Use this command-line interface to manage the CDS components and the contents of your namespace.

**Note:** Most of the `cdscp` command functions are also available through the `dcecp` administrative tool. See [Chapter 1. DCE Commands](#) on page 1. The `cdscp` tool was available in previous versions of DCE and is provided in this version to ease migration. Be aware that `cdscp` might not be available in future releases of DCE. Tools and scripts that depend on `cdscp` need to be converted to use `dcecp` and Tool Command Language (TCL). For more information on `dcecp`, see [Description](#) on page 451.

The following commands are typically started automatically by scripts that execute as part of normal system startup procedures. See the reference pages for these commands before using them.

**cdsadv**

Starts the advertisement and solicitation daemon on the local system and then starts clerks as needed by applications. Use this command only when troubleshooting, because it creates and automatically starts the CDS clerk whenever the host system is rebooted.

**cdsd**

Starts the CDS server. Use this command only when troubleshooting, because it starts the CDS server process automatically whenever the host system is rebooted.

**gdad**

Starts the Global Directory Agent (GDA) daemon. GDA enables intercell communication, serving as a connection to other cells through the global naming environment. GDA is typically started automatically by scripts that execute as part of normal system startup and shutdown procedures.

Related Information

Commands: `cdsadv(8cds)`, `cdsbrowser(8cds)`, `cdscp(8cds)`, `cdsd(8cds)`, `gdad(8cds)`, `dced(8dce)`.

add directory

Purpose

Adds a value to a modifiable, set-valued attribute of a directory

Synopsis

`cdscp add directory directory-name attribute-name = attribute-value`

Arguments

`directory-name`
The full name of the directory.

`attribute-name`
The name of a particular attribute. Specify only one attribute at a time. See the `cds_attributes` file for the list of attributes that your application uses.

`attribute-value`
The value of a particular attribute. The value of an application-defined attribute is dependent on the type of attribute. See the `cds_attributes` file for the list of attributes and corresponding data types that your application uses. If you enter a byte data type, you must enter an even number of digits in length. You can enter only pairs of hexadecimal values for user-defined attributes.

Description

The `add directory` command adds a value to a modifiable, set-valued attribute (including application-defined attributes) of a directory. If the attribute does not exist, this command creates it. Usually, this task is performed through the client application. See the `IBM DCE Version 3.1 for AIX and Solaris: Administration Guide` for more information about attributes.

Privileges Required

You must have `w (write)` permission to the directory.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

To add the value `ontario` to the attribute `myname` of a directory named `/./sales`, read the `cds_attributes` file to verify that the attribute exists, as shown in the following:

```
OID     LABEL       SYNTAX
1.3.22.1.3.91 myname    char
```

Then enter the following command to assign the value `ontario` to the attribute `myname`:
add directory(8cds)

cdscp add directory /.:sales myname = ontario

Related Information

Commands: remove_directory(8cds), show_directory(8cds).

add object

Purpose

Adds a value to a modifiable, set-valued attribute of an object entry

Synopsis

```shell
cdscp add object object-name attribute-name = attribute-value
```

Arguments

- `object-name`
  - The full name of the object entry.
- `attribute-name`
  - The name of a particular attribute. Specify only one attribute at a time. See the `cds_attributes` file for the list of attributes and corresponding data types that your application uses.
- `attribute-value`
  - The value of a particular attribute. The value of an application-defined attribute is dependent on the type of attribute.

Description

The `add object` command adds a value to a modifiable, set-valued attribute (including application-defined attributes) of an object entry. If the attribute does not exist, this command creates it. Usually, this task is performed through the client application. See the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide* for more information about attributes.

Privileges Required

You must have `w` (write) permission to the object entry.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

To add the value `ps` to the attribute `printcap` of an object entry named `./:/subsys/deskprinter`, read the `cds_attributes` file to verify that the attribute exists, as shown in the following:

<table>
<thead>
<tr>
<th>OID</th>
<th>Label</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.22.1.3.70</td>
<td>printcap</td>
<td>char</td>
</tr>
</tbody>
</table>

Then enter the following command to assign the value `ps` to the attribute `printcap`:

```shell
cdscp> add object ./:/subsys/deskprinter printcap = ps
```
add object(8cds)

Related Information

Commands: create object(8cds), delete object(8cds), list object(8cds), remove object(8cds), set object(8cds), show object(8cds).

catraverse

Purpose
Traverses the clerk cache

Synopsis

    catraverse [-m cacheid] [-n filename]

Options

- **-m cacheid**
  Specifies the shared memory ID (obtain with the `ipcs` AIX command).

- **-n filename**
  Specifies the clerk cache file name.

Description

The `catraverse` command traverses the clerk cache data. It is used for debugging purposes only.

**Privileges Required**

No special privileges are needed to use the `catraverse` command.

Examples

None.

Related Information

None.
cdsadv

Purpose

Starts the CDS client daemon

Synopsis

```
```

Options

- `-c size` Specifies cache size in kilobytes. Changing cache sizes causes previously cached information to be discarded, including information about cached servers, so use of this option might necessitate defining a new cached server.
- `-D` For debugging use only. Causes the `cdsadv` process to not fork.
- `-I` Caches all advertisements, even from other cells on the same LAN. Normally, the advertiser only caches information within the advertiser’s local cell.
- `-P` Enables the proxy function. The proxy function allows one or more advertisers in your cell to advertise on behalf of CDS servers in your cell. The servers you might want to proxy on behalf of are those that normally cannot broadcast directly because they are located on a WAN or on a separate LAN segment that prevents broadcasting. See the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide—Core Components* for more information about the proxy function.
- `-s` Causes the `cdsadv` process not to send or receive advertisements. This argument can be used for diagnostic work involving multiple servers on the same local area network to limit access to those servers identified with the `define cached server` command.
- `-w route` Routes serviceability messages.

Description

The `cdsadv` command starts the Cell Directory Service (CDS) client daemon.

Privileges Required

You must log in as superuser (root).

Notes

This command is ordinarily executed by a DCE configuration or startup script. You should use this command interactively only when the `cdsadv` process fails to start automatically after a reboot, or if you want to restart the `cdsadv` process after disabling it to perform a backup or to do diagnostic work on the host system.
Examples

To restart the `cdsadv` process, follow these steps:
1. Log in to the clerk system as **superuser** (root).
2. Verify that the `dced` process is running.
3. Enter the following command to restart the `cdsadv` process:

```
cdsadv
```

Related Information

Commands: `gdad(8cds)`, `dced(8dce)`.

cdsbrowser

Purpose

Starts the CDS browser utility on the local system

Synopsis

cdsbrowser

Description

The **cdsbrowser** command starts the Cell Directory Service (CDS) browser utility on the local system. This utility runs on workstations with windowing software based on the OSF/Motif® graphical user interface. Using a mouse to manipulate pull-down menus, you can view the directory structure of a namespace, view child directories of a particular directory, view the object entries and soft links in a directory, and set a filter to display only object entries of a particular class. (Similar functions are available within the CDS control program, **cdscp**, for users who do not have windowing software.) When you use the CDS browser, it sets the confidence level of clerk calls to **low**. This means that the information displayed is obtained from the local CDS client cache if available.

Related Information

cdsclerk

Purpose

Manages the interface between clients and the CDS server

Synopsis

```bash
cdsclerk [-D] [-n] [-w route]
```

Options

- **-D** For debugging use only. Causes the `cdsadv` clerk process not to fork.
- **-n** Runs the CDS clerk without the Advertiser. This configuration is also referred to as the “Slim Client” and reduces the amount of memory required for the CDS client. It does not support the full advertisement capability; therefore, some functionality is reduced. See the *IBM DCE Version 3.1 for AIX: Quick Beginnings* or *IBM DCE Version 3.1 for Solaris: Quick Beginnings* for more information.
- **-w route** Routes serviceability messages.

Description

The `cdsclerk` command manages the interface between clients and the Cell Directory Service (CDS) server.

**Privileges Required**

You must log in as **superuser** (*root*).

Notes

This command is used by the advertiser on the system on which the CDS clerk is running. You should use this command interactively only to do diagnostic work on the host system.

Examples

Before you start the `cdsclerk` process, you must make sure that other clerks are not running. To start the `cdsclerk` process, follow these steps:

1. Make sure that a CDS server is already running somewhere within the cell.
2. Log into the system as **superuser** (*root*).
3. Log into DCE as the machine principal of the local host. Enter the principal name in the format `/hosts/hostname/self`, as shown in the following example for a host named `orion` whose password is `smith`:

   ```bash
dce_login hosts/orion/self smith
   ```

4. Enter the following command to see whether the `dced` process is already running:

   ```bash
   ps
   ```
5. If the **dced** process appears on the list of active processes, proceed to step 6. If the **dced** process does not appear on the list of active processes, enter the following command to start the process:

```
dced
```

6. Enter the following command to start the **cdsadv** process:

```
cdsadv
```

7. The **cdsclerk** process will start when the user issues any command that requires access to cds.

**Related Information**

cdscp

Purpose

Starts the CDS control program

Synopsis

**cdscp** [**cdscp-command**]

Arguments

**cdscp-command**

Optionally specifies one of the following control commands:

- **add directory**
  - Adds a value to a modifiable, set-valued attribute (including application-defined attributes) of a directory

- **add object**
  - Adds a value to a modifiable, set-valued attribute (including application-defined attributes) of an object entry

- **clear cached server**
  - Removes knowledge of a server that you had specifically defined from the local clerk’s cache

- **clear clearinghouse**
  - Removes knowledge of the specified clearinghouse from the server’s memory

- **create child**
  - Creates a child pointer at the master replica of the parent directory

- **create clearinghouse**
  - Creates a clearinghouse on the local server system or makes an existing clearinghouse available

- **create directory**
  - Creates a directory

- **create link**
  - Creates a soft link and optionally specifies an expiration time and an extension time

- **create object**
  - Creates a new object entry

- **create replica**
  - Creates a replica of an existing directory in the specified clearinghouse

- **define cached server**
  - Creates knowledge of a server in the local clerk’s cache

- **delete child**
  - Deletes a child pointer from the namespace

- **delete clearinghouse**
  - Deletes the specified clearinghouse from the local server system
cdscp(8cds)

delete directory
    Deletes a directory

delete link
    Deletes a soft link

delete object
    Deletes an object entry

delete replica
    Deletes a read-only replica of a directory from a clearinghouse

disable clerk
    Stops the clerk on the local system

disable server
    Stops the server on the local system

dump clerk cache
    Displays the contents of the clerk cache

help
    Displays a list of the CDS control program commands

list child
    Displays a list of all the child pointers whose names match the specified child name

list clearinghouse
    Displays a list of all the clearinghouses whose names match the specified clearinghouse name

list directory
    Displays a list of all the directories whose names match the specified directory name

list link
    Displays a list of all the soft links whose names match the specified link name

list object
    Displays a list of all the object entries (including clearinghouse object entries) whose names match the specified object entry name

remove directory
    Removes a value from a set-valued or single-valued attribute (including application-defined attributes) of a directory

remove link
    Removes a soft link's timeout value attribute

remove object
    Removes a value from a set-valued or single-valued attribute (including application-defined attributes) of an object entry

set cdscp confidence
    Sets the confidence level of clerk calls issued as a result of CDS control program commands

set cdscp preferred clearinghouse
    Specifies a preferred clearinghouse to use for satisfying read requests that result from CDS control program commands

set directory
    Changes the value of a modifiable, single-valued attribute of a directory
set directory to new epoch
Reconstructs a directory’s replica set, allowing you to designate a
new master replica or to exclude a replica

set directory to skulk
Starts the skulk of a directory immediately

set link
Changes the value of a modifiable, single-valued attribute of a soft
link

set object
Changes the value of a modifiable, single-valued attribute of an
object entry

show cached clearinghouse
Displays current information about the specified cached
clearinghouse

show cached server
Displays address information of a server in the local clerk’s cache

show cdscp confidence
Displays the current confidence level of clerk calls resulting from
CDS control program commands

show cdscp preferred clearinghouse
Displays the preferred clearinghouse for satisfying read requests
that result from CDS control program commands

show cell
Displays the information you need to create a cell entry in either
DNS or GDS

show child
Displays attribute information about the specified child pointer

show clearinghouse
Displays attribute information about the specified clearinghouse

show clerk
Displays attribute information about the CDS clerk on the local
system

show directory
Displays attribute information about the specified directory

show link
Displays attribute information about the specified soft link

show object
Displays attribute information about the specified object entry

show replica
Displays attribute information about the specified replica

show server
Displays attribute information about the server running on the local
system

Description

Note:
With the exception of the following subcommands, this command was replaced at DCE Version 1.1 by the `dcecp` command. This command might be fully replaced by the `dcecp` command in a future release of DCE, and might no longer be supported at that time.

- `disable clerk`
- `disable server`
- `help`
- `set cdscp confidence`
- `set directory to new epoch`
- `show cdscp confidence`
- `show cell`
- `show clerk`
- `show server`

This control program has been superseded by `dcecp`. It is not designed for international use, and might give unexpected or undesirable results when used in non-English environments. If you are working with non-English data, you should use `dcecp`.

The Cell Directory Service (CDS) control program, `cdscp`, is a command-line interface for managing CDS components and the contents of the namespace.

You can use the control program commands from within the control program or from the system prompt. To use the control program commands from inside the control program, start the control program by using the `cdscp` command without any argument. This enters the control program, which displays the control program prompt, `cdscp>`, as shown in the following:

```
cdscp
```

```
cdscp>
```

At this prompt, you can enter any control program command, for example:

```
cdscp> show server
```

Enter `do filename` at the control program prompt to read commands from the file `filename`.

To leave the control program and return to the system prompt, enter `quit`.

To use the control program commands from the system prompt, enter the `cdscp` command with an a control program command as the argument. The control program executes the command immediately, without displaying the control program prompt. For example, you can enter the `show server` command as follows:

```
cdscp show server
```

### Elements of a CDS Command

All CDS control program commands must include a verb, an entity name, and all required arguments. Depending on the command, you can also specify optional arguments and attributes. A space must separate more than one attribute or argument. A space must precede and follow any use of `=` (equal sign).

### Control Program Verbs

The following is a list of the definitions of verbs used in control program commands:
add
Add a value to a modifiable, set-valued attribute

clear
Removes knowledge of a cached clearinghouse or cached server from memory

create
Creates an entity

define
Creates knowledge of a locally cached server

delete
Deletes an entity

disable
Stops operation of a clerk or server

dump
Displays the contents of a clerk cache

list
Displays a list of specified entity names

remove
Removes a value from a set-valued or single-valued attribute

set
Changes the value of a modifiable, single-valued attribute

show
Displays attribute information

CDS Entities

Any individually manageable piece of CDS is called an entity. A set of commands exists for each entity. The following is a list of the entities and a description of what each entity represents:

Cached Clearinghouse
A cached clearinghouse is a clearinghouse that a clerk has discovered and cached. A clerk can learn about clearinghouses as a result of configuration information or advertisements received on a local area network (LAN), or during the process of finding a name.

Cached Server
A cached server is a server that a clerk has cached as a result of manual configuration through the control program.

Child
A child pointer connects a parent and child directory in a hierarchical namespace. The child pointer is stored in the parent directory and has the same name as the child directory.

Clearinghouse
A clearinghouse is a database containing a collection of directory replicas at a particular server.

Clerk
The clerk is the interface between client applications and servers.

Directory
A directory contains child, object, and link entries that are logically stored under one name (the directory name).

Link
A soft link is a pointer providing an alternate name for an object entry, directory, or other soft link.

Object
An object entry represents a resource (for example, an application) that is named in the namespace.

Replica
A replica is a copy of a directory. Each copy, including the original or master, is referred to as a replica.
Server
A server handles lookup requests from clerks and maintains the contents of the clearinghouse or clearinghouses at its node.

CDS Entity Attributes
Every CDS entity has attributes, which are pieces or sets of data associated with that entity. Attributes can reflect or affect the operational behavior of an entity, record the number of times a particular event or problem occurred since the entity was last enabled, and uniquely distinguish an entity from any other entity. Some attributes have a single value; others contain a set of values.

CDS attributes are identified by ISO object identifiers (OIDs). Every CDS attribute name maps to an OID and a corresponding data type. Usually, client applications define the name of an attribute and its data type. Application programmers should never need to modify (except for the purpose of foreign language translation) the existing CDS labels associated with the unique OIDs in the `cds_attributes` file. However, programmers can obtain new OIDs from the appropriate allocation authority, create new attributes for their own object entries, and then append them to the existing list. The OID and data type of each attribute are stored in the file `/opt/dcelocal/etc/cds_attributes`. Descriptions of the CDS data types that applications can use are in the `cdsclerk.h` file.

All entities have `show` commands that you can use to display the names and values of specific attributes or all attributes. When you display an attribute that has more than one value, the `show` command lists each value for the attribute separately. When there are multiple values for an attribute, the command first lists the attribute name on a line ending with a colon, then the parts of the value.

For more information about CDS attributes, see the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide*.

Editing CDS Control Program Commands
You can abbreviate commands, continue a command beyond one line, or redirect output to a file within the control program.

To abbreviate any command name, enter only the first four characters. You can abbreviate a command name to fewer than four characters as long as the abbreviated name remains unique among all command names in the control program. For example, the following commands are equivalent:

```
cdscp> show directory ./:/sales
cdscp> sh dir ./:/sales
```

To continue a long command line onto the next line, enter a space and then a `\` (backslash) at the end of the first line, for example:

```
cdscp> set link ./:/sales CDS_LinkTimeout \ > (1991-12-31-12:00:00 090-00:00:00)
```

To add a comment, use the `#` (number sign). Everything following the first number sign on a line is ignored.

To redirect output to a file, most UNIX shell users can enter `>` `filename` at the shell prompt. To redirect output of error text to a file, most UNIX shell users can enter `>&`
cdscp(8cds)

FILENAME at the shell prompt. For example, the following command redirects the
display produced by the show directory command to a new text file named
directory_names:

cdscp show directory ./:* > directory_names

Using Wildcard Characters

When entering a name in show and list commands, you can use wildcard
characters in the rightmost simple name (the name to the right of the last / (slash)
in the full pathname). The * (asterisk) matches zero or more characters in a simple
name. The ? (question mark) matches exactly one character in a simple name.

When you use an asterisk or a question mark as a normal character in the
rightmost simple name of a show or list command, precede it with a \ (backslash).
Otherwise, the character is interpreted as a wildcard.

You cannot use wildcard characters in show clerk and show server commands.

Privileges Required

CDS supports the following DCE permissions: r (read), w (write), i (insert), d
(delete), t (test), c (control), and A (Admin). Each permission has a slightly
different meaning, depending on the kind of CDS name with which it is associated.
In general, the permissions are defined as follows:

read Allows a principal to look up a name and view the attribute values
associated with it.
write Allows a principal to change the modifiable attributes associated with a
name, except the name’s access control list (ACL) entries.
insert For use with directory entries only. Allows a principal to create new names
in a directory.
delete Allows a principal to delete a name from the namespace.
test Allows a principal to test whether an attribute of a name has a particular
value without being able to actually see any of the values (that is, without
having read permission to the name).

Test permission provides application programs a more efficient way to verify
a CDS attribute value. Rather than reading an entire set of values, an
application can test for the presence of a particular value.

control
Allows a principal to modify the ACL entries associated with a name. (Note
that read permission is also necessary for modifying a CDS entry’s ACLs;
otherwise, acl_edit will not be able to bind to the entry.) Control permission
is automatically granted to the creator of a CDS name.

Admin
For use with directory entries only. Allows a principal to issue CDS control
program commands that control the replication of directories.

The creator of a name is automatically granted all permissions appropriate for the
type of name created. For example, a principal creating an object entry is granted
read, write, delete, test, and control permission to the object entry. A principal
creating a directory is granted read, write, insert, delete, test, control, and Admin
permission to the directory.
Examples

The following command starts the CDS control program:

```
cdscp
```
```
cdscp>
```

The following command displays the attributes of the CDS clerk on the local system:

```
cdscp show clerk
```

Related Information

cdsd

Purpose

Starts the CDS server

Synopsis


Options

- **-a**  Creates a new namespace if there is not an existing namespace. This flag is meaningful only when the cell is first configured (that is, during the initial creation of the namespace).
- **-D**  For debugging use only. Causes the cdscd process not to fork.
- **-l principal**  Sets locksmith mode. Allows the specified principal to have full access to all information stored with this server.
- **-v directory version**  Causes the cdscd to create new directories with the specified CDS directory version number. IBM supports versions 3.0 and 4.0.
- **-w route**  Routes serviceability messages.

Description

The cdscd command starts the Cell Directory Service (CDS) server.

Privileges Required

You must log in as superuser (root).

Notes

This command is ordinarily executed by a DCE configuration or startup script. You should use this command interactively only when a cdscd server fails to start automatically after a reboot, or if you want to restart a cdscd server after disabling it to perform a backup or to do diagnostic work on the host system.

Use the "v 4.0" option when all CDS clearinghouses and directories in the cell are based on OSF DCE release 1.1 or later. This enables the use of features such as cell aliasing and CDS recognition of extended privilege attribute certificates on ACLs in namespace entries. If you are creating a new cell based on OSF DCE release 1.1 or later and you do not use the "v 4.0" option, you must manually upgrade the CDS_DirectoryVersion attribute of the cell root directory to 4.0 to use the release 1.1 features in CDS. Refer to the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide—Core Components.
cdsd(8cds)

Examples

To restart a cdsd server, follow these steps:
1. Log in to the server system as superuser (root).
2. Verify that the dced and cdsadv processes are running.
3. Enter the following command to restart the CDS server:

   `cdsd`

When the server process starts, it starts all clearinghouses on the system.

Related Information

Commands: cdsadv(8cds), dced(8dce).

cds_dbdump

Purpose

Dumps the CDS server database.

Synopsis

```
cds_dbdump {-c | -t} file_spec
```

```
cds_dbdump -c checkpoint_file
```

```
cds_dbdump -t transaction_log
```

Options

- `-c` Specifies that the checkpoint file should be dumped.
- `-t` Specifies that the transaction log file should be dumped.

Description

This utility is used to dump the server checkpoint and transaction log files.

Privileges Required

No special privileges are needed to use the `cds_dbdump` command.

Examples

None.

Related Information

`IBM DCE Version 3.1 for AIX and Solaris: Administration Guide—Core Components`
Purpose

Deletes recursively the namespace of cells.

Synopsis

```bash
```

Options

- `[-C]` Specifies a confidence level to use (low, medium, or high).
- `[-d]` Specifies a directory.
- `[-o]` Specifies an object.
- `[-l]` Specifies a link.
- `[-r]` Indicates recursive delete including the directory specified.
- `[-R]` Indicates to delete entries recursively, not including the directory specified.

`entity` A valid namespace entity.

Description

Directory and target can be in `/` or `/.../cellname` format. Directory and target can contain the wildcard characters `*` and `?`. Only trailing wildcarding is allowed.

When specifying `-R` or `-r` with `cdsdel`, the entity must be a directory. Any combination of the `-c`, `-o`, or `-l` options with `-R` or `-r` will result in a user error.

A DCE cell must be running in order for `cdsdel` to function.

Privileges Required

No special privileges are needed to use the `cdsdel` command.

Examples

The following example will delete object `./:foobar`.

```bash
cdsdel -o ./:foobar
```

The following example will delete everything under `./:old_dir` as well as `./:old_dir` itself.

```bash
cdsdel -r ./:old_dir
```

Related Information

Commands: `cdsli`
Purpose

Starts the CDS Diagnostic utility for the server running on the local system.

Synopsis

```
cdsd_diag
```

Description

The `cdsd_diag` command starts the CDS Diagnostic utility for the server running on the local system.

Privileges Required

You must have superuser (root) privileges on the local system.

Notes

To use this command you must have an extensive knowledge of DCE. The `cdsd_diag` command is intended to be used by service personnel for diagnostic purposes.

Examples

The following command starts the CDS Diagnostic utility on the local system:

```
cdsd_diag
```

Related Information

None.
cdsli

Purpose

Lists recursively the namespace of cells.

Synopsis

`cdsli [ -c [ -C ] [ -d ] [ -o ] [ -l ] [ -w ] [ -R ] [ -r ] directory ]`

Options

- `-c` Specifies a clearinghouse
- `-C` Specifies a confidence level to use (low, medium, or high).
- `-d` Specifies a directory.
- `-o` Specifies an object.
- `-l` Specifies a link.
- `-w` Specifies a long listing.
- `-R` Indicates to list entries recursively, not including the directory specified.
- `-r` Indicates to list entries recursively, including the directory specified.

`directory` A valid namespace directory.

Description

Directory and target can be in `/:` or `/.../cellname` format. Directory and target can contain the wildcard characters `*` and `?`. Only trailing wildcarding is allowed. See Examples.

Note: The user can leave the directory field blank and the enumeration will occur on the root directory.

A DCE cell must be running in order for `cdsli` to function.

Privileges Required

No special privileges are needed to use the `cdsli` command.

Examples

```
cdsli -d /:.
/.:/hosts
/.:/subsys

cdsli -dR /:
/.:/hosts
/.:/hosts/smith.austin.ibm.com
/.:/subsys
/.:/subsys/dce/dfs
/.:/subsys/dce/dfs/bak
/.:/subsys/dce/sec
```
cdsl

**cdsl -oR /:/hosts**

/:/hosts
/:/hosts/smith.austin.ibm.com/cds-clerk
/:/hosts/smith.austin.ibm.com/cds-server
/:/hosts/smith.austin.ibm.com/profile
/:/hosts/smith.austin.ibm.com/self

**cdsl -wodR**

o  /:/cell-profile
o  /:/fs
o  /:/lan-profile
o  /:/sec
d  /:/hosts
d  /:/subsys

**Related Information**

Commands: **cdsdel**

Chapter 3. Cell Directory Service Commands 463
clear cached server

Purpose

Removes knowledge of a user-defined server from the local clerk’s cache.

Synopsis

```bash
cdscp clear cached server simplename
```

Arguments

```bash
simplename
```

The simple name given to the cached server when it is created.

Description

The `clear cached server` command removes knowledge of a server from the local clerk’s cache. You can clear only those servers that you have created with the `define cached server` command.

Privileges Required

You must have `w` (write) permission to the clerk.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command removes knowledge of the server `nrl` from the clerk cache:

```
cdscp> clear cached server nrl
```

Related Information

Commands: `define_cached_server(8cds)`, `dump_clerk_cache`, `show_cached_server(8cds)`. 
clear clearinghouse

Purpose

Removes knowledge of a clearinghouse from the server’s memory

Synopsis

\texttt{cdscp clear clearinghouse clearinghouse-name}

Arguments

\textit{clearinghouse-name}

The full name of the clearinghouse.

Description

The \texttt{clear clearinghouse} command removes knowledge of the specified clearinghouse from the server’s memory. The clearinghouse files are not deleted. This ensures that the clearinghouse is not automatically enabled on server restarts. If you issue a \texttt{list clearinghouse} command, the clearinghouse will still be listed.

Before you can delete a cleared clearinghouse, you must use the \texttt{create clearinghouse} command to recreate it. After recreating the clearinghouse, you can use the \texttt{delete clearinghouse} command to remove it.

This command is part of the process of relocating a clearinghouse. See the \textit{IBM DCE Version 3.1 for AIX and Solaris: Administration Guide} for more information.

\textbf{Note:} Once a clearinghouse has been disabled, skulks can fail until the clearinghouse is enabled again.

Privileges Required

You must have \texttt{w} (write) permission to the server on which the clearinghouse resides.

Notes

This command was replaced at DCE Version 1.1 by the \texttt{dcecp} command and might not be provided in future releases of DCE.

Examples

The following command clears the clearinghouse /.:\texttt{Paris2\_CH} before moving it to another server:

\texttt{cdscp clear clearinghouse /.:\texttt{Paris2\_CH}}

Related Information

Commands: \texttt{create clearinghouse(8cds)}, \texttt{delete clearinghouse(8cds)}, \texttt{list clearinghouse(8cds)}, \texttt{set cdscp preferred clearinghouse(8cds)}, \texttt{show cdscp preferred clearinghouse(8cds)}, \texttt{show clearinghouse(8cds)}. 
clear clearinghouse(8cds)

create child

Purpose

Creates a child pointer at the master replica of the parent directory

Synopsis

`cdscp create child child-name clearinghouse clearinghouse-name`

Arguments

- `child-name`  
  The full name of the child pointer.
- `clearinghouse clearinghouse-name`  
  The full name of a clearinghouse that contains a replica of the child directory.

Description

The `create child` command creates a child pointer at the master replica of the parent directory. When the Cell Directory Service (CDS) looks up a name in the namespace, it uses child pointers to locate directory replicas. Use the `set cdscp preferred clearinghouse` command before issuing this command to ensure that the request is directed to the master replica.

Privileges Required

You must have `i (insert)` permission to the parent directory.

Notes

Use the `create child` command only to recreate a child pointer that is accidentally deleted. This command is designed for troubleshooting only.

This command will fail if the associated directory does not exist. If the associated directory exists, this command will return successfully.

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command creates the child pointer in the parent directory `./:subsys`. It uses the replica located at the `./:subsys/NY_CH` clearinghouse to fill in its replica set.

```
cdscp> create child ./:subsys clearinghouse ./:subsys/NY_CH
```

Related Information

Commands: `delete_child(8cds)`, `list_child(8cds)`, `show_child(8cds)`. 
create clearinghouse

Purpose

Creates a clearinghouse or makes an existing clearinghouse available

Synopsis

`cdscp create clearinghouse clearinghouse-name`

Arguments

`clearinghouse-name`

The full name of the clearinghouse.

Description

The `create clearinghouse` command creates a clearinghouse on the local server system or makes an existing clearinghouse available. The server startup command usually creates a new clearinghouse when you configure a new Cell Directory Service (CDS) server. Occasionally, you might need to create a second clearinghouse on a particular server; for example, if you are temporarily relocating a clearinghouse on a different server. See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information about relocating a clearinghouse.

Clearinghouses should be named only in the root. When you enter the `create clearinghouse` command, CDS creates a read-only replica of the root directory and stores it in the new clearinghouse as the initial replica. Because the process that creates the new clearinghouse initiates a skulk of the root directory, all replicas of the root should be reachable when you enter the command.

Privileges Required

You need `w` (write) permission to the server on which you intend to create the clearinghouse and `A` (Admin) permission to the cell root directory. The server principal needs `r` (read), `w` (write), and `A` (Admin) permissions to the cell root directory.

Notes

This command is usually executed only by the network configuration procedure. To ensure that all replicas of the root are reachable, perform an immediate skulk of `./:` prior to issuing this command.

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command creates a clearinghouse named `./:Boston_CH` on the local server system:

```
cdscp> create clearinghouse ./:Boston_CH
```
Related Information

Commands: clear_clearinghouse(8cds), delete_clearinghouse(8cds), list_clearinghouse(8cds), set_cdscp_preferred_clearinghouse(8cds), show_cached_clearinghouse(8cds), show_cdscp_preferred_clearinghouse(8cds), show_clearinghouse(8cds).

create directory

Purpose

Creates a directory

Synopsis

```
cdscp create directory directory-name [clearinghouse clearinghouse-name]
```

Arguments

- `directory-name`:
  The full name of the directory.
- `clearinghouse` `clearinghouse-name`:
  The optional name of the clearinghouse in which to create the directory.

Description

The `create directory` command creates a directory with the name that you specify. If you do not specify a clearinghouse, the Cell Directory Service (CDS) creates the master replica of the directory in the same clearinghouse as the new directory’s parent directory.

Privileges Required

In order to create a directory, you must have `r` (read) and `i` (insert) permission to the parent directory, and `w` (write) permission to the clearinghouse in which the master replica of the new directory is to be stored. In addition, the server principal must have `r` (read) and `i` (insert) permission to the parent directory.

Notes

To ensure that all replicas are consistent, perform an immediate skulk of the parent directory after issuing this command.

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command creates a directory named `./:/sales`.

```
cdscp create directory ./:/sales
```

Related Information

Commands: `delete_directory(8cds)`, `list_directory(8cds)`, `set_directory(8cds)`, `set_directory_to_skulk(8cds)`, `show_directory(8cds)`.
create link

Purpose

Creates a soft link, optionally specifying expiration time and extension time

Synopsis

```bash
cdscp create link link-name
CDS_LinkTarget = target-name [CDS_LinkTimeout = (expiration-time extension-time)]
```

Arguments

- `link-name`
  The full name of the soft link.
- `CDS_LinkTarget = target-name`
  The full name of the entry to which the soft link points.
- `CDS_LinkTimeout = (expiration-time extension-time)`
  The expiration-time argument specifies a date and time after which CDS checks for existence of the soft link’s target and either extends or deletes the soft link. The value is specified as follows:
  ```
  yyyy-mm-dd-hh:mm:ss
  ```
  You can abbreviate this value.
  
  The extension-time argument specifies a period of time by which to extend the soft link’s expiration time (if the server has validated that the target still exists). The value is specified as follows:
  ```
  dd-hh:mm:ss
  ```
  You can abbreviate this value.

Description

The `create link` command creates a soft link. If you specify the `CDS_LinkTimeout` attribute, you must specify an expiration time and an extension time. If you omit the `CDS_LinkTimeout` attribute, the soft link is permanent and must be explicitly deleted.

Privileges Required

You must have `i` (insert) permission to the directory in which you intend to create the soft link.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.
create link(8cds)

Examples

The following command creates a permanent soft link named /.:sales/tokyo/price-server that points to an object entry named /.:sales/east/price-server. The expiration value indicates that the Cell Directory Service (CDS) will check that the destination name /.:sales/east/price-server still exists on June 25, 1995, at 12:00 p.m. If the destination name still exists, the soft link remains in effect another 90 days. Thereafter, CDS will check that the destination name exists every 90 days.

cdscp> create link /.:sales/tokyo/price-server \
CDS_LinkTarget = /.:sales/east/price-server \
CDS_LinkTimeout = (1995-06-25-12:00:00 90-00:00:00)

Note: If you are entering the cdscp create link command directly from the command line, remember to escape the ( ) metacharacters by using the forward slash (/).

Related Information

Commands: delete_link(8cds), list_link(8cds), set_link(8cds), show_link(8cds).
create object

Purpose

Creates an object entry

Synopsis

```
cdscp create object object-name [CDS_Class = class-name] [CDS_ClassVersion = value]
```

Arguments

- `object-name`
  The full name of the object entry.
- `CDS_Class = class-name`
  The class of object entry being created. You can specify an application-defined class name. A class is specified as a simple name limited to 31 characters.
- `CDS_ClassVersion = value`
  The version of the class assigned to the object entry. Specify the value as `v.n`, where `v` defines the major release number and `n` specifies the minor version number. Specifying a class version is useful in that it allows the definition of a class to evolve as the application is revised.

Description

The `create object` command creates an object entry. This task is usually done through a client application.

Privileges Required

You must have `i` (insert) permission to the parent directory.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command creates an object entry named `./:sales/east/floor1cp`. The object entry describes a color printer on the first floor of a company’s eastern sales office.

```
cdscp> create object ./:sales/east/floor1cp \ 
CDS_Class = printer CDS_ClassVersion = 1.0
```

Related Information

Commands: `delete_object(8cds)`, `list_object(8cds)`, `set_object(8cds)`, `show_object(8cds)`.
create replica

Purpose

Creates a replica of an existing directory in the specified clearinghouse.

Synopsis

```
cdscp create replica directory-name clearinghouse clearinghouse-name
```

Arguments

- `directory-name`
  The full name of the directory.
- `clearinghouse` `clearinghouse-name`
  The full name of the clearinghouse in which you want to create the replica.

Description

The `create replica` command creates a replica of an existing directory in the specified clearinghouse.

Privileges Required

You must have A (Admin) permission to the directory you intend to replicate and w (write) permission to the clearinghouse that stores the new replica. The server principal needs r (read), w (write), and A (Admin) permission to the directory you intend to replicate.

Notes

This command is usually executed only by the network configuration procedure. To ensure that all replicas are consistent, perform an immediate skulk of the parent directory after issuing this command.

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command creates a replica of the `./:mfg` directory in the clearinghouse `./:Paris_CH`:

```
cdscp create replica ./:mfg clearinghouse ./:Paris1_CH
```

Related Information

Commands: `delete_replica(8cds)`, `show_replica(8cds)`, `dcecp(8cds)`.
**define cached server**

**Purpose**

Creates knowledge of a server in the local clerk's cache

**Synopsis**

```
cdscp define cached server name tower value
```

**Arguments**

- **name**  
  A simple name for the cached server.

- **tower value**  
  The protocol sequence and network address of the server node. The format is `protocol-sequence: network-address`. A `protocol-sequence` is a character string identifying the network protocols used to establish a relationship between a client and server. There are two choices of protocol sequence, depending on the network address that is supplied in the binding: `ncacn_ip_tcp` or `ncadg_ip_udp`. For the `network-address`, specify an Internet address using the common Internet address notation. For more information about this format, see the *IBM DCE Version 3.1 for AIX and Solaris: Application Development Reference*.

**Description**

The `define cached server` command creates knowledge of a server in the local clerk's cache. This command is typically used to manually provide configuration information to a clerk that cannot automatically configure itself. This is required, for instance, to give the clerk addressing information about a server across a wide area network (WAN). Once the clerk knows about one server, it can find other servers through referrals. The server being added to the cache must be running and reachable when this command is issued.

**Privileges Required**

You must have `w` (write) permission to the clerk.

**Notes**

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

**Examples**

The following command creates knowledge of the server `nrl` in the local clerk's cache:

```
cdscp> define cached server nrl tower ncacn_ip_tcp:16.20.15.25
```
define cached server(8cds)

Related Information

Commands: clear_cached_server(8cds), dump_clerk_cache(8cds), show_cached_server(8cds).

delete child

Purpose

Deletes a child pointer from the namespace

Synopsis

```bash
cdscp delete child child-name
```

Arguments

- `child-name`
  - The full name of the child pointer.

Description

The `delete child` command deletes a child pointer from the namespace.

Privileges Required

You must have `d` (delete) permission to the child pointer or `A` (Admin) permission to the parent directory.

Notes

Use the `delete child` command only when the directory to which the child pointer refers is deleted and the child pointer accidentally remains.

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command deletes the child pointer that accidentally remains after the `/./sales/east` directory is deleted:

```bash
cdscp> delete child /./sales/east
```

Related Information

Commands: `create_child(8cds)`, `list child(8cds)`, `show child(8cds)`.
delete clearinghouse

Purpose

Deletes the specified clearinghouse from the local server system.

Synopsis

\texttt{cdscp delete clearinghouse clearinghouse-name}

Arguments

\texttt{clearinghouse-name}

The full name of the clearinghouse.

Description

The \texttt{delete clearinghouse} command deletes a clearinghouse from the local server system. The Cell Directory Service (CDS) does not permit you to delete a cleared clearinghouse. Before you can delete a cleared clearinghouse, you must recreate it using the \texttt{create clearinghouse} command.

The \texttt{delete clearinghouse} command automatically deletes all read-only replicas from a clearinghouse. CDS does not permit you to delete a clearinghouse that contains a master replica. See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information about handling master replicas when deleting a clearinghouse.

Permissions Required

You must have \texttt{w (write)} and \texttt{d (delete)} permission to the clearinghouse and \texttt{A (Admin)} permission to all directories that store replicas in the clearinghouse. The server principal needs \texttt{d (delete)} permission to the associated clearinghouse object entry and \texttt{A (Admin)} permission to all directories that store replicas in the clearinghouse.

Notes

It is recommended that you delete all replicas except the root before issuing this command.

This command was replaced at DCE Version 1.1 by the \texttt{dcecp} command and might not be provided in future releases of DCE.

Examples

The following command deletes a clearinghouse named \texttt{./sales/Orion_CH} from the local server system:

\texttt{cdscp delete clearinghouse ./sales/Orion_CH}
Related Information

Commands: clear_clearinghouse(8cds), create_clearinghouse(8cds), list_clearinghouse(8cds), set_cdscp_preferred_clearinghouse(8cds), show_clearinghouse(8cds), show_cdscp_preferred_clearinghouse(8cds).

delete directory

Purpose

Deletes a directory

Synopsis

\texttt{cdscp delete directory directory-name}

Arguments

\textit{directory-name}

The full name of the directory.

Description

The \texttt{delete directory} command deletes a directory. The directory cannot contain any object entries, soft links, or child pointers. The master replica must be the only remaining replica in the cell. Use the \texttt{delete replica} command if you need to remove read-only replicas.

Privileges Required

You must have \texttt{d (delete)} permission to the directory and \texttt{w (write)} permission to the clearinghouse that stores the master replica of the directory. The server principal needs \texttt{A (Admin)} permission to the parent directory or \texttt{d (delete)} permission to the child pointer that points to the directory you intend to delete.

Notes

This command was replaced at DCE Version 1.1 by the \texttt{dcecp} command and might not be provided in future releases of DCE.

Examples

The following command deletes the directory \\
\texttt{./eng} from the namespace:

\texttt{cdscp> delete directory ./eng}

Related Information

Commands: \texttt{create_directory(8cds)}, \texttt{delete_replica(8cds)}, \texttt{list_directory(8cds)}, \texttt{set_directory(8cds)}, \texttt{set_directory_to_skulk(8cds)}, \texttt{show_directory(8cds)}. 
delete link

Purpose

Deletes a soft link

Synopsis

`cdscp delete link link-name`

Arguments

`link-name`

The full name of the soft link.

Description

The `delete link` command deletes a soft link.

Privileges Required

You must have `d` (delete) permission to the soft link, or `A` (Admin) permission to the directory that stores the soft link.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command deletes the soft link `././sales/asia`.

```
  cdscp> delete link ././sales/asia
```

Related Information

Commands: `create_link(8cds)`, `list_link(8cds)`, `set_link(8cds)`, `show_link(8cds)`.
delete object

Purpose

Deletes an object entry

Synopsis

cdscp delete object object-name

Arguments

object-name

The full name of the object entry.

Description

The delete object command deletes an object entry. This task is usually performed through the client application, except under certain circumstances—for example, if the application is obsolete or no longer has access to the namespace.

Note: This command deletes the object entry only in the namespace. It does not delete the actual object that the name represents.

Privileges Required

You must have d (delete) permission to the object entry, or A (Admin) permission to the directory that stores the object entry.

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

The following command deletes the object entry /./sales/east/floor1pr2:

cdscp delete object /./sales/east/floor1pr2

Related Information

Commands: create_object(8cds), list_object(8cds), set_object(8cds), show_object(8cds)
delete replica

Purpose

Deletes a read-only replica of a directory from a clearinghouse

Synopsis

```
cdscp delete replica directory-name clearinghouse clearinghouse-name
```

Arguments

- `directory-name` -- The full name of the directory
- `clearinghouse clearinghouse-name` -- The full name of the clearinghouse

Description

The `delete replica` command deletes a read-only replica of a directory from a clearinghouse. Use the `delete directory` command to delete the master replica of the directory.

Privilgues Required

You must have `A` (Admin) permission to the directory whose replica you want to delete and `w` (write) permission to the clearinghouse from which you are deleting the replica.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command deletes a read-only replica of the directory `./:mfg` from the `./:/Paris1_CH` clearinghouse:

```
cdscp> delete replica ./:mfg clearinghouse ./:/Paris1_CH
```

Related Information

Commands: `create_replica(8cds)`, `delete_directory(8cds)`, `show_replica(8cds)`.
disable clerk

Purpose

Stops the clerk on the local system

Synopsis

\texttt{cdscp disable clerk}

Description

The \texttt{disable clerk} command stops the clerk on the local system, causing all active communication with any server to be aborted and all client calls in progress to fail. The clerk cache is copied to disk.

\textbf{Privileges Required}

You must have \texttt{w (write)} permission to the clerk.

Notes

If you are disabling a clerk on a system where a server is running, make sure you disable the server first.

This command might be replaced in future releases of DCE by the \texttt{dcecp} command, and might no longer be supported at that time.

Examples

The following command stops the clerk on the local server system:

\texttt{cdscp> disable clerk}

Related Information

Command: \texttt{show clerk(8cds)}. 
disable server

Purpose

Stops the server on the local system

Synopsis

`cdscp disable server`

Description

The `disable server` command stops the server on the local system. The server is disabled after all transactions in progress are completed.

Note: To be able to restart the server successfully, you must also stop and restart the advertiser first. Use the `disable clerk` command to stop the advertiser and the `start.dce` command to restart the advertiser and server together

Privileges Required

You must have `w` (write) permission to the server.

Notes

This command might be replaced in future DCE releases by the `dcecp` command, and might no longer be supported at that time.

Examples

The following command stops the server on the local system:

`cdscp disable server`

Related Information

Command: `show server(8cds)`.
dump clerk cache

Purpose

Displays the contents of the clerk cache

Synopsis

cdscp dump clerk cache

Description

The dump clerk cache command displays the contents of the clerk cache on the screen. Use this command when solving Cell Directory Service (CDS) problems.

Privileges Required

You must have superuser (root) privileges on the clerk system. No CDS permissions are required.

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

The following command displays the contents of the clerk cache on the screen:

cdscp> dump clerk cache

Related Information

Command: show clerk.
gdad

Purpose

Starts the GDA daemon

Synopsis

```
```

Options

- **-b** This option disables the ability to use DNS as the global directory service.

- **-d routing**

  Sets the debug routing. This option is for debugging use only.

- **-D** This option is used for debugging only. It causes the gdad process not to fork.

- **-r resolv.conf file**

  This option can be used to specify the file name of the resolve configuration file. The default is /etc/resolv.conf.

- **-s named.ca.file**

  This option can be used to indicate the file name of the bind server named data file. The default is /etc/named.data.

- **-u** This option can be specified to prevents gdad from updating the GDA Parent Pointer on/.. This option is for debugging use only.

- **-w route**

  This option routes serviceability messages.

- **-x** This option disables the ability to use X.500 as the global directory service.

- **-l** This option disables the ability to use LDAP as the global directory service.

- **-h addr_LDAP_server**

  Address of the LDAP server in which DCE cell information is registered. The value addr_LDAP_server can be specified as:

  host[:port] where host is the hostname running the LDAP server. Alternatively, the host can be specified as an IP address in dotted decimal format.

  port is the port on which the LDAP server is listening; this is needed if the default port [389] is not used.

  This option -h is required when -l is not present.

- **-a authentication_DN**

  The distinguished name (DN), specified in LDAP name syntax that will be authenticated for successive operations to use.

- **-p password_for_authentication_DN**

  The password that is used to authenticate the distinguished name (DN). This option is used with LDAP only.
The `gdad` command starts the Global Directory Agent (GDA) daemon. The GDA enables intercell communication, serving as a connection to other cells through the global naming environment. The default, `gdad`, starts all services, that is BIND, X.500, and LDAP.

**Privileges Required**

You must log in as **superuser (root)**.

**Notes**

This command is ordinarily executed by a DCE configuration or startup script. You should use this command interactively only when a `gdad` process fails to start automatically after a reboot, or if you want to restart the GDA daemon after disabling it to perform a backup or do diagnostic work on the host system.

**Examples**

To restart the GDA daemon process, follow these steps:

1. Log in to the system as **superuser (root)**.
2. Verify that the `dced` and `cdsadv` processes are running.
3. Enter the following command to restart the `gdad` process:

   ```
gdad
   ```

   To stop the GDA, enter the following command:

   ```
kill pid
   ```

   where `pid` is the process identifier of the `gdad` process.

**Related Information**

**ldap_addcell**

**Purpose**

Registers DCE cell information in a server which supports LDAP.

**Synopsis**

```bash
ldap_addcell -h ldap_server -a authentication_DN -p password [-o object_class, object_class...] [-d]
```

**Options**

- `-h ldap_server`
  The name of the LDAP server targeted to hold the binding.

- `-a authentication_DN`
  The distinguished name (DN), specified in LDAP name syntax that will be authenticated for successive operations to use.

- `-p password`
  The password that is used to authenticate the distinguished name (DN).

- `-o object_class, object_class...`
  Value or values of the attribute object_class for the entry (the registration) being created or modified. Note that if you are listing more than one object_class value, you must separate them with commas.

  If no object_class value is specified, it is assumed that the object exists and an attempt is made to modify its CDS_CELL and CDS_REPLICAS attributes. If this attempt fails, another attempt is made with the addition of the object_class value of dceCellInfo.

  **Note:** dceCellInfo is the recommended auxiliary object_class for the CDS_CELL and CDS_REPLICAS attributes.

- `-d`
  Deletes the DCE cell information attributes from the entry in the directory. It does not remove the entire directory entry.

**Description**

The `ldap_addcell` command registers DCE cell information in a server which supports LDAP. It returns a 0 on success and a 1 on error.

**Privileges Required**

You must log in as **superuser** (root) to run the `ldap_addcell` command.

**Examples**

The following ldap_add cell examples assume the following:

- bermuda.austin.ibm.com is the ldap server machine name
- gdatest is a user that has write access to the ldap server
- gdatest is also the password of the user gdatest
- an organizationalUnit is allowed to contain the auxiliary object dceCellInfo
- the ldap server does schema checking
**ldap_addcell**

Example 1 shows the normal creation of the cell bindings in the ldap server.

```bash
ldap_addcell -h bermuda.austin.ibm.com -a "cn=gdatest,ou=austin,o=ibm,c=us" \ 
  -p "gdatest" -o organizationalUnit,dceCellInfo
```

Example 2 shows the deletion of the CDS CELL and CDS REPLICAS attributes.

```bash
ldap_addcell -h bermuda.austin.ibm.com -a "cn=gdatest,ou=austin,o=ibm,c=us" \ 
  -p "gdatest" -d
```

Example 3 shows the changing of the CDS CELL and CDS REPLICAS attributes in an object that already exists.

```bash
ldap_addcell -h bermuda.austin.ibm.com -a "cn=gdatest,ou=austin,o=ibm,c=us" \ 
  -p "gdatest"
```

Environment variables used: Each parameter of the `ldap_addcell` command has a corresponding environment variable which is used when the corresponding parameter is not present on the `ldap_addcell` command invocation. The `ldap_addcell` parameters and the corresponding environment variable are as follows:

<table>
<thead>
<tr>
<th><code>ldap_addcell</code> parameter</th>
<th>environment variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h</td>
<td>LDAP_SERVER</td>
</tr>
<tr>
<td>-a</td>
<td>LDAP_AUTH_DN</td>
</tr>
<tr>
<td>-p</td>
<td>LDAP_AUTH_DN_PW</td>
</tr>
<tr>
<td>-o</td>
<td>LDAP_OBJECT_CLASS</td>
</tr>
</tbody>
</table>

If the cell entry is already registered, the CDS CELL and CDS REPLICAS attributes are replaced with new values for this cell.

**Related Information**

None.
list child

Purpose

Displays a list of all child pointers matching a specified child name

Synopsis

\texttt{cdscp list child \textit{child-name} [with attribute-name = attribute-value]}

Arguments

\textit{child-name} \\
The full name of a specific child pointer. The last simple name can contain wildcard characters.

\textbf{with attribute-name} \\
The name of a particular attribute.

\textit{attribute-value} \\
The value of a particular attribute.

Description

The list child command displays a list of all the child pointers whose names match the specified child name. The last simple name can contain wildcard characters. You can use a \texttt{with attribute-name= attribute-value} clause to limit output only to child pointers whose attributes have values equal to the specified values. Spaces must precede and follow the = (equal sign).

Privileges Required

You must have \texttt{r (read)} permission to the directory that stores the child pointer. If you use a \texttt{with attribute-name= attribute-value} clause in the command, you also need \texttt{r (read)} or \texttt{t (test)} permission to the selected child pointers.

Notes

This command was replaced at DCE Version 1.1 by the \texttt{dcecp} command and might not be provided in future releases of DCE.

Examples

The following command displays a list of all the child pointers named in the \texttt{./sales} directory:

\begin{verbatim}
  cdscp> list child ../sales/*
  LIST
  CHILD  /.../abc.com/sales
  AT  1991-10-15-15:56:00
  Q1
  Q2
  Q3
  Q4
\end{verbatim}
list child(8cds)

Related Information

Commands: create_child(8cds), delete_child(8cds), show_child(8cds).
list clearinghouse

Purpose

Displays a list of all clearinghouses matching a specified clearinghouse name

Synopsis

```
cdscp list clearinghouse clearinghouse-name [with attribute-name = attribute-value]
```

Arguments

- **clearinghouse-name**: The full name of a specific clearinghouse. The last simple name can contain wildcard characters.
- **with attribute-name**: The name of a particular attribute.
- **attribute-value**: The value of a particular attribute.

Description

The `list clearinghouse` command displays a list of all the clearinghouses whose names match the specified name. The last simple name can contain wildcards. You can use a `with attribute-name = attribute-value` clause to limit output only to clearinghouses whose attributes have values equal to the specified values. Spaces must precede and follow the `=` (equal sign).

Privileges Required

You must have `r` (read) permission to the directory that stores the associated clearinghouse object entry. If you use a `with attribute-name = attribute-value` clause in the command, you also need `r` (read) or `t` (test) permission to the selected clearinghouses.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command displays a list of all the clearinghouses named in the root directory:

```
cdscp> list clearinghouse /.*
```

```
LIST
CLEARINGHOUSE /.../abc.com/*
AT 1991-10-15-15:56:00
/.../abc.com/Munich_CH
/.../abc.com/Paris_CH
```
list clearinghouse(8cds)

Related Information

Commands: clear_clearinghouse(8cds), create_clearinghouse(8cds),
delete_clearinghouse(8cds), set_cdscp_preferred_clearinghouse(8cds),
show_cdscp_preferred_clearinghouse(8cds), show_clearinghouse(8cds).
list directory

Purpose

Displays a list of all directories matching a specified directory name

Synopsis

`cdscp list directory directory-name [with attribute-name = attribute-value]`

Arguments

directory-name
   The full name of a specific directory. The last simple name can contain wildcard characters.

with attribute-name
   The name of a particular attribute.

attribute-value
   The value of a particular attribute.

Description

The `list directory` command displays a list of all the directories whose names match the specified directory name. The last simple name can contain wildcards. You can use a `with attribute-name = attribute-value` clause to limit output only to directories whose attributes have values equal to the specified values. Spaces must precede and follow the `= (equal sign)`.

Privileges Required

You must have `r (read)` permission to the parent directory. If you use a `with attribute-name = attribute-value` clause in the command, you also need `r (read)` or `t (test)` permission to the selected directories.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command displays the names of all the directories in the `./sales` directory:

```
cdscp> list directory ./sales/*
```

```
LIST
DIRECTORY    /.../abc.com/sales
atlanta
austin
boston
```
list directory(8cds)

   chicago
   ontario
   ny
   seattle

Related Information

Commands: add_directory(8cds), create_directory(8cds),
delete_directory(8cds), remove_directory(8cds), set_directory(8cds),
set_directory_to_skulk(8cds), show_directory(8cds).
list link

Purpose

Displays a list of all soft links matching a specified link name

Synopsis

`cdscp list link link-name [with attribute-name = attribute-value]`

Arguments

`link-name`

The full name of a specific soft link. The last simple name can contain wildcard characters.

`with attribute-name`

The name of a particular attribute.

`attribute-value`

The value of a particular attribute.

Description

The `list link` command displays a list of all the soft links whose names match the link name that you specify. The last simple name can contain wildcard characters. You can use a `with attribute-name = attribute-value` clause to limit output only to soft links whose attributes have values equal to the specified values. Spaces must precede and follow the `=` (equal sign). This command does not list the name of the directory, object entry, or other soft link to which the soft link points.

Privileges Required

You must have `r` (read) permission to the directory that stores the soft link. If you use a `with attribute-name = attribute-value` clause in the command, you also need `r` (read) or `t` (test) permission to the selected soft links.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command displays a list of all the soft links whose names begin with the letter `l` in the directory `/.:admin`:

```
cdscp> list link /.:admin/l*

LIST
SOFTLINK /.../abc.com/admin
1nk01
1nk02
```
list link(8cds)

1nk03
1nk04
1nk05
1nk06

Related Information

Commands: create link(8cds), delete link(8cds), remove link(8cds), set link(8cds), show link(8cds).
list object

Purpose

Lists the specified object entries matching a specified object entry name

Synopsis

```
cdscp list object object-name [with attribute-name = attribute-value]
```

Arguments

- `object-name`
  
  The full name of a specific object entry. The last simple name can contain wildcard characters.

- `with attribute-name`
  
  The name of a particular attribute.

- `attribute-value`
  
  The value of a particular attribute.

Description

The `list object` command displays a list of all the object entries (including clearinghouse object entries) whose names match the object entry name that you specify. The last simple name can contain wildcard characters. You can use a `with attribute-name = attribute-value` clause to limit output only to object entries whose attributes have values equal to the specified values. Spaces must precede and follow the `=` (equal sign).

Privileges Required

You must have `r` (read) permission to the directory that stores the object entry. If you use a `with attribute-name = attribute-value` clause in the command, you also need `r` (read) or `t` (test) permission to the selected object entries.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command displays a list of all the object entries named in the directory `./.:eng`:

```
cdscp> list object ./.:eng/*
LIST
OBJECT /.../abc.com/eng
juno
test_stats
work_disk1
work_disk2
```
list object(8cds)

Related Information

Commands: add_object(8cds), create_object(8cds), delete_object(8cds), remove_object(8cds), set_object(8cds), show_object(8cds).
remove directory

Purpose

Removes a value from a set-valued or single-valued attribute of a directory

Synopsis

```
cdscp remove directory directory-name attribute-name [= attribute-value ]
```

Arguments

directory-name
  The full name of the directory.

attribute-name
  The name of a particular attribute. Specify only one attribute at a time. See the `cds_attributes` file for the list of attributes and corresponding data types that your application uses.

attribute-value
  The value of a particular attribute. The value of an application-defined attribute is dependent on the type of attribute.

Description

The `remove directory` command removes a value from a set-valued or single-valued attribute (including application-defined attributes) of a directory. If the attribute is set-valued, one value at a time can be removed. When the last value is removed, the result is <empty set>. If you do not specify a value, the command removes the entire attribute. If you do not specify a value, the command removes the entire attribute. This command can delete attributes created by the `add directory` and `set directory` commands. Usually this task is performed through the client application. See the `IBM DCE Version 3.1 for AIX and Solaris: Administration Guide` for more information about attributes.

Privileges Required

You must have `w` (write) permission to the directory.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

To remove the value 1 from the user-defined, set-valued attribute `dirregion` of a directory named `./sales`, follow these steps:

1. Read the `cds_attributes` file to make sure that the attribute `dirregion` is listed, as shown in the following:

<table>
<thead>
<tr>
<th>OID</th>
<th>LABEL</th>
<th>SYNTAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.22.1.3.66</td>
<td>dirregion</td>
<td>small</td>
</tr>
</tbody>
</table>

2. Use the `remove directory` command with the appropriate attributes.

```bash
cdscp remove directory ./sales dirregion 1
```
2. Enter the following command to remove the value 1 from the attribute `dirregion`:

```
    cdscp> remove directory /.:sales dirregion = 1
```

Related Information

Commands: `add_directory(8cds)`, `list_directory(8cds)`, `set_directory(8cds)`, `set_directory_to_skulk(8cds)`, `show_directory(8cds)`.

remove link

Purpose

Removes a soft link’s timeout value attribute

Synopsis

cdscp remove link link-name CDS_LinkTimeout

Arguments

link-name

The full name of the soft link.

Description

The remove link command removes a soft link’s timeout value attribute, CDS_LinkTimeout, causing the soft link to become permanent.

Privileges Required

You must have w (write) permission to the soft link.

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

The following command removes the timeout value attribute of a soft link named /.:/eng/link01:

cdscp remove link /.:/eng/link01 CDS_LinkTimeout

Related Information

Commands: create_link(8cds), delete_link(8cds), list_link(8cds), set_link(8cds), show_link(8cds).
remove object

Purpose

Removes a value from a set-valued or single-valued attribute of an object entry

Synopsis

cdscp remove object object-name attribute-name [= attribute-value ]

Arguments

object name
The full name of the object entry.

attribute-name
The name of a particular attribute. Specify only one attribute at a time. See the cds_attributes file for the list of attributes and corresponding data types that your application uses.

attribute-value
The value of a particular attribute. The value of an application-defined attribute is dependent on the type of attribute.

Description

The remove object command removes a value from a set-valued or single-valued attribute (including application-defined attributes) of an object entry. If the attribute is set-valued, one value at a time can be removed. When the last value is removed, the result is <empty set>. If you do not specify a value, the command removes the entire attribute. If you do not specify a value, the command removes the entire attribute. This command can delete attributes created by the add object and set object commands. Usually, this task is performed through the client application. See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information about attributes.

Privileges Required

You must have w (write) permission to the object entry.

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

To remove the value ps from the attribute printcap of an object entry named /.:mih/deskprinter, follow these steps:

1. Read the cds_attributes file to check that the attribute printcap is listed, as shown in the following:

<table>
<thead>
<tr>
<th>OID</th>
<th>LABEL</th>
<th>SYNTAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.22.1.3.50</td>
<td>printcap</td>
<td>char</td>
</tr>
</tbody>
</table>

2. Execute the command:

```
cdscp remove object /.:mih/deskprinter printcap=ps
```
2. Enter the following command to remove the value \texttt{ps} from the attribute \texttt{printcap}:

\texttt{cdscp> remove object /.:mlh/deskprinter printcap = ps}

\textbf{Related Information}

Commands: \texttt{add\_object(8cds)}, \texttt{list\_object(8cds)}, \texttt{set\_object(8cds)}, \texttt{show\_object(8cds)}.

set cdscp confidence

Purpose

Sets the confidence level of clerk calls

Synopsis

cdscp set cdscp confidence = value

Arguments

value Specifies one of the following confidence levels: low, medium, or high. A low confidence level means the clerk obtains information from caches or the most convenient server. A medium level means the clerk obtains information directly from a server. A high level means the clerk obtains information only at master replicas. The initial value is medium.

Description

The set cdscp confidence command sets the confidence level of clerk calls issued as a result of Cell Directory Service (CDS) control program commands. You must use this command within cdscp. Exiting from cdscp removes the confidence level setting. You must reset the confidence level each time you enter cdscp.

Notes

This command might be replaced in future DCE releases by the dcecp command, and might no longer be supported at that time.

Examples

The following command sets the confidence level of clerk calls to high:

cdscp> set cdscp confidence = high

Related Information

Commands: show_cdscp_confidence(8cds).
set cdscp preferred clearinghouse

Purpose

Specifies a preferred clearinghouse to use for satisfying read requests

Synopsis

`cdscp set cdscp preferred clearinghouse [clearinghouse-name]`

Arguments

`clearinghouse-name`

The full name of the preferred clearinghouse. If you omit this argument, the command causes CDS to revert to the default, which is to use any clearinghouse.

Description

The `set cdscp preferred clearinghouse` command specifies a preferred clearinghouse to use for satisfying read requests that result from Cell Directory Service (CDS) control program commands. You cannot specify a preferred clearinghouse for making modifications, because these requests always use the master replica. You must use this command within `cdscp`. Exiting from `cdscp` removes the preferred clearinghouse setting. You must reset the preferred clearinghouse each time you enter `cdscp`.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command specifies `./:/Paris_CH` as the preferred clearinghouse:

```
cdscp> set cdscp preferred clearinghouse ./:/Paris_CH
```

Related Information

Commands: `show_cdscp_preferred_clearinghouse(8cds)`.
set directory

Purpose

Changes the value of a modifiable, single-valued attribute of a directory

Synopsis

cdscp set directory directory-name attribute-name = attribute-value

Arguments

directory-name
The full name of the directory.

attribute-name
The name of a particular attribute. Specify only one attribute at a time. See the cds_attributes file for the list of attributes and corresponding data types that your application uses.

attribute-value
The value of a particular attribute. The value of an application-defined attribute is dependent on the type of attribute.

Description

The set directory command changes the value of a modifiable, single-valued attribute of a directory. If the attribute does not exist, this command creates it. Usually, this task is performed through the client application. See the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide for more information about attributes. You can specify an application-defined attribute or the following attributes:

CDS_Convergence = value
Specifies the degree of consistency among replicas. By default, every directory inherits the convergence of its parent at creation time. The default setting on the root directory is medium. You can define one of the following for value:

low The Cell Directory Service (CDS) does not immediately propagate any updates. The next skulk distributes all updates that occurred since the previous skulk. Skulks occur at least once every 24 hours.

medium CDS attempts to immediately propagate an update to all replicas. If the attempt fails, the software lets the next scheduled skulk make the replicas consistent. Skulks occur at least once every 12 hours.

high CDS attempts to immediately propagate an update to all replicas. If that attempt fails (for example, if one of the replicas is unavailable), a skulk is scheduled for within one hour. Background skulks occur at least once every 12 hours. Use this setting temporarily and briefly because it uses extensive system resources.

CDS_UpgradeTo = v. n
Controls the upgrading of a directory from one version of CDS to another. By modifying this attribute, you can initiate the upgrading of a directory to a
set directory(8cds)

higher version of CDS. Specify the value as v. n, where v indicates the
major version number and n specifies the minor version number. There is
no default.

Privileges Required

You must have w (write) permission to the directory.

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might
not be provided in future releases of DCE.

Examples

The following sets a low convergence value on the /./mfg directory:

cdscp> set directory /./mfg CDS_Convergence = low

Related Information

Commands: create_directory(8cds), delete_directory(8cds),
list_directory(8cds), remove_directory(8cds), set_directory_to_skulk(8cds),
show_directory(8cds).

set directory to new epoch

Purpose

Reconstructs a directory’s replica set

Synopsis

```
cdscp set directory directory-name to new epoch master clearinghouse-name [readonly clearinghouse-name] [exclude clearinghouse-name]
```

Arguments

- `directory-name`
  The full name of the directory.
- `master clearinghouse-name ...
  The full name of the clearinghouse in which an individual replica is located.
  The first `clearinghouse-name` specifies where the master replica is stored.
- `readonly clearinghouse-name ...
  Designates the replicas in the specified clearinghouses as read-only.
- `exclude clearinghouse-name ...
  Excludes the replicas in the specified clearinghouses.

Description

The `set directory to new epoch` command reconstructs a directory’s replica set, allowing you to designate a new master replica, designate a replica as read-only, or exclude a replica. You must list each existing replica and indicate whether an existing replica needs to be included in or excluded from the new replica set. You can include or exclude more than one replica. You can specify multiple clearinghouse names, separated by spaces.

When you set a new epoch on a directory, you must disable the clearinghouse containing the replica that is being excluded. To do this, use the `disable server` command (if the server has more than one clearinghouse, all its clearinghouses will be disabled). Note that all clearinghouses that are not excluded must be enabled and available before you issue the `disable server` command.

Privileges Required

You must have A (Admin) permission to the directory, and the server principal needs r (read), w (write), and A (Admin) permissions to the directory. When designating a new master replica, you also need w (write) permission to the clearinghouse that stores the new master replica, and the server principal needs w (write) permission to each clearinghouse where the replica type is changed to read-only.

Notes

This command might be replaced in future DCE releases by the `dcecp` command, and might no longer be supported at that time.
set directory to new epoch(8cds)

Examples

The following command sets a new epoch for the directory ./mfg. The master replica is in the clearinghouse ./Paris1_CH, and read-only replicas are in the clearinghouses ./Chicago1_CH, ./Seattle_CH, and ./NY1_CH. The new replica set excludes the replica in the clearinghouse ./NY1_CH.

cdscp> set directory ./mfg to new epoch master ./Paris1_CH \readonly ./Chicago1_CH./Seattle_CH exclude ./NY1_CH

Related Information

Commands: set_directory_to_skulk(8cds), show_directory(8cds), show_replica(8cds).
set directory to skulk

Purpose

Starts the skulk of a directory immediately

Synopsis

cdscp set directory directory-name to skulk

Arguments

directory-name
The full name of the directory.

Description

The set directory to skulk command starts the skulk of a directory immediately. The Cell Directory Service (CDS) control program prompt cdscp> does not return until the skulk is complete. The amount of time for the skulk to complete is dependent on the location, number, and availability of replicas of the directory.

Privileges Required

You must have A (Admin), w (write), i (insert), or d (delete) permission to the directory. The server principal needs A (Admin), r (read), and w (write) permission to the directory.

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

The following command initiates a skulk on the ./admin directory:

cdscp set directory ./admin to skulk

Related Information

Commands: add_directory(8cds), create_directory(8cds), delete_directory(8cds), list_directory(8cds), remove_directory(8cds), set_directory_to_new_epoch(8cds), show_directory(8cds).
set link

Purpose

Changes the value of a modifiable, single-valued attribute of a soft link

Synopsis

```
cdscp set link link-name attribute-name = attribute-value
```

Arguments

- `link-name`
  The full name of the soft link.
- `attribute-name`
  The name of the attribute to be modified. Specify only one attribute at a time. See Description for valid attribute names.
- `attribute-value`
  The value of a particular attribute.

Description

The `set link` command changes the value of a modifiable, single-valued attribute of a soft link. The following are valid attributes:

- **CDS_LinkTarget = fullname**
  Specifies the full name of the directory, object entry, or other soft link to which the soft link points.

- **CDS_LinkTimeout = (expiration-time extension-time)**
  Specifies a timeout value after which the soft link is either checked or deleted. The timeout value contains both an expiration time and an extension time. If a soft link expires and its target entry is deleted, the soft link is deleted. If the soft link still points to an existing entry, its life is extended by the expiration time. Specify `expiration-time` in the following format:

  `yyyy-mm-dd hh:mm:ss`

  Specify `extension-time` in the following format:

  `ddd hh:mm:ss`

Privileges Required

You must have `w` (write) permission to the soft link.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.
set link(8cds)

Examples

The following command redirects a soft link named "/.:admin/work_disk" from its current destination name, "/.:admin/work_disk01", to a new destination name, "/.:admin/work_disk03":

```
set link ./.:admin/work_disk CDS_LinkTarget = ./.:admin/work_disk03
```

Related Information

Commands: create_link(8cds), delete_link(8cds), list_link(8cds), show_link(8cds).
set object

Purpose

Changes the value of a modifiable, single-valued attribute of an object entry

Synopsis

```bash
cdscp set object object-name attribute-name = attribute-value
```

Arguments

- `object-name`
  The full name of the object entry.
- `attribute-name`
  The name of the attribute to be modified. Specify only one attribute at a time. See the `cds_attributes` file for the list of attributes and corresponding data types that your application uses.
- `attribute-value`
  The value of a particular attribute. The value of an application-defined attribute is dependent on the type of attribute.

Description

The `set object` command changes the value of a modifiable, single-valued attribute of an object entry. If the attribute does not exist, this command creates it. Usually, this task is performed through the client application. See the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide* for more information about attributes.

Privileges Required

You must have `w` (write) permission to the object entry.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

To change the value of the `sales_record` attribute to `region2` of an object entry named `./Q1_records`, follow these steps:

1. Read the `cds_attributes` file to make sure that the attribute `sales_record` is listed, as shown in the following display:

```
+-------+----------+----------+
| OID   | LABEL    | SYNTAX   |
| 1.3.22.1.3.66 | sales_record | char    |
```

2. Enter the following command to assign the value `region2` to the attribute `sales_record` of an object entry named `./Q1_records`:

```
cdscp> set object ./Q1_records sales_record = region2
```
set object(8cds)

Related Information

Commands: add_object(8cds), create_object(8cds), delete_object(8cds), list_object(8cds), remove_object(8cds), show_object(8cds).

show cached clearinghouse

Purpose

Displays current information about the specified cached clearinghouse

Synopsis

```
cdscp show cached clearinghouse clearinghouse-name
```

Arguments

```
clearinghouse-name
```
A specific clearinghouse name. The name can contain wildcard characters.

Description

The `show cached clearinghouse` command displays all the names and values of the attributes in the specified cached clearinghouse. The following are valid attributes:

**Creation Time**
Specifies the time at which this clearinghouse was added to the cache.

**Miscellaneous Operations**
Specifies the number of operations other than read and write (that is, skulks, new epochs, and so on) performed by this clerk on the cached clearinghouse.

**Read Operations**
Specifies the number of lookup operations of any sort performed by the clerk on the cached clearinghouse.

**Towers**
Specifies the protocol sequence and Internet address of the server that maintains the cached clearinghouse.

**Write Operations**
Specifies the number of write operations performed by this clerk on the cached clearinghouse.

**Privileges Required**

You must have r (read) permission to the clerk.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command displays all attributes of the cached clearinghouse `/./Paris2_CH`:

```
cdscp> show cached clearinghouse /./Paris2_CH
SHOW
```
show cached clearinghouse(8cds)

    CACHE CLEARINGHOUSE  /.../abc.com/Paris2_CH
    Creation Time = 1991-10-01-17:03:32.32
    Read Operations = 412
    Towers = ndag_ip_udpi:129.35.69.7[
    Write Operations = 618
    Miscellaneous Operations = 278
    Rank = 5000

Related Information

Commands: list_clearinghouse(8cds).
show cached server

Purpose

Displays address information of a server in the local clerk's cache

Synopsis

show cached server name

Arguments

name A simple name for the cached server. The name can contain wildcard characters.

Description

The show cached server command displays address information of a server in the local clerk's cache. The following list describes the valid attributes:

Name The directory cell name.

Towers The protocol sequence and network address of the server node.

Privileges Required

You must have r (read) permission to the clerk.

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

The following command displays all attributes of the cached server emv:

cdscp> show cached server emv*

SHOW

SHOW

CACHE NAMENAMESERVER emv_udp
Name = /.../emv.abc.com
Tower = ncadg_ip_udp:14.20.14.32
Tower = ncacn_ip_tcp:14.20.14.32

SHOW

CACHE NAMENAMESERVER emv_ip
Name = /.../emv.abc.com
Tower = ncadg_ip_udp:14.20.14.32
Tower = ncacn_ip_tcp:14.20.14.32

Related Information

Commands: clear_cached_server(8cds), define_cached_server(8cds).
show cdscp confidence

Purpose

Displays current confidence level of clerk calls

Synopsis

cdscp show cdscp confidence

Description

The show cdscp confidence command displays the current confidence level of clerk calls. A low confidence level means the clerk obtains information from caches or the most convenient server. A medium level means the clerk obtains information directly from a server. A high level means the clerk obtains information only at master replicas.

You must use this command within the Cell Directory Service (CDS) control program. Exiting from cdscp removes the confidence level setting. You must reset the confidence level each time you enter cdscp.

Notes

This command might be replaced in future DCE releases by the dcecp command, and might no longer be supported at that time.

Examples

The following command displays the current confidence level of clerk calls:

cdscp> show cdscp confidence
Confidence used is medium

Related Information

Commands: set_cdscp_confidence(8cds).
show cdscp preferred clearinghouse

Purpose

Displays the preferred clearinghouse for satisfying read requests.

**Note:** This command will show only the preferred clearinghouse that was set with the `set preferred clearinghouse` command. It does not read the `cds_serv_pref` file.

Synopsis

```
cdscp show cdscp preferred clearinghouse
```

Description

The `show cdscp preferred clearinghouse` command displays the preferred clearinghouse for satisfying read requests that result from Cell Directory Service (CDS) control program commands. You can only read attribute values for entries stored in the specified clearinghouse.

You must use this command within `cdscp`. Exiting from `cdscp` removes the preferred clearinghouse setting. You must reset the preferred clearinghouse each time you enter `cdscp`.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command displays the current clearinghouse:

```
cdscp> show cdscp preferred clearinghouse
read attribute values from clearinghouse /.../abc.com/Paris_CH
```

Related Information

Commands: `set_cdscp_preferred_clearinghouse(8cds)`. 
show cell

Purpose

Displays the information you need to create a cell entry in either DNS or GDS.

Synopsis

cdscp show cell cell-name [as type]

Arguments

cell-name
  The global name of the cell.

as type
  The global namespace in which you want to define the cell. Specify either
dns or gds. The default is gds.

Description

The show cell command displays the information you need to create a cell entry in
either the Domain Name System (DNS) or the Global Directory Service (GDS).
DCE does not support cells registered simultaneously in GDS and DNS. If you want
to define a cell in DNS, you can use this command to produce a preformatted set of
resource records. You can then edit the appropriate DNS data file and copy the
output directly into the file. In GDS, cell information is contained in two attributes:
CDS-Cell and CDS-Replica. If you want to define a cell in GDS, you can use this
command to obtain the data you need to supply when creating the CDS-Cell and
CDS-Replica attributes. For details, see the IBM DCE Version 3.1 for AIX and
Solaris: Administration Guide.

Note: GDS is not provided in the IBM DCE 3.1 product, but GDS provided by IBM
DCE 1.0.3 or other products can be used.

Privileges Required

You must have r (read) permission to the cell root directory.

Notes

This command might be replaced in future DCE releases by the dcecp command,
and might no longer be supported at that time.

Examples

The following command displays the GDS-formatted output in the local cell:

cdscp> show cell /.../abc.com as gds
SHOW
  CELL /.../abc.com
  Namespace Uuid = 2d2d50ad-8b1a-11ba-8983-08002b0f79aa
  Clearinghouse Uuid = 2ab024a8-8b1a-11ba-8983-08002b0f79aa
  Clearinghouse Name = /.../abc.com/NY_CH
  Replica Type = Master

Note: GDS is not provided in the IBM DCE 3.1 product, but GDS provided by IBM
DCE 1.0.3 or other products can be used.

Privileges Required

You must have r (read) permission to the cell root directory.

Notes

This command might be replaced in future DCE releases by the dcecp command,
and might no longer be supported at that time.

Examples

The following command displays the GDS-formatted output in the local cell:
Related Information


show cell(8cds)

```
Tower 1  = ncadg_ip_udp:16.18.17.33
Tower 2  = ncacn_ip_tcp:16.18.17.33

Namespace Uuid = 2d2d50ad-8b1a-11ba-8983-08002b0f79aa
Clearinghouse Uuid = 49757f28-8b1a-11ba-8983-08002b0f79aa
Clearinghouse Name = /.../abc.com/Boston_CH
Replica Type = Readonly
Tower 1  = ncadg_ip_udp:16.18.17.33
Tower 2  = ncacn_ip_tcp:16.18.17.33
```
show child

Purpose

Displays attribute information about the specified child pointer

Synopsis

cdscp show child child-name [attribute-name] [with attribute-name = attribute-value]

Arguments

child-name
The full name of a specific child pointer. The last simple name can contain wildcard characters.

with attribute-name
The name of a particular attribute; see Description for valid attribute names.

attribute-value
The value of a particular attribute.

Description

The show child command displays the names and values of the attributes specified in attribute-name. You can use a combination of attributes in a single command. Use a space to separate multiple attributes. You can use a with attribute-name = attribute-value clause to limit output only to child pointers whose attributes have values equal to the specified values. Spaces must precede and follow the = (equal sign).

If you do not supply any attributes, the command displays all attributes and their values. The following is a description of child pointer attributes:

**CDS_CTS**
Specifies the creation timestamp (CTS) of the specified child pointer.

**CDS_ObjectUUID**
Specifies the unique identifier of the directory to which the child pointer refers.

**CDS_Replicas**
Specifies the address, Unique Universal Identifier (UUID), and name of a set of clearinghouses where a copy of the child directory referenced by the child pointer is located. This attribute also specifies whether the directory in a particular clearinghouse is a master or read-only replica.

**CDS_UTS**
Specifies the timestamp of the most recent update to an attribute of the child pointer.

Privileges Required

You must have r (read) permission to the child pointer. If you specify a wildcard child name, you also need read permission to the parent directory.
show child(8cds)

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

The following command displays all of the attributes and values of the child directory to which the child pointer ./:.admin refers:

```
cdscp> show child ./:.admin
SHOW
  CHILD  /.../abc.com/admin
  CDS_CTS = 1991-10-15-19:55:52.000000003/08-00-2b-1c-8f-1f
  CDS_UTS = 1991-10-15-19:55:52.000000006/08-00-2b-1c-8f-1f
  CDS_ObjectUUID = 6b5362e8-8b1c-11ca-8981-08002b0f79aa
  CDS_Replicas = :
  Clearinghouse's UUID = 2ab024a8-8b1a-11ca-8981-08002b0f79aa
    Tower = ncadg_ip_udp:16.18.16.32
    Tower = ncacn_ip_tcp:16.18.16.32
  Replica type = master
  Clearinghouse's Name = /.../abc.com/Boston_CH
```

Related Information

Commands: `create_child(8cds)`, `delete_child(8cds)`, `list_child(8cds)`.
show clearinghouse

Purpose

Displays attribute information about the specified clearinghouse.

Synopsis

```
cdscp show clearinghouse clearinghouse-name [attribute-name] [with attribute-name = attribute-value]
```

Arguments

- `clearinghouse-name`
  - The full name of a specific clearinghouse. The last simple name can contain wildcard characters.
- `with attribute-name`
  - The name of a particular attribute; see Description for valid attribute names.
- `attribute-value`
  - The value of a particular attribute.

Description

The `show clearinghouse` command displays the names and values of the attributes specified in `attribute-name`. You can use a combination of attributes in any sequence in a single command. Use a space to separate multiple attributes. You can use a `with attribute-name = attribute-value` clause to limit output only to clearinghouses whose attributes have values equal to the specified values. Spaces must precede and follow the `=` (equal sign).

If you do not supply any attributes, the command displays all attributes and their values. The following list describes the clearinghouse attributes:

- **CDS_AllUpTo**
  - Indicates the date and time the clearinghouse object has been updated to reflect the `CDS_CHDirectories` attribute.

- **CDS_CHDirectories**
  - Specifies the full name and Universal Unique Identifier (UUID) of every directory that has a replica in this clearinghouse.

- **CDS_CHLastAddress**
  - Specifies the current reported network address of the clearinghouse.

- **CDS_CHName**
  - Specifies the full name of the clearinghouse.

- **CDS_CHState**
  - Specifies the state of the clearinghouse. The state `on` indicates the clearinghouse is running and available.

- **CDS_NSCellname**
  - Specifies the name of the cell in which the clearinghouse resides.

- **CDS_CTS**
  - Specifies the creation timestamp (CTS) of the clearinghouse.
show clearinghouse(8cds)

CDS_DirectoryVersion
Specifies the directory version for new directories that are created in the clearinghouse.

CDS_ObjectUUID
Specifies the unique identifier of the clearinghouse.

CDS_ReplicaVersion
Specifies the current version of the replica in which the directory was created.

CDS_UTS
Specifies the timestamp of the most recent update to an attribute of the clearinghouse.

The following counters and their values are displayed only when you use this command to display all attributes and their values:

Data Corruption Count
Specifies the number of times that the data corruption event was generated.

Enables
Specifies the number of times that the clearinghouse was enabled since it was last started.

Read Accesses
Specifies the number of read operations directed to this clearinghouse.

References Returned
Specifies the number of requests directed to this clearinghouse that resulted in the return of a partial answer instead of satisfying the client’s request.

Skulk Failures
Specifies the number of times that a skulk of a directory, initiated from this clearinghouse, failed to complete — usually because one of the replicas in the replica set was unreachable.

Entry Missing Count
Specifies the number of times the clearinghouse entry missing event was generated.

Root Not Reachable Count
Specifies the number of times the root lost event was generated.

Upgrades Failed Counts
Specifies the number of times that upgrades failed.

Write Accesses
Specifies the number of write operations directed to this clearinghouse.

Disables
Specifies the number of times that the clearinghouse was disabled since it was last started.

Privileges Required
You must have r (read) permission to the clearinghouse. If you specify a wildcard clearinghouse name, you also need r (read) permission to the cell root directory.
show clearinghouse(8cds)

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

The following command displays the current values of the CDS_UTS and CDS_ObjectUUID attributes associated with the /./Chicago1_CH clearinghouse:

cdscp> show clearinghouse /./Chicago1_CH CDS_UTS CDS_ObjectUUID
SHOW
CLEARINGHOUSE /.../abc.com/Chicago1_CH
AT 1991-10-21-13:12:30
CDS_UTS = 1991-10-21-13:04.000000009/08-00-2b-1c-8f-1f
CDS_ObjectUUID = 3706d70c-8b05-11ca-9002-08002b1c8f1f

Related Information

Commands: clear_clearinghouse(8cds), create_clearinghouse(8cds), delete_clearinghouse(8cds), list_clearinghouse(8cds), set_cdscp_preferred_clearinghouse(8cds), show_cdscp_preferred_clearinghouse(8cds).
show clerk

Purpose

Displays attribute information about the CDS clerk on the local system

Synopsis

cdscp show clerk

Description

The show clerk command displays all the names and values of the clerk attributes on the local system. The clerk must be enabled when you use this command. The following are valid attributes:

Authentication Failures
Specifies the number of times a requesting principal failed authentication procedures.

Cache Bypasses
Specifies the number of requests to read attributes for which the clerk was specifically directed by the requesting application to bypass its own cache. Instead, a server is contacted to get the requested information. This attribute does not account for requests that the clerk is unable to satisfy from the cache or for requests to look up names or enumerate the contents of directories.

Cache Hits
Specifies the total number of read requests directed to this clerk that were satisfied entirely by the information contained in its own cache. This attribute accounts only for requests to read attribute values and does not include requests to look up names or enumerate the contents of directories.

Creation Time
Specifies the time when this entity was created.

Miscellaneous Operations
Specifies the number of operations other than read and write (that is, skulks, enumerating contents of directories, and so on) performed by this clerk.

Read Operations
Specifies the number of lookup operations performed by this clerk. This attribute accounts only for requests to read attributes and does not include requests to look up names or enumerate the contents of directories.

Write Operations
Specifies how many requests to modify data were processed by this clerk.

Privileges Required

You must have r (read) permission to the clerk.

Notes

This command might be replaced in future DCE releases by the dcecp command, and might no longer be supported at that time.
show clerk(8cds)

Examples

The following command displays the attributes of the clerk on the local system:

cdscp> show clerk

SHOW
CLERK
Authentication failures = 0
Read Operations = 1068
Cache Hits = 137
Cache bypasses = 433
Write operations = 1250
Miscellaneous operations = 590

Related Information

Commands: disable_clerk(8cds).
show directory

Purpose

Displays attribute information about the specified directory

Synopsis

```
cdscp show directory directory-name [attribute-name] [with attribute-name = attribute-value]
```

Arguments

- **directory-name**
  The full name of a specific directory. The last simple name can contain wildcard characters.

- **with attribute-name**
  The name of a particular attribute; see Description for valid attribute names.

- **attribute-value**
  The value of a particular attribute.

Description

The `show directory` command displays the names and values of the attributes specified in `attribute-name`. You can use a combination of attributes in any sequence in a single command. Use a space to separate multiple attributes. You can use a `with attribute-name = attribute-value` clause to limit output only to directories whose attributes have values equal to the specified values. Spaces must precede and follow the `=` (equal sign). If you do not supply any attributes, the command displays all attributes and their values. In addition to the following directory attributes, application-specific attributes can exist for a directory:

- **CDS_AllUpTo**
  Indicates the date and time of the last successful skulk on the directory. All replicas of the directory are guaranteed to receive all updates whose timestamps are less than the value of this attribute.

- **CDS_Convergence**
  Specifies the degree of consistency among replicas. This attribute’s value is defined as one of the following:
  - **low**
    CDS does not immediately propagate an update. The next skulk distributes all updates that occurred since the previous skulk. Skulks occur at least once every 24 hours.
  - **medium**
    CDS attempts to immediately propagate an update to all replicas. If the attempt fails, the next scheduled skulk makes the replicas consistent. Skulks occur at least once every 12 hours.
  - **high**
    CDS attempts to immediately propagate an update to all replicas. If the attempt fails (for example, if one of the replicas is unavailable), a skulk is scheduled for within one hour. Skulks usually occur at least once every 12 hours. Use this setting temporarily and briefly, because it uses extensive system resources.
show directory(8cds)

By default, every directory inherits the convergence setting of its parent at creation time. The default setting on the root directory is medium.

**CDS_CTS**
Specifies the creation timestamp (CTS) of the CDS directory.

**CDS_DirectoryVersion**
Specifies the minimum of all the values of the **CDS_ReplicaVersion** attribute on the directory replicas.

**CDS_Epoch**
A UUID that identifies a particular incarnation of the directory.

**CDS_LastSkulk**
Records the timestamp of the last skulk performed on this directory.

**CDS_LastUpdate**
Records the timestamp of the most recent change to any attribute of a directory replica, or any change to an entry in the replica.

**CDS_ObjectUUID**
Specifies the unique identifier of the directory.

**CDS_ParentPointer**
Contains a pointer to this directory’s parent in the namespace.

**CDS_Replicas**
Specifies the address, UUID, and name of every clearinghouse where a copy of this directory is located. This attribute also specifies whether the replica in a particular clearinghouse is a master or read-only replica.

**CDS_ReplicaState**
Specifies whether a directory replica can be accessed.

**CDS_ReplicaType**
Indicates whether a directory replica is a master or read-only replica.

**CDS_ReplicaVersion**
Specifies the version of a replica of the directory.

**CDS_RingPointer**
Specifies the UUID of a clearinghouse containing another replica of this directory. This attribute is written by the system and is read-only to users. It will appear on older directories, but not on DCE 1.1 directories.

**CDS_UpgradeTo**
Controls the upgrading of a directory from one version of CDS to another. By modifying this attribute, you can initiate the upgrading of a directory to a new version of CDS.

**CDS_UTS**
Specifies the timestamp of the most recent update to an attribute of the directory.

**RPC_ClassVersion**
Specifies the RPC runtime software version that can be used to import on the directory.

**Note:** Since replicas of the directory can exist in multiple clearinghouses, some of the values displayed will be different depending on which clearinghouse the control program chooses to read the information from.
show directory(8cds)

For example, the value displayed for the CDS_Replica Type attribute reflects the replica type on the clearinghouse from which the information is being read.

Unless a cdscp preferred clearinghouse has been set (see the set cdscp preferred clearinghouse command), the control program randomly chooses the clearinghouse it uses to retrieve the information.

Privileges Required

You must have r (read) permission to the directory. If you specify a wildcard directory name, you also need r (read) permission to the directory’s parent directory.

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

The following command displays the current values of all the attributes associated with the ./admin directory:

cdscp> show directory ./admin

SHOW
DIRECTORY  ../../../abc.com/admin

RPC_ClassVersion = 0100
CDS_CTS = 1991-10-15-13:09:47.000000003/08-00-2b-1c-8f-1f
CDS_UTS = 1991-10-17-08:59:50.000000006/08-00-2b-1c-8f-1f
CDS_ObjectUUID = ba700c98-8b1a-11ca-8981-08002b0f79aa
CDS_Replacs = :
Clearinghouse's UUID = 2ab024a8-8b1a-11ca-8981-08002b0f79aa
Tower = ncagd_ip_udp:16.20.16.32
Tower = ncacn_ip_tcp:16.20.16.32
Replica type = master
Clearinghouse's Name = ../../../abc.com/Paris_CH
CDS_AllUpTo = 1991-10-17-08:51:18.000000003/08-00-2b-1c-8f-1f
CDS_Convergence = medium
CDS_ParentPointer = :
Parent's UUID = b773525c-8b1a-11ca-8981-08002b0f79aa
Timeout = :
Expiration = 1991-10-16-19:43:50.516
Extension = +1-00:00:00.000
CDS_DirectoryVersion = 3.0
CDS_ReplicaState = on
CDS_ReplicaType = master
CDS_LastSkulk = 1991-10-17-08:51:18.000000003/08-00-2b-1c-8f-1f
CDS_LastUpdate = 1991-10-21-13:04:02.000000044/08-00-2b-1c-8f-1f
CDS_RingPointer = 2ab024a8-8b1a-11ca-8981-08002b0f79aa
CDS_Epoch = bd8b2c50-8b1a-11ca-8981-08002b0f79aa
CDS_ReplicaVersion = 3.0

Related Information

Commands: add_directory(8cds), create_directory(8cds), delete_directory(8cds), list_directory(8cds), remove_directory(8cds), set_directory(8cds).
show link

Purpose

Displays attribute information about the specified soft link

Synopsis

```
cdscp show link link-name [attribute-name] [with attribute-name = attribute-value]
```

Arguments

- **link-name**
  The full name of a specific soft link. The last simple name can contain wildcard characters.

- **with attribute-name**
  The name of a particular attribute; see **Description** for valid attribute names.

- **attribute-value**
  The value of a particular attribute.

Description

The **show link** command displays the names and values of the attributes specified in **attribute-name**. You can use a combination of attributes in any sequence in a single command. Use a space to separate multiple attributes. You can use a **with attribute-name = attribute-value** clause to limit output only to soft links whose attributes have values equal to the specified values. Spaces must precede and follow the **=** (equal sign). If you do not supply any attributes, the command displays all attributes and their values. The following is a description of soft link attributes:

- **CDS_CTS**
  Specifies the creation timestamp (CTS) of this soft link.

- **CDS_LinkTarget**
  Specifies the full name of the directory, object entry, or other soft link to which the soft link points.

- **CDS_LinkTimeout**
  Specifies a timeout value after which the soft link is either checked or deleted.

- **CDS_UTS**
  Specifies the timestamp of the most recent update to an attribute of the soft link.

Privileges Required

You must have **r** (read) permission to the soft link. If you specify a wildcard soft link name, you also need **read** permission to the directory that stores the soft link.

Notes

This command was replaced at DCE Version 1.1 by the **dcecp** command and might not be provided in future releases of DCE.
Examples

The following command displays the current values of all the attributes associated with the soft link "/.:sales/region1".

cdscp> show link /.:sales/region1

SHOW

SOFTLINK /.../abc.com/sales/region1
CDS_CTS = 1991-10-15-19:54:35.00000000000000003/08-00-2b-1c-8f-1f
CDS_UTS = 1991-10-15-19:54:35.00000000006/08-00-2b-1c-8f-1f
CDS_LinkTarget = /.../abc.com/sales/service

SHOW

SOFTLINK /.../abc.com/sales/region1
CDS_CTS = 1991-10-15-19:54:36.0000000000000000077/08-00-2b-1c-8f-1f
CDS_UTS = 1991-10-15-19:54:36.00000000000009/08-00-2b-1c-8f-1f
CDS_LinkTarget = /.../abc.com/sales/software
CDS_LinkTimeout = :
Expiration = 1991-10-15-00:00:00.0
Extension = +1-00:00:00.000

Related Information

Commands: create_link(8cds), delete_link(8cds), list_link(8cds), remove_link(8cds), set_link(8cds).
show object(8cds)

show object

Purpose

Displays attribute information about the specified object entry

Synopsis

cdscp object object-name [attribute-name] [with attribute-name = attribute-value]

Arguments

object-name
    The full name of a specific object entry. The last simple name can contain
    wildcard characters.

with attribute-name
    The name of a particular attribute; see Description for valid attribute
    names.

attribute-value
    The value of a particular attribute.

Description

The show object command displays the names and values of the attributes
specified in attribute-name. You can use a combination of attributes in a single
command. Use a space to separate multiple attributes. You can use a with
attribute-name = attribute-value clause to limit output only to object entries whose
attributes have values equal to the specified values. Spaces must precede and
follow the = (equal sign). If you do not supply any attributes, the command displays
all attributes and their values. In addition to the following attributes, any
application-defined attributes that might exist will be included in the output of this
command. The following is a description of object entry attributes:

CDS_Class
    Specifies the class to which an object belongs.

CDS_ClassVersion
    Contains the version number of the object’s class. This allows applications
to build in compatibility with entries created by earlier versions.

CDS_CTS
    Specifies the creation timestamp (CTS) of this object entry.

CDS_ObjectUUID
    Specifies a unique identifier for the object being referenced.

CDS_UTS
    Specifies the timestamp of the most recent update to an attribute of the
    object entry.

When RPC object UUIDs are exported into a CDS object, they are stored by RPC
into CDS as binary data. Because the UUID is binary, CDS does not display these
UUIDs in standard UUID string representation when you view them using the CDS
Control Program (cdscp). The following is an example:
Notice that the RPC_ObjectUUIDs are in binary instead of standard UUID representation.

The RPC Control Program (rppcp) recognizes that this attribute is a UUID and displays it in standard UUID representation. An example follows:

```
rppcp> show mapping -o faf2e540-58b8-11ca-a040a-08802b12a70d
mappings:
<object> faf2e540-58b8-11ca-a040a-08802b12a70d
<string binding> ncadg_ip_tcp:129.35.69.7[4960]
<annotation> cdspd [16232]
</object>
<object> faf2e540-58b8-11ca-a040a-08802b12a70d
<string binding> ncadg_ip_udp:129.35.69.7[4824]
<annotation> cdspd [16232]
</object>
```

Privileges Required

You must have r (read) permission to the object entry. If you specify a wildcard object entry name, you also need r (read) permission to the directory that stores the object entry.

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

The following command lists all the attributes of the object entry /./sales/east/floor1cp, and their values:

```
cdscp> show object /./sales/east/floor1cp
SHOW
OBJECT /./abc.com/sales/floor1cp
CDS_CTS = 1991-10-15-19:53:03.00000003/08-00-2b-1c-8f-1f
CDS_UTS = 1991-10-15-19:53:03.00000006/08-00-2b-1c-8f-1f
```

Chapter 3. Cell Directory Service Commands

537
show object(8cds)

Related Information

Commands: add_object(8cds), create_object(8cds), delete_object(8cds), list_object(8cds), remove_object(8cds), set_object(8cds).
show replica

Purpose

Displays attribute information about the specified replica

Synopsis

`cdscp show replica directory-name clearinghouse clearinghouse-name [attribute-name]`

Arguments

- `directory-name`
  The full name of the directory
- `clearinghouse clearinghouse-name`
  The full name of the clearinghouse
- `attribute-name`
  The name of a particular attribute; see Description for valid attribute names.

Description

The `show replica` command displays the directory-specific attributes as well as the per-replica attributes of the specified directory. If you do not supply any attributes, the command displays all attributes and their values; any application-defined attributes that might exist will be included in the output of this command. You can enter one or more of the following attributes:

**CDS_AllUpTo**
Indicates the date and time of the last successful skulk on the directory. All replicas of the directory are guaranteed to have received all updates whose timestamps are less than the value of this attribute.

**CDS_Convergence**
Specifies the degree of consistency among replicas. This attribute’s value is defined as one of the following:

- **low**
  CDS does not immediately propagate an update. The next skulk distributes all updates that occurred since the previous skulk. Skulks occur at least once every 24 hours.

- **medium**
  CDS attempts to immediately propagate an update to all replicas. If the attempt fails, the next scheduled skulk makes the replicas consistent. Skulks occur at least once every 12 hours.

- **high**
  CDS attempts to immediately propagate an update to all replicas. If the attempt fails (for example, if one of the replicas is unavailable), a skulk is scheduled for within one hour. Skulks usually occur at least once every 12 hours. Use this setting temporarily and briefly, because it uses extensive system resources.

By default, every directory inherits the convergence setting of its parent at creation time. The default setting on the root directory is **medium**.
show replica(8cds)

CDS_CTS
Specifies the creation timestamp (CTS) of the directory of which this replica is a copy.

CDS_DirectoryVersion
Specifies the minimum of all the values of the CDS_ReplicaVersion attribute on the directory replicas.

CDS_Epoch
A Universal Unique Identifier (UUID) that identifies a particular incarnation of the directory.

CDS_LastSkulk
Records the timestamp of the last skulk performed on this particular replica of a directory.

CDS_LastUpdate
Records the timestamp of the last update to any attribute of the replica, or any change to the contents of the replica, including object entries, child pointers, and soft links.

CDS_ObjectUUID
Specifies the unique identifier of the directory of which this replica is a copy.

CDS_ParentPointer
Contains a pointer to this directory’s parent in the namespace.

CDS_Replicas
Specifies the address, UUID, and name of every clearinghouse where a replica of this directory is located. This attribute also specifies whether the replica in a particular clearinghouse is a master or read-only replica.

CDS_ReplicaState
Specifies the internal state of a replica. When you create or delete a replica, it goes through various states.

CDS_ReplicaType
Specifies the replica type of a directory.

CDS_ReplicaVersion
Specifies the replica version of a directory.

CDS_RingPointer
Specifies the UUID of a clearinghouse containing another replica of this directory. This attribute is written by the system and is read-only to users. It will appear on older directories, but not on DCE 1.1 directories.

CDS_UTS
Specifies the timestamp of the most recent update to an attribute of the directory.

RPC_ClassVersion
Specifies the RPC runtime software version that can be used to import on the directory.

Privileges Required
You must have r (read) permission to the directory from which the replica is created.
Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

The following command displays the current values of all the attributes of the replica of the `./.eng` directory in the `./.Chicago2_CH` clearinghouse:

```
cdscp> show replica ././eng clearinghouse ././Chicago2_CH
SHOW
REPlica /.../abc.com/eng
RPC_ClassVersion = 0100
CDS_CTS = 1991-10-15-12:09:47.000000003/08-00-2b-1c-8f-1f
CDS_UTS = 1991-10-17-07:59:50.000000006/08-00-2b-1c-8f-1f
CDS_ObjectUUID = 5816da70-8b1c-11ca-8981-08002b0f79aa
CDS_Resticas = :
Clearinghouse's UUID = 2ab024a8-8b1a-11ca-8981-08002b0f79aa
Tower = ncadg_ip_udp:16.20.16.32
Tower = ncacn_ip_tcp:16.20.16.32
Replica type = master
Clearinghouse's Name = /.../abc.com/Chicago1_CH
CDS_Resticas = :
Clearinghouse's UUID = 4975f28-8b1a-11ca-8981-08002b0f79aa
Tower = ncadg_ip_udp:16.20.16.32
Tower = ncacn_ip_tcp:16.20.16.32
Replica type = readonly
Clearinghouse's Name = /.../abc.com/Chicago2_CH
CDS_AllUpTo = 1991-10-17-07:51:18.000000032/08-00-2b-1c-8f-1f
CDS_Convergence = medium
CDS_ParentPointer = :
Parent's UUID = 560f1ad0-8b1c-11ca-8981-08002b0f79aa
Timeout = :
Extension = +1-00:00:00.000
CDS_DirectoryVersion = 3.0
CDS_ReplicaState = on
CDS_ReplicaType = readonly
CDS_LastSkulk = 1991-10-17-07:51:18.000000032/08-00-2b-1c-8f-1f
CDS_LastUpdate = 1991-10-21-12:04:02.000000044/08-00-2b-1c-8f-1f
CDS_RingPointer = 2ab024a8-8b1a-11ca-8981-08002b0f79aa
CDS_Epoch = 58472144-8b1c-11ca-8981-08002b0f79aa
CDS_ReplicaVersion = 3.0
```
show server

Purpose

Displays attribute information about the server running on the local system

Synopsis

cdscp show server

Description

The show server command displays all the names and values from the attributes named in this entity. The server must be enabled when you use this command. The following are valid attribute names:

Child Update Failures
Specifies the number of times the server was unable to contact all the clearinghouses that store a replica of a particular child directory’s parent directory and apply the child updates that have occurred since the last skulk. This counter is incremented by the Cannot Update Child Pointer event.

Creation Time
Specifies the time when the cdscp process was started.

Crucial Replicas
Specifies the number of times a user attempted (from this server) to remove a replica that is crucial to the connectivity of a directory hierarchy. The server background process prevents users from accidentally disconnecting lower-level directories from higher-level directories. When it detects an attempt to remove a crucial replica, it does not execute the command to do so. This counter is incremented by the Crucial Replica event.

Future Skew Time
Specifies the maximum amount of time that a timestamp on a new or modified entry can vary from local system time at the server system.

Known Clearinghouses
Specifies the clearinghouse or clearinghouses known to the server.

Read Operations
Specifies the number of read operations directed to this Cell Directory Service (CDS) server.

Security Failures
Specifies the number of times a server principal for this server was found to have inadequate permissions to perform a requested operation.

Skulks Completed
Specifies the number of skulks successfully completed by this CDS server.

Skulks Initiated
Specifies the number of skulks initiated by this CDS server.

Times Lookup Paths Broken
Specifies the number of broken connections between clearinghouses on this server and clearinghouses closer to the root. Incoming requests to this server that require a downward lookup in the directory hierarchy still
show server(8cds)

succeed, but requests that require a lookup in directories closer to the root will fail. This counter is incremented by the **Broken Lookup Paths** event.

**Write Operations**
Specifies the number of write operations to this CDS server.

**Privileges Required**
You must have **r (read)** permission to the server.

**Notes**
This command might be replaced in future releases by the **dcecp** command, and might no longer be supported at that time.

**Examples**
The following command displays the current values of all the attributes associated with the server running on the local system:

```
cdscp> show server
SHOW
SERVER
Future Skew Time = 300
Read Operations = 757
Write Operations = 542
Skulks Initiated = 219
Skulks Completed = 219
Times Lookup Paths Broken = 1
Crucial Replicas = 0
Child Update Failures = 1
Security Failures = 0
Known Clearinghouses = /.../abc.com/Boston_CH
                  = /.../abc.com/NY_CH
```

**Related Information**
Commands: disable_server(8cds).
show server(8cds)
dts_intro

Purpose

Introduction to the DCE DTS commands

Description

The DCE Distributed Time Service (DTS) provides the following facilities:

- **dtsd**  The DTS daemon
- **dtscp**  The DTS control program
- **dtsdate**  The DTS local clock setting program

DTS is implemented in the **dtsd** process. Both clerks and servers use the same daemon. The behavior of **dtsd** is determined by **dtscp**.

The DTS control program allows you to synchronize, adjust, and maintain the system clocks in a distributed network. The **dtscp** commands are as follows:

- **advertise**  Configures the DTS server as a global server.
- **change**  Modifies the epoch and sets the local time to a new time.
- **create**  Establishes a DTS entity (a clerk or server).
- **delete**  Causes DTS to exit on the local node.
- **disable**  Suspends a DTS entity.
- **enable**  Starts a DTS entity.
- **exit**  Ends the **dtscp** management session and returns you to the system prompt.
- **help**  Invokes the **dtscp** help service.
- **quit**  Ends the **dtscp** management session and returns you to the system prompt.
- **set**  Modifies characteristics of a DTS entity.
- **show**  Displays characteristics of a DTS entity.
- **synchronize**  Synchronizes the system clock with the time obtained from DTS servers in the network.
- **unadvertise**  Removes the global server entry.
- **update**  Gradually adjusts the system clock to a new time.

For more information on any of the **dtscp** commands, see the appropriate reference page.
The `dtsd` command restarts the DTS daemon (clerk or server process). When the host system is rebooted, this command is automatically executed as part of the overall DCE configuration procedure.

Invocation of `dtsd` leaves it in an idle state. In order for it to assume an identity, it must be created with the `dtscp create` command.

After the DTS entity is created, it is still in a nonfunctioning state. To put it into operation, you must invoke `dtscp enable`, which causes an immediate synchronization to take place.

To bring down a DTS entity, you must first stop it with `dtscp disable` and then delete it with `dtscp delete`.

The `dtsdate` command sets the local clock of a system to be the same as the host `remote_host`, running a `dtsd` server.

**Related Information**

Commands: `advertise(8dts)`, `change(8dts)`, `create(8dts)`, `delete(8dts)`, `disable(8dts)`, `dtscp(8dts)`, `dtsd(8dts)`, `dtsdate(8dts)`, `enable(8dts)`, `exit(8dts)`, `help(8dts)`, `quit(8dts)`, `set(8dts)`, `show(8dts)`, `synchronize(8dts)`, `unadvertise(8dts)`, `update(8dts)`.

advertise

Purpose

Configures the system as a global server

Synopsis

dtscp advertise

Description

The `dtscp advertise` command configures the system as a global server by adding the server's entry to the cell profile. It causes the Distributed Time Service (DTS) to forward the name and attributes of the server to the Cell Directory Service (CDS) by binding the server's protocol tower to the CDS object and adding an entry for the server in the cell profile. Once the server's entry is in the cell profile, it is configured as a global server, and servers outside of the local area network (LAN) can access it.

Privileges Required

You must have `w (write)` permission on the access control list (ACL) associated with the DTS entity in order to execute the command.

Notes

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

dtscp> advertise

Related Information

Commands: `dtscp(8dts)`.
change

Purpose

Alters the epoch number and time on the local node

Synopsis

dtscp change epoch integer [time absolute-time]

Arguments

epoch integer
   Specifies the new epoch number, an integer from 0 to 255. This argument
   is required.

time absolute-time
   Specifies a clock setting for the new epoch. If you do not supply this
   argument and a value, the server uses the current clock time with an
   unspecified inaccuracy and initiates a synchronization. This argument is
   optional.

Description

The dtscp change command sets the time and changes the epoch of the
Distributed Time Service (DTS) server on which it is entered. Use this command to
isolate a server from the rest of the servers in the network before changing the
time.

Permissions Required

You must have w (write) permission on the access control list (ACL) associated
with the DTS entity in order to execute the command.

Notes

This command is valid only for servers. The new epoch number you specify must
be different from the current epoch number.

This command was replaced at DCE Version 1.1 by the dcecp command and might
not be provided in future releases of DCE.

Examples

1. The following command changes the epoch number:

   dtscp> change epoch 1

2. The following command changes the epoch number and time:

   dtscp> change epoch 1 time 1990-11-30-10:58:00.000-05:0010.000

Related Information

Commands: dtscp(8dts).
create(8dts)

create

Purpose

Creates the DCE DTS entity on the specified node

Synopsis

dtscp create type type

Arguments

type type

Specifies the type of DTS entity to be created on the specified node. Specify one of the following for type:

clerk The DTS entity is created as a clerk. (This is the default setting.)

server The DTS entity is created as a server.

Description

The create command creates a time server or time clerk entity on the system on which the command is entered.

After the Distributed Time Service (DTS) entity is created, it is still in a nonfunctioning state. To put it into operation, you must invoke dtscp enable, which causes an immediate synchronization to take place. For more information, see the enable(8dts) reference page.

Privileges Required

You must have w (write) permission on the access control list (ACL) associated with the DTS entity in order to execute the command.

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

dtscp> create type server

Related Information

Commands: dtscp(8dts, enable(8dts)).
delete

Purpose

Deletes the DCE DTS entity

Synopsis

`dtscp delete`

Description

The `dtscp delete` command deletes the DCE Distributed Time Service (DTS) entity from the system on which the command is entered. When `delete` is executed, the DTS daemon process completes execution. To restart the DTS daemon, use the `start.dce` shell command.

Privileges Required

You must have `w` (write) permission on the access control list (ACL) associated with the DTS entity in order to execute the command.

Notes

The DTS entity cannot be deleted until you enter the `disable` command, which causes the status attribute `state` to be set to `off`.

This command was replaced at DCE Version 1.1 by the `dcecp` command and might not be provided in future releases of DCE.

Examples

```
dtscp> delete
```

Related Information

Commands: `disable(8dts)`, `dtscp(8dts)`, `dce_config(8dce)`.
disable(8dts)

disable

Purpose

Stops the DCE DTS entity on the local node

Synopsis

dtscp disable

Description

The disable command turns off the Distributed Time Service (DTS) entity on the system on which the command is entered. When the command is executed, the status attribute state is set to off.

Privileges Required

You must have w (write) permission on the access control list (ACL) associated with the DTS entity in order to execute the command.

Notes

The DTS entity cannot be disabled until it is enabled with the enable command. You must enter the disable command before you can delete the entity with the delete command.

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

dtscp> disable

Related Information

Commands: delete(8dts), dtscp(8dts), enable(8dts).
dtscp

Purpose

Starts the DTS control program

Synopsis

dtscp

Description

Note:

With the exception of the following subcommands, this command was replaced at DCE Version 1.1 by the dcecp command. This command might be fully replaced by the dcecp command in a future release of DCE, and might no longer be supported at that time.

1. exit
2. help
3. quit

This control program has been superseded by dcecp. It is not designed for international use, and might give unexpected or undesirable results when used in non-English environments. If you are working with non-English data, you should use dcecp.

The *(8dts) reference pages describe the commands for the Distributed Time Service (DTS) control program, dtscp. The DTS control program is a command-line interface that enables you to synchronize, adjust, and maintain the system clocks in a distributed network. For a detailed explanation of system clock synchronization and management, see the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide.

The DTS control program commands are as follows:

advertise
   Configures the DTS server as a global server.

change
   Modifies the epoch and sets the local time to a new time.

create
   Establishes a DTS entity (a clerk or server).

delete
   Causes DTS to exit on the local node.

disable
   Suspends a DTS entity.

enable
   Starts a DTS entity.

exit
   Ends the dtscp management session and returns you to the system prompt.

help
   Invokes the dtscp help service.
**dtscp(8dts)**

**quit**  Ends the dtscp management session and returns you to the system prompt.

**set**  Modifies characteristics of a DTS entity.

**show**  Displays characteristics of a DTS entity.

**synchronize**

Synchronizes the system clock with the time obtained from DTS servers in the network.

**unadvertise**

Removes the global server entry.

**update**

Gradually adjusts the system clock to a new time.

For more information on any of the above dtscp commands, see the appropriate reference page.

You can use control program commands from within the control program or from the system prompt. To enter DTS commands from within the control program, first start the control program by entering the dtscp command. For example:

```
dtscp
```

At this prompt you can enter any control program command. For example:

```
dtscp> show current time
```

To leave the control program and return to the system prompt, enter the `exit` command.

To enter DTS commands from the system prompt, interactively or in a command procedure, enter the dtscp command with an internal command of the control program as the first argument. The control program executes the command without displaying the control program prompt. For example, you can enter the synchronize command as follows:

```
dtscp synchronize
```

Some dtscp commands have optional arguments or attributes, and there might also be optional variables for the arguments and attributes. This is shown in the following diagram:

```
dtscp> update time 1990-08-03-05:45:28.000+01:00I00.500
```

<table>
<thead>
<tr>
<th>Command</th>
<th>Argument</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>------------</td>
<td>----------</td>
<td>----------</td>
</tr>
</tbody>
</table>

**Abbreviations**

You can enter as few as three characters for each DTS command or argument; DTS commands and arguments are unique for three characters or more. For example, rather than entering the command `enable set clock true`, you can enter the following abbreviated command:

```
dtscp> ena set clo tru
```
Attributes

The dtscp set and show commands have several attributes—pieces or sets of data—associated with them. The attribute groups are categorized as follows:

Characteristics
Set or show the entity’s operation.

Counters
Show the number of occurrences of an event since the entity was enabled.

Status
Show the current state of the entity. (The DTS entity has four status attributes.)

Global Servers
Show the global servers known by this DTS entity.

Local Servers
Show the local servers known by this DTS entity.

Individual attributes within each of the previously listed groups are described in the set(8dts) and show(8dts) reference pages. The show command also allows you to specify attribute groups.

Time Stamps

All responses to commands contain a timestamp. The following is a typical DTS time display:

1993-03-16-14:29:47.52000-05:001000.003

The timestamp uses the DTS format as explained in the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide—Core Components. In this example, the year is 1993, the day is March 16, and the time is 14 hours, 29 minutes, and 47.52 seconds. A negative Time Differential Factor (TDF) of 5 hours and an inaccuracy of 3 milliseconds is included in the timestamp.

Note:

An inaccuracy value of I----- indicates an infinite inaccuracy. This value appears in time displays before a node’s initial synchronization, or after you enter the change command without specifying an inaccuracy value.

Related Information

Commands: advertise(8dts), change(8dts), create(8dts), delete(8dts), disable(8dts), enable(8dts), exit(8dts), help(8dts), quit(8dts), set(8dts), show(8dts), synchronize(8dts), unadvertise(8dts), update(8dts).

dtbsd(8dts)

dtbsd

Purpose

Restarts the DTS daemon

Synopsis

```
dbsd [-d] [-w serviceability] [-s [-k courier | noncourier] [-g] [-o] [-c]
```

Options

- **-d** Specifies debug mode. The command runs in the foreground.
- **-w serviceability**
  
  See the `svcroute(5dce)` reference page for the full description of the appropriate format for this entry. Only the three-field format is used, as follows:

  ```
  severity: how: where
  ```

  The following is an example:

  ```
  FATAL: TEXTFILE:/dev/console
  ```
- **-s** Runs `dtbsd` as a server. Default is backup, courier, local server
- **-g** Runs `dtbsd` as a global server.
- **-k courier | noncourier**
  
  Runs `dtbsd` as a courier or a noncourier.
- **-g** Runs `dtbsd` as a global server.
- **-o** When enabling as a server, sets the clock immediately. Equivalent to the command `enable set clock true` in `dtscp` or to the command `dcecp dts activate -abruptly`.
- **-c** Runs `dtbsd` as a clerk.

Description

The `dtbsd` command invokes the Distributed Time Service (DTS) daemon (clerk or server process). This command is usually executed as part of the overall DCE startup script, `start.dce`.

You can enter the command manually under the following conditions:

1. If a DTS daemon fails to start automatically upon reboot
2. If you want to restart a daemon that you shut down to perform a backup or do diagnostic work

In normal rebooting, the `start.dce` script automatically provides arguments appropriate to the choice of configuration options.

The command line options shown here can also be provided to `dced` as part of the fixed configuration strings, if `dced` is configured to automatically start `dtbsd`. 
If `dtsd` is started with no options other than `-d` and `-w`, then the server must be started with the `dcecp dts` command. The following configures a local server:

```
  dcecp> dts configure -notglobal
  dcecp> dts activate
```

**Privileges Required**

DTS runs as the host machine principal, which is usually `root`. See the security reference pages for information about principals.

**Notes**

Use `dtsd` interactively only when troubleshooting; otherwise use the `start.dce` script. On some systems the superuser is associated with the machine principal.

**Examples**

To restart the daemon, follow these steps:

1. Log into the system as `superuser` (`root`).
2. Use the `ps` command to make sure that `dced` and `cdsadv` are running. (The DCE daemon, `dced`, provides the endpoint mapping and security services, and `cdsadv` provides CDS.)
3. Enter the following command to restart the `dts` daemon as a clerk:

   ```
   dtsd -c
   ```

   Enter the following command to restart the `dts` daemon as a server:

   ```
   dtsd -s
   ```

   To restart the `dts` daemon as a global server, setting the clock on startup, use the following command:

   ```
   dtsd -s -g -o
   ```

**Related Information**

Commands: `dtscp (8dts)`, `dtsdate (8dts)`, `dcecp (8dce)`.

Files: `svcroute(5dce)`.

The `dtsdate` command sets the local clock of a system to be the same as the host `remote_host`, running a `dtss` server. The purpose of `dtsdate` is to ensure that clock skew is minimized at initial cell configuration or at host instantiation, because it is difficult to start DCE and its components if the skew is too great.

Clocks among all DCE components must be within five minutes of each other, to prevent failure of the Cell Directory Service (CDS) and of security. Some DCE components have even more stringent requirements. For instance, a Distributed File Service (DFS) file server cannot start if its local host differs from other DFS hosts by more than ten seconds.

The `dtsdate` command can be used for adjusting a clock backwards, before DCE is running on a host. Adjusting a clock backwards while DCE is running can cause many difficulties, because security and file system software generally require system time to increase monotonically.

The remote host must be running as a Distributed Time Service (DTS) server. This means that the `dtss` on that system must have registered the DTS management interface, because `dtsdate` uses the management call to get the current time from that host.
For `dtsdate` to be able to set the clock, it must run as a privileged user (root).

**Exit Values**

If the `-q` argument is given, `dtsdate` returns 2 if the remote time and local time differ by more than `nsecs`, 1 if there was an error, and 0 otherwise.

If the `-q` argument is not given, `dtsdate` returns 1 if there was an error, and 0 otherwise.

**Examples**

1. To run `dtsdate` with only the host argument, enter the following command:

   ```
   dtsdate remotehost
   ```

   As a result, `dtsdate` prints out the time on `remotehost`.

2. In the following example, `dtsdate` indicates that local host’s and remote host’s times differ by more than 10 seconds, without showing the time:

   ```
   dtsdate -s -q remotehost 10
   1
   ```

3. In the following example, `dtsdate` resets the clock if it differs from the remote clock by more than 10 seconds. It does this work silently due to the `-s` option.

   ```
   dtsdate -s remotehost 10
   ```

4. The following example shows a shell script that uses the return value of `dtsdate`:

   ```
   dtsdate -s -q remhost 10
   result = $?
   if [ $result -eq 0 ]; then
      echo "Time is within tolerance."
   elif [ $result -eq 1 ]; then
      echo "Could not contact remote host." >&2
   else # result = 2
      if dtsdate remhost 10; then # it failed!
         echo "Could not set the clock." >&2
      fi
   fi
   ```

**Related Information**

Commands: `dtsd(8dts)`. 
enable(8dts)

enable

Purpose

Starts the DTS entity on the local node

Synopsis

dtscp enable set clock {true | false}

Arguments

set clock {true | false}

Specifies whether the clock is abruptly set (true) or gradually adjusted to the computed time (false, the default). This argument is optional.

Description

After the Distributed Time Service (DTS) entity is created with the dtscp create command, it is still in a nonfunctioning state. To put it into operation, you must invoke dtscp enable, which causes an immediate synchronization to take place. When the command is executed, the status attribute state is set to on.

In addition, you can use the enable command to activate a DTS entity that has previously been deactivated with the disable command. See the disable(8dts) reference page for more information.

Privileges Required

You must have w (write) permission on the access control list (ACL) associated with the DTS entity in order to execute the command.

Notes

The DTS entity cannot be enabled until it is created with the create command; the DTS entity must be in the off state.

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

1. The following command enables the entity and adjusts the clock gradually to the computed time following the first synchronization:

   dtscp> enable

2. The following command enables the entity and abruptly sets the clock to the computed time following the first synchronization:

   dtscp> enable set clock true

Related Information

Commands: create (8dts), disable (8dts), dtscp(8dts).
exit

Purpose

Causes the DTS control program to complete execution

Synopsis

dtscp exit

Description

The `exit` command causes the Distributed Time Service (DTS) control program, `dtscp`, to complete execution and returns operation to the parent process.

Notes

This command might be replaced in future DCE releases by the `dcecp` command, and might no longer be supported at that time.

Examples

To leave `dtscp` and return to the parent process, enter the following:

```
dtscp> exit
```

Related Information

Commands: `dtscp(8dts)`, `quit(8dts)`.
help(8dts)

help

Purpose

Displays help information for DTS control program commands

Synopsis

dtscp help [topic]

Arguments

topic Specifies the topic for which help information is to be displayed. The following are valid help topics:
1. advertise
2. change
3. create
4. delete
5. disable
6. enable
7. set
8. show
9. synchronize
10. unadvertise
11. update

Description

The help command displays information about dtscp commands.

Notes

This command might be replaced in future DCE releases by the dcecp command, and might no longer be supported at that time.

Examples

To display information about the dtscp command unadvertise, enter the following command:

dtscp help unadvertise

Related Information

Commands: dtscp(8dts).
**quit**

**Purpose**

Causes the DTS control program to complete execution

**Synopsis**

```
dtscp quit
```

**Description**

The `quit` command causes the Distributed Time Service (DTS) control program, `dtscp`, to complete execution and returns operation to the parent process.

**Notes**

This command might be replaced in future DCE releases by the `dcecp` command, and might no longer be supported at that time.

**Examples**

To leave `dtscp` and return to the parent process, enter the following:

```
dtscp> quit
```

**Related Information**

Commands: `dtscp(8dts)`, `exit(8dts)`. 
Purpose

Modifies characteristics for the DTS entity

Synopsis

dtscp set characteristic

Arguments

characteristic

The name and value of one or more characteristics to be modified. Valid values for characteristic are described in the following list. These values are described in more detail in the Description section.

Description

The set command modifies the characteristics you specify for the Distributed Time Service (DTS) entity.

Note: Time must never be set backwards in the DCE environment. Backwards movement in the clock causes a server to be unable to determine event ordering, resulting in inconsistency in the server’s database and the corruption of the timestamps. For more information, see the IBM DCE Version 3.1 for AIX and Solaris: Administration Guide—Core Components.

The modifiable characteristics and their values are as follows:

check interval [relative-time]

Specifies the amount of time between checks for faulty servers. Applicable only for servers that have external time-providers.

Default: 0-01:30:00.000
Value: 0-00:00:30.000 – 10675199-02:48:05.000

courier role [role]

Specifies a server’s interaction with the set of global servers.

Default: backup courier

The following values are valid:

backup courier

The local server becomes a courier if none are available on the local area network (LAN).

courier

The local server synchronizes with the global set of servers.

noncourier

The local server does not synchronize with the global set of servers.
error tolerance [relative-time]
  Specifies the maximum separation allowed between the local clock and the
  computed time before synchronizations become abrupt rather than gradual
  (monotonic).
  Default: 0-00:10:00.000
  Value: 0-00:00:00.500 – 10675199-02:48:05.000

global set timeout [relative-time]
  Specifies the amount of time the node waits for a response to a global
  synchronization request before sending another request or declaring a
  global server to be unavailable. The number of attempts made to reach the
  server is controlled by the query attempts characteristic.
  Default: 0-00:00:15.000
  Value: 0-00:00:00.000 – 0-00:10:00.000

local set timeout [relative-time]
  Specifies the amount of time the node waits for a response to a local
  synchronization request before sending another request or declaring a
  server to be unavailable. The number of attempts made to reach the server
  is controlled by the query attempts characteristic.

  Note that the local set timeout value controls only the initial contact with a
  time-provider. During this initial contact, the time-provider itself determines
  the timeout value for actually reporting back times. This allows a
  time-provider attached to a slow source like a modem to request that dtsd
  wait for a longer interval.
  Default: 0-00:00:05.000
  Value: 0-00:00:00.000 – 0-00:01:00.000

maximum inaccuracy [relative-time]
  Specifies the inaccuracy limit for the node. When the node exceeds the
  maximum inaccuracy setting, it attempts to synchronize.
  Default: 0-00:00:00.100
  Value: 0-00:00:00.000 – 10675199-02:48:05.000

query attempts [integer]
  Specifies the number of attempts that a node makes to contact a server
  before the node considers the server unavailable.
  Default: 3
  Value: 1 – 10

server entry name [name]
  Specifies a server’s CDS entry name; dce_hostname represents the name
  of the system or node that is the server’s client. The default setting is the
  recommended value.
  Default: /./hosts/ dce_hostname /dts-entity

server group name [name]
  Specifies the name of the security group that DTS uses for authentication
  checks. DTS clerks and servers do not accept time values from DTS
  servers that are not included in this group.

server principal name [dce_hostname]
  Specifies a server’s principal name for authentication purposes; hostname
represents the name of the system or node that is the server's client. The
default setting is the recommended value.
Default: /.:hosts/ dce_hostname /self

servers required [integer]
Specify the minimum number of servers required for a synchronization.
Settings of 1 or 2 might cause unreliable computed times.
Default: 1 (clerks) 3 (servers)
Value: 1 – 10

synchronization hold down [relative-time]
Specify the interval a node must wait to synchronize. Also specifies
synchronization frequency when a node reaches the value specified by the
maximum inaccuracy characteristic.

Clerks:
Default: 0-00:10:00.000
Value: 0-00:00:30.000 – 01-00:00:00.000

Servers:
Default: 0-00:02:00.000
Value: 0-00:00:30.000 – 01-00:00:00.000

Privileges Required

You must have w (write) permission on the access control list (ACL) associated
with the DTS entity in order to execute the command.

Notes

This command was replaced at DCE Version 1.1 by the dcecp command and might
not be provided in future releases of DCE.

The following two commands are obsolete. Use the replacements shown.

set lan timeout
This command is the same as set local set timeout.

set wan timeout
This command is the same as set global set timeout.

Examples

1. The command in the following example sets the check interval characteristic
to 30 seconds:

   dtscp> set check interval 00-00:00:30.000

2. The following command sets the number of servers required before the entity
can synchronize:

   dtscp set servers required 4

3. The following command sets the courier role for a server:

   dtscp> set courier role backup courier
4. The command in the following example sets the **error tolerance** characteristic to seven minutes:

```
   dtscp> set error tolerance 0-00:07:00.000
```

5. The following command sets the **global set timeout** characteristic to 45 seconds:

```
   dtscp set global set timeout 0-00:00:45.000
```

6. The following command sets the local **set timeout** characteristic to five seconds:

```
   dtscp> set local set timeout 0-00:00:05.000
```

7. The following command sets the **maximum inaccuracy** characteristic to three milliseconds:

```
   dtscp> set maximum inaccuracy 0-00:00:00.300
```

8. The following command sets the **server entry name** characteristic to 
   `/./hosts/orion/dts-entity`:

```
   dtscp> set server entry name /./hosts/orion/dts-entity
```

9. The command in the following example sets the **server principal name** characteristic to 
   `/./hosts/vega/dts-entity`:

```
   dtscp set server principal name /./hosts/vega/dts-entity
```

10. The following command sets the **synchronization hold down** characteristic to
    15 minutes:

```
    dtscp> set synchronization hold down 0-00:15:00.000
```

**Related Information**

Commands: `dtscp(8dts)`, `show (8dts)`. 
Purpose

Displays current information about the DTS entity

Synopsis

```
dtscp show attribute-group attribute-name
```

Arguments

- **attribute-group**
  - The name of an attribute group to be displayed. The following values are valid:
    1. all
    2. all characteristics
    3. all counters
    4. all status
    5. global servers
    6. local servers

- **attribute-name**
  - The name of a specific attribute from the characteristics, counters, or status groups. The attribute specifiers global servers and local servers do not contain any other attributes.

Description

The `show` command displays the names and values of the specified attributes or attribute groups. For attribute groups, if you do not supply a group name with the all argument, all characteristics and their values are displayed. The names of individual attributes, categorized by group, are listed in the following sections.

Note that the attributes displayed by the `show` command might differ depending upon whether you have requested information about a server or a clerk.

Characteristics

Characteristic arguments can contain a maximum of 80 characters and are recalculated to a normalized date format. For example:

Input value: 0-0025:10:99.99999999

Result: 1-01:11:39.990

**acting courier role**

Specifies whether a backup courier is currently functioning as a courier. If the role is noncourier, the node is not attempting to synchronize with global servers. This characteristic is shown only for servers.

Default: noncourier

Value: courier or noncourier
show(8dts)

**automatic tdf change**
Specifies whether automatic changes to the time differential factor are enabled or disabled; the value is determined by the operating system.

Default: `true`
Value: `true/false`

**check interval**
Specifies the amount of time between checks for faulty servers. Applicable only to servers that have external time-providers. This characteristic is shown only for servers.

Default: `0-01:30:00.00`
Value: `0-00:00:30.000 – 10675199-02:48:05.478`

**clock adjustment rate**
Specifies the rate at which the DTS server or clerk entity adjusts the node’s clock during a synchronization.

**clock resolution**
Specifies the amount of time between system clock ticks. The value is determined by the operating system.

**courier role**
Specifies a server’s interaction with the set of global servers. This characteristic is shown only for servers.

Default: `noncourier`
Possible values are as follows:

- **backup courier**
  The local server becomes a courier if none are available on the local area network (LAN).

- **courier**
  The local server synchronizes with the global set of servers.

- **noncourier**
  The local server does not synchronize with the global set of servers.

**DTS version**
Specifies the DTS software version installed on the node.

**epoch number**
Specifies the server’s epoch number. The `change` command modifies this characteristic. This characteristic is shown only for servers.

Default: `0`
Value: `0–255`

**error tolerance**
Specifies the maximum separation allowed between the local clock and the computed time before synchronizations become abrupt rather than gradual (monotonic).

Default: `0-00:10:00.000`
Value: `0-00:00:00.500 – 10675199-02:48:05.478`

**global set timeout**
Specifies the amount of time the node waits for a response to a wide area
show(8dts)

network (WAN) synchronization request before sending another request or
declaring a global server to be unavailable. The number of attempts made
to reach the server is controlled by the query attempts characteristic.

Default: 0-00:00:15.000
Value: 0-00:00:00.000 – 0-00:10:00.000

local set timeout
Specifies the amount of time the node waits for a response to a
synchronization request before sending another request or declaring a
server to be unavailable. The number of attempts made to reach the server
is controlled by the query attempts characteristic.

Default: 0-00:00:05.000
Value: 0-00:00:00.000 – 0-00:10:00.000

local time differential factor
Specifies the Time Differential Factor (TDF), which is the amount of time the
server varies from Greenwich mean time (GMT) or Universal Time
Coordinated (UTC) time.

Default: 0-00:00:00.000
Value: -13-00:00:00 – 13-00:00:00

maximum clock drift rate
Specifies the worst-case drift rate of the node’s clock, in nanoseconds per
second, as determined by the manufacturer’s specifications.

maximum inaccuracy
Specifies the inaccuracy limit for the node. When the node exceeds the
maximum inaccuracy setting, it attempts to synchronize.

Default: 0-00:00:00.100
Value: 0-00:00:00.0 – 10675199:02:48:05.478

next tdf change
Specifies the future time at which the time differential factor is automatically
changed. The value is determined by the operating system.

query attempts
Specifies the number of attempts that a node makes to contact a server
before the node considers the server unavailable.

Default: 3
Value: 1–10

server entry name
Specifies a server’s ACL entry name; hostname represents the name of the
system or node that is the server’s client. The default setting is the
recommended value. This characteristic is shown only for servers.

Default: /./:hosts/: hostname/dts-entity

server group name
Specifies the security group name for the time servers within the cell.

Default: /./:subsys/dce/dts-servers

server principal name
Specifies a server’s principal name for authentication purposes; hostname
represents the name of the system or node that is the server’s client. The default setting is the recommended value. This characteristic is shown only for servers.

Default: /.:/hosts/ hostname/self

servers required
Specifies the minimum number of servers required for a synchronization. Settings of 1 or 2 might cause unreliable computed times.

Default: 3
Value: 1–10

synchronization hold down
Specifies the interval a node must wait to synchronize. Also specifies synchronization frequency when a node reaches the value specified by the maximum inaccuracy characteristic.

Clerks:
Default: 0-00:10:00.0
Value: 0-00:00:30.0 – 01 00:00:00.00

Servers:
Default: 0-00:02.00.0
Value: 0-00:00:30.0 – 01 00:00:00.00

time-provider present
Specifies whether or not the entity used an external time-provider at the last successful synchronization. This attribute applies to servers only.

time representation version
Specifies the timestamp format used by the node.

type
Specifies whether the node is a DTS server or clerk. The create command modifies this characteristic.

Counters

clock settings
Specifies the number of times the node clock has been set nonmonotonically (abruptly).

creation time
Specifies the time at which the DTS entity was created and the counters were initialized.

different epochs detected
Specifies the number of times the node received time response messages from servers or clerks that had epoch numbers different from its own. This counter is shown only for servers.

disable directives completed
Specifies the number of times the DTS has been disabled.

enable directives completed
Specifies the number of times the DTS has been enabled.

epoch changes completed
Specifies the number of times the server’s epoch has changed.

insufficient resources detected
Specifies the number of times the node has been unable to allocate virtual memory.
local servers not in group
   Specifies the number of times that a local server was contacted, but it was
   not in the dts security group.

local times not intersecting
   Specifies the number of times the node’s time interval failed to intersect
   with the computed interval of the servers.

no global servers detected
   Specifies the number of times the courier server could not contact any
   global servers. This counter is shown only for servers.

protocol mismatches detected
   Specifies the number of times the local node failed to process a received
   message containing an incompatible protocol version.

servers not in group
   Specifies the number of times that a nonlocal server was contacted, but it
   was not in the dts security group. This counter is shown only for servers.

servers not responding
   Specifies the number of times the courier server could not contact a specific
   global server. This counter is shown only for servers.

servers times not intersecting
   Specifies the number of times a server has detected faulty servers (other
   than itself). This counter is shown only for servers.

synchronizations completed
   Specifies the number of times the node successfully synchronized.

system errors detected
   Specifies the number of times a DTS operation detected a system error.

time-provider failures detected
   Specifies the number of times the external time-provider signaled a failure
   or the node was unable to access the time-provider.

time-provider timeouts detected
   Specifies the number of times a dtsd server process initiated contact with a
   time-provider and did not receive the initial response within the interval
   specified by local set timeout (the default interval is 5 seconds). This
   counter is shown only for servers.

time representation version mismatches detected
   Specifies the number of times the local node failed to process a received
   message containing an incompatible timestamp format.

too few servers detected
   Specifies the number of times a node failed to synchronize because it could
   not contact the required minimum number of servers.

updates initiated
   Specifies the number of times a server has attempted to update its clock.
   This counter is shown only for servers.

Status

current time
   Specifies the current time on the node.

global servers
   Specifies the set of global servers known by the node.
last synchronization
   Specifies the computed time at the last attempted synchronization.

local servers
   Specifies the set of local servers known by the node.

state
   Specifies the state of the DTS entity. Valid values are as follows:
   off   The DTS entity is disabled.
   on    The DTS entity is enabled.
   synchronizing
      The DTS entity is synchronizing.
   updating
      The DTS entity is updating the time.

uid
   Specifies the entity’s unique identifier, which is generated when the entity is
   created. This attribute is shown only for servers.

Privileges Required
   You must have r (read) permission on the access control list (ACL) associated
   with the DTS entity in order to execute the command.

Notes
   This command was replaced at DCE Version 1.1 by the dcecp command and might
   not be provided in future releases of DCE.

Examples

1. The following command displays the current time:

   dtscp> show current time
   Current Time = 1990-11-30-12:11:41.718-05:00I0.359 EST

2. The following command displays all of the entity’s characteristic attribute
   settings:

   dtscp> show all
   Check Interval = +0-01:30:00.000I-----
   Epoch Number = 0
   Error Tolerance = +0-00:10:00.000I-----
   Local Time Differential Factor = -0-04:00:00.000I-----
   Maximum Inaccuracy = +0-00:00:00.100I-----
   Servers Required = 3
   Query Attempts = 3
   Local Set Timeout = +0-00:00:05.000I-----
   Global Set Timeout = +0-00:00:15.000I-----
   Synchronization Hold Down = +0-00:02:00.000I-----
   Type = Server
   Courier Role = NonCourier
   Acting Courier Role = NonCourier
   Clock Adjustment Rate = 40000000 nsec/sec
   Maximum Clock Drift Rate = 1000000 nsec/sec
   Clock Resolution = 10000000 nsec
   DTS Version = V1.0.1
   Time Representation Version = V1.0.0
   Time Provider Present = FALSE
   Automatic TDF Change = FALSE
   Next TDF Change = 1993-10-31-06:00:00.000+00:00I0.000
   Server Principal Name = hosts/system1/self
   Server Entry Name = hosts/system1/dts-entity
   Server Group Name = subsys/dce/dts-servers
3. The following command displays the current values of all characteristic attributes. It produces the same output as does the `show all` command.

```
dtscp> show all characteristics
```

4. The following command displays all of the local servers known to the entity:

```
dtscp show local servers
Known Servers
============================================================================
Local /.../sisyphus.osf.org/hosts/system2/self
 Last Time Polled = 1993-10-15-21:01:46.124+00:00 I0.809
 Last Observed Time = 1993-10-15-21:03:09.041+00:00 I-----
 Last Observed Skew = +0-00:01:22.917 I-----
 Used in Last Synchronization = TRUE
 Transport Type = RPC

Local /.../sisyphus.osf.org/hosts/system3/self
 Last Time Polled = 1993-10-15-21:01:46.124+00:00 I0.817
 Last Observed Time = 1993-10-15-21:01:46.143+00:00 I0.817
 Last Observed Skew = +0-00:00:00.019 I1.625
 Used in Last Synchronization = TRUE
 Transport Type = RPC
```

5. The following displays the current values of all counter attributes:

```
dtscp> show all counters
Creation Time = 1993-10-14-16:23:57.801+00:00 I0.767
Local Times Not Intersecting = 0
Server Times Not Intersecting = 0
Different Epochs Detected = 0
Too Few Servers Detected = 0
Time Provider Timeouts Detected = 1
Protocol Mismatches Detected = 0
Time Representation Mismatches Detected = 0
No Global Servers Detected = 0
Servers Not Responding = 0
Clock Settings = 0
Epoch Changes Completed = 0
System Errors Detected = 0
Synchronizations Completed = 865
Updates Initiated = 0
Enable Directives Completed = 1
Disable Directives Completed = 0
Insufficient Resources Detected = 0
Time Provider Failures Detected = 0
Local server not in group = 0
Servers not in group = 0
```

6. The following command displays the current values of all status attributes:

```
dtscp> show all status
UID = 00004e34-5e1c-2c87-8500-08005a0d4582
Last Synchronization = 1993-10-15-21:05:43.023+00:00 I-----
State = On
```

7. The following command displays the current value of the courier role attribute:

```
dtscp show courier role
Courier Role = NonCourier
```

8. The following command displays the server entry name for this server:

```
dtscp> show server entry name
Server Entry Name = hosts/system1/dts-entity
9. The following command displays the current state of the DTS entity:

    dtscp> show state
    State = On

10. The following displays the current value of the check interval attribute:

    dtscp> show check interval
    Check Interval = +0-01:30:00.000I-----

11. The following command displays the current value of the servers times not intersecting counter:

    dtscp show servers times not intersecting
    Server Times Not Intersecting = 0

Related Information

Commands: `dtscp(8dts)`, `set(8dts)`. 
synchronize(8dts)

synchronize

Purpose

Causes the DTS entity to synchronize the clock

Synopsis

dtscp synchronize set clock {true | false}

Arguments

set clock {true | false}
  Specifies whether the clock is abruptly set (true) or gradually adjusted to
  the computed time (false, the default). This argument is optional.

Description

The synchronize command causes the Distributed Time Service (DTS) clerk or
server to solicit time intervals from servers, compute the intersection of the time
intervals, and adjust the system clock to the midpoint of the computed time interval.
This command overrides the functions of the synchronization hold down
characteristic.

Privileges Required

You must have w (write) permission on the access control list (ACL) associated
with the DTS entity in order to execute the command.

Notes

The synchronize command does not execute if the entity is already synchronizing
or is disabled; the entity must be in the on state.

This command was replaced at DCE Version 1.1 by the dcecp command and might
not be provided in future releases of DCE.

Examples

1. The following command initiates a synchronization for the entity, followed by a
   gradual clock adjustment:

   dtscp> synchronize

2. The following command initiates a synchronization for the entity, followed by an
   abrupt reset of the clock:

   dtscp> synchronize set clock true

Related Information

Commands: dtscp(8dts).
unadvertise

Purpose

Removes the global server entry from the cell profile

Synopsis

\texttt{dtscp unadvertise}

Description

The \texttt{unadvertise} command causes the Distributed Time Service (DTS) to remove the server’s name from the cell profile and binding from the related Cell Directory Service (CDS) entry, deleting the server’s global status.

Privileges Required

You must have \texttt{w (write)} permission on the access control list (ACL) associated with the DTS entity in order to execute the command.

Notes

This command was replaced at DCE Version 1.1 by the \texttt{dcecp} command and might not be provided in future releases of DCE.

Examples

\texttt{dtscp> unadvertise}

Related Information

Commands: \texttt{dtscp(8dts)}. 
update(8dts)

update

Purpose

Gradually adjusts the clock on the local node to the specified time

Synopsis

dtscp update time absolute-time

Arguments

time absolute-time

Specifies the absolute time to which the clock is adjusted. This argument is required.

Description

The update command gradually adjusts the system clock to a new time, beginning at the time specified in the argument. The difference between the current clock value and the absolute time specified in the argument is used to adjust the clock.

Privileges Required

You must have w (write) permission on the access control list (ACL) associated with the Distributed Time Service (DTS) entity in order to execute the command.

Notes

The update command is valid only for servers. The combined time and inaccuracy value you specify must be contained within the interval formed by the current time and inaccuracy. That is, the new setting must be more accurate than the current time.

This command was replaced at DCE Version 1.1 by the dcecp command and might not be provided in future releases of DCE.

Examples

The following command updates the time for a server, with the clock being gradually adjusted to the specified time:

    dtscp> update time 1993-12-30-11:24:00.000-05:00I0.000

Related Information

Commands: dtscp(8dts).
Introduction to the DCE Security Service administrative files

The *(5sec)* reference pages describe the DCE Security Service files for system administration. The reference pages are as follows:

- **aud_audit_events(5sec)**
  Describes the auditable events for the audit service.

- **cds_audit_events(5sec)**
  Describes the auditable events for the directory service.

- **dts_audit_events(5sec)**
  Describes the auditable events for the time services.

- **event_class(5sec)**
  Describes event class files. Each event class file contains the declaration of a single event class, a logical group of auditable events.

- **group_override(5sec)**
  Describes the *group_override* file, which contains entries that let you override group UNIX ID and member list entries in the registry database for a local machine.

- **passwd_override(5sec)**
  Describes the *passwd_override* file, which contains entries that let you override password, GECOS, home directory, and shell entries in the registry database for a local machine.

- **sec_audit_events(5sec)**
  Describes the auditable events for the security service.

- **v5srvtab(5sec)**
  Describes the *v5srvtab* file, which contains server machine passwords on the local machine.

### Related Information

- Files: *aud_audit_events(5sec), dts_audit_events(5sec), event_class(5sec), passwd_override(5sec), sec_audit_events(5sec), v5srvtab(5sec)*.

Purpose

Auditable events for the audit services

Description

The DCE Security Service supports the auditing of audit service-significant events. Among these events are:

1. Administrative operations
   These are subdivided into modify and query operations.
2. Filter operations
   These are subdivided into modify and query operations.

Event class definitions, together with filters, control the auditing execution at these code points. Filters can be updated dynamically. Filter files are maintained by a per-host audit daemon, and are shared among all the audit clients on the same host. The DCE control program, dcecp, is used for maintaining the filters. (See the dcecp(8dce) reference page.) The dcecp command is executable by all users and system administrators. The ability to modify filters is controlled through audit daemon’s access control list (ACL), which maintains the filters.

The audit service remote procedure call (RPC) interfaces include audit_control and audit_filter operations.

Administrative Operations

The dce_audit_admin_modify and dce_audit_admin_query event classes lump together the administrative operations that are performed on the audit daemon.

The dce_audit_admin_modify event class has the following events that modify the operation of the audit daemon:

EVT_MODIFY_STATE
   Enables or disables the audit daemon for logging.

EVT_MODIFY_SSTRATEGY
   Modifies the storage strategy for the central trail. This can be any of the following:
   Save   If the trail is full, it is backed up and renamed with a timestamp then writes on the original trail again.
   Wrap   If the trail is full, goes back to the beginning of the file, overwriting previously written records.

EVT_REWIND
   Rewinds the audit daemon’s central trail file.

EVT_STOP
   Stops the audit daemon.

The following are the audit code points in the audit service interfaces. Each entry shows the event type, followed by the event number and event classes, and then any event-specific information.
The `dce_audit_admin_query` event class has two events:

**EVT_SHOW_SSTRATEGY**

Shows the storage strategy.
EVT_SHOW_STATE
Shows the state of the audit daemon.

Following are the details of this event class:

**EVT_SHOW_SSTRATEGY (0x309, dce_audit_admin_query)**

Generated in:

```c
audit_control_show_sstrategy()
dcecp: aud show
```

Event-specific information:
```
[if (outcome != aud_c_esl_cond_success) ]
ulong_int last_error_status
[ end if ]
```

**EVT_SHOW_STATE (0x30a, dce_audit_admin_query)**

Generated in:

```c
audit_control_show_state()
dcecp: aud show
```

Event-specific information:
```
[if (outcome != aud_c_esl_cond_success) ]
ulong_int last_error_status
[ end if ]
```

**Filter Operations**

The `dce_audit_filter_modify` and `dce_audit_filter_query` event classes are the filter operations that the audit daemon handles.

The `dce_audit_filter_modify` event class has the following events:

**EVT_ADD_FILTER**

Adds a filter.

**EVT_DELETE_FILTER**

Removes all guides for a specific subject.

**EVT_REMOVE_FILTER**

Removes a specific guide for a specific subject.

Following are the details of this event class:

**EVT_ADD_FILTER (0x303, dce_audit_filter_modify)**

Generated in:

```c
audit_control_add_filter()
dcecp: audfilter create
dcecp: audfilter modify -add
```

Event-specific information:

```
Format ID: 1
ulong_int esl_type (aud_esl_type_t)
    [ if (esl_type != aud_e_esl_world) &&
      (esl_type != aud_e_esl_word_overridable) ]
char_string subject_name
    [ end if ]
short_int num_new_guides
```
audit_events(5sec)

short_int num_new_guides
[ 0 <= i < num_new_guides ]
short_int num_new_evt_classes(i)
[ 0 <= j < num_new_evt_classes(i) ]
ulong_int new_evt_class(i,j)
[ end iteration ]
ulong_int new_condition(i) (aud_esl_cond_t)
ulong_int new_action(i) (aud_esl_act_t)
[ end iteration ]
if (outcome == aud_c_esl_cond_success)
short_int num_old_guides
[ 0 <= i < num_old_guides ]
short_int num_old_evt_classes(i)
[ 0 <= j < num_old_evt_classes(i) ]
ulong_int old_evt_class(i,j)
[ end iteration ]
ulong_int old_condition(i) (aud_esl_cond_t)
ulong_int old_action(i) (aud_esl_act_t)
[ end iteration ]
end if

EVT_DELETE_FILTER (0x300, dce_audit_filter_modify)

Generated in:

audit_control_delete_filter()
dcecp: audfilter delete

Event-specific information

Format ID: 1
ulong_int esl_type (aud_esl_type_t)
[if (esl_type != aud_e_esl_world) &&
(esl_type != aud_e_esl_word_overridable) ]
char_string subject_name
[ end if ]
if (outcome == aud_c_esl_cond_success)
short_int num_guides
[ 1 <= i <= num_guides ]
short_int num_evt_classes(i)
[ 1 <= j <= num_evt_classes(i) ]
ulong_int evt_class(i,j)
[ end iteration ]
ulong_int condition(i) (aud_esl_cond_t)
ulong_int action(i) (aud_esl_act_t)
[ end iteration ]
else
ulong_int last_error_status
[ end if ]

EVT_REMOVE_FILTER (0x304, dce_audit_filter_modify)

Generated in:

audit_control_remove_filter()
dcecp: audfilter modify -remove

Event-specific information

Format ID: 1
ulong_int esl_type (aud_esl_type_t)
[if (esl_type != aud_e_esl_world) &&
(esl_type != aud_e_esl_word_overridable) ]
char_string subject_name
[ end if ]
short_int num_guides
The `dce_audit_filter_query` contains two events:

**EVT_LIST_FILTER**
Lists all subjects that have filters.

**EVT_SHOW_FILTER**
Shows all filters for a specific principal.

Following are the details of this event class.

**EVT_LIST_FILTER (0x302, dce_audit_filter_query)**

Generated in:

```c
audit_control_list_filter()
dcecp: audfilter catalog
```

Event-specific information:
Format ID: 1
- `long_int` `esl_type`
  - `[if (outcome != aud_c_esl_cond_success) ]`
- `ulong_int` `last_error_status`
  - `[ end if ]`

**EVT_SHOW_FILTER (0x301, dce_audit_filter_query)**

Generated in:

```c
audit_control_show_filter()
dcecp: audfilter { catalog | show }
```

Event-specific information:
Format ID: 1
- `ulong_int` `esl_type (uad_esl_type_t)`
  - `[ if (esl_type != aud_e_esl_world) &&
    (esl_type != aud_e_esl_word_overridable) ]`
- `char_string` `subject_name`
  - `[ end if ]`
- `ulong_int` `last_error_status`
  - `[ end if ]`
Related Information

Commands: `dcecp(8dce)`.

Files: `event_class(5sec)`.
Purpose

Auditable events for directory services.

Description

Code is in place for auditing significant events in the CDS server and client. Among these events are the following:

1. Deletion, creation, and update of directory objects including:
   a. Objects
   b. Directories
   c. Links
   d. Attributes
2. Attempts to invoke operations that involve access control
3. Attempts to cache information at the client

Event class definitions, together with filters, control the auditing execution at these code points. Filters can be updated dynamically. Filter files are maintained by per-host audit daemon, and are shared among all the audit clients on the same host. The DCE control program, dcecp, is used for maintaining the filters. (See "dcecp" on page 118.) The dcecp command is executable by all users and system administrators. The ability to modify filters is controlled through audit daemon’s access control list (ACL), which maintains the filters.

DACL Management Interface (rdaclif) Operations

The rdacl_lookup() operation retrieves an ACL of an object in the directory server. Review of the ACL associated with an object in the directory server is allowed if the caller has access to the object.

Event Type (Event Number, Event Classes)

CSD_ACL_Lookup(0x420, dce_cds_control)

Generated in:

rdacl_replace()
dcecp: acl delete
dcecp: acl modify
dcecp: acl replace

Event-specific information

Format ID: 1

char_string entry_name
uuid manager_type
ulong_int acl_type (sec_acl_type_t)
acl ack (sec_acl_t)

The rdacl_get_access() operation determines the caller’s access to a specified object. This call is authorized if the caller has any access to the object.

Event Type (Event Number, Event Classes)

CDS_ACL_GetAccess(0x422, dce_cds_control)

Generated in:
The rdacl_test_access() operation determines if the caller has the requested access. The return value of the call indicates whether the caller has the requested access to the object.

**Event Type (Event Number, Event Classes)**

CDS_ACL_TestAccess(0x423, dce_cds_control)

**Generated in:**

rdacl_test_access()
dcecp: acl check

**Event-specific information**

Format ID: 1

char_string entry_name
uuid manager_type
ulong_int desired_permset (sec_acl_permset_t)

The rdacl_test_access() operation determines if a user (not the caller) has the requested access. The return value of the call indicates whether the user has the requested access to the object.

**Event Type (Event Number, Event Classes)**

CDS_ACL_TestAccessOnBehalf(0x424, dce_cds_control)

**Generated in:**

rdacl_test_access_on_behalf()
dcecp: acl check

**Event-specific information**

Format ID: 1

char_string entry_name
uuid manager_type
ulong_int desired_permset (sec_acl_permset_t)
pac subject_pac

The rdacl_get_manager_types() operation lists the types (UUIDs) of ACLs protecting an object. The caller must have some permissions on the object for each of the manager types that are defined for the object. Otherwise, not manager type is returned.

**Event Type (Event Number, Event Classes)**

CDS_ACL_GetMgrTypes(0x425, dce_cds_control)

**Generated in:**

rdacl_get_manager_types()
rdacl_get_mgr_types_semantics()
dcecp: acl (all operations)

**Event-specific information**

Format ID: 1
The `rdacl_get_referral()` operation obtains a referral to an ACL update site.

**Event Type (Event Number, Event Classes)**

**CDS_ACL_GetReferral(0x426, dce_cds_control)**

**Generated in:**

- `rdacl_get_referral()`
- `dcecp: acl delete`
- `dcecp: acl modify`
- `dcecp: acl replace`

**Event-specific information**

Format ID: 1

- `char_string entry_name`
- `uuid manager_type`
- `ulong_int acl_type (sec_acl_type_t)`

---

**CDS Server Database (cds_clerkserver) Operations**

The `cds_NewEpoch()` operation declares a new epoch. It rebuilds the replica set from user input and updates the epoch attributes of each replica.

**Event Type (Event Number, Event Class)**

**CDS_NewEpoch (0x0401, dce_cds_server_modify)**

**Generated in:**

- `cdsNewEpoch()`
- `dcecp: directory modify`
- `cdscp: set dir to new epoch`

**Event-specific information:**

- `char_string - FullName of directory`
- `[if (outcome != aud_c_esl_cond_success)]`
- `char_string - 'master'`
- `uuid - uuid of master`
- `char_string - 'readonlyies'`
  - `[for # readonly clearinghouses]`
  - `char_string - name of readonly clearinghouse`
  - `[end for]`
  - `[end if]`

---

The `cds_DeleteObject()` operation tries to find an object entry and if found, it marks it as absent. Background processing will actually remove it from the database.

**Event Type (Event Number, Event Class)**

**CDS_DelObject (0x0402, dce_cds_server_modify)**

**Generated in:**

- `cdsDeleteObject()`
- `cdsdel: -b`
- `cdsdel: -g`
- `cdsdel: -p`
- `cdsdel: -o`
- `cdsdel: -r`
cds_del: -R
dcecp: object delete
cdscp: delete object

Event-specific information:
char_string - FullName of Object

The cds_DeleteSoftLink() operation tries to find a softlink entry and if found, it marks it as absent. Background processing will actually remove it from the database.

Event Type (Event Number, Event Class)
CDS_DelSoftLnk (0x0403, dce_cds_server_modify)

Generated in:

cdsDelSLink()
cdsdel: -l
cdsdel: -r
cdsdel: -R
dcecp: link delete
cdscp: delete link

Event-specific information:
char_string - FullName of SoftLink

The cds_DeleteDirectory() operation tries to find a directory entry and if found, it marks it as absent. Background processing will actually remove it from the database.

Event Type (Event Number, Event Class)
CDS_DelDir (0x0404, dce_cds_server_modify)

Generated in:

cdsDelDir()
cdsdel: -d
cdsdel: -r
cdsdel: -R
dcecp: directory delete
cdscp: delete directory

Event-specific information:
char_string - FullName of Directory

The cds_DeleteChild() operation tries to find a child entry and if found, it marks it as absent. Background processing will actually remove it from the database.

Event Type (Event Number, Event Class)
CDS_DelChild (0x0405, dce_cds_server_modify)

Generated in:

cds_DelChild()
dcecp directory add
cdscp delete child
cdsdel: -d
cdsdel: -r
Event-specific information:
char_string - FullName of Directory

The \texttt{cds\_Skulk()} operation issues calls to perform a skulk on the specified directory.

**Event Type (Event Number, Event Class)**

\texttt{CDS\_Skulk (0x0406, dce\_cds\_server\_modify)}

*Generated in:*

\begin{itemize}
  \item \texttt{cdsSkulkDir()}
  \item \texttt{background()}
  \item \texttt{cdscp set dir to skulk}
  \item \texttt{dcecp directory synchronize}
\end{itemize}

Event-specific information:
char_string - FullName of Directory

The \texttt{cds\_ReadAttribute()} operation attempts for find and entry, and if found will return a set or a single valued attribute.

**Event Type (Event Number, Event Class)**

\texttt{CDS\_ReadAtt (0x0407, dce\_cds\_server\_query)}

*Generated in:*

\begin{itemize}
  \item \texttt{cdsReadAttrSet()}
  \item \texttt{cdsReadAttrValue()}
  \item \texttt{cdsReadMbr()}
  \item Many \texttt{cds}d functions
\end{itemize}

Event-specific information:
char_string - Object \{object, directory, clearinghouse, link\}
char_string - Attribute Name

The \texttt{cds\_EnumerateAttributes()} operation attempts to find an entry, and if found it returns a set of attribute specifiers for the entry.

**Event Type (Event Number, Event Class)**

\texttt{CDS\_EnumAtt (0x0408, dce\_cds\_server\_query)}

*Generated in:*

\begin{itemize}
  \item \texttt{cdsEnumAttr()}
  \item \texttt{dcecp directory delete}
  \item \texttt{dcecp [ object | link | clearinghouse | dir ] show}
  \item \texttt{cdscp show [ object | link | clearinghouse | dir | replica ]}
\end{itemize}

Event-specific information:
char_string - Object \{object, directory, clearinghouse, link\}

The \texttt{cds\_ModifyAttribute()} operation attempts to locate an entry and make an update to it.

**Event Type (Event Number, Event Class)**

\texttt{CDS\_ModAtt (0x0409, dce\_cds\_server\_query)}

*Generated in:*

\begin{itemize}
  \item \texttt{cdsAuditEvents}
\end{itemize}
The `cdsEnumerateChild()` operation builds a set of object entries in the named directory based on the wildcard string.

**Event Type (Event Number, Event Class)**

**CDS_ENUMChildren (0x040A, dce_cds_server_query)**

Generated in:

```
cdsEnumChild()
```

**Event-specific information:**

```
char_string - Object Name
char_string - '{present/absent,attributeName}'
[ if (aud_c_esl_cond_success) ]
Atomic Value
[ end if ]
```

The `cdsEnumerateSoftLink()` operation builds a set of object entries in the named directory based on the wildcard string.

**Event Type (Event Number, Event Class)**

**CDS_ENUMSoftLnk (0x040B, dce_cds_server_query)**

Generated in:

```
cdsEnumSLink
dcecp list dir
cdscp list link
cdsli -l
cdsdel -l
```

**Event-specific information:**

```
char_string - Directory Name
[ if (!aud_c_esl_cond_success) ]
char_string - Wildcard
[ end if ]
```

The `cdsEnumerateObject()` operation builds a set of object entries in the named directory based on the wildcard string and class.

**Event Type (Event Number, Event Class)**

**CDS_ENUMObject (0x040C, dce_cds_server_query)**

Generated in:

```
cdsEnumObject
```

**Event-specific information:**

```
char_string - Directory Name
[ if (!aud_c_esl_cond_success) ]
char_string - Wildcard
[ end if ]
```
The `cds_T estAttribute()` operation tries to find an entry. If the entry is found and the specified attribute is present, it compares the set of values with the supplied value and returns the results.

The `cds_CreateObject()` operation makes sure an object does not exist and then creates it and its ACLs.

The `cds_CreateSoftLink()` creates a new soft link.
The `cds_CreateDirectory()` operation creates the directory in the CDS database specified clearinghouse or in the clearinghouse of the parent if not specified.

**Event Type (Event Number, Event Class)**

`CDS_CreateDir (0x0410, dce_cds_server_modify)`

**Generated in:**

```c
cdsCreateDir()
dcecp directory create
cdscp create directory
```

**Event-specific information:**

- `char_string - Directory Name`
- `uuid - Clearinghouse UUID`

The `cds_CreateChild()` operation creates a child pointer in the parent directory.

**Event Type (Event Number, Event Class)**

`CDS_CreateChild (0x0411, dce_cds_server_modify)`

**Generated in:**

```c
cdsCreateChild()
dcecp directory create
dcecp directory add
cdscp create directory
cdscp create child
```

**Event-specific information:**

- `char_string - Child Name`
- `uuid - Parent UUID`
- `uuid - Directory UUID`

The `cds_AddReplica()` operation creates a directory replica. It is used by clerk initiated transactions and management when a new clearinghouse is created.

**Event Type (Event Number, Event Class)**

`CDS_AddRep (0x0412, dce_cds_server_modify)`

**Generated in:**

```c
cdsAddReplica()
dcecp directory create
cdscp create replica
```

**Event-specific information:**

- `char_string - Replica Name`
- `char_string - type 'readonly'`
- `uuid - Clearinghouse UUID`

...
The cds_RemoveReplica() operation removes a replica.

**Event Type (Event Number, Event Class)**

CDS_RemoveRep (0x0413, dce_cds_server_modify)

Generated in:

- cdsRmReplica()
- dcecp directory delete
- cdscp delete replica

**Event-specific information:**

- char_string - Replica Name
- uuid - Clearinghouse UUID

The cds_DoUpdate() function applies an update packet to an entry. These update packets are sent from the CDS server with the master replica of a directory to the CDS servers with read-only replicas.

**Event Type (Event Number, Event Class)**

CDS_DoUpdate (0x0414, dce_cds_server_update)

Generated in:

- cdsDoUpdate()
- various cdsd functions

**Event-specific information:**

- char_string - Directory Fullname
- uuid - CDS Epoch
- utc - Creation TimeStamp
- short_int - entry type
  - #define ET_directory 1
  - #define ET_object 2
  - #define ET_childPointer 3
  - #define ET_softlink 4
  - #define ET_clearinghouse 5
  - #define ET_anyDirectName 6
  - #define ET_firstLink 7
  - #define ET_dirOrObj 8
- short_int - update count
  - [ ( while (update_count-- > 0 ) ]
- utc - timestamp
- char_string - Attribute Name
  - Atomic Value
  - [ ( end while ) ]
  - [ ( end if ) ]

The CDS_Combine() function copies all updates within a time period back to the requesting clearinghouse for one directory.

**Event Type (Event Number, Event Class)**

CDS_Combine (0x0415, dce_cds_server_modify)

Generated in:

- cdsCombine()
- dcecp directory synchronize
- cdscp set dir to skulk
- various cdsd functions
The cds_LinkReplica() function adds or removes a replica of a directory. It is used to complete the creation of a new replica by adding the replica to the cds_REPLICAS attribute of the directory entry.

**Event Type (Event Number, Event Class)**
CDS_LinkRep (0x0416, dce_cds_server_modify)

Generated in:

cdsLinkReplica()
dcecp clearinghouse create
cdscp create clearinghouse

The cds_ModifyReplica() function applies an update to an attribute.

**Event Type (Event Number, Event Class)**
CDS_ModifyRep (0x0417, dce_cds_server_modify)

Generated in:

cdsModifyReplica()

**Atomic Values**

<table>
<thead>
<tr>
<th>VT</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>VT_none</td>
<td>None</td>
</tr>
<tr>
<td>01</td>
<td>VT_long</td>
<td>long</td>
</tr>
<tr>
<td>02</td>
<td>VT_short</td>
<td>short</td>
</tr>
<tr>
<td>03</td>
<td>VT_small</td>
<td>small</td>
</tr>
<tr>
<td>04</td>
<td>VT_uuid</td>
<td>uuid</td>
</tr>
<tr>
<td>05</td>
<td>VT_Timestamp</td>
<td>utc</td>
</tr>
<tr>
<td>06</td>
<td>VT_Timeout</td>
<td>char_string</td>
</tr>
<tr>
<td>07</td>
<td>VT_Version</td>
<td>char_string</td>
</tr>
</tbody>
</table>
CDS Advertiser Cache (cprpc) Operations

Calls to modify CDS clients or their caches will access these interfaces. Commands cdscp and dcecp call the function dnscp_clerk() to control the clients themselves and their caches.

clerk_acl_resolve() is an internal cdsadv function that initializes ACLs.

Event Type (Event Number, Event Class)
CDS_CheckAdvAclType (0x440, dce_cds_adver)

Generated in:

cdsadv initialization

Event-specific information:
char_string - requesting client
UUID - requesting client clerk's uuid
long_int - sec acl type of object

dnscp_clerk(DISCLK) disables and shuts down the advertiser.
cds_audit_events

Event Type (Event Number, Event Class)
CDS_DisableClerk (0x441, dce_cds_adver)
Generated in:

dnscp_clerk()
dcecp cdsclient disable
cdscp disable clerk

Event-specific information:
char_string - host name of local clerk
UUID - uuid of local clerk
long_int - sec acl type of object

dnscp_clerk(DEFCLKSRV) adds cached nameserver information to the client cache.

Event Type (Event Number, Event Class)
CDS_DefineCaSvr (0x442, dce_cds_adver)
Generated in:

dnscp_clerk()
dcecp cdscache create
cdscp define cached server

Event-specific information:
char_string - server being defined
UUID - uuid of server, if known
long_int - sec acl type of object
char_string - protocol to defined server

dnscp_clerk(CLEARCLKSRV) removes cached nameserver information from the client cache.

Event Type (Event Number, Event Class)
CDS_ClearCaSvr (0x443, dce_cds_adver)
Generated in:

dnscp_clerk()
dcecp cdscache delete
cdscp clear cached server

Event-specific information:
char_string - server being removed from cache
UUID - uuid of server, if known
long_int - sec acl type of object

TestAdverUserAccess() determines if the client has the specified permissions to the clerk.

Event Type (Event Number, Event Class)
CDS_TestAdverAcl(0x443, dce_cds_adver)
Generated in:
All dnscp_clerk functions and serviceablity functions.

Event-specific information:
char_string - name of requesting client
UUID - clerk's management dacl uuid
long_int - sec acl type of object
GDA Acl Operations

These interfaces are reached by issuing calls when commands are issued that require permission check on GDA acls.

**Event Type (Event Number, Event Class)**

CDS_CheckGdaAclType (0x0430, dce_cds_gda)

*Generated in:*

- gda_acl_resolve()
- gda initialization

*Event-specific information:*

- char_string - requesting client
- UUID - requesting client clerk's uuid
- long_int - sec acl type of the object

CTestUserAccess() determine if the client identified has the specified permissions to the gdad.

**Event Type (Event Number, Event Class)**

CDS_TestGdaAcl (0x0431, dce_cds_gda)

*Generated in:*

- TestUserAccess()
- gda serviceablity setup functions

*Event-specific information:*

- char_string - local clerk
- UUID - gda management dacl uuid
dts_audit_events

Purpose

Auditable events for the time services

Description

The DCE Security Service supports the auditing of security-significant events in the time server. Among these events are:

1. Time service processes
2. Clock readings
3. Global-set membership (in the cell service profile)
4. Time service attributes

Event class definitions, together with filters, control the auditing execution at these code points. Filters can be updated dynamically. Filter files are maintained by a per-host audit daemon, and are shared among all the audit clients on the same host. The DCE control program, `dcecp`, is used for maintaining the filters. (See the `dcecp(8dce)` reference page.) The `dcecp` command is executable by all users and system administrators. The ability to modify filters is controlled through audit daemon’s access control list (ACL), which maintains the filters.

The time server remote procedure call (RPC) interfaces that manage the Distributed Time Service (DTS) and request and provide the time include `time_control`, `time_service`, `gbl_time_service`, and `time_provider`.

The following are the audit code points in these time service interfaces. Each entry shows the event type, followed by the event number and event classes, and then any event-specific information.

Control Interface (time_control) Operations

The `CreateCmd()` operation creates the time service as a server or a clerk. The caller must have write access to the management interface.

**EVT_CREATE_CMD (0x200, dce_dts_mgt_modify)**

- Generated in:
  - dtscp: create

- Event-specific information:
  - `signed32` `servType`

The `DeleteCmd()` operation deletes the time service entity from the system where the command is entered. This command stops the process. The caller must have write access to the management interface.

**EVT_DELETE_CMD (0x201, dce_dts_mgt_modif)**

- Generated in:
  - dcecp: dts stop

- Event-specific information:
  - None.
The **EnableCmd**() operation starts the DTS entity on the local node. This command makes the server available to the network. The *clockSet* argument tells the time service whether or not to set the clock after the first synchronization. The caller must have write access to the management interface.

**EVT_ENABLE_CMD (0x202, dce_dts_mgt_modify)**

- **Generated in:**
  - dcecp: dts activate

- **Event-specific information:**
  - `signed32 clockSet`

The **DisableCmd** operation disables the time service by making it unavailable to the network. In the case of servers, it makes it unavailable to the RPC client trying to talk to it. For clerks, it stops synchronizing with servers. The caller must have write access to the management interface.

**EVT_DISABLE_CMD (0x203, dce_dts_mgt_modify)**

- **Generated in:**
  - dcecp: dts { deactivate | stop }

- **Event-specific information:**
  - None.

The **UpdateCmd**() operation gradually adjusts the clock on the local node to the specified time. The caller must have write access to the management interface.

**EVT_UPDATE_CMD (0x204, dce_dts_synch)**

- **Generated in:**
  - dcecp: clock set -to

- **Event-specific information:**
  - `utc_t old_time`
  - `utc_t new_time`

The **ChangeCmd** operation changes the epoch number on the server and optionally sets the time to a new time. These values are passed in the argument *changeDir*. The caller must have write access to the management interface.

**EVT_CHANGE_CMD (0x205, dce_dts_synch)**

- **Generated in:**
  - dcecp: clock set -{ abruptly | epoch }

- **Event-specific information:**
  - Format ID: 1
  - `long_int epoch`
  - `boolean new_time_present`
  - `[ if (new_time_present) ]`
  - `utc_t old_time`
  - `utc_t new_time`
  - `[ end if ]`

The **SynchronizeCmd**() operation causes the time service to synchronize immediately. If the argument *clockSet* is true, the clock is set to the new value after a synchronization. The caller must have write access to the management interface.

**EVT_SYNCHRONIZE_CMD (0x206, dce_dts_synch)**

- **Generated in:**

---

Chapter 5. Security Service Files and Commands  601
**dts_audit_events(5sec)**

```
dcecp:  clock synchronize [-abruptly] [-dtsd]
dcecp:  dts synchronize
```

**Event-specific information:**

```
signed32  setClock
```

The **AdvertiseCmd**() operation adds (advertises) this time server node as a member of the global set in the cell service profile. The caller must have write access to the management interface.

**EVT_ADVERTISE_CMD (0x207, dce_dts_mgt_modify)**

- **Generated in:**
  - dcecp: dts configure -global

  **Event-specific information:**
  - None.

The **UnadvertiseCmd**() operation removes (unadvertises) this time server node as a member of the set of global servers in the cell service profile. The caller must have write access to the management interface.

**EVT_UNADVERTISE_CMD (0x208, dce_dts_mgt_modify)**

- **Generated in:**
  - dcecp: dts configure -notglobal

  **Event-specific information:**
  - None.

The **SetDefaultCmd**() operation, when an attribute with no accompanying value is passed, sets an attribute to its default value. The attribute type is passed in the `setAttr` argument. The caller must have write access to the management interface.

**EVT_SET_DEFAULT_CMD (0x209, dce_dts_mgt_modify)**

- **Generated in:**
  - dtscp: set <attribute>

  **Event-specific information:**
  - Format ID: 1
  - long_int  attribute (aud_log_dts_attr)

The **SetAttrCmd**() operation, when an attribute and an accompanying value is passed, sets an attribute to a value given. The attribute type is passed in the `setAttr` argument and the attribute value in the `AttrValue` argument. The caller must have write access to the management interface.

**EVT_SET_ATTR_CMD (0x20A, dce_dts_mgt_modify)**

- **Generated in:**
  - dts modify

  **Event-specific information:**
  - Format ID: 1
  - long_int  attribute_type (aud_log_dts_attr)  
  [ if (attribute_type == aud_l_dts_attr_svr_princ_name)  
  (attribute_type == aud_l_dts_attr_svr_entry_name)  
  (attribute_type == aud_l_dts_attr_svr_name)  
  (attribute_type == aud_l_dts_attr_dns_version)  
  (attribute_type == aud_l_dts_attr_dcedts_version)  
  (attribute_type == aud_l_dts_attr_time_rep)  
  (attribute_type == aud_l_dts_attr_svr_group_name) ]
The `ShowAttrCmd( )` operation, when passed an attribute name, queries the time service for the attribute's value. The attribute value is passed back in the argument `attrValue`. The caller must have read access to the management interface.

**EVT_SHOW_ATTR_CMD (0x20B, dce_dts_mgt_query)**

Generated in:

```
dcecp: clock show [-dtsd]
dcecp dts show [-{ attributes | all }]
```
SNMP get { aDtsdCurrTime | aDtsSvrRoles }
SNMP next { aDtsSyncCmpltd | aDtsSvrProviderErrors }
SNMP (daemon startup)

Event-specific information:
Format ID: 1
long_int attribute

The ShowAllCharsCmd( ) operation, when not passed a group name with the all value, queries the time service for the values of all the characteristic attributes and values. The caller must have read access to the management interface.

EVT_SHOW_ALL_CHARS_CMD (0x20C, dce_dts_mgt_query)

Generated in:
dts show [-{ attributes | all }]

Event-specific information:
None.

The ShowAllStatusCmd( ) operation, when passed the all status value, queries the time service for the values of all the status attributes. The caller must have read access to the management interface.

EVT_SHOW_ALL_STATUS_CMD (0x20D, dce_dts_mgt_query)

Generated in:
dtscp: show all status

Event-specific information:
None.

The ShowAllCntrsCmd( ) operation, when passed the all counters value, queries the time service for the values of all the counters. The caller must have read access to the management interface.

EVT_SHOW_ALL_CNTRS_CMD (0x20E, dce_dts_mgt_query)

Generated in:
dcecp: dts show [-{ counters | all }]

Event-specific information:
None.

The ShowLocServersCmd( ) operation, when passed the local servers value, queries the time service for the servers in the local set. A variable conformant array is used to return the set of local servers available. The size of the array transmitted over RPC is determined at run-time. The caller must have read access to the management interface.

EVT_SHOW_LOC_SERVERS_CMD (0x20F, dce_dts_mgt_query)

Generated in:
dcecp: dts show [-{ attributes | all }]

Event-specific information:
None.

The ShowGblServersCmd( ) operation, when passed the global servers value, queries the time service for the servers in the global set. A variable conformant array is used to return the set of global servers available. The caller must have read access to the management interface.

EVT_SHOW_GBL_SERVERS_CMD (0x20F, dce_dts_mgt_query)

Generated in:
dcecp: dts show [-{ attributes | all }]

Event-specific information:
None.
Time-Provider Interface (time_provider) Operations

Auditable events in the RPC-based Time-Provider Program (TPP) interfaces are defined here. These events are invoked by a time service daemon running as a server (in this case it makes an RPC client call to the TPP server).

The **ContactProvider( )** operation sends initial contact message to the TPP. The TPP server responds with a control message. This operation might cause modification of the time server’s (not the provider’s) clock and should be defined to be an auditable event in the time server. There is no access control in the provider for this operation, but the integrity of the messages is protected.

**EVT_CONTACT_PROVIDER** (0x211, dce_dts_time_provider)

Generated in:

dtsd, configured with:
dcecp: dts modify /<cell_name>/hosts/<hostname>/dts-entity
                    -checkinterval

Event-specific information:
None.

The **ServerRequestProviderTime( )** operation has the client send a request to the TPP for times. The TPP server responds with an array of time stamps obtained by querying the time-provider hardware that it polls. There is no access control in the time-provider for this operation, but the integrity of the message is protected.

**EVT_REQUEST_PROVIDER_TIME** (0x212, dce_dts_time_provider)

Generated in:

dtsd, configured with:
dcecp: dts modify /<cell_name>/hosts/<hostname>/dts-entity
                    -checkinterval

Event-specific information:
None.

Related Information

Commands: aud(8dce), audfilter(8dce), dcecp(8dce), advertise(8dts),
change(8dts), create(8dts), delete(8dts), disable(8dts), dts_intro(8dts),
dtsd(8dts), enable(8dts), exit(8dts), help(8dts), quit(8dts), set(8dts),
show(8dts), synchronize(8dts), unadvertise(8dts), update(8dts).

Files: aud_audit_events(5sec), event_class.5sec, sec_audit_events(5sec).
event_class

Purpose

Contains the declaration of an event class

Description

Audit events can be logically grouped into event classes. Event classes are defined in event class files. An event class file contains an event class number and a list of event numbers corresponding to audit events.

All event class files must be created in the `dcelocal/etc/audit/ec` directory.

The name of the event class file becomes the name of the event class. The recommended naming convention for event class files is as follows:

```
dce_server-name_class
```

In this format, `class` is a descriptive text that characterizes the event class.

Event class files must be write-protected by the local operating system (that is, only administrators should have write access to these files). Audit clients read these files to maintain an event table in their address space.

Optionally, an event class file can contain a `SEP` line. This line contains a list of prefixes of the event numbers in the file. The SEP line speeds up the scanning performed by the audit clients. Audit clients which do not have events with one of the prefixes listed will not scan the event list. If the SEP line is not provided in the file, audit clients will have to read the entire file to find out if the event class file contains any of their events.

Empty lines are ignored in the event class file. Comments are designated by a `#` (number sign) placed before the comment text.

**The Event Class File Format**

The format of an event class file is as follows:

```
ECN = event_class_number
SEP = prefix_1 prefix_2 ...
# comments start with a number sign
event_number_1
# another comment
event_number_2 ...
```

**Examples**

The following is an example of an event class file for the event class `ec_local_authentication`:

```
ECN = 0x00000001
SEP = 0x100
# AS_Request
0x00000010
# TGS_TicketReq
```

---

606 IBM DCE Version 3.1 for AIX and Solaris: Administration Commands Reference
Related Information

Files: aud_audit_events(5sec), dts_audit_events(5sec), sec_audit_events(5sec)
group_override(5sec)

group_override

Purpose

Overrides group passwords, UNIX IDs, or member lists from the registry database.

Description

The `dcelocal/etc/group_override` administrative file lets you override the group UNIX ID and member list for a group in the network registry database. The `passwd_override` file serves a similar function for principals; see the `passwd_override(5sec)` reference page for more information.

Each host machine contains its own `group_override` file. Override entries contained in the file take effect transparently, but on the local machine only; they have no effect on the centralized registry. You might find `group_override` especially useful for overriding the default group definitions supplied with the registry if they do not match your local UNIX system’s group definitions.

Note: The `group_override` file will not work on a slim client configuration because the slim client configuration does not run with all the necessary daemons to support `group_override`.

The group_override File Format

The format of a `group_override` entry is similar to entries in the UNIX group file. The format is as follows:

```
group_name: passwd: group_uid: members
```

In an override entry, `group_name` and `group_uid` are keyfields. You must enter one of them to identify the group to which the override applies. The keyfield is used to perform a lookup in the override file. The lookup is performed in order as the fields are specified in an entry: first by group name, then by group UNIX ID. If you specify both keyfields in an override entry, the `group_name` keyfield is used as the lookup key; the `group_uid` key field is used as an override.

Field Descriptions

The fields contained in a `group_override` file are described in the following:

- `group_name`
  
  A keyfield that contains the name that identifies the group to which the override applies.

- `passwd`
  
  The encrypted password. If you specify an override in this field, the password you enter is in effect for the local machine only.

  You can also specify OMIT in the `passwd` field to disallow use of the `newgrp` command specifying this group on the local machine. The use of OMIT along with an option to the `passwd_export` command also omits this group from the group file created by `passwd_export`. (See the section Using OMIT for details.)

- `group_uid`

  A group UNIX ID. This field can function as a keyfield when the
group_name keyfield is not entered, or as a field specifying an override when entered in conjunction with group_name. When used in an override entry, this field specifies the ID to be used for the group on the local machine.

members
This field specifies a comma-separated list of members of the group. The contents of this field overrides information in the registry when passwd_export creates an /etc/group file. Note that to specify a null membership, as opposed to indicating that no override is required (see Leaving Fields Blank), you must specify a * (asterisk) in this field.

Leaving Fields Blank
If you do not want to override an item, leave its field blank, being sure to use a : (colon) to separate blank fields. (You must enter one of the keyfields, however, to identify the group for which you are creating overrides.) You are required to enter the colons associated with any blank trailing fields. Note that to override a group with a null membership list, you must enter an asterisk in the members field.

Using OMIT
If you specify OMIT and run passwd_export with the -x option, the named group or set of groups will not appear in the /ect/group file produced by passwd_export.

On a system with AIX/DCE integrated security features, the group_override file can also be used to prevent local access by DCE users. This is done by specifying a groupname and OMIT as a password in a group_override entry. Then, any user who is a member of that group is denied local access. For example, the entry

temps:OMIT::

will prevent any DCE user that belongs to group temps from logging onto the local system. It does not matter whether temps is the user’s primary group or an arbitrary group in the user’s group set. Access denial by the group_override file is in effect only for integrated BOS functions (not DCE only functions like dce_login) and denial is only on a group name basis. To deny access based on GID, use the passwd_override file mechanism.

Examples

1. To override the group ID of group kmem and change it to 3 on the local machine, include the following the entry in the group_override file:

   kmem::3:

2. To override the membership of group system so that it includes only the single account root on the local machine, include the following entry:

   system:::root

Related Information

Commands: dcecp(8sec), rgy_edit(8sec), passwd_export(8sec).

Files: passwd_override(5sec).
passwd_override

Purpose

Registry database user override file

Description

The dcelocal/etc/passwd_override administrative file lets you override the password, GECOS, home directory, login shell, group membership, and principal UNIX ID information stored in the network registry database. The group_override file serves a similar function for groups; see the group_override(5sec) reference page for more information.

Each host machine contains its own passwd_override file. Override entries contained in the file take effect transparently, but on the local machine only; they have no effect on the centralized registry. You might find passwd_override especially useful for excluding users from logging into certain machines, establishing local root passwords, or tailoring local user environments.

Note: The passwd_override file will not work on a slim client configuration because the slim client configuration does not run with all the necessary daemons to support passwd_override.

The passwd_override File Format

The format of a passwd_override entry is similar to entries in the UNIX password file. The format is as follows:

```
principal_name: passwd: principal_uid: group_id: GECOS: home_dir: login_shell
```

In an override entry, principal_name, principal_uid, and group_id fields are keyfields. You must enter one of them to identify the principal or group to which the overrides apply. The keyfield is used to perform a lookup in the override file. The lookup is performed in order as the entries are specified in an override entry: first by principal name, then by principal UNIX ID, and finally by group UNIX ID. If you specify more than one keyfield in an override entry, the first keyfield specified is used as the lookup key; subsequent keyfields are used as overrides.

Field Descriptions

The fields contained in a passwd_override file entry are described in the following:

principal_name

A keyfield that contains a principal name identifying the account to which the overrides apply. Enter principal_name to apply the override only to the account for the principal’s primary name and not to any accounts for the principal’s aliases.

 passwd

The encrypted password. If you specify an override in this field, the password you enter is in effect for this local machine only.

When you override a principal’s password, only the principal’s local credentials are obtained at login, not the principal’s network credentials. Without network credentials, the principal cannot access the network

Note: The passwd_override file will not work on a slim client configuration because the slim client configuration does not run with all the necessary daemons to support passwd_override.
registry and obtain the information normally provided at network login. Therefore, you must supply all this information in the password_override file entry. For overrides to passwords, you must enter all fields in the override entry, including all keyfields.

You can also specify OMIT in the passwd field to disallow login on the local machine. The use of OMIT in conjunction with an option to the passwd_export command also omits this principal from the password file created by passwd_export. (See the section Using OMIT for details.)

principal_uid
A UNIX user ID. This field can function as a keyfield (when the principal_name keyfield is not entered) or as an override field (when the principal_name keyfield is entered). Enter principal_uid and not principal_name when you want to apply the overrides to all of a principal’s accounts, including any accounts for the principal’s aliases. The principal_uid keyfield is especially useful for overrides to root. For example, if root has an alias of virtuoso, an override keyed by principal name applies only when root logs in as root. An override keyed by root’s principal_uid applies when root logs in as root, as virtuoso, and under any other alias.

Enter principal_uid and principal_name to override the UNIX ID of the named principal.

group_id
A UNIX group ID. This field can function as a keyfield, when no other keyfields are entered, or as a field containing an override, when entered in conjunction with principal_name or principal_uid.

Enter group_uid and no other keyfield (principal_name or principal_uid) to apply the override to all members of the group identified by group_uid. In this instance the group_uid field functions as a keyfield, identifying the accounts to which to apply the overrides (that is, accounts whose principal is a member of the specified group).

Enter group_uid and principal_name to change the group of the principal identified by principal_name to the group identified by group_uid. The change applies only to the account for the principal’s primary name, not to any accounts for the principal’s aliases. Enter group_uid and principal_uid to apply the group override to all of the principal’s accounts, including any for the principal’s aliases. In these instances the group_uid field functions as a field supplying override information, not as a keyfield.

GECOS
The account’s GECOS field. You can specify an override in this field. To keep it unchanged, leave it empty.

home_dir
The account’s home directory. You can specify an override in this field. To keep it unchanged, leave it empty.

login_shell
The account’s login shell. You can specify an override in this field. To keep it unchanged, leave it empty.

Leaving Fields Blank
If you do not want to override an item, leave its field blank, being sure to use a : (colon) to separate blank fields. (You must enter one of the keyfields; however, to
passwd_override(5sec)

identify the principal or group for which you are creating overrides.) You are required to enter the colons associated with any blank trailing fields.

Using OMIT

If you enter either the word OMIT or another invalid password string (such as * (asterisk) or NO GOOD) in the passwd field, the principal (or set of principals) will be unable to log into the local machine. If you specify OMIT and run passwd_export with the -x option, the named principal (or set of principals) will not appear in the /etc/passwd file produced by passwd_export.

You should also be aware that, if you have omitted principals from the /etc/passwd file, information about those principals will not be available to any programs that use the password file. For example, the ls -l and the finger commands both access the password file to obtain further information about a principals. If the principal is omitted, no password entry will exist and no information will be available. For this reason, you should use OMIT to omit principals from the /etc/passwd file only if your user community is very large and either of the following conditions occur:

1. The passwd file is taking up too much space.
2. User-ID-to-name mapping is too slow (during ls -l, for example).

Examples

1. To prevent the principal with a UNIX ID of 52 from logging into the local machine, include the following entry in the passwd_override file:

   :exclude:52:::

2. To prevent members of the group identified by a UNIX ID of 25 from logging into a node and to omit them from inclusion in the password file, put OMIT in the passwd field as follows:

   :OMIT::25:::

   Then run the passwd_export command with the -x option to omit these principals from the /etc/passwd file, as follows:

   dcelocal/etc/passwd_export
   -x

3. To change the password, home directory, and initial shell for user mozart’s account, include the following entry in the passwd_override file:

   mozart:sq1Rc1Urrb1L6:678:893:Wolfgang A. Mozart:/aria/wolfgang:/bin/csh

4. To override the home directory for user mozart’s account, include the following entry in the passwd_override file:

   mozart::::aria/wolfgang:

Related Information

Commands: dcecp(8sec), rgy_edit(8sec), passwd_export(8sec).

Files: group_override(5sec).
sec_audit_events

Purpose

Auditable events for the security services

Description

Code is in place for auditing security-significant events in the security server. Among these events are the following:

1. Attempts to invoke authentication server/ticket-granting server/privilege server (AS/TGS/PS) operations.
2. Deletion of security server objects, including the following:
   a. access control lists (ACLs)
   b. accounts
   c. pgo items
   d. registry properties
   e. registry/organization policies
   f. registry master key
3. Attempts to invoke an operation that modifies security server objects or updates an ACL.
4. Attempts to invoke operations that involve access control.
5. Failed client responses to the server’s challenge, detected replays and invalid ticket requests.
6. The use of cryptographic keys in the remote procedure call (RPC) runtime.
7. Attempts to change the maintenance/operation states of the registry server.

Event class definitions, together with filters, control the auditing execution at these code points. Filters can be updated dynamically. Filter files are maintained by a per-host audit daemon, and are shared among all the audit clients on the same host. The DCE control program, dcecp, is used to maintain the filters. (See the dcecp(8dce) reference page.) The dcecp command is executable by all users and system administrators. The ability to modify filters is controlled through the audit daemon’s ACL, which maintains the filters.

Security server RPC interfaces include krb5rpc, rdacilif, rdacilftpmp, rpriv, rs_acct, rs_query, rs_rpladmn, rs_update, and rsec_cert. All the RPC interfaces are offered using the rpc_c_authn_dce_secret authentication service. The security server’s RPC runtime uses dce-rgy as its authentication identity. Within the same process, the security server’s UDP/IP interface provides Kerberos AS/TGS functions, with krbtgt/cell_name as its authentication identity.

The following are the audit code points in these security service interfaces. Each entry shows the event type, followed by the event number and event classes, and then any event-specific information.

Authentication Interface (krb5rpc) Operations

The rsec_krb5rpc_sendto_kdc( ) function is an RPC interface operation for accessing Kerberos AS/TGS services. Ticket-granting tickets and application tickets
are requested and returned. There is no access control on this interface other than what is within the Kerberos ticket-granting mechanism itself; that is, the TGS request verification.

Generated in:

dce_login
dcecp: login
(any authentication)

Event Type (Event Number, Event Classes)
AS_Request (0x101, dce_sec_authent)

Event-specific information
Format ID: 2
char_string username

Note: username is the name under which the authentication attempt occurred. The username has the form /.../<cellname>/<userid>.

Event Type (Event Number, Event Classes)
TGS_TicketReq (0x102, dce_sec_authent)

Generated in:

process_tgs_req()
rsec_krb5rpc-sendto_kdc()
(kerberos ticket operations requiring authentication)

Event-specific information
Format ID: 2
char_string username

Note: username is the name under which the authentication attempt occurred. The username has the form /.../<cellname>/<userid>.

Event Type (Event Number, Event Classes)
TGS_RenewReq (0x103, dce_sec_authent)

Generated in:

process_tgs_req()
rsec_krb5rpc-sendto_kdc()
(kerberos ticket operations requiring authentication)

Event-specific information
Format ID: 2
char_string username

Note: username is the name under which the authentication attempt occurred. The username has the form /.../<cellname>/<userid>.

Event Type (Event Number, Event Classes)
TGS_ValidateReq (0x104, dce_sec_authent)

Generated in:
process_tgs_req()
sec_krb5rpc-sendto_kdc()
( kerberos ticket operations requiring authentication )

Event-specific information
Format ID: 2
char_string username

Note: username is the name under which the authentication attempt occurred. The username has the form /.../<cellname>/userid.

DACL Management Interface (rdaclif) Operations

The rdacl_lookup() operation retrieves an ACL of an object in the security server. Review of ACL associated with an object in security server is allowed if the caller has any access to the object.

Event Type (Event Number, Event Classes)
ACL_Lookup (0x105, dce_sec_control, dce_sec_query)

Generated in:

rdacl_lookup()
sec_acl_lookup()
any namespace operation
dcecp: acl (all commands)

Event-specific information
Format ID: 2
char_string entry_name
uuid manager_type
long_int acl_type
[ if (outcome != aud_c esl_cond_success) ]
ulong_int last_error_status
[ end if ]

The rdacl_replace() operation replaces the ACL of an object in the security server. The client must have the sec_acl_perm_owner permission for the update to be carried out.

Event Type (Event Number, Event Classes)
ACL_Replace (0x106, dce_sec_control, dce_sec_modify)

Generated in:

rdacl_replace()
sec_acl_replace()
dcecp: acl delete
dcecp: acl modify
dcecp: acl replace

Event-specific information
Format ID: 2
char_string entry_name
uuid manager_type
ulong_int acl_type (sec_acl_type_t )
acl (sec_acl_t)
[ if (outcome != aud_c esl_cond_success) ]
ulong_int last_error_status
[ end if ]
The `rdacl_get_access()` operation determines the caller’s access to a specified object. This call is authorized if the caller has any access to the object.

**Event Type (Event Number, Event Classes)**

`ACL_GetAccess (0x107, dce_sec_control, dce_sec_query)`

Generated in:

```c
rdacl_get_access()
sec_acl_get_access()
dcecp: acl_check
```

**Event-specific information**

Format ID: 2

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>char_string</td>
<td>entry_name</td>
<td></td>
</tr>
<tr>
<td>uuid</td>
<td>manager_type</td>
<td></td>
</tr>
<tr>
<td>ulong_int</td>
<td>last_error_status</td>
<td></td>
</tr>
</tbody>
</table>

The `rdacl_test_access()` operation determines if the caller has the requested access. The return value of the call indicates whether the caller has the requested access to the object.

**Event Type (Event Number, Event Classes)**

`ACL_TestAccess (0x108, dce_sec_control, dce_sec_query)`

Generated in:

```c
rdacl_test_access()
sec_acl_test_access()
dcecp: acl_check
```

**Event-specific information**

Format ID: 2

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>char_string</td>
<td>entry_name</td>
<td></td>
</tr>
<tr>
<td>uuid</td>
<td>manager_type</td>
<td></td>
</tr>
<tr>
<td>ulong_int</td>
<td>desired_permset</td>
<td></td>
</tr>
<tr>
<td>ulong_int</td>
<td>last_error_status</td>
<td></td>
</tr>
</tbody>
</table>

**Event Type (Event Number, Event Classes)**

`ACL_TestOnBehalf`

Generated in:

```c
rdacl_test_access_on_behalf()
sec_acl_test_access_on_behalf()
```

**Event-specific information**

Format ID: 1

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>char_string</td>
<td>entry_name</td>
<td></td>
</tr>
<tr>
<td>uuid</td>
<td>manager_type</td>
<td></td>
</tr>
<tr>
<td>ulong_int</td>
<td>desired_permset</td>
<td></td>
</tr>
<tr>
<td>pac</td>
<td>subject_pac</td>
<td></td>
</tr>
<tr>
<td>ulong_int</td>
<td>last_error_status</td>
<td></td>
</tr>
</tbody>
</table>
The `rdacl_get_manager_types()` operation lists the types (UUIDs) of ACLs protecting an object. The caller must have some permissions on the object for each of the manager types that is defined for the object. Otherwise, no manager type is returned.

**Event Type (Event Number, Event Classes)**

ACL_GetMgrTypes (0x10A, dce_sec_control, dce_sec_query)

**Generated in:**

- rdacl_get_manager_types()
- rdacl_get_mgr_types_semantics()
- dcecp: acl (all operations)

**Event-specific information**

Format ID: 2

- char_string entry_name
- ulong_int acl_type (sec_acl_type_t)
  
  [ if (outcome != aud_c_esl_cond_success) ]

- ulong_int last_error_status
  
  [ end if ]

The `rdacl_get_referral()` operation obtains a referral to an ACL update site. This function is used when the current ACL site yields a `sec_acl_site_readonly` error. Some replication managers will require all updates for a given object to be directed to a given replica. Clients of the generic ACL interface might know they are dealing with an object that is replicated in this way. This function allows them to recover from this problem and rebind to the proper update site. The client is required to have execute access on the parent of the object named by `component_name`.

**Event Type (Event Number, Event Classes)**

ACL_GetReferral (0x10B, dce_sec_control, dce_sec_query)

**Generated in:**

- rdacl_get_referral()
- rdacl_replace()
- sec_acl_replace
- dcecp: acl delete
- dcecp: acl modify
- dcecp: acl replace

**Event-specific information**

Format ID: 2

- char_string entry_name
- uuid manager_type
- ulong_int acl_type (sec_acl_type_t)
  
  [ if (outcome != aud_c_esl_cond_success) ]

- ulong_int last_error_status
  
  [ end if ]

**Privilege Server Interface (rpriv) Operations**

The `rpriv_get_ptgt()` operation returns a privilege certificate to the ticket-granting service. The caller supplies the group set, and the privilege server seals the group set in the authorization portion of a privilege ticket-granting ticket (TGT), after first
rejecting any groups that are not legitimately part of the caller credentials. A group will be rejected if the caller is not a member of the group, or the group is not allowed on project lists (the projlist_ok flag is not set).

There is no access control on this interface other than what was within the Kerberos ticket-granting mechanism itself; that is, the TGS request verification. This call might result in growth of potential access set. Note that this is a pre-DCE Version 1.1 routine.

Event Type (Event Number, Event Classes)
PRIV_GetPtgt (0x10C, dce_sec_authent, dce_sec_control)

Generated in:

rpriv_get_ptgt()
(privilege service requests)

Event-specific information
Format ID: 1
char_string client_address
short_int num_groups
[0 <= i < num_groups]
uuid_t groups(i)
[end iteration]

Note: num_groups is the number of local groups in the PAC.

Registry Server Account Interface (rs_acct) Operations

The rs_acct_add( ) operation adds an account with a specified login name. The caller needs m, a, and u (mgmt_info, auth_info, and user_info) permissions on the principal of the account that is to be added. The constituent principal, group, and organization (PGO) items for an account must be added before the account can be created. Also, the principal must have been added as a member of the specified group and organization.

Event Type (Event Number, Event Classes)
ACCT_Add (0x10D, dce_sec_control, dce_sec_modify)

Generated in:

rs_acct_add()
sec_rgy_acct_add()
dcecp: account create
dcecp: registry connect

Event-specific information
char_string login_name
[ if (outcome != aud_c_esl_cond_success)]
char_string primary_group_name
char_string primary_org_uuid
char_string key_parts
char_string gecos
char_string homedir
char_string shell
long_int expiration_date (sec_timeval_sec_t)
long_int good_since_date (sec_timeval_sec_t)
ulong_int user_flags (sec_rgy_acct_users_flags_t)
ulong_int admin_flags (sec_rgy_acct_users_flags_t)
ulong_int auth_flags (sec_rgy_acct_users_flags_t)
The rs_acct_delete( ) operation deletes an account with a specified login name. The caller must have m, a, and u (mgmt_info, auth_info, and user_info) permissions on the principal of the account that is to be deleted.

**Event Type (Event Number, Event Classes)**

ACCT_Delete (0x10E, dce_sec_control, dce_sec_modify)

Generated in:

- rs_acct_delete()
- sec_rgy_acct_delete()
- dcecp: account delete
- dcecp: principal delete
- dcecp: group delete
- dcecp: organization delete
- passwd_import

**Event-specific information**

Format ID: 2

- char_string login_name
- char_string primary_group_name
- char_string primary_org_name
- [ if (outcome == aud_c esl cond success) ]
  - char_string gecos
  - char_string homedir
  - char_string shell
- long_int creation_date (sec_timeval_sec_t)
- boolean foreign_creator
- uuid creatorUuid
  - [ if (foreign_creator) ]
    - uuid creatorCellUuid
      - [ (end if) ]
- long_int last_change_date (sec_timeval_sec_t)
- boolean foreign_changer
- uuid last_changerUuid
  - [ if (foreign_changer) ]
    - uuid last_changer_cellUuid
      - [ (end if) ]
- long_int expiration_date (sec_timeval_sec_t)
- long_int good_since_date (sec_timeval_sec_t)
- ulong_int user_flags (sec_rgy_acct_users_flags_t)
- ulong_int admin_flags (sec_rgy_acct_users_flags_t)
- ulong_int auth_flags (sec_rgy_acct_users_flags_t)
  - [ end if ]
    - [ if (outcome != aud c esl cond success) ]
      - ulong_int last_error_status
  - [ end if ]

**Note:** If the outcome is successful but the retrieval of the existing account information is unsuccessful, the error that occurred will be stored as the third and final event-specific item:

- long_int acct_retrieval_status

The rs_acct_rename( ) operation changes the account login name. The caller has to have the m (mgmt_info) permission on the account’s principal to be renamed (old_login_name.pname).
sec_audit_events(5sec)

Event Type (Event Number, Event Classes)
ACCT_Rename (0x10F, dce_sec_control, dce_sec_modify)

Generated in:

    rs_acct_rename()
    sec_rgy_acct_rename()

dcecp: account modify (changing primary group or organization)
dcecp: registry connect (if foreign cell account already exists)

Event-specific information
Format ID: 2
char_string login_name
char_string existing_group_name
char_string existing_org_name
long_int rejected_key_parts
[ if (outcome != aud_c_esl_cond_success) ]
ulong_int changes (aud_log_name_domain_t)
[ end if ]
[ if (changes & aud_l_name_person) ||
  (outcome != aud_c_esl_cond_success) ]
char_string princ_name
[ end if ]
[ if (changes & aud_l_name_group) ||
  (outcome != aud_c_esl_cond_success) ]
char_string group_name
[ end if ]
[ if (changes & aud_l_name_org) ||
  (outcome != aud_c_esl_cond_success) ]
char_string org_name
[ end if ]
[ if (outcome != aud_c_esl_cond_success) ]
ulong_int last_error_status
[ end if ]

The rs_acct_lookup( ) operation returns data for a specified account. The caller
must have the r (read) permission according to the ACL of the account’s principal in
order to be viewed.

Event Type (Event Number, Event Classes)
ACCT_Lookup (0x110, dce_sec_control, dce_sec_query)

Generated in:

    rs_acct_replace()
    sec_rgy_acct_lookup()

Event-specific information
Format ID: 2
char_string princ_login_name
char_string group_name
char_string org_name
ulong_int last_error_status
[ end if ]

The rs_acct_replace( ) operation replaces both the user and administrative
information in the account record specified by the input login name. The
administrative information contains limitations on the account’s use and privileges.
The user information contains such information as the account home directory and
default shell. The administrative information can only be modified by a caller with
the **a** (auth_info) privilege for the account’s principal. The user information can be modified by a caller with the **u** (user_info) privileges for the account’s principal.

**Event Type (Event Number, Event Classes)**

ACCT_Replace (0x111, dce_sec_control, dce_sec_modify)

**Generated in:**

- `rs_acct_replace()`
- `sec_rgy_acct_user_replace()`
- `sec_rgy_acct_admin_replace()`
- `sec_rgy_acct_replace_all()`
- `dce_login -n`
- integrated login: chpass
- `dced: key mgmt for host DCE keytab`
- `dcecp: registry connect`
- `dcecp: account modify`

**Event-specific information**

**Format ID: 2**

```plaintext
char_string princ_login_name
[ if (outcome == aud_c_es1_con_success) 
  [ if (changes & aud_l_acct_f_gecos) ]
  char_string gecos
  [ end if ]
[ if (changes & aud_l_acct_f_homedir) ]
  char_string homedir
  [ end if ]
[ if (changes & aud_l_acct_f_shell) ]
  char_string shell
  [ end if ]
[ if (changes & aud_l_acct_f_expires) ]
  long_int expires_date (sec_timeval_sec_t)
  [ end if ]
[ if (changes & aud_l_acct_f_good_since) ]
  long_int good_since_date (sec_timeval_sec_t)
  [ end if ]
[ if (changes & aud_l_acct_f_user_flags) ]
  ulong_int user_acct_flags (sec_rgy_acct_user_flags_t)
  [ end if ]
[ if (changes & aud_l_acct_f_admin_flags) ]
  admin_acct_flags (sec_rgy_acct_admin_flags_t)
  [ end if ]
[ if (changes & aud_l_acct_f_auth_flags) ]
  auth_acct_flags (sec_rgy_acct_auth_flags_t)
  [ end if ]
[ else ]
  char_string group_name
  char_string org_name
  long_int key_parts (sec_rgy_acct_key_t)
  ulong_int modify-parts
  [ if (modify_parts & rs_acct_part_user) ]
    char_string gecos
    [ end if ]
  [ if (modify_parts & rs_acct_part_user) ]
    char_string homedir
    [ end if ]
  [ if (modify_parts & rs_acct_part_user) ]
    char_string shell
    [ end if ]
  [ if (modify_parts & rs_acct_part_admin) ]
    long_int expires_date (sec_timeval_sec_t)
    [ end if ]
  [ if (modify_parts & rs_acct_part_admin) ]
    long_int good_since_date (sec_timeval_sec_t)
  [ end if ]
[ end if ]
```

**Chapter 5. Security Service Files and Commands**

621
The rs_acct_get_projlist() operation returns members of the project list for the specified account. This operation requires the caller to have the r (read) permission on the account principal for which the project list data is to be returned.

Event Type (Event Number, Event Classes)

ACCT_GetProjlist (0x112, dce_sec_control, dce_sec_query)

Event-specific information

<table>
<thead>
<tr>
<th>Format ID: 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>char_string</td>
</tr>
<tr>
<td>char_string</td>
</tr>
<tr>
<td>char_string</td>
</tr>
<tr>
<td>ulong_int</td>
</tr>
</tbody>
</table>

Registry Miscellaneous Operation Interface (rs_misc) Operations

The rs_login_get_info() operation returns login information for the specified account. This information is extracted from the account’s entry in the registry database. This operation requires the caller to have the r (read) permission on the account’s principal from which the data is to be returned.

Event Type (Event Number, Event Classes)

LOGIN_GetInfo (0x113, dce_sec_control, dce_sec_query)

Generated in:

rs_login_get_info()
sec_rgy_login_get_info()
sec_login_validate_identity()
sec_login_valid_and_cert_ident()
sec_login_get_pwent()
sec_login_get_groups
sec_login_valid_from_keytable()
dce_server_sec_begin()
dce_login [-c]
dcecp: login [-certify]
any DCE authentication

Event-specific information

<table>
<thead>
<tr>
<th>Format ID: 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>char_string</td>
</tr>
<tr>
<td>char_string</td>
</tr>
<tr>
<td>char_string</td>
</tr>
<tr>
<td>ulong_int</td>
</tr>
</tbody>
</table>
Registry PGO Interface (rs_pgo) Operations

The rs_pgo_add() operation adds a PGO item to the registry database. This operation requires the caller to have the i (insert) permission on the parent directory in which the PGO item is to be created.

**Event Type (Event Number, Event Classes)**

PGO_Add (0x114, dce_sec_control, dce_sec_modify)

**Generated in:**

- rs_pgo_add()
- sec_rgy_pgo_add()
- passwd_import
- dcecp: principal create
- dcecp: group create
- dcecp: organization create
- dcecp: registry create

**Event-specific information**

Format ID: 2

- ulong_int name_domain (sec_rgy_domain_t)
- char_string pgo_name
- [ if (outcome != aud_c_esl_cond_success) ]
- uuid pgo_uuid
- long_int pgo_unix_id
- long_int quota
- ulong_int flags (sec_rgy_pgo_flags_t)
- char_string fullname
- [ end if ]
- char_string princ_login_name
- [ if (outcome != aud_c_esl_cond_success) ]
- ulong_int last_error_status
- [ end if ]

The rs_pgo_delete() operation deletes a PGO item from registry database. Any account depending on the deleted PGO item is also deleted. The deletion operation requires the caller to have d (delete) permission on the parent directory that contains the PGO item to be deleted and the D (Delete_object) permission on the PGO item itself.

**Event Type (Event Number, Event Classes)**

PGO_Delete (0x115, dce_sec_control, dce_sec_modify)

**Generated in:**

- rs_pgo_delete()
- sec_rgy_pgo_delete()
- dcecp: principal delete
- dcecp: group delete
- dcecp: organization delete
- dcecp: registry connect

**Event-specific information**

Format ID: 2

- long_int name_domain (sec_rgy_domain_t)
- char_string pgo_name
- [ if (outcome == aud_c_esl_cond_success) ]
- uuid pgo_uuid
- long_int pgo_unix_id
- long_int quota
- long_int flags (sec_rgy_pgo_flags_t)
The `rs_pgo_replace()` operation replaces the data associated with a PGO item in the registry database. The caller needs to have the `m` (mgmt_info) permission on the PGO item, if `quota`, `flags`, or `unix_num` is being set. (Only a cell principal's `unix_num` is modifiable.) The caller needs to have the `f` (full name) permission to modify the full name of the PGO item.

**Event Type (Event Number, Event Classes)**

`PGO_Replace (0x116, dce_sec_control, dce_sec_modify)`

Generated in:

- `rs_pgo_replace()`
- `sec_rgy_pgo_replace()`
- `dcecp: principal modify`
- `dcecp: group modify`
- `dcecp: organization modify`

**Event-specific information**

- Format ID: 2

```plaintext
ulong_int name_domain (sec_rgy_domain_t)
char_string pgo_name
[ if (outcome == aud_c_esl_cond_success) ]
ulong_int changes (aud_log_pgo_field_t)
  [ if (changes & aud_l_pgo_f_uid) ]
  long_int unix_id
  [ end if ]
  [ if (changes & aud_l_pgo_f_quota) ]
  long_int object_creation_quota
  [ end if ]
  [ if (changes & aud_l_pgo_f_flags) ]
  long_int flags (sec_rgy_pgo_flags_t)
  [ end if ]
char_string fullname
[ end if ]
[ else ]
uuid pgo_uuid
long_int unix_id
long_int object_creation_quota
long_int flags (sec_rgy_pgo_flags_t)
char_string fullname
ulong_int last_error_status
[ end if ]
```

The `rs_pgo_rename()` operation renames a PGO item in the registry database. The caller needs to have the `n` (name) permission on the old name of the PGO item, if performing a rename within a directory. In order to move a PGO item between directories, the caller needs to have the `n` (name) permission on the old name of the PGO item as well as the `d` (delete) permission on the old parent directory and the `i` (insert) permission on the new parent directory in which the PGO item is being added under the new name.

**Event Type (Event Classes)**

`PGO_Rename (0x117, dce_sec_control, dce_sec_modify)`

Generated in:
The \texttt{rs\_pgo\_rename()} operation renames a PGO item. The desired item is identified by a query key, which can be a \texttt{name}, a \texttt{uuid}, a \texttt{unix\_num}, or a \texttt{sequential\_search} flag. The caller needs to have the \texttt{r (read)} permission on the PGO item to be viewed.

\textbf{Event Type (Event Number, Event Classes)}

\texttt{PGO\_Get (0x118, dce\_sec\_control, dce\_sec\_query)}

\textbf{Generated in:}

\begin{verbatim}
rs\_pgo\_get()
sec\_rgy\_pgo\_get\_by\_name()
sec\_rgy\_pgo\_get\_by\_id()
sec\_rgy\_pgo\_get\_by\_unix\_num()
sec\_rgy\_pgo\_get\_next()
sec\_rgy\_pgo\_get\_by\_eff\_unix\_num()
dcecp: principal \{ catalog | show | modify \}
dcecp: group \{ catalog | show | modify \}
dcecp: organization \{ catalog | show | modify \}
integrated login
password import
password export
\end{verbatim}

\textbf{Event-specific information}

\textbf{Format ID: 1}

\begin{verbatim}
long\_int \hspace{1em} name\_domain (sec\_rgy\_domain\_t)
long\_int \hspace{1em} key (rs\_pgo\_query\_t)
[ if (key == rs\_pgo\_query\_name) ]
char\_string \hspace{1em} pgo\_name
[ else if (key == rs\_pgo\_query\_id) ]
uuid \hspace{1em} pgo\_uuid
[ else if (key == rs\_pgo\_query\_unix\_num) ]
long\_int \hspace{1em} pgo\_unix\_num
[ else if (key == rs\_pgo\_query\_next) ]
char\_string \hspace{1em} scope
[ end if ]
[ if (outcome != aud\_c\_esl\_cond\_success) ]
boolean \hspace{1em} allow\_aliases
ulong\_int \hspace{1em} last\_error\_status
[ end if ]
\end{verbatim}

The \texttt{rs\_pgo\_key\_transfer()} operation performs a specified key transfer between the \texttt{uuid}, \texttt{unix\_num}, and \texttt{name} of a PGO item. The caller needs to have some permission on the PGO item for \texttt{id->name} and \texttt{unix\_num->name} transfers.

\textbf{Event Type (Event Number, Event Classes)}

\texttt{PGO\_KeyTransfer (0x119, dce\_sec\_control)}
The `rs_pgo_add_member()` operation adds a member to a group or an organization. The caller must have the `M` (Member_list) permission on the group or organization. Additionally, if this call is for adding a group member, the caller must have the `g` (groups) permission on the principal to be added.

Event Type (Event Number, Event Classes)

PGO_AddMember (0x11A, dce_sec_control, dce_sec_modify)

Generated in:

rs_pgo_add_member()
sec_rgy_pgo_add_member()
dcecp: group add
dcecp: organization add
dcecp: registry connect
passwd_import

Event-specific information

<table>
<thead>
<tr>
<th>Security Domain</th>
<th>Name Domain</th>
<th>Group or org's name</th>
</tr>
</thead>
<tbody>
<tr>
<td>name_domain</td>
<td>pgo_name</td>
<td>uuid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgo_unix_num</td>
</tr>
<tr>
<td></td>
<td></td>
<td>last_error_status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The `rs_pgo_delete_member()` operation deletes a principal from a group or an organization in the registry database. The caller must have the `M` (Member_list) permission on the group or organization. Note that the caller does not need to have the `g` (groups) permission when deleting the principal from a group.

Event Type (Event Number, Event Classes)

PGO_DeleteMember (0x11B, dce_sec_control, dce_sec_modify)

Generated in:
The `rs_pgo_is_member( )` operation tests whether a specified principal is a member of a specified group or organization. The caller must have `t` (test) permission on the group or organization.

**Event Type (Event Number, Event Classes)**

PGO_IsMember (0x11C, dce_sec_control, dce_sec_query)

**Generated in:**

- rs_pgo_is_member()
- sec_rgy_pgo_is_member()

**Event-specific information**

```
sec_rgy_domain_t name_domain
char *go_name /* Group or org's name */
char *person_name /* Principal's name */
    [ if (outcome != aud_c_esl_cond_success) ]
ulong_int last_error_status
    [ end if ]
```

The `rs_pgo_get_members( )` operation, if the specified domain is group or organization, lists the members of a specified group or organization. If the domain is principal, list the groups in which the principal is a member. The caller must have the `r` (read) permission on the principal, group, or organization.

**Event Type (Event Number, Event Classes)**

PGO_GetMembers (0x11D, dce_sec_control, dce_sec_query)

**Generated in:**

- rs_pgo_get_members()
- sec_rgy_pgo_get_members()
- dcecp: principal show
- dcecp: group list
- dcecp: organization list
- passwd_export
- integrated login

**Event-specific information**

```
sec_rgy_domain_t name_domain
char *go_name /* PGO's uuid */
    [ if (outcome != aud_c_esl_cond_success) ]
ulong_int last_error_status
    [ end if ]
```
Registry Policy Interface (rs_policy) Operations

The rs_properties_get_info() operation returns a list of registry properties. The caller must have the r (read) permission on the policy object from which the property information is to be returned.

Event Type (Event Number, Event Classes)
PROP_GetInfo (0x11E, dce_sec_control, dce_sec_query)

Generated in:

rs_properties_get_info()
sec_rgy_properties_get_info()
dce_aud_start()
dcecp: regisry connect
dcecp: registry show
dcecp: registry modify
passwd_export
passwd_import

Event-specific information

[ if (outcome != aud_c_esl_cond_success) ]
ulong_int last_error_status
[ end if ]

The rs_properties_set_info() operation sets the registry properties. The caller must have the m (mgmt_info) permission on the policy object for which the property information is to be set.

Event Type (Event Number, Event Classes)
PROP_SetInfo (0x11F, dce_sec_control, dce_sec_modify)

Generated in:

rs_properties_set_info()
sec_rgy_properties_set_info()
dcecp: registry modify

Event-specific information

Format ID: 2

[ if (outcome == aud_c_esl_cond_success) ]
ulong_int changes (aud_log_prop_field_t)
[ if (changes & aud_l_prop_f_read_ver) ]
long_int read_version
[ end if ]
[ if (changes & aud_l_prop_f_write_ver) ]
long_int write_version
[ end if ]
[ if (changes & aud_l_prop_f_min_tkt_life) ]
long_int minimum_ticket_lifetime (sec_timeval_sec_t)
[ end if ]
[ if (changes & aud_l_prop_f_def_cert_life) ]
long_int default_ticket_lifetime (sec_timeval_sec_t)
[ end if ]
[ if (changes & aud_l_prop_f_min_uid_princ) ]
long_int min_unix_id_principal
[ end if ]
[ if (changes & aud_l_prop_f_min_uid_group) ]
long_int min_unix_id_group
[ end if ]
[ if (changes & aud_l_prop_f_min_uid_org) ]
long_int min_unix_id_org
[ end if ]
The rs_policy_get_info() operation returns the policy for a specified organization or the registry (if no organization name is specified). The caller must have the r (read) permission on the policy object or organization item from which the data is to be returned. Note that the rs_policy_get_effective() operation uses the same audit event (POLICY_GetInfo) as the rs_policy_get_info() operation.

**Event Type (Event Number, Event Classes)**

**POLICY_GetInfo (0x120, dce_sec_control, dce_sec_query)**

Generated in:

- rs_policy_get_info()
- sec_rgy_policy_get_info()
- sec_rgy_plcy_get_effective()
- dcecp: organization { show | modify }
- dcecp: registry { show | modify }

**Event-specific information**

Format ID: 2

- char_string organization_name
- ulong_int last_error_status

The rs_policy_set_info() operation sets the policy for a specified organization or the registry (if no organization name is specified). The caller must have the m (mgmt_info) permission on the policy object or organization item for which the data is to be set.

**Event Type (Event Number, Event Classes)**

**POLICY_SetInfo (0x121, dce_sec_control, dce_sec_modify)**

Generated in:

Chapter 5. Security Service Files and Commands 629
sec_audit_events(5sec)

rs_policy_set_info()
sec_rgy_plcy_set_info()
dcecp: organization { show | modify }
dcecp: registry modify

Event-specific information
Format ID: 2

char_string organization_name
[ if (outcome == aud_c_esl_cond_success) ]
ulong_int changes (aud_log_plcy_field_t)
[ if (changes & aud_l_plcy_f_passwd_min_len) ]
long_int password_minimum_length
[ end if ]
[ if (changes & aud_l_plcy_f_passwd_life) ]
long_int password_lifetime (sec_timeval_period_t)
[ end if ]
[ if (changes & aud_l_plcy_f_passwd_exp) ]
long_int password_expiration_date (sec_timeval_sec_t)
[ end if ]
[ if (changes & aud_l_plcy_f_acct_life) ]
long_int account_lifespan (sec_timeval_period_t)
[ end if ]
[ if (changes & aud_l_plcy_f_passwd_flags) ]
ulong_int password_flags(sec_rgy_plcy_passwd_flags_t)
[ end if ]
[ else ]
long_int password_minimum_length
long_int password_lifetime (sec_timeval_period_t)
long_int password_expiration_date (sec_timeval_sec_t)
long_int account_lifespan (sec_timeval_period_t)
ulong_int password_flags(sec_rgy_plcy_passwd_flags_t)
ulong_int last_error_status
[ end if ]

The rs_auth_policy_get_info() operation returns the authentication policy for a specified account or the registry (if no account is specified). The caller must have the r (read) permission on the policy object or account's principal from which the data is to be returned.

Event Type (Event Number, Event Classes)

AUTHPOLICY_GetInfo (0x122, dce_sec_control, dce_sec_query)

Generated in:

rs_auth_policy_get_info()
rs_auth_policy_get_effective()
sec_auth_policy_get_info()
sec_auth_policy_get_effective()
sec_key_mgmt_change_key()
sec_key_mgmt_generate_key()
sec_key_mgmt_mange_key()
sec_key_mgmt_garbage_collect()
dced_keytab_change_key()
dcecp: account { show | modify }
dcecp: registry { show | modify }
integrated login (authentication)
security client (key maintenance for DCE host keys)

Event-specific information
Format ID: 2
The `rs_auth_policy_get_effective` operation returns the effective authentication policy for an account. If no account is specified, the authentication policy for the registry is returned. The caller must have `r` (read) permission on the policy object of the registry. If an account is specified, the caller must also have `r` (read) permission on the account's principal.

**Event Type (Event Number, Event Classes)**

No new event is defined for this operation. `AUTHPOLICY_GetInfo` is used here.

The `rs_auth_policy_set_info` operation sets the authentication policy for an account or the registry (if no account is specified). The caller must have `a` (auth_info) permission on the account's principal or policy object of the registry.

**Event Type (Event Number, Event Classes)**

`AUTHPOLICY_SetInfo` (0x123, dce_sec_control, dce_sec_modify)

**Generated in:**

- `rs_auth_policy_set_info`
- `sec_rgy_auth_pcly_set_info`
- `dcecp: account { create | modify }
- `dcecp: registry modify`

**Event-specific information**

Format ID: 2

```c
char_string principal_name
    [ if (outcome != aud_c ESL_cond_success) ]
char_string group_name
char_string org_name
ulong_int last_error_status
    [ end if ]
```

**Note:** If no account name is given to the rs-layer functions that generate this event, the account name of the local cell's registry (krbtgt/<cell_name>) is used.

**Registry Administration Interface Operations**

The `rs_rep_admin_stop` operation directs the registry server to stop servicing remote procedure calls. The caller must have `A` (Admin) permission on the registry policy object.
Event Type (Event Number, Event Classes)
REPADMIN_Stop (0x124, dce_sec_control, dce_sec_server)
Generated in:

rs_repadmin_stop()
sec_rgy_rep_admin_stop()
dcecp: registry stop

Event-specific information

ulong_int last_error_status
[ if (outcome != aud_c_esl_cond_success) ]
[ end if ]

The rs_rep_admin_maint() operation directs the registry server into (checkpoint the database, close files, and so on) or out of maintenance state. The caller must have A (Admin) permission on the registry policy object.

Event Type (Event Number, Event Classes)
REPADMIN_Maint (0x125, dce_sec_control, dce_sec_server)
Generated in:

rs_repadmin_maint()
sec_rgy_rep_admin_maint()
dcecp: registry { disable | enable }

Event-specific information

boolean in_maintenance
ulong_int last_error_status
[ if (outcome != aud_c_esl_cond_success) ]
[ end if ]

The rs_rep_admin_mkey() operation directs the registry to change its master key and reencrypt account keys using the new master key. The caller must have A (Admin) permission on the registry policy object.

Event Type (Event Number, Event Classes)
REPADMIN_Mkey (0x126, dce_sec_control, dce_sec_server)
Generated in:

rs_repadmin_mkey()
sec_rgy_rep_admin_mkey()
dcecp: registry modify -key

Event-specific information

ulong_int last_error_status
[ if (outcome != aud_c_esl_cond_success) ]
[ end if ]

The rs_rep_admin_destroy() operation directs the registry server replica to destroy its database and exit. The caller must have A (Admin) permission on the registry policy object.

Event Type (Event Classes)
REPADMIN_Destroy (0x127, dce_sec_control, dce_sec_server)
Generated in:
The rs_rep_admin_init_replica( ) operation directs the registry server to (re)initialize the slave identified by rep_id. This is a master server only operation. The caller must have A (Admin) permission on the registry policy object.

Event Type (Event Classes)
REPADMIN_Init (0x128, dce_sec_control, dce_sec_server)
Generated in:
rs_rep_admin_init_replica()
sec_rgy_rep_admin_init_replica()
dcecp: registry synchronize

Event-specific information
Format ID: 2
uuid replica_uuid

The rs_rep_admin_set_sw_rev( ) operation directs the registry server to change its software version. The caller must have A (Admin) permission on the registry policy object.

Event Type (Event Number, Event Classes)
REPADMIN_SetSwRev(0x013A, dce_sec_control, dce_sec_server)
Event-specific information
unsigned32 sw_rev

Registry Server Attributes Manipulation Interface (rs_attr)
Operations
The rs_attr_update( ) operation updates (writes/creates) an attribute. The caller must have, for each attribute defined in attr_keys, the q (query_permset) permission on the registry object specified.

Event Type (Event Classes)
ERA_Update (0x12B, dce_sec_control, dce_sec_modify)
Generated in:
rs_attr_update()
sec_rgy_attr_update()
dcecp: { principal | group | organization } create -attribute
dcecp: { principal | group | organization } modify -{ add | change | remove }
dcecp: account { create | modify } —pkmechanism
The `rs_attr_delete()` operation deletes specified attributes. The caller must have `delete_permset` permission for each attribute specified.

**Event Type (Event Classes)**

**ERA_Delete (0x12C, dce_sec_control, dce_sec_modify)**

**Generated in:**

- `rs_attr_delete()`
- `sec_rgy_attr_delete()`
- `dcecp: { principal | group | organization } modify -remove`

**Event-specific information**

Format ID: 2

```c
char_string component_name
ulong_int num_to_delete
  [ if (outcome == aud_c esl_cond_success) ]
  [ if (next item type is uhyper_int) ]
  [ if (outcome == aud_c esl_cond_success) ]
  [ if (next item type is ulong_int) ]
ulong_int lookup_error_status
  [ 1 <= i <= num_to_write ]
ulong_int attr_uuid(i)
  [ end iteration ]
  [ else ]
  [ if (next item type is uuid) ]
ulong_int new_attr_uuid(i)
  [ else ]
ulong_int attr uuid(i)
  [ end if ]
  [ else ]
  [ if (next item type is uhyper_int) ]
ulong_int existing_attr_sentinel(i)
  [ end iteration ]
ulong_int old_attr(i)
  [ else ]
ulong_int failure_index
ulong_int last_error_status
  [ end if ]
```

The `rs_attr_delete()` operation deletes specified attributes. The caller must have `delete_permset` permission for each attribute specified.
The rs_attr_lookup_by_id( ) operation performs a lookup of the attributes by attribute type ID. If the number of query attribute keys is 0, this operation will return all attributes that the caller is authorized to use. The caller must have, for each attribute specified, the q (query_permset) permission on the registry object specified.

Event Type (Event Classes)

ERA_LookupById (0x12E, dce_sec_control)

Generated in:

rs_attr_lookup_by_id()
sec_rgy_attr_lookup_by_id()
integrated login
dcecp: { principal | group | organization } show -{ xattr | all }
dcecp: { principal | group | organization } modify -{ add | change | remove }
dcecp: account { create | modify } -(pkmechanism | pkkeycipherusage | pksignatureusage)
password strength (changing a password)
dce_login (public key)

Event-specific information

cchar_string global_component_name
ulong_int num_attrs
[ 1 <= i <= num_attrs ]
[ if (next item type is uuid) ]
ulong_int attr_uuid(i)
[ else ]
ulong_int attr(i)
[ end if ]
[ end iteration ]

The rs_attr_lookup_no_expand( ) operation performs a lookup of the attributes by attribute type ID without expanding attribute sets to their constituent member attributes. If the number of query attribute keys is 0, this operation will return all attributes that the caller is authorized to use. The caller must have, for each attribute specified, the q (query_permset) permission on the registry object specified.

Event Type (Event Classes)

ERA_LookupNoExpand (0x12F, dce_sec_control)

Generated in:
The rs_attr_lookup_by_name() operation performs a lookup of an attribute by name. The caller must have, for the attribute specified, a \(q\) (query_permset) permission on the registry object specified.

**Event Type (Event Classes)**

ERA_LookupByName (0x130, dce_sec_control)

Generated in:

- rs_attr_lookup_by_name()
- sec_rgy_attr_lookup_by_name()
- dce_login
- password strength operations
- dcecp: account generate
- sec_attr_trig_query()

**Event-specific information**

<table>
<thead>
<tr>
<th>Format ID: 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>char_string component_name</td>
</tr>
<tr>
<td>char * attr_name</td>
</tr>
<tr>
<td>ulong_int last_error_status</td>
</tr>
</tbody>
</table>

Registry Server Attributes Schema Manipulation Interface (rs_attr_schema) Operations

The rs_attr_schema_create_entry() operation creates a new schema entry. The caller must be authorized to add entries to the specified schema.

**Event Type (Event Classes)**

ERA_SchemaCreate (0x131, dce_sec_control, dce_sec_modify)

Generated in:

- rs_attr_schema_create_entry()
- sec_rgy_attr_schema_create_entry()
- dcecp: xattrschema create

**Event-specific information**

<table>
<thead>
<tr>
<th>Format ID: 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>char_string schema_name</td>
</tr>
<tr>
<td>uuid schema_entry_uuid</td>
</tr>
<tr>
<td>[ if (outcome != aud_c_esl_cond_success) ]</td>
</tr>
<tr>
<td>char_string schema_entry_name</td>
</tr>
<tr>
<td>long_int encoding_type (sec_attr_encoding_t)</td>
</tr>
</tbody>
</table>
The rs_attr_schema_delete_entry( ) operation deletes a schema entry. The caller must be authorized to delete schema entries.

Event Type (Event Classes)
ERA_SchemaDelete (0x132, dce_sec_control, dce_sec_modify)

Generated in:
rs_attr_schema_delete_entry()
sec_rgy_attr_sch_delete_entry()
dcecp: xattrschema delete

Event-specific information
Format ID: 2

char_string  schema_name
uuid         schema_entry_uuid
            [ if (outcome == aud_c_esl_cond_success) ]
char_string  schema_entry_name
long_int     encoding_type (sec_attr_encoding_t)
ulong_int    num_acl_mgrs
            [ 1 <= i <= num_acl_mgrs ]
uuid         acl_mgr_uuid(i)
ulong_int    query_permset(i) (sec_acl_permset_t)
ulong_int    update_permset(i) (sec_acl_permset_t)
ulong_int    test_permset(i) (sec_acl_permset_t)
ulong_int    delete_permset(i) (sec_acl_permset_t)
[ end iteration]
ulong_int    schema_entry_flags (sec_attr_sch_entry_flags_t)
long_int     intercell_action (sec_attr_intercell_action_t)
ulong_int    trig_types (sec_attr_trig_types_flags_t)
            [ if (trig_types != sec_attr_trig_type_none) ]
long_int     info_type (sec_attr_bind_auth_info_type_t)
            [ if (info_type == sec_attr_bind_auth_dce) ]
char_string  server_princ_name
ulong_int    protect_level
ulong_int    authn_svc
ulong_int    authz_svc
            [ end if ]
ulong_int    num_bindings
            [ 1 <= j <= num_bindings ]
long_int     bind_type (j) (sec_attr_bind_type_t)
            [ if (bind_type(j) == sec_attr_bind_type_string) ]
char_string  string_binding(j)
            [ else if (bind_type(j) == sec_attr_bind_type_twrs) ]
ulong_int    count(j)
            [ 1 <= k <= count(j) ]
byte_array   tower_octet_string (j,k)
            [ end iteration ]
            [ else if (bind_type(j) == sec_attr_bind_type_svrname) ]
ulong_int    name_syntax(j)
char_string  name(j)
            [ end if ]
            [ end iteration ]
            [ end if ]
char_string  scope
char_string  comment
ulong_int    last_error_status
[ end if ]
ulong_int schema_entry_flags (sec_attr_sch_entry_flags_t)
long_int intercell_action (sec_attr_intercell_action_t)
ulong_int trig_types (sec_attr_trig_type_flags_t)
    [ if (trig_types != sec_attr_trig_type_none) ]
long_int info_type (sec_attr_bind_auth_info_type_t)
    [ if (info_type == sec_attr_bind_auth_dce) ]
char_string server_princ_name
ulong_int protect_level
ulong_int authn_svc
ulong_int authz_svc
    [ end if ]
ulong_int num_bindings
    [ 1 <= j <= num_bindings ]
long_int bind_type(j) (sec_attr_bind_type_t)
    [ if (bind_type(j) == sec_attr_bind_type_string) ]
char_string string_binding(j)
    [ else if (bind_type(j) == sec_attr_bind_type_twrs) ]
ulong_int count(j)
    [ 1 <= k <= count(j) ]
byte_array tower_octet_string(j,k)
    [ end iteration ]
    [ else if (bind_type(j) == sec_attr_bind_type_svrname) ]
ulong_int name_syntax(j)
char_string name(j)
    [ end if ]
    [ end iteration ]
    [ end if ]
char_string scope
char_string comment
    [ else ]
ulong_int last_error_status
    [ end if ]

The rs_attr_schema_update_entry() operation updates the modifiable fields of a
schema entry. The caller needs to have m(mgmt_info) permissions on the schema
entry that is to be modified.

Event Type (Event Classes)
ERA_SchemaUpdate (0x133, dce_sec_control, dce_sec_modify)

Generated in:
rs_attr_schema_update_entry()
sec_rgy_attr_sch_update_entry()
dcecp: xattrschema { modify | rename }

Event-specific information
Format ID: 2
char_string schema_name
uuid schema_uuid
    [ if (outcome == aud_c_es1_cond_success) &&
        (next item type is ulong_int) ]
ulong_int changes (sec_attr_schema_entry_parts_t)
    [ if (changes & sec_attr_schema_part_name) ]
char_string schema_entry_name
    [ end if ]
    [ if (changes & sec_attr_schema_part_acl_mgrs) ]
ulong_int num_new_acl_mgrs
    [ 1 <= i <= num_new_acl_mgrs ]
uuid new_acl_mgr_uuid(i)
ulong_int query_permset(i) (sec_acl_permset_t)
ulong_int update_permset(i) (sec_acl_permset_t)
ulong_int test_permset(i) (sec_acl_permset_t)
ulong_int delete_permset(i) (sec_acl_permset_t)
sec_audit_events(5sec)

[ end iteration ]
[ end if ]
[ if (changes & sec_attr_schema_part_reserved) ||
  (changes & sec_attr_schema_part_defaults) ]
ulong_int sch_entry_flags (sec_attr_sch_entry_flags_t)
[ end if ]
[ if (changes & sec_attr_schema_part_intercell) ]
ulong_int intercell_action (sec_attr_intercell_action_t)
[ end if ]
[ if (changes & sec_attr_schema_part_trig_bind) ]
long_int info_type (sec_attr_bind_auth_info_type_t)
[ if (info_type == sec_attr_bind_auth_dce) ]
char_string server_princ_name
ulong_int protect_level
ulong_int authn_svc
ulong_int authz_svc
[ end if ]
[ end iteration ]
ulong_int num_bindings
[ 1 <= j <= num_bindings ]
long_int bind_type(j) (sec_attr_bind_type_t)
[ if (bind_type(j) == sec_attr_bind_type_string) ]
char_string string_binding(j)
[ else if (bind_type(j) == sec_attr_bind_type_twrs) ]
ulong_int count(j)
[ 1 <= k <= count(j) ]
byte_array tower_octet_string(j,k)
[ end iteration ]
[ else if (bind_type(j) == sec_attr_bind_type_srvname) ]
ulong_int name_syntax(j)
char_string name(j)
[ end if ]
[ end iteration ]
[ else if (changes & sec_attr_schema_part_scope) ]
char_string scope
[ end if ]
[ else if (changes & sec_attr_schema_part_comment) ]
char_string comment
[ end if ]
[ else if (outcome != aud_c_esl_cond_success) &&
  (next item type == uhyper_int] 
ulhyper_int lookup_error_status
[ else if (outcome != aud_c_esl_cond_success) ]
char_string schema_entry_name
long_int encoding_type (sec_attr_encoding_t)
ulong_int num_acl_mgrs
[ 1 <= i <= num_acl_mgrs ]
uuid acl_mgr_uuid(i)
ulong_int query_permset(i) (sec_acl_permset_t)
ulong_int update_permset(i) (sec_acl_permset_t)
ulong_int test_permset(i) (sec_acl_permset_t)
ulong_int delete_permset(i) (sec_acl_permset_t)
[ end iteration ]
ulong_int schema_entry_flags (sec_attr_sch_entry_flags_t)
long_int intercell_action (sec_attr_intercell_action_t)
ulong_int trig_types (sec_attr_trig_type_flags_t)
[ if (trig_types != sec_attr_trig_type_none) ]
long_int info_type (sec_attr_bind_auth_info_type_t)
[ if (info_type == sec_attr_bind_auth_dce) ]
char_string server_princ_name
ulong_int protect_level
ulong_int authn_svc
ulong_int authz_svc
[ end if ]
ulong_int num_bindings
[ 1 <= j <= numBindings ]
long_int bind_type(j) (sec_attr_bind_type_t)
[ if (bind_type(j) == sec_attr_bind_type_string) ]
char_string string_binding(j)
The rs_attr_schema_lookup_by_id( ) operation retrieves the schema entry identified by the attribute type uuid. The caller must have r (read) permissions on the schema entry specified.

**Event Type (Event Classes)**

ERA_SchemaLookupId (0x134, dce_sec_control)

**Generated in:**

rs_attr_schema_lookup_by_id()
sec_attr_sch_lookup_by_id()
dcecp: { principal | group | organization } show -{ all | xattrs }
dcecp: { principal | group | organization } modify -change
dcecp: server modify

dcecp: { principal | group | organization } create -attributes
dcecp: xattrschema { delete | rename | modify } \\
<local_cell>/sec/xattrschema/<schema_entry_name>
dcecp: xattrschema modify -aclmgr
password strength (checking)

event-specific information

The rs_attr_schema_lookup_by_name( ) operation retrieves the schema entry identified by the attribute name. The caller must have r (read) permissions on the schema entry specified.

**Event Type (Event Classes)**

ERA_SchemaLookupName (0x135, dce_sec_control)

**Generated in:**

rs_attr_schema_lookup_by_name()
sec_attr_sch_lookup_by_name()
dcecp: account { create | modify } -pkmechanism
dcecp: { principal | group | organization } create -attributes
dcecp: { principal | group | organization } modify -change
dcecp: xattrschema { delete | rename | modify } \\
<local_cell>/sec/xattrschema/<schema_entry_name>
dcecp: xattrschema modify -aclmgr
password strength (checking)

event-specific information
Version 1.1 Privilege Server Manager Interface (rpriv_v1_1)

Operations

The rpriv_get_eptgt() operation constructs and returns an extended privilege certificate to the ticket granting service. The caller supplies the extended privilege attributes in the form of an encoded extended privilege attribute certificate (EPAC). The procedure by which the requested privilege attributes are verified depends on how the call is authenticated and whether the request is local (that is, is a request from a client in this privilege server’s cell) or is intercell (that is, is from a foreign privilege service).

If the request is local, then the ticket to the privilege server is based on a Kerberos V5 TGT and the requested_privs consists of a single encoded EPAC. The privilege server decodes the requested_privs and verifies that the requested privileges are valid by performing the necessary database queries.

If the request is foreign, then the ticket to the privilege service is based on a DCE extended privilege TGT and the privilege server retrieves the EPAC seal from the DCE authorization data contained in the ticket, and uses it to verify that the requested privileges are valid.

Event Type (Event Classes)

PRIV_GetEptgt (0x136, dce_sec_control, dce_sec_authent)

Generated in:

rpriv_get_eptgt()
sec_login_cred_get_initiator()
sec_login_become_initiator()
sec_login_cred_get_delegate()
sec_login_become_initiator()
sec_login_inquire_net_info()
gssdce_login_context_to_cred()
gss_acquire_cred()
gss_accept_sec_context()
gss_init_sec_context()
gss_inquire_cred()

event-specific information

Format ID: 2

boolean local_request
  [ if (local_request) ]

uuid req_princ_uuid

uuid req_primary_group_uuid

ushort_int num_local_groups
  [ 0 <= i < num_local_groups ]

uuid local_group_uuid(i)
  [ end iteration ]

[ else if (!local_request) ]

ushort_int num_epacs_in_chain
  [ 0 <= j < num_epacs_in_chain ]

uuid PAC_realm_uuid(j)

uuid PAC_prince_uuid(j)

ushort_int num_groups_in_PAC(j)
  [ 0 <= k < num_groups_in_PAC ]
The `rpriv_become_delegate()` operation permits an intermediate server to become a delegate for its caller. The caller supplies extended privilege attributes in the form of an encoded EPAC. The privilege server verifies that the delegation token for this EPAC chain is correct and then creates a new chain from the existing one with the intermediary’s EPAC as a new delegate.

**Event Type (Event Classes)**

PRIV_BecomeDelegate (0x138, dce_sec_control, dce_sec_authent)

**Generated in:**

- `rpriv_become_delegate()`
- `sec_login_become_delegate()`
- `gss_accept_sec_context()`

**Event-specific information**

- `uuid req_princ_id`
- `uuid req_group_id`
- `ushort_int num_local_groups`
  - `[1 <= i <= num_local_groups ]`  
  - `uuid groups_uuid(i)`
  - `[ end iteration ]`
- `ushort_int num_epacs`
  - `[ 1 <= j <= num_epacs ]`
  - `uuid pa_realm_uuid(j)`
  - `uuid pa_princ_uuid(j)`
  - `uuid pa_num_groups(j)`
    - `[ 1 <= k <= pa_num_groups(j) ]`
    - `uuid pa_group_uuid(j,k)`
    - `[ end iteration ]`

The `rpriv_become_impersonator()` operation permits an intermediate server to become an impersonator for its caller. The caller supplies extended privilege attributes in the form of an encoded EPAC. The privilege server verifies that the delegation token for the initiator’s EPAC is correct and also that the intermediary is allowed to impersonate the initiator.

**Event Type (Event Classes)**

PRIV_BecomeImpersonator (0x139, dce_sec_control, dce_sec_authent)

**Generated in:**

- `rpriv_become_impersonator()`
- `sec_login_become_impersonator()`
- `gss_accept_sec_context()`

**Event-specific information**

- `uuid req_princ_id`
- `uuid req_group_id`
- `ushort_int num_local_groups`
  - `[1 <= i <= num_local_groups ]`  
  - `uuid groups_uuid(i)`
  - `[ end iteration ]`
- `ushort_int num_epacs`
  - `[ 1 <= j <= num_epacs ]`
  - `uuid pa_realm_uuid(j)`
  - `uuid pa_princ_uuid(j)`
Password Strength Operations

The password history database is read every time a password is checked against the strength rules involving old passwords.

**Event Type (Event Classes)**

HIS\_dbRead (0x154, dce\_pwd\_strength\_check)

**Event-specific information**

Format ID: 1

char\_string history\_database
char\_string user

[if (outcome != aud\_c\_esl\_cond\_success) ]
ulong\_int last\_error\_status
[ end if ]

The password history database is updated every time a password is successfully updated.

**Event Type (Event Classes)**

HIS\_dbWrite (0x155, dce\_pwd\_strength\_check)

**Event-specific information**

Format ID: 1

char\_string history\_database
char\_string user

[if (outcome != aud\_c\_esl\_cond\_success) ]
ulong\_int last\_error\_status
[ end if ]

The password history database is opened and closed every time a password is checked against the strength rules.

**Event Type (Event Classes)**

HIS\_dbOpen (0x153, dce\_pwd\_strength\_check)

**Event-specific information**

Format ID: 1

char\_string history\_file

[if (outcome != aud\_c\_esl\_cond\_success) ]
ulong\_int last\_error\_status
[ end if ]

**Event Type (Event Classes)**

HIS\_dbClose (0x156, dce\_pwd\_strength\_check)

**Event-specific information**

Format ID: 1

char\_string history\_file

[if (outcome != aud\_c\_esl\_cond\_success) ]
ulong\_int last\_error\_status
[ end if ]

Dictionary files are opened and closed every time passwords are checked against strength rules that disallow the use of commonly used words as passwords.
Event Type (Event Classes)
DIC_FileOpen (0x157, dce_pwd_strength_check)
Event-specific information
Format ID: 1
char_string dictionary_filename
ulong_int error
[ if (outcome != aud_c_esl_cond_success) ]
[ end if ]

Event Type (Event Classes)
DIC_FileClose (0x158, dce_pwd_strength_check)
Event-specific information
Format ID: 1
char_string dictionary_filename
ulong_int error
[ if (outcome != aud_c_esl_cond_success) ]
[ end if ]

User-defined libraries are loaded and unloaded every time a password is checked against strength rules that allow DCE administrators to have their own software to ensure strong passwords.

Event Type (Event Classes)
UDL_LibLoad (0x159, dce_pwd_strength_check)
Event-specific information
Format ID: 1
char_string library_path
ulong_int integer_error
char_string string_error
[ if (outcome != aud_c_esl_cond_success) ]
[ end if ]

Event Type (Event Classes)
UDL_LibUnload (0x15A, dce_pwd_strength_check)
Event-specific information
Format ID: 1
char_string library_path
ulong_int integer_error
char_string string_error
[ if (outcome != aud_c_esl_cond_success) ]
[ end if ]

Event Type (Event Classes)
PWS_ServerStopped (0x151, dce_pws_server)
Event-specific information
Format ID: 1

Event Type (Event Classes)
PWS_ServerStarted (0x152, dce_pws_server)
Event-specific information
Format ID: 1

Related Information

Commands: dcecp(8dce).

Files: dts_audit_events(5sec), event_class.5sec.
v5srvtab

Purpose

Server and machine keytab file

Description

The /krb5/v5srvtab file is a file on the local node created by the rgy_edit command, the sec_create_db command, or any application that makes sec_key_mgmt( ) calls. The file contains passwords for servers and machine accounts. To view or manipulate the contents of this file, use the sec_key_mgmt API, described in the IBM DCE Version 3.1 for AIX and Solaris: Application Development Guide—Core Components and the sec_key_mgmt_ *(3sec) reference pages. You can also view and manipulate the contents of this file using the dcecp keytab create, keytab show, and keytab delete commands.

Note: This file is read-writable only by the local root user.

Related Information

Commands: rgy_edit(8sec).

sec_intro

Purpose

Introduction to DCE Security Service administrative commands

Description

The *(8sec)* reference pages describe the DCE Security Service commands for system administration. These commands are as follows:

**acl_edit**
Manages access control lists (ACLs) for DCE objects.

**auditd**
Starts the DCE audit daemon.

**dce_login**
Validates a principal’s identity and obtains a principal’s network credentials.

**dceunixd**
Acts as an intermediary between base operating system security commands and the DCE security services.

**kdestroy**
Destroys your login context and credentials.

**kinit**
Obtains and caches a ticket granting ticket.

**klist**
Lists cached tickets.

**k5dcelogin**
Promotes a Kerberos V5 credential to a DCE credential so that the credential can be used to access DCE objects.

**passwd_export**
Updates local password and group files from DCE registry data.

**passwd_import**
Creates DCE registry entries based on password and group file entries.

**pwd_strengthd**
Sample password management server.

**rgy_edit**
Edits the registry database.

**sec_admin**
Administers the security server.

**sec_create_db**
Creates registry databases.

**secd**
The security server daemon.

See each command’s reference page for further information.

Related Information

Commands: acl_edit(8sec), auditd(8sec), dce_login(8sec), dceunixd(8sec), kdestroy(8sec), kinit(8sec), klist(8sec), passwd_export(8sec), passwd_import(8sec), pwd_strengthd(8sec), rgy_edit(8sec), sec_admin(8sec), sec_create_db(8sec), sec_intro(8sec), secd(8sec), .
acl_edit

Purpose

Edits or lists an object's ACLs

Synopsis

```
ACL EDIT [-e]pathname | [-addr string_binding component_name ] [-ic | -io] [-n | -c]
[command_line_subcommands ] [-ngui | -v]
```

Options

- **-e pathname**
  Specifies that the access control list (ACL) on the directory service entry is to be edited. You must specify the `pathname` argument if you use the `-e` option.

  The `-e` option is especially useful in case of an ambiguous `pathname`. The `pathname` argument can be interpreted in two ways if it is the name of a leaf object in the directory service (that is, if it is not the name of a directory). It can be interpreted as the directory service entry itself, or as the object (whatever it is) referred to by that directory service entry. When such a `pathname` is specified, the `-e` option directs `acl_edit` to the ACL on the directory service entry. However, the `-e` option can always be used, provided only that `pathname` specifies a leaf object in the Directory Service.

- **-addr string_binding component_name**
  The `-addr` option lets you identify the object whose ACLs you want to edit by supplying the remote procedure call (RPC) binding handle of the ACL manager that controls access to the object (with the `string_binding` argument) and the relative pathname of the object (with the `component_name` argument). Because you have identified the RPC binding handle, you can specify only the object’s relative pathname for `component_name`.

  The most common way to identify the object whose ACLs you want to manipulate is through the `pathname` argument, described below. The `-addr` option is used primarily by applications that do not use the directory service, but do use the generic ACL manager. It can also be used if the directory service is unavailable.

- **-ic**
  For container objects only, specifies that the object's Initial Container Creation ACL is to be edited. The Initial Container Creation ACL is applied by default to any containers created within the ACL'd container. If this option is specified and the object named in `pathname` is not a container, an error is returned.

- **-io**
  For container objects only, specifies that the object's Initial Object Creation ACL is to be edited. The Initial Object Creation ACL is applied by default to any simple objects (that is, objects that are not containers) created within the ACL'd container. If this option is specified and the object is not a container, an error is returned.

- **-n**
  Specifies that a new mask should not be calculated. This option is useful only for objects that support the `mask_obj` entry type and that are required to recalculate a new mask after they are modified.
If a modify operation creates a mask that unintentionally adds permissions to an existing ACL entry, the modify causing the mask recalculation will abort with an error unless you specify either the -c or -n option.

-c Creates or modifies the object's mask_obj type entry with permissions equal to the union of all entries other than type user_obj, other_obj, and unauthenticated. This creation or modification is done after all other modifications to the ACL are performed. The new mask is set even if it grants permissions previously masked out. It is recommended that you use this option only if not specifying it results in an error. This option is useful only for objects that support the mask_obj entry type and are required to recalculate a new mask after they are modified.

If a modify operation creates a mask that unintentionally adds permissions to an existing ACL entry, the modify causing the mask recalculation will abort with an error unless you specify either the -c or -n option.

If you specify the -c option for an ACL that does not support mask_obj entry type, acl_edit returns an error when it attempts to save the ACL, aborting all subcommands supplied on the command line.

-ngui Specifies that a graphical user interface (GUI) should not be used even if a GUI is available. If your version of acl_edit supports a GUI and your terminal is capable of using it, invoking acl_edit without this option will bring up the GUI mode. Use the -ngui option to bring up command-line mode. However, if a GUI is not available, or the terminal is not capable of using the GUI, acl_edit comes up in command-line mode regardless of whether you supply this option or not.

-v Runs in verbose mode.

Arguments

pathname
The full pathname of the object whose ACL is to be viewed or edited. If the object is in another cell, pathname must be fully qualified to include the cell identifier.

command_line_subcommands
The command-line subcommands, which act on the object specified by pathname, are entered as part of the command string that invokes acl_edit. Only one command-line subcommand can be specified per invocation. The commands follow. See the description of the equivalent interactive subcommand for a more detailed description of the command functions.

-m acl_entry...
Adds a new ACL entry or changes the permissions of an existing entry. You can enter multiple entries, separated by spaces.

-p Purges all masked permissions (before any other modifications are made). This option is useful only for ACLs that contain an entry of type mask_obj. Use it to prevent unintentionally granting permissions to an existing entry when a new mask is calculated as a result of adding or modifying an ACL entry.

-d acl_entry ...
Deletes an existing entry from the ACL associated with the specified object. You can enter multiple entries, separated by spaces.

-s acl_entry ...
Replaces (substitutes) the ACL information associated with this
acl_edit(8sec)

object with acl_entry. All existing entries are removed and replaced by the newly specified entries. If you specify the -s subcommand, you cannot specify the -f or -k subcommand. You can enter multiple entries, separated by spaces.

-f file Assigns the ACL information contained in file to the object. All existing entries are removed and replaced by the entries in the file. If you specify the -f subcommand, you cannot specify the -s or -k subcommand.

-k Removes all entries, except entries of type user_obj (if they are present). If you specify the -k subcommand, you cannot specify the -f or -s subcommand.

-l Lists the entries in the object's ACL.

The command-line subcommands are evaluated in the following order:
1. -p
2. -s or -f or -k
3. -d
4. -m
5. -l

Description

Note:

With the exception of the following subcommands, this command is replaced at DCE Version 1.1 by the dcecp command. This command might be fully replaced by the dcecp command in a future release of DCE, and might no longer be supported at that time.
1. abort
2. commit
3. exit
4. help
5. test access

This control program has been superseded by dcecp. It is not designed for international use, and might give unexpected or undesirable results when used in non-English environments. If you are working with non-English data, you should use dcecp.

The acl_edit command is a client program that, when invoked, binds to the specified object's access control list (ACL) manager (which is implemented in the object's server), and allows the user to manipulate the object's ACL through the standard DCE ACL interface. This is the sec_acl_*(3sec) interface documented in the IBM DCE Version 3.1 for AIX and Solaris: Application Development Reference.

All acl_edit subcommands act on the object specified by pathname when you invoked acl_edit. You can invoke acl_edit in either command-line or interactive mode, as follows:
To invoke *acl_edit* in command-line mode, enter the command, the object’s pathname, options, and the command-line subcommand on the line that invokes *acl_edit*. Only one command-line subcommand can be entered per *acl_edit* invocation.

To invoke *acl_edit* in interactive mode, enter only *acl_edit*, the object’s pathname, and options. The *acl_edit* prompt is then displayed. In this mode, you enter interactive subcommands that let you edit and view entries in the object’s ACL and view help information about the *acl_edit* command itself.

Changes you make in command-line mode are saved when you enter the command. In interactive mode, you must explicitly save your changes. To do so, use the *commit* subcommand to save the changes without exiting *acl_edit* or the *exit* subcommand to save the changes and exit *acl_edit*. Use the *abort* subcommand to exit *acl_edit* and save none of the changes you have made.

Note: When you invoke *acl_edit* for a specific object’s ACL, that ACL is not locked. This means that it is possible for multiple users to edit the ACL simultaneously, with each change overwriting the previous changes. For this reason, the number of users assigned rights to change a particular ACL should be tightly controlled and limited to one user if possible.

### Interactive Subcommands

The following subcommands are available when *acl_edit* is invoked in interactive mode. All of the commands act on the ACL associated with the object specified by *pathname* when *acl_edit* was invoked.

- **?** Displays the available *acl_edit* subcommands.
- **ab [ort]** Exits *acl_edit* without saving the changes to the object’s ACL.
- **as [sign] filename** Applies the ACL entries in *filename* to the specified object. This subcommand removes existing entries and replaces them with the entries in the file.
- **c [ell] name** For ACL entry types of *user_obj*, *group_obj*, *other_obj*, and *group* sets the cell name to be assigned to the principal or group in place of the local cell name. This subcommand is used primarily to facilitate copying ACLs to different cells. The default cell name stays in place until you run the subcommand again to change it.
- **co [mmitt]** For ACL entry types of *user_obj*, *group_obj*, *other_obj*, and *group* sets the cell name to be assigned to the principal or group in place of the local cell name.
- **d [elete] acl_entry** Deletes the specified ACL entry.
- **e [xit]** Exits from *acl_edit*, saving any changes to the object’s ACL.
- **g [et_access]** Displays the permissions granted in the specified object’s ACL to the principal that invoked *acl_edit*.
- **h [elp] [command...]** Initiates the *help* facility. If you enter only the command *help, acl_edit*. 
acl_edit(8sec)
displays a list of all commands and their functions. If you enter help and a
cmd (or commands separated by a space), acl_edit displays help
information on the specified commands. Entering help sec_acl_entry
displays information about ACL entries.

k [ill_entries]
Removes all ACL entries except the user_obj entry if it exists.

l [ist] Lists the entries in the object’s ACL.

m [odify] acl_entry [-n | -c]
Adds a new ACL entry or replaces an existing ACL entry. This command
affects a single ACL entry. To add or replace all of an object’s ACL entries,
see the su[bstitute] subcommand.

For objects that support the mask_obj entry type and are required to
calculate a new mask when their ACLs are modified, the -n option specifies
that a new mask should not be calculated; the -c option specifies that the
object’s mask_obj entry should have permissions equal to the union of all
entries other than user_obj, other_obj, and unauthenticated. The mask is
calculated after the ACL is modified.

If you use the -c option, the new mask is set even if it grants permissions
previously masked out. It is recommended that you use the -c option only if
not specifying it results in an error. If the new mask unintentionally grants
permissions to an existing entry, the modify operation causing the mask
recalculation will abort with an error unless you specify either the -c or -n
option.

p [ermissions]
Lists the available permission tokens and explanations.

pu [rage]
Purges all masked permissions. This option is useful only for ACLs that
contain an entry of type mask_obj. Use it to prevent unintentionally
granting permissions to an existing entry when a new mask is calculated as
a result of adding or modifying an ACL entry.

su [bstitute] acl_entry ...
Replaces all ACL entries with the one or ones specified. This subcommand
removes all existing entries and adds the ones specified by acl_entry. To
replace only a single ACL entry, see the m[odify] subcommand.

t [est_access] [permissions ...]
Tests whether or not the permissions specified in the command are granted
to the principal under whose DCE identity the acl_edit command was
invoked. The option returns Granted if the permissions are granted or
Denied if they are not. The test_access subcommand works only on
Object ACLs, not on Initial Object ACLs or Initial Container ACLs. Thus, if
you invoke acl_edit with the -io or -ic option and then run the test_access
subcommand, the response you get is based on testing your access
against the object ACL.

sec_acl_entry
Describes the syntax and semantics of ACL entry types.

ACL Entries

An ACL entry has the following syntax:

type[:key]:permissions
where:

- **type** Identifies the role of the ACL entry.
- **key** Identifies the specific principal or group to whom the entry applies. For an entry type of `extended`, `key` contains the ACL data.
- **permissions** Specifies the ACL permissions.

A thorough description of each syntax component follows.

**type Component**

The `type` tag identifies the role of the ACL entry. Valid types are the following:

- **user_obj** Permissions for the object’s real or effective user.
- **group_obj** Permissions for the object’s real or effective group.
- **other_obj** Permissions for others in the local cell who are not otherwise named by a more specific entry type.
- **user** Permissions for a specific principal user in the ACL’s cell. This type of ACL entry must include a key that identifies the specific principal.
- **group** Permissions for a specific group in the ACL’s cell. This type of ACL entry must include a key that identifies the specific group.
- **foreign_user** Permissions for a specific, authenticated user in a foreign cell. This type of ACL entry must include a key that identifies the specific principal and the principal’s cell.
- **foreign_group** Permissions for a specific, authenticated group in a foreign cell. This type of ACL entry must include a key that identifies the specific group and the group’s cell.
- **foreign_other** Permissions for all authenticated principals in a specific foreign cell, unless those principals are specifically named in an ACL entry of type `foreign_user` or members in a group named in an entry of type `foreign_group`. This type of ACL entry must include a key that identifies the specific foreign cell.
- **any_other** Permissions for all authenticated principals unless those principals match a more specific entry in the ACL. If the ACL refers to DCE LFS objects, this entry applies to all unauthenticated users.
- **mask_obj** Permissions for the object mask that is applied to all entry types except `user_obj`, `other_obj`, and `unauthenticated`.
- **unauthenticated** Maximum permissions applied when the accessor does not pass authentication procedures. This entry is used for principals that have failed authentication due to bad keys, principals who are entirely outside of any authentication cell, and principals who choose not to use authenticated
access. Permissions granted to an unauthenticated principal are masked
with this entry, if it exists. If this entry does not exist, access to
unauthenticated principals is always denied.

**extended**

A special entry that allows client applications running at earlier DCE
versions to copy ACLs to and from ACL Managers running at the current
DCE version without losing any data. The extended entry allows the
application running at the lower version to obtain a printable form of the
ACL. The extended ACL entry has the following form:

```
extended:uuid. ndr. ndr. ndr. number_of_byte. data:permissions
```

where:

**uuid** Identifies the type extended ACL entry. (This Universal Unique
Identifier, or UUID, can identify one of the ACL entry types
described here or an as-yet-undefined ACL entry type.)

**ndr.ndr.ndr**

Up to three Network Data Representation (NDR) format labels (in
hexadecimal format and separated by periods) that identify the
encoding of data.

**number_of_bytes**

A decimal number that specifies the total number of bytes in data.

**data** The ACL data in hexadecimal form. (Each byte of ACL data is two
hexadecimal digits.) The ACL data includes all of the ACL entry
specifications except the permissions (described later) that are
entered separately. The data is not interpreted; it is assumed that
the ACL manager to which the data is being passed can understand
that data.

**user_obj_delegate**

Delegated permissions for the real or effective user of the object.

**group_obj_delegate**

Delegated permissions for the real or effective group of the object.

**other_obj_delegate**

Delegated permissions for others in the local cell who are not
otherwise named by a more specific entry type.

**user_delegate**

Delegated permissions for a specific principal user in the ACL’s cell.
This type of ACL entry must include a key that identifies the specific
principal.

**group_delegate**

Delegated permissions for a specific group in the ACL’s cell. This
type of ACL entry must include a key that identifies the specific
group.

**foreign_user_delegate**

Delegated permissions for a specific, authenticated user in a foreign
cell. This type of ACL entry must include a key that identifies the
specific principal and the principal’s cell.

**foreign_group_delegate**

Delegated permissions for a specific, authenticated group in a
foreign cell. This type of ACL entry must include a key that identifies
the specific group and the group’s cell.

any_other_delegate
Delegated permissions for all authenticated principals unless those
principals match a more specific entry in the ACL.

key Component
The key identifier (principal or group name) specifies the principal or group to which
the ACL entry applies. For entries of entry type extended, key is the data passed
from one ACL manager to another. A key is required for the following types of ACL
entries:
user Requires a principal name only.
group Requires a group name only.
foreign_user Requires a fully qualified cell name in addition to the principal name.
foreign_group Requires a fully qualified cell name in addition to the group name.
foreign_other Requires a fully qualified cell name.

permissions Component
The permissions argument specifies the set of permissions that defines the access
rights conferred by the entry. Since each ACL manager defines the permission
tokens and meanings appropriate for the objects it controls, the actual tokens and
their meanings vary. For example, the Distributed File Service (DFS), the directory
service, and the security registry service each implement a separate ACL manager,
each can use a different set of tokens and permissions. This means that file
system objects, objects in the namespace, and registry objects could each use
different permissions. Use the p [permissions] subcommand to display the currently
available tokens and their meanings. See the documentation for the DCE
component you are using to obtain a more detailed description of its specific
permissions.

Examples
1. The following example uses the interactive interface to set permissions for the
unauthenticated and mask_obj entry type:

   sec_acl_edit> m mask_obj:rwx
   sec_acl_edit> m unauthenticated:r

2. The following example uses the interactive interface to set permissions for the
effective user, group, and others in the ACL's cell:

   sec_acl_edit> m user_obj:crwx
   sec_acl_edit> m group_obj:rwrx
   sec_acl_edit> m other_obj:rwrx

3. The following example uses the command-line interface to invoke acl_edit and
assign permissions for the file progress_chart to the authenticated user mike
in the local cell:

   acl_edit /.../dresden.com/fs/walden/progress_chart -m user:mike:crwx
acl_edit(8sec)

Note that because this entry will be filtered through the object mask (mask_obj), which specifies only rwx permissions, the actual permissions will be rwx, not crwx. The [list] subcommand will show those permissions as follows:

user:mike:crwx #effective -rwx---

4. The following example uses the interactive interface to set permissions for the authenticated foreign user named burati in the cell named /.../usc-cs.uscal.edu:

  sec_acl_edit> m foreign_user:/.../usc-cs.uscal.edu/sailing/staff/burati:rwx

5. The following example uses the command-line interface to invoke acl_edit and set the Initial Container Creation permissions for the directory walden:

  acl_edit /.../dresden.com/fs/walden -ic -m /user:walden:crwxid

Related Information

Commands: acl(8dce).
auditd

Purpose

Starts the DCE audit daemon

Synopsis

```
```

Options

- `-t` Specifies the path of the audit trail file used by `auditd`. The default path is `dcelocal/var/aud/adm/central_trail`. If an audit trail filename is specified instead of an absolute pathname, the file is created in the `dcelocal/var/aud/adm/` directory.
- `-a` Audits access to the audit daemon's control interface.
- `-s size` Sets a warning threshold on the size of the audit trail file. The audit daemon displays a warning each time an audit record is appended to the audit trail after the threshold has been reached.
- `-wrap` Wraps the recording of audit events to the beginning of the audit trail file when its size limit is reached. The default action when the size limit has been reached is to stop auditing.
- `-w svc_route` Specifies where each level of serviceability messages are routed. The `svc_route` argument contains three fields, separated by colons: the level, a routing identifier, and a routing parameter, as follows:

  severity: how: where

See `svcroute(5dce)` for possible values for these fields.
- `-d debug_level` Specifies debugging level of subcomponents. The `debug_level` argument contains four fields separated by colons, as follows:

  component: flags: how: where

See `svcroute(5dce)` for possible values of these fields.

Description

The `auditd` command starts the audit daemon. The audit daemon must be run on the host before the audit clients. The audit daemon can only service audit clients that are on the host where it is running. Thus, an audit daemon must be installed and run on every host in the cell that has audit clients (audit clients include DCE servers and user-written application servers).

The audit daemon has two functions. It maintains the filter files which are shared by all audit clients running on the host. It also provides an audit record logging service to these clients.
The audit daemon runs under the local host’s machine principal identity (host/hostname/self).

A DCE host daemon (dced) must be running on the local host when auditd is started. Typically, dced and auditd are started at boot time. The auditd process places itself in the background and sends messages indicating it is ready to service requests for updating or querying filters and logging audit records.

Privileges Required

You must be logged into a privileged account (cell_admin or a member of the audit-admin group) to run auditd.

Examples

1. The following command starts the audit daemon using the default audit trail file (dcelocal/var/aud/adm/central_trail):

   auditd

2. The following command starts the audit daemon and specifies dcelocal/var/aud/adm/my_trail_file as the audit trail file:

   auditd -t my_trail_file

3. The following command starts the audit daemon and specifies where each level of serviceability messages is to be routed:

   auditd -w FATAL:FILE:/dev/console \
   -w NOTICE:FILE:/opt/dcelocal/var/audit/adm/svc_log

4. The following starts auditd and specifies the debugging level:

   auditd -d ";*.9;FILE:/opt/dcelocal/var/audit/audit.dbg"

Related Information

Commands: aud(8dce), audevents(8dce), audfilter(8dce), audtrail(8dce), dcecp(8dce).

Files: svcroute(5dce).
**Purpose**

Note: The `dce_login` command also describes the `dce_login_noexec` command. Validates a principal’s identity and obtains the principal’s network credentials.

**Synopsis**

```bash
```

**Options**

- **-c** Causes the principal’s identity to be certified. If you do not specify `-c`, the principal’s identity is validated only. (You must be system user `root` to use this option.)

- **-k eyfile filename**
  Validates the principal_name identity by using a password-derived key obtained from a keytable file identified by file_name.

- **-r** Refreshes and validates the current login ID.

- **-e [exec] cmd_string**
  Executes the command supplied as `cmd_string`.

- **-n [ewpass]**
  Indicates intent to update the password for this principal.

- **-f** Makes the ticket forwardable.

- **-p** Makes the ticket proxiable.

- **-R renewable_life**
  Allows the ticket to be renewed for the renewable life period. The `renewable_life` can be specified as a number of hours, minutes, or days (that is 2h, 60m, 1d). The maximum value, however specified, is 28 days.

**Arguments**

- **principal_name**
  The name of the principal to log in as.

- **password**
  The password for `principal_name`.

**Description**

The `dce_login` command validates the principal identity and obtains the principal network credentials.

Note: If you are using DFS, you must `dce_login` after the DFS client is configured and running on your machine in order to have authenticated access to DFS objects. Any credentials acquired before DFS is running will not be recognized by DFS.

The `-exec` option runs the command specified by `cmd_string` after login. If `cmd_string` is specified without a full pathname, the path prefix is obtained by

---

**Chapter 5. Security Service Files and Commands**

659
dce_login(8sec)

searching the directories according to the PATH variable. If this option is not specified, the dce_login command runs the shell specified in the SHELL environment variable.

The principal_name argument specifies the name of the DCE principal logging in. The password argument specifies the password of the principal. If you do not supply a principal name or a principal password, dce_login prompts for them, unless the -r option is specified. If the -r option is specified, a principal_name should not be specified. If you enter them both on the command line, you must specify the principal name first, followed by the password.

The -keyfile option causes the principal_name identity to be validated using a password-derived key obtained from the file file_name, (instead of using a password supplied as a command line option). The file_name must include full path name of the file together with the characters FILE: prepended to the path name, for example:

FILE:D:/opt/dcelocal/krb5/mykeyfile

and this file must have been created by the appropriate rgy_edit, dcecp, or sec_key_mgmt_set_key functions. When using this option, do not include a password as a command-line option. This option is incompatible with the -c option.

The -r option causes the network credentials for the currently logged in user to be refreshed, that is, the acquisition of a new Ticket Granting Ticket (TGT), currently associated with the process in which this command will be executed. When using this option, do not include principal_name as a command-line option.

The -newpass option indicates the desire to update the password for the named principal. dce_login will issue a prompt for the new password. If the option is specified,

1. Do not specify either the [-system], [-keyfile], or [-newpass] options
2. Do ensure that you include both principal_name and password.

The dce_login_noexec command does not run a new shell as does dec_login, but results in the creation of a new credentials file. dce_login_noexec returns the name of this file.

Note: If you are running DFS, do not use dce_login_noexec. The necessary information needed by DFS to recognize you as an authenticated DCE user is not established. You must dce_login after the DFS client is configured and running on your machine in order to have authenticated access to DFS objects. Any credentials acquired before DFS is running will not be recognized by DFS.

Note that if the clocks on the Security server and client machines are not synchronized to within two or three minutes of each other, you can receive a password validation error and be unable to log into DCE.

Examples

dce_login myname mypwd -f -p -R 5d

The default cell policy allows tickets to be forwardable and renewable. However, to set the proxiable flag the /.../cell_name/dce-rgy and the /.../cell_name/dce-ptgt
accounts must be modified to allow proxiable tickets. The following dcecp commands can be used to modify the accounts.

dcecp> account modify dce-rgy -proxiabletkt yes
dcecp> account modify dce-ptgt -proxiabletkt yes

The user account must also enable the appropriate ticket options.
**dcecred_* Files**

**Purpose**

- **dcecred_XXXXXXXX**
  - Contains the tickets issued to a DCE principal.

- **dcecred_XXXXXXXX.data**
  - Contains fixed registry information pertaining to a DCE principal.

- **dcecred_XXXXXXXX.data.db**
  - Contains the extended privilege attribute certificate (EPAC) pertaining to a DCE principal.

- **dcecred_XXXXXXXX.nc**
  - Contains cached entries of global principal or group names parsed into cellnames and cell-relative principal or group names.

**Synopsis**

- **dcecred_***

**Description**

All ticket caches (also called credentials caches) reside in the directory `dcelocal/var/security/creds`. Each `dcelocal/var/security/creds/dcecred_XXXXXXXX` file stores the tickets issued to a particular DCE principal (XXXXXXXX is an eight-digit hexadecimal number). A `dcecred_XXXXXXXX` file is created by invoking `dce_login` (or `dce_login_noexec`), or by invoking an AIX command (such as `login` or `su`) that is integrated with the DCE security services. Only the user who invoked `dce_login` has read-write permission to the file. The environment variable `KRB5CCNAME` (which stands for "Kerberos version 5 credentials cache name") is set to `FILE:/opt/dcelocal/var/security/creds/dcecred_XXXXXXXX`.

The `dcelocal/var/security/creds/dcecred_fffffff` ticket cache is special. It stores the tickets for the local machine's principal, known in the DCE registry as `hosts/dce_hostname/self`.

**Note:** The DCE host name is case sensitive. The root user on a machine automatically inherits these credentials and so appears as DCE entity `self` until reauthenticating as a different DCE principal. A root user who has not reauthenticated will share the `dcecred_fffffff` ticket cache with other daemons using the machine's identity (for example, `dced` and `dtsd`) and must be careful not to inadvertently destroy the ticket cache (by issuing `kdestroy` or deleting the file, for example). If this special file is destroyed, it can be re-created by restarting `dced` with the `-p` option.

The `dcelocal/var/security/creds/dcecred_XXXXXXXX.data` files contain fixed principal data such as UID, GID, registry quota, and GECOS information. A `dcecred_XXXXXXXX.data.db` file contains a principal's extended privilege attribute certificate (EPAC). These files are created along with a `dcecred_XXXXXXXX` file during `dce_login`.

The `dcelocal/var/security/creds/dcecred_XXXXXXXX.nc` files are name caches that contain entries in which a global principal or group name has been parsed into...
a cellname and a cell-relative principal or group name. Each entry also contains a
timestamp and can contain the UUIDs of the cell and principal or group. A
dcecred_XXXXXXXXX.nc name cache is created when a DCE principal makes a
call to the sec_id API. On subsequent calls to the sec_id API, the cache is
checked first for the requested information.

Related Information

Commands: dce_login(8sec), dce_login_noexec.
dceunixd

**dceunixd (AIX only)**

**Purpose**

Acts as an intermediary between AIX base operating system security commands and the DCE security services. This daemon must be running on any machine in which integrated security operations are needed.

**Synopsis**

```
dceunixd -l lifetime -d debug_level -h -i behavior -n numdaemons -s -t
```

**Options**

- **-l lifetime**
  Specifies the lifetime, in minutes, of cached `passwd struct` and `group struct` entries. Valid lifetimes range from 2 to 120 minutes, inclusive. The default is two minutes.

- **-d debug_level**
  Runs the daemon in the foreground for debugging purposes. `debug_level` is a nonzero, single-digit value that determines the content of debugging output. Each level subsumes the levels below it:
  1. Displays code flow and irrecoverable error conditions such as `malloc` errors.
  2. Displays return statuses from DCE RPCs and internal routines.
  3. Displays input and output arguments.
  4. Displays socket buffer contents.
  5 through 9
     Currently undefined; displays the information of levels 1 through 4.

- **-h**
  Displays usage.

- **-i behavior**
  Specifies behavior for intercell operations. Valid intercell behaviors are 0 and 1.
  1. Causes the `dceunixd` routines to check the home cell ERA, to bind to foreign cells to satisfy requests, and to allow intercell integrated security operations.
  0. Disallows any intercell operations. 0 is the default value.

- **-n numdaemons**
  Specifies the number of daemons to be started. Valid numbers of daemons range from 1 to 5. The default is 1. A larger number of daemons will enhance the performance of the `dceunixd` daemon. The -n option is mutually exclusive with the -d option.

- **-s**
  Specifies that you are running on a slim client configuration. You must specify this flag when you want to run integrated login on a slim client configuration.

- **-t**
  Sets the appropriate TGT options as specified in the user’s account information.
Description

The `dceunixd` daemon is necessary to enable integration between AIX base operating system security and DCE security services. Without this daemon, no AIX BOS program (for example, `login`, `passwd`, or `id`) can access information in the DCE registry. Typically, `dceunixd` runs on any DCE client machine which requires integrated security operations, and is started after all DCE core daemons. `dceunixd` operates under the local machine’s identity and so must be started after `dced` sets up the machine DCE context (that is, valid `dcelocal/var/security/creds/dcecred_ffffffff` and `dcelocal/var/security/creds/dcecred_ffffffff.data` files must exist).

Under normal circumstances, `dceunixd` is started with no flags and runs as a background process. If debugging is necessary, it can be stopped and restarted with the `-d` option. `dceunixd` must be started by the local root user who should have no explicitly established DCE context (that is, the `KRB5CCNAME` environment variable must not be set).

For intercell operations the `dceunixd` daemon must be started with the `-i` option set to 1 to enable the intercell integrated security services.

To set the forwardable, proxiable, and renewable flags and the renewable time in the TGT during a login, the `dceunixd` daemon must be started with the `-t` option. The default cell policy allows tickets to be forwardable and renewable. However, to set the proxiable flag the `./cell_name/dce-rgy` and the `./cell_name/dce-ptgt` accounts must be modified to allow proxiable tickets. The following `dcecp` commands can be used to modify the accounts.

```
dcecp> account modify dce-rgy -proxiabletkt yes
dcecp> account modify dce-ptgt -proxiabletkt yes
```

The user account must also enable the appropriate ticket options. During login `dceunixd` queries the user account and sets the enabled options in the TGT. If `dceunixd` is not started with the `-t` option, the user account information is ignored and the TGT will not have any flags set.

For a slim client configuration, the `dceunixd` daemon must be started with the `-s` option to allow execution.

Notes

UNIX limits the DCE username to eight bytes. Any DCE user with a cell-relative name longer than eight bytes cannot take advantage of AIX integration.

Related Information

Files:

```
dcelocal/var/security/creds.dcecred_ffffffff
   The credentials file that contains the local machine’s context. `dceunixd` uses the identity defined in this credentials file.
```

```
dcelocal/var/security/adm.dceunixd/dceunixd.skt
   The UNIX domain socket file used to communicate between `dceunixd` and AIX base operating system programs.
```
dceunixd

/usr/lib/security/DCE

A dynamically loadable module incorporated into AIX base operating system programs when access to the DCE security services is necessary. This file must be present on any machine which requires integrated security operations.
kdestroy

Purpose

Destroys a principal’s login context and associated credentials

Synopsis

kdestroy [-c cache_name] [-f] [-e time]

Options

- **c cache_name**
  Specifies that the login context and associated credentials for *cache_name* should be destroyed instead of the default cache.

- **f**
  Forces the operation. It is necessary to specify this option if you want to destroy the machine principal’s credentials.

- **e time**
  Deletes credentials expired before the relative time specified by the time string. The *time* format is *x w x d x h x m x s*, where *x* = a number, *w* = weeks, *d* = days, *h* = hours, *m* = minutes and *s* = seconds. The default is hours. For example, *kdestroy -e 5m* deletes credentials that expired before 5 minutes in the past. The relative time is computed as now negative time, which is in the past.

Description

kdestroy destroys the principal login context and the principal credentials. Until the credentials are re-established, the principal and any processes created by the principal are limited to unauthenticated access.

Notes

If you accidentally destroy your machine credentials, you need to stop and then restart dced to obtain new machine credentials.

Files

*dcelocal*/var/security/creds/dcecred_XXXXXXXX
This is the form of a DCE credentials cache file. XXXXXXXX is a random hex number. Authenticating to DCE also sets the KRBS5CCNAME environment variable equal to the string

FILE:/opt/dcelocal/var/security/creds/dcecred_XXXXXXXX. If this environment variable is set when kdestroy is invoked (without the -c option), its setting determines the name of the credentials cache file destroyed. Note that even though kdestroy destroys the cache file, it does not unset the KRBS5CCNAME environment variable.

Related Information

Commands: *kinit(8sec), klist(8sec).*
kinit

Purpose

granting ticket

Synopsis


Options

-c cachename
  Specifies an alternative credentials cache, cachename, to be used in place of the default credentials cache. The kinit command overwrites the contents of the alternative cache with the current credentials.

  The name of the default credentials cache might vary between systems. However, if the KRB5CCNAME environment variable is set, its value is used to name the default cache.

-f
  Requests the FORWARDABLE option. This option allows a ticket-granting ticket with a different network address than the present ticket-granting ticket to be issued to the principal. For forwardable tickets to be granted, the principal's account in the registry must specify that the principal can be granted forwardable tickets.

-l lifetime
  Specifies the lifetime of the ticket-granting ticket in hours. If this option is not specified, the default ticket lifetime (set by each site using the rgy_edit command) is used.

  Note: Input to this flag must be >= 5m. Registry policy can limit the lifetime interval specified with this flag.

-p
  Requests the PROXIABLE option. This option allows a ticket with a different network address than the present ticket to be issued to the principal. For proxiable tickets to be granted, the principal's account in the registry must specify that the principal can be granted proxiable tickets.

-r lifetime
  Requests the RENEWABLE option. This option allows the tickets issued to the principal to be renewed. For renewable tickets to be granted, the principal's account in the registry must specify that the principal can be granted renewable tickets. The lifetime of the ticket-granting ticket is specified in hours by lifetime.

-s start_time
  Sets the ticket valid time to the time specified by the start_time variable. The start_time can be specified in various time formats including the following:
  hh:mm
  hh:mm:ss
  yy:mm:dd:hh:mm
  yy:mm:dd:hh:mm:ss
-v Specifies verbose mode.

-V Validates a postdated ticket. The user account must enable the postdatedtkt option. The -V option will not succeed until after the time specified by the -s option start_time variable.

Arguments

principal

Specifies the name of the principal for whom the ticket-granting ticket should be obtained. If principal is omitted, the principal name from the existing cache is reused.

password

The password for the specified principal.

Description

The kinit command can be used to refresh the principal ticket-granting ticket. All tickets in the target cache file are replaced by a single refreshed ticket-granting ticket. When you invoke kinit, it prompts for your password. You must be authenticated as a DCE user to use this command to refresh your credentials.

Note: Do not use kinit to refresh credentials for a principal requiring third-party preauthentication (that is, the principal has a pre_auth_req extended registry attribute set at level 2). Use dce_login -r to refresh such credentials.

The ticket lifetime and renewable lifetime is set in the following format:

{num {interval}}...

where:

num is a number that specifies the number of the interval; interval is:

w Weeks
d Days
h Hours
m Minutes
s Seconds

For example, to set the lifetime to 3 weeks, 5 days, and 10 hours, the entry would be as follows:

3w5d10h

Note: If you are using DFS, you must dce_login after the DFS client is configured and running on your machine in order to have authenticated access to DFS objects. Any credentials acquired before DFS is running will not be recognized by DFS.

Files

FILE:/opt/dcelocal/var/security/creds/dcecred_XXXXXXXX

This is the form of a DCE credentials cache file. XXXXXXXXX is a random hex number. Authenticating to DCE also sets the KRB5CCNAME
environment variable equal to the string

FILE:/opt/dcelocal/var/security/creds/dcecredXXXXXXXX. If this
environment variable is set when the kinit command is invoked (without the
-c option), its setting determines the name of the credentials cache file on
which kinit operates.

/tmp/krb5cc_UID

This is the form of the cache file created when the kinit command is
invoked in the absence of a DCE login context (that is, the KRB5CCNAME
environment variable is unset) and with a principal name specified. UID is
the UNIX ID of the local user who invoked kinit.

Related Information

Commands: dce_login, dce_login_noexec, kdestroy(8sec), klist(8sec).

Files: dcelocal/var/security/creds/dcecredXXXXXXXX
klist

Purpose

Lists cached tickets

Synopsis

klist [-c cachename] [-e] [-f]

Options

-c cachename
  Specifies that the contents of the cache identified by cachename should be
  displayed instead of the contents of the default cache. The input to this flag
  must be in the form FILE:cachename.

-e
  Includes expired tickets in the display. Without this option, only current
  tickets are displayed.

-f
  Displays option settings on the tickets. The options are as follows:
  D    Postdatable
  d    Postdated
  F    Forwardable
  f    Forwarded
  I    Initial
  i    Invalid
  P    Proxiable
  p    Proxy
  R    Renewable

Description

The klist command lists the primary principal and tickets held in the default
credentials cache (the cache indicated by the name KRBS5CCNAME environment) ,
or in the cache identified by cachename if the -c option is used.

Files

dcelocal/var/security/creds/dcecred_XXXXXXXX
  If the KRBS5CCNAME environment variable is not set, the name of the
default credentials cache is in the form shown, where XXXXXXXX is a
random hexadecimal number. If the KRBS5CCNAME environment variable is
set, its value is used to name the default cache.

Related Information

Commands: kdestroy(8sec), kinit(8sec).
k5dcelogin

Purpose

Promotes a Kerberos V5 credential to a DCE credential so that the credential can be used to access DCE objects, such as Distributed File System (DFS) files.

Synopsis

Arguments

username
The name of the remote user attempting to access the remote host.

cmd
The remote command to be invoked by k5dcelogin after the DCE credentials have been established. Typically this is the login (1) command or a shell command.

cmd_parameters
Optional parameters for the cmd option.

Description

The k5dcelogin command promotes a principal’s Kerberos V5 credentials to DCE credentials without prompting for a password. It is intended to be called by the kerberized rlogind and rshd servers as the last step of the authentication process when ticket forwarding is requested.

The DCE credentials are destroyed when the command finishes.

Only the remote owner should be granted write and execute permissions to k5dcelogin.

Related Information

Commands: rlogind(8sec), rshd(8sec).
passwd_export

Purpose

Creates local password and group files

Synopsis

passwd_export [-l] [-g group1, group2, ...][-u user1, user2, ...] [-f][-d dir_name] [-h] [-m max_entries] [-n] [-r] [-s] [-v] [-x]

Options

- **-l**  (AIX only) Specifies that operations are to be performed only on users in the local /etc/passwd file.
- **-g group1, group2, ...**
  Specifies the groups on which to perform the operation. If this option is not specified, all groups are included.
- **-u user1, user2, ...**
  Specifies the users (or principals) on which to perform the operation. If this option is not specified, all users (or principals) are included.
- **-f**  (AIX only) Forces the update by overriding the dce_export attribute in the /etc/security/user file.
- **-d dir_name**
  Specifies the name of the directory in which to store the password, group, and organization files created by passwd_export. If you do not enter a directory name, the files are stored in the /etc directory.
- **-h**  Displays help information.
- **-m max_entries**
  Specifies the maximum number of entries that can be stored in the local files.
- **-n**  Ignores passwd_override and group_override file entries. Without this option, passwd_export applies the override entries from both files to the local password and group files it creates. Consult the passwd_override(5sec) and group_override(5sec) reference pages for further information.
- **-r**  (Solaris only) Forces the update of the root user entry. If this option is not specified, the root user entry is created from the current /etc/passwd file.
- **-s**  Sorts the local password and group file entries by UNIX number. If you do not specify this option, the entries are in order as they are retrieved from the registry.
- **-v**  Runs in verbose mode.
- **-x**  Omits users and groups with the specified passwd OMIT in their passwd_override and group_override file entries from the local password and group files created. Consult the passwd_override(5sec) and group_override(5sec) reference pages for further information.
**passwd_export(8sec)**

**Description**

The `dceshared/bin_passwd_export` command creates local password and group files from registry data. Use `passwd_export` to keep these local files consistent with the registry database.

If you do not specify the `-n` option, `passwd_export` reads override entries in the `passwd_override` and `group_override` files and modifies accordingly the local and group files it creates. See the `passwd_override(5sec)` and `group_override(5sec)` reference pages for further information.

When `passwd_export` runs, it makes backup copies of the current password and group files, if they exist. The files are named, respectively, `passwd.bak` and `group.bak`. The new files created by `passwd_export` are named `passwd` and `group`. By default, the backups are stored and the new files created in the `/etc` directory. You can override the default by supplying a directory name with the `-d` option.

The `dce_export` value must be set to TRUE on the local machine for each user and group to be downloaded. If this value is FALSE, the information is not downloaded unless the force option (`-f`) is specified. The `dce_export` attribute is located in the `/etc/security/user` and `/etc/security/group` files and is initialized to FALSE at installation time.

If the `-d` option is specified, a `passwd` and `group` file is created in the directory specified. The `passwd` file created contains encrypted passwords. The `passwd_export` command does not create or make additions or changes to the `/etc/security/passwd` or `/etc/security/group` files.

If the password property is set to HIDDEN in the registry, a warning is displayed and no action occurs unless the `-d` option is used. If the `-d` option is specified, the `passwd` and `group` files are created but an * (asterisk) is displayed in the password field of the `passwd` file.

If the principal name is greater than `L_cuserid` defined in the `stdio.h` header file, that user is not added to the local `passwd` files. A warning indicating the problem is displayed if the verbose option (`-v`) is specified.

The CIPHER and OMIT strings are changed to an * (asterisk) to be consistent with AIX. Both accomplish the same thing.

**Note:**

- UNIX limits the DCE username to eight bytes. Any DCE user with a cell-relative name longer than eight bytes cannot take advantage of AIX integration.
- The `passwd_export` command will not work on a slim client configuration because the slim client does not run with all the necessary daemons to support the command.

**Running passwd_export**

The `passwd_export` command is commonly run with an entry in `/usr/lib/crontab`. For example, to update the files every hour, the entry is as follows:

```
0 * * * * dceshared/bin/passwd_export
```
In large network environments, it is a good idea to stagger the times at which `passwd_export` is run.

To download all users in the `/etc/passwd` file, enter:

```
passwd_export [-l] [-f]
```

The `-l` option specifies that the `passwd_export` command reads the `/etc/passwd` file to determine which users to download. The `-f` option is optional; it overrides the `dce_export = FALSE` entries for the default stanza or in each individual user’s stanzas.

To protect some user entries in the `/etc/passwd` file, but download all other DCE users, enter:

```
passwd_export
```

This entry downloads all users in the DCE registry but honors the `dce_export = FALSE` entry.

To obtain specified users, enter:

```
passwd_export [-l] -u user1, user2, ... [-f]
```

Related Information

Commands: `dcecp(8sec), rgy_edit(8sec)`.

Files: `passwd_override(5sec), group_override(5sec), group(5), passwd(5)`.
passwd_import

Purpose

Creates registry database entries from group and password files

Synopsis


Options

-c Runs in check mode: processes the command, showing all conflicts, but makes no requests for resolution.

-d path Specifies the path of the directory containing the foreign password and group files to be imported.

-h Displays help information.

-i Ignores name conflicts. Names in the registry and the group and password files represent the same identity.

-o org Specifies the name of an organization to be assigned to all imported entries. If the specified organization does not exist, it is created. The default organization is none.

-p password Specifies the password for the account with whose privileges passwd_import runs.

-u username Specifies the principal name of the account with whose privileges passwd_import will run. This account must have the privileges to access the registry and add principals, groups, accounts, and organizations, and to add members to groups and organizations. The principal name and password are used to obtain network authentication. If you do not supply them, passwd_import prompts for them, even if you have already performed a network login.

-v Runs in verbose mode, generating a verbose transcript of passwd_import activity.

Description

The passwd_import command is a mechanism for creating registry database entries that are consistent with foreign password and group file entries. Use passwd_import to ensure consistency between DCE and foreign protection mechanisms when you do any of the following:

1. Attach DCE node(s) to an existing UNIX network
2. Attach UNIX node(s) to a DCE network
3. Connect DCE and UNIX networks

If the password and group file entries do not exist in the DCE registry, passwd_import creates them. If there are duplicate entries, passwd_import follows your directions on how to handle them.
**Note:** The `passwd_import` command will not work on a slim client configuration because the slim client does not run with all the necessary daemons to support the command.

**The Process**

The DCE registry database must exist and be running before you can use `passwd_import`. If you are simply adding a few DCE nodes to a foreign network, you can create a new, but empty, registry to meet this requirement.

As `passwd_import` processes, it performs the following steps:

1. It opens the group and password files and establishes a connection to the registry.
2. It compares the group file entries to groups in the registry. If there are no conflicts, it creates groups in the registry corresponding to the groups in the group file.
3. It compares the entries in the password file to principals in the registry. Again, if there are no conflicts, it does the following:
   a. Creates principals in the registry corresponding to the entries in the password file.
   b. Adds the newly created principals to the appropriate groups.
   c. Creates accounts for the newly created principals.
4. It reexamines the group file and adds the principals as members of any additional groups it finds there.

The changes to the registry are made individually as each step is processed. If you do not specify the organization, the principals are added to the organization `none`.

**Conflicts**

During this process, `passwd_import` can find conflicts in name strings (for example, in the password file: `joe 102`; in the registry database: `joe 555`) and in UNIX IDs (for example, in the password file: `joe 102`; in DCE: `carmelita 102`). When `passwd_import` finds a conflict, it prompts for changes to make to the `/etc/passwd` and `/etc/group` entries. No changes are made to the registry entries. In other words, all conflicts are resolved in favor of the registry entry.

The `-i` option specifies that duplicate names are not in conflict but, in fact, represent the same identity. Therefore, when duplicate names arise, no action is necessary. If you do not use the `-i` option, `passwd_import` prompts for how to handle the name conflicts.

**Resolving Conflicts**

The `passwd_import` command prompts for instructions to resolve the conflicts it finds. You have the following choices:

1. You can create an alias to resolve a UNIX ID conflict. This action creates an alias for the registry object in conflict. The `passwd_import` command assigns this alias the same name as the conflicting entry in the `/etc/group` or `/etc/passwd` file. For example, if the entry `joe 555` exists in the registry and the entry `tim 555` exists in the `/etc/passwd` file, choosing this option creates the alias `tim` for `joe 555`.
2. You can generate a new UNIX ID automatically or enter a new one explicitly to resolve a UNIX ID conflict. For example, if there is a conflict
between the entry joe 555 in the registry and tim 555 in the /etc/passwd file, you can generate a new UNIX ID for tim.

3. You can enter a new name to resolve a name conflict. For example, if there is a conflict between the entry joe 555 in the registry and joe 383 in the /etc/passwd file, you can generate a new name for joe 383. This new name will then be added to the registry.

In addition, you are given the option of ignoring the conflict and skipping the entry.

Generally, you should run passwd_import with the -c option. Using the results of this run, you can determine how to handle the conflicts. If there are many conflicts, it might be more efficient to manually edit either the registry or the group and password files to resolve some of them before you run passwd_import.

Registry Database Entries

New registry entries created by passwd_import are assigned the following values:

For Principal and Group Entries:

alias/primary
If the /etc/passwd file contains two entries with the same UNIX number, passwd_import creates a primary name entry for the first occurrence of the UNIX number and alias entries for each subsequent occurrence.

fullname
A blank string; no full name is added for the entry.

membership list
For new groups only; all principals listed in the group file, and all principals with accounts in the password file with that group.

projlist_ok
Yes (for groups only).

For Account Entries:

Account expiration date
None.

Account_valid
False.

Client flag
True.

Duplicate certificate flag
False.

Forwardable certificate flag
True.

Gecos
Same as password file.

Good since date
Time of account creation.

Homedir
Same as password file.
Maximum certificate lifetime
Default to registry authentication policy.

Maximum certificate renewable
Default to registry authentication policy.

Passwd
Randomly generated. Note that you must modify or reset randomly generated passwords before user authentication is possible.

Passwd_dtm
Date and time passwd_import was run.

Passwd_valid
False.

Postdated certificate flag
False.

Proxiable certificate flag
False.

Renewable certificate flag
True.

Server flag
True.

Shell
Same as password file.

TGT authentication flag
True.

Related Information

Commands: dcecp, rgy_edit, sec_admin, secd.
pwd_strengthd(8sec)

pwd_strengthd

Purpose

The password management server

Synopsis


Options

+all_spaces

Allows passwords to be all spaces. If this option is not set, the effective registry policy is used.

-all_spaces

Prevents passwords from being all spaces. If this option is not set, the effective registry policy is used.

+alpha_num

Allows passwords to consist of only alphanumeric characters. If this option is not set, the effective registry policy is used.

-alpha_num

Prevents passwords from consisting of only alphanumeric characters. If this option is not set, the effective registry policy is used.

-cache_size size

Specifies the number of hash buckets in the password cache. The password cache is used to store generated passwords which are retrieved when the password is strength checked. The password cache is a hash table with a linked list for collisions. The size should be set to a reasonable value based on how large the cache will be on average. The default value if not specified is 100.

-debug

Runs pwd_strengthd in the foreground. Messages are written to either the standard output or the serviceability logs or both. Be sure the notice level in the serviceability routing file, routing, is set.

-min_len pwd_min_len

Specifies the minimum length of a password. If this option is not set, the effective registry policy is used.

-timeout minutes

Specifies the time, in minutes, that generated passwords remain in the cache before they are deleted from memory. If this option is not specified, the default time is 30 minutes.

-verbose

More informational messages are sent to the serviceability notice log file or standard output or both. You must enable routing of notice messages. See the svcroute section in IBM DCE Version 3.1 for AIX and Solaris: Application Development Reference.
-userdef
Allows user-defined library checking.

-server_princ name
Specifies the name of the password strength server principal. The default password strength server principal is **pwd_strengthd**. If you use the -server_princ option, you must also set the config.dce options, -pwdstr_principal and -pwdstr_cmd to the same password strength server principal name. For example:

```bash
cfg.dce -pwdstr_arg "-v -s pws_id
-pwdstr_principal pws_id
-pwdstr_cmd pwd_strengthd pw_strength_srv
```

where pws_id is the password strength server principal name.

### Description

The **pwd_strengthd** command is a password management server. The IBM DCE Enhanced Password Strength Server performs expanded password checking and generation. The server can validate passwords against composition rules, age rules, reuse history rules, dictionary lists, and user-defined rules. It exports the **rsec_pwd_mgmt** application programming interface.

The **pwd_strengthd** command generates passwords or strength-checks them. It enforces the security registry policy for password strength-checking. Administrators can override the security registry policy via the command-line options +/−alpha, +/−all, and -m.

Administrators can specify enhanced checking rules with **IBM_pwd_*_rules** extended registry attributes (ERAs). Administrators can subject principals to **password strength** and **password generation** policies by attaching the following ERAs. These ERAs can be set on a server-wide basis, organizational basis, or on a user principal's basis.

**pwd_val_type**
Specifies the password management policy the user must conform to when selecting passwords.

**pwd_mgmt_binding**
Specifies information required in order to connect to the password management server.

**IBM_pwd_comp_rules**
Specifies what types and combinations of characters can be used in a password.

**IBM_pwd_age_rules**
Specifies how soon a password can be changed.

**IBM_pwd_hist_rules**
Specifies the amount of time before a password can be used again and the number of different passwords that must be used before a password can be used again.

**IBM_pwd_dict_rules**
Specifies a list of dictionary files to used during password verification.

**IBM_pwd_userdef_rules**
Specifies a list of the customer’s own password checking libraries.
pwd_strengthd(8sec)

See the *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide—Core Components* for more information and examples.

Notes

You might want your password strength server to support your site’s policies for password strength and generation. If so, start with the example code located in `/opt/dcelocal/examples/pwdstren`, and read the README located in the same directory for instructions on how to modify and build your custom strength server.

Related Information

Commands: *passwd_export, passwd_import.*
rgy_edit

Purpose

Edits the registry database

Synopsis

rgy_edit [[-a | -p | -g | -o] [-s name] [-up[date]] [-v [-f] [name | -un[ix__name]] [-nq]] | -l]

Options

The following options are available when rgy_edit is invoked. You can specify only one of the options -a, -p, -g, and -o. If you specify the -l option, you can specify no other options.

-a (default)
- Edits or views accounts.
-p
- Edits or views principals.
-g
- Edits or views groups.
-o
- Edits or views organizations.
-s name
- Binds to the registry site specified by name. The name argument is either the fully qualified name of the cell that contains the registry to which you want access, or the fully qualified name of a specific registry server.

-up[date]
- Binds to a read-write registry site in the cell specified by the -s option.

-v
- Views the registry entry specified its name or unix_name arguments. If no entry is specified, all entries are viewed.
-f
- Displays in full the entry (or entries) selected by the -v option. The full entry includes all fields except the membership list and organization policy.

-nq
- Specifies that delete operations will not be queried. The default is to prompt the user for verification when a delete operation is requested.

-l
- Edits or views entries in local registry.

Description

Note:

With the exception of the following subcommands, this command is replaced at DCE Version 1.1 by the dcecp command. This command might be fully replaced by the dcecp command in a future release of DCE, and might no longer be supported at that time.
1. defaults
2. delete
3. domain
4. exit
5. help
rgy_edit(8sec)

6. purge
7. quit
8. scope
9. view

This control program has been superseded by dcecp. It is not designed for international use, and might give unexpected or undesirable results when used in non-English environments. If you are working with non-English data, you should use dcecp.

The rgy_edit tool views and edits information in the registry database. You can invoke rgy_edit from any node.

You can edit and view principals, groups, organization, accounts, and policies in the network registry (the default) or perform a subset of those functions on the local registry (by using the -l option). Changes made by rgy_edit apply only to the registry. They do not apply to the local override file or the local password and group files, both of which can be edited manually. You can view and change only those registry objects to which you are granted the appropriate permissions.

Invoking rgy_edit

When you invoke rgy_edit, it displays the following prompt:

rgy_edit=>

You can enter any of the rgy_edit subcommands at this prompt; rgy_edit will prompt you for the required information. Alternatively, you can enter the subcommand followed by all the options required to perform a specific operation. The rgy_edit command might prompt you for any required information you do not enter.

Subcommands Syntax

In the rgy_edit subcommands that follow, use " " (empty double quotation marks) to indicate a null fullname, password, misc, homedir, or shell. Use double quotation marks to embed spaces or dashes in fullname, misc, and homedir if you specify the argument on the command line.

Principal, Group, and Organization Subcommands

v[iew] [name | -u unix_number] [-f] [-m] [-po]

Views registry entries. Whether name applies to a principal, group, or organization depends on the domain in which you run rgy_edit. Use the do [main] subcommand (see Miscellaneous Commands) to change domains.

If you specify the -u unix_number option, rgy_edit displays all matching entries, including any aliases.

The -f option displays entries in full (all fields except the membership list and organization policy).
rgy_edit(8sec)

If you are viewing groups or organizations, -m displays the membership list. For principals, -m lists all groups of which the principal is a member, including groups that cannot appear in a project list.

If you are viewing organizations, -po displays policy information. If you do not enter the -po option, rgy_edit shows only the organization’s name and the UNIX number.

```
a[dd] [principal_name [unix_number] [-f fullname] [-al] [-q quota]]
a[dd] [group_name [unix_number] [-f fullname [-nl]]] [-al]
a[dd] [organization_name [unix_number] [-f fullname]]
```

Creates a new name entry.

If you do not specify principal_name, group_name, or organization_name, the `add` subcommand prompts you for each field in the entry. If you are adding organizations, the command prompts you for policy information as well. If you specify only principal_name, group_name, or organization_name and no other arguments, the object’s full name defaults to "" (that is, blank), the object’s UNIX number is assigned automatically, and the object’s creation quota defaults to unlimited.

For principals and groups, the -al option creates an alias entry. If `unix_number` is already assigned to a principal and you do not specify -al, an error occurs and you must either choose a different `unix_number` or specify -al. If you use -al to create an alias and `unix_number` is not already associated with a primary name, rgy_edit issues a warning but creates the alias. The -q option specifies the total number of registry objects that can be created by the principal.

The -q option specifies the principal’s object creation quota, the total number of registry objects that can be created by the principal. If you do not specify this option, the object creation quota defaults to unlimited.

For groups, the -nl option indicates that the group is not to be included on project lists; omitting this option allows the group to appear on project lists.

Use quotation marks to embed spaces (or quotation marks) in a fullname. A single space between quotation marks indicates a null fullname.

```
c[change] [principal_name [-n name] [-f fullname] [-al | -pr] [-q quota]]
c[change] [group_name [-n name] [-f fullname] [-nl | -l]] [-al | -pr]
c[change] [organization_name [-n name] [-f fullname]]
```

Changes a principal, group, or organization.

Specify the entry to change with principal_name, group_name, or organization_name. If you do not specify a principal_name, group_name, or organization_name, the change subcommand prompts you for a name. If you do not specify any fields, the subcommand prompts you for each field in succession.
To leave a field unchanged, press `<Return>` at the prompt. If you are changing organization entries in the interactive mode, the subcommand prompts you for policy information as well.

Use `-n name` and `-f fullname`, to specify a new primary name or full name, respectively.

For principals and groups, the `-al` option changes a primary name into an alias, and the `-pr` option changes an alias into a primary name. This change can be made only from the command line, not in the interactive mode.

The `-q` option specifies the total number of registry objects that can be created by the principal.

For group entries, the `-nl` option disallows the group from appearing in project lists, while the `-l` option allows the group to appear in project lists.

For organization entries, you can change policy information only in the interactive mode.

A single space between quotation marks indicates a null `fullname`.

Changes to a principal name are reflected in membership lists that contain the principal name. For example, if the principal `ludwig` is a member of the group `composers` and the principal name is changed to `louis`, the membership list for `composers` is automatically changed to include `louis` but not `ludwig`.

For reserved names, you can change only `fullname`.

```
m[ember] [group_name | organization_name [-a member_list] [-r member_list]]
```

Edits the membership list for a group or organization.

If you do not specify a group or organization, the `member` subcommand prompts you for names to add or remove.

To add names or aliases to a membership list, use the `-a` option followed by the names (separated by commas). To delete names from a membership list, use the `-r` option followed by the names (separated by commas). If you do not include either the `-a` or `-r` option on the command line, `rgy_edit` prompts you for names to add or remove.

Adding a principal to a membership list permits creation of a login account for that principal with that group or organization.

Removing names from the membership list for a group or organization has the side effect of deleting the login account for removed member (and, of course, eliminating any permissions granted as a result of the membership the next time the member’s ticket-granting ticket is renewed).

```
del [ete] pgo_name
```

Deletes a registry entry.
If you delete a principal, `rgy_edit` deletes the principal’s account. If you delete a group or organization, `rgy_edit` deletes any accounts associated with the group or organization. You cannot delete reserved names.

```bash
adopt uuid principal_name
[-u unix_number] [-f fullname] [-q object_creation_quota]
adopt uuid group_name [-u unix_number]

[-f]
fullname [-nl]
adopt uuid organization_name
[-u unix_number]
[--f fullname]
```

Creates a principal, group, or organization for the specified Universal Unique Identifier (UUID).

The UUID must be an orphan (a UUID for which no name exists in any domain). The `uuid` is hexadecimal numbers in RPC string format.

An error occurs if you specify a name or UNIX number that is already defined within the same domain of the database.

A single space between quotation marks indicates a null `fullname`.

**Account Subcommands**

```bash
v[iew] [pname [gname [oname]]] [-f]
```

Displays login accounts.

Without the `-f` option, `view` displays only the user fields in each account entry. These fields include the following for each account:

1. Principal, group, and organization name
2. Encrypted password
3. Miscellaneous information
4. Home directory
5. Login shell

With `-f`, `view` displays the full entry, including the administrative fields as well as the user fields. Administrative information includes:

1. Who created the account
2. When the account was created
3. Who last changed the account
4. When the account was last changed
5. When the account expires
6. Whether the account is valid
7. Whether the account principal’s password is valid
8. When the account principal’s password was last changed

```bash
add [pname [-g gname -o oname]
-mp password {-rp | -pw password] [-m]
```
rgy_edit(8sec)

[-h homedir] [-s shell] [-pnv | -pv] [-x account_exp | none] [-anv | -av]
([-ena[ble] option | -dis[able] option]...) [-gs
date_and_time] [-mcr lifespan]
[-mcl lifespan]]

Creates a login account.

If you enter the subcommand only or the subcommand and the optional 
(principal name) argument, rgy_edit prompts you for all information. If you enter the
subcommand, the pname argument, and the gname (group name) argument or the
the pname, gname and oname (organization name) arguments, you must also enter the
-mr, and -pw or -rp options. All other options are optional.

The pname argument specifies the principal for whom the account should be
created. The -g and -o options specify the account’s group and organization. If the
principal specified in pname is not already a member of the specified group and
organization, rgy_edit automatically attempts to add the principal to the
membership lists. If you do not have the appropriate permissions for the group and
organization, the attempt will fail and the account will not be created.

The -rp option generates a random password for the account. The primary use of
this option is to create passwords for accounts that will not be logged into (since the
random password can never be supplied.) The -pw option is used to supply a
password for the account on the command line.

If you use the -rp option or the -pw option, you must also use the -mp option to
supply your password so your identity can be validated.

If you do not specify the -rp option or the -pw option, rgy_edit prompts for the
account’s password twice to ensure you did not make a typing mistake. Then it
prompts for your password to verify your identity.

If the user’s password management policy allows the selection of generated
passwords, specifying * (asterisk) as the argument to the -pw option or at the
account’s password prompt automatically generates a plaintext password.

If the user’s password management policy requires the selection of generated
passwords, specifying the -pw option is an error; rgy_edit displays a generated
password and then prompts for the password for confirmation. The format of
password must adhere to the policy of the associated organization or the policy of
the registry as a whole, whichever is more restrictive.

The information supplied with the -m option is used to create the GECOS field for
the account in the /etc/passwd file. If you run the passwd_export command, this
entry contains the concatenation of the principal’s full name and the information
specified with the -m option. Use quotes to include spaces and hyphens.

The -h option specifies the pathname of the principal’s home directory. The default
homedir is /. The -s option specifies the pathname of the principal’s login shell. The
default shell is a null string.

Use a single space between quotation marks to indicate a null password, misc_info,
homedir, or shell.
The `-pnv` (password not valid) option specifies that the password has expired. Generally, users must change their passwords when the passwords expire. However, the policy to handle expired passwords and the mechanism by which users change their passwords are defined for each platform, usually through the login facility. The `-pv` option indicates the password is not expired (the default).

The `-x` option sets an expiration date for the account and the password of the account.

The default is `none`, meaning that the account and the password will never expire.

The `-anv` (account not valid) option specifies that the account is not currently valid for login. The `-av` option indicates the account is currently valid (the default).

The `-enable` and `-disable` options set or clear the following options:

1. The `client` option, if enabled, allows the principal to act as a client and log in, acquire tickets, and be authenticated. If you disable `client`, the principal cannot act as a client. The default is enabled.
2. The `server` option, if enabled, allows the principal to act as a server and engage in authenticated communication. If you disable `server`, the principal cannot act as a server that engages in authenticated communication. The default is enabled.
3. The `stdated` option, if enabled, allows tickets with a start time some time in the future to be issued to the account’s principal. The default is disabled.
4. The `forwardable` option, if enabled, allows a new ticket-granting ticket with a network address that differs from the present ticket-granting ticket address to be issued to the account’s principal. The default is enabled.
5. The `oxiable` option, if enabled, allows a new ticket with a different network address than the present ticket to be issued to the account’s principal. The default is disabled.
6. The `GT_authentication` option, if enabled, specifies that tickets issued to the account’s principal can use the ticket-granting-ticket authentication mechanism. The default is enabled.
7. The `renewable` option, if enabled, allows tickets issued to the principals ticket-granting ticket to be renewed for as long as its lifetime is valid. The default is enabled.
8. The `session_key` option allows tickets issued to the account’s principal to have duplicate keys. The default is disabled.

The `-gsd` (good since date) is the date and time the account was last known to be valid. When accounts are created, this date is set to the account creation time. If you change the good since date, any tickets issued before the changed date are invalid. Enter the date in the following format:

```
yy/mm/dd hh:mm
```

The `-mcr` (maximum certificate renewable) option is the number of hours before a session with the principal’s identity expires and the principal must log in again to reauthenticate. The default is 4 weeks.

The `-mcl` (maximum certificate lifetime) option is the number of hours before the Authentication Service must renew a principal’s service certificates. This is handled automatically and requires no action on the part of the principal. The default is 1 day.
Changes an account.

The -p, -g, and -o options identify the account to change. The -np, -ng, and -no options change the account’s, principal, group, and organization, respectively.

All other options have the same meaning as described in the add command for accounts. Note that the -rp option can be used to change the random passwords of the reserved accounts created by sec_create_db when the registry database is created.

de[lete] -p pname [-g gname] [-o oname]

Deletes the specified account.

Enter the -p option to delete the specified principal’s account. Enter the -g or -o option to delete accounts associated with the specified group or organization. If you enter the -g or -o option, rgyledit prompts individually for whether to delete each account associated with the group or organization.


Creates a cross-cell authentication account in the local and foreign cells.

This account allows local principals to access objects in the foreign cell as authenticated users and vice versa. The administrator in the foreign cell must have also set up a standard account, whose ID and password the administrator of the foreign cell must supply to you. This foreign account must have a non-zero quota, because it will be used to create a principal and account necessary for cross-cell authentication. You must also invoke the cell subcommand authenticated as a DCE principal who has the privileges and quota necessary to allow creation of a principal and account in the local cell registry.

The cellname variable specifies the cell to which cross-cell authentication will be established. The cellname variable should be a DCE cellname in global format.

The -ul option specifies the UNIX number for the local cell’s principal. The -uf option specifies the UNIX number for the foreign cell’s principal. If you do not specify these UNIX numbers, they are generated automatically.
The -gl and -ol options specify the local account’s group and organization. The -gf and -of options specify the foreign account’s group and organization.

The -mp option specifies the password of the person who invoked rgy_edit.

The -fa option specifies the name identifying the account in the foreign cell, and the -fp option specifies the account’s password.

The -q option specifies the object creation quota for the new cell accounts.

The -x option specifies the account expiration date for both the local and foreign accounts. The default for this option is none.

Note that the object creation quota for the local account defaults to 0 (zero), meaning that principals in the foreign cell cannot create objects in the local cell. You can change this with the change subcommand.

**Key Management Subcommands**

The key management subcommands must be run in command-line mode.

```
ktadd [dd [ ] -p principal_name [ -pw password ]
[ -a[uto] ] [ -r[egistry] ] [ -f
keyfile ]
```

Creates a password for a server or machine in the keytab file on the local node.

The -p option specifies the name of the server or machine principal for which you are creating a password.

The -pw option lets you supply the password on the command line. If you do not enter this option or the -auto option, ktadd prompts for the password.

The -a option generates the password randomly. If you use this option, you must also use the -r option. If you do not specify the -auto or the -pw option, you are prompted for a password.

The -r option updates the principal’s password in the registry to match the string you enter (or automatically generate) for the password in the keytab file. Use it to ensure that the principal’s password in the registry and the keytab file are in synch when you change a principal’s password in the keytab file. To use this option, a password for the principal must exist in the default keytab file or the keytab file named by the -f option.

The -f option specifies the name of the server keytab file on the local node to which you are adding the password. If you do not specify a keytab filename, /krb5/v5srvtab is used. You must have the appropriate access rights to add entries in this or other keytab files.

```
ktl [lst] [ -p principal_name ] [ -f
keyfile ]
```

Displays principal names and password version numbers in the local keytab file.

The -p option specifies the name of the server or machine principal for which you are displaying passwords.
rgy_edit(8sec)

The -f option specifies the name of the server keytab file on the local node for which you want to display entries. If you do not specify a keytab filename, /krb5/v5srvtab is used.

**ktd [delete] -p principal_name -v version_number [-f keyfile]**

Deletes a server or machine principal’s password entry from a keytab file.

The -p option specifies the name of the server or machine principal for whom you are deleting a password entry.

The -v option specifies the version number of the password you want to delete. Version numbers are assigned to a principal’s password whenever the principal’s password is changed. This allows any servers or machines still using tickets granted under the old password to run without interruption until the ticket expires naturally.

The -f option specifies the name of the server keytab file on the local node from which you want to delete passwords. If you do not specify a keytab filename, /krb5/v5srvtab is used. Note that you must be root to delete entries in the default keytab file. You must have the appropriate access rights to delete entries in other keytab files.

**Miscellaneous Commands**

**do [main] [p | g | o | a]**

Changes or displays the type of registry information being viewed or edited.

You can specify p for principals, g for groups, o for organizations, or a for accounts. If you supply no argument, rgy_edit displays the current domain.

**si[te] [[name]] [-u[date] | -q[uey]]**

Changes or displays the registry site being viewed or edited.

The name variable is the fully qualified name of the cell that contains the registry to which you want access. If you supply no argument, rgy_edit displays the current site.

The -u option indicates you want to edit the registry. The -q indicates that you want to only view the registry.

**prop[erties]**

Changes or displays registry properties.

This command prompts you for changes. Press <Return> to leave information unchanged.

Changes or displays registry standard policy or the policy for an organization. (Note that policies for an organization are set with the change command in the organization domain.)

This command prompts you for changes. Press <Return> to leave information unchanged.

Enter organization_name to display or change policy for that specific organization. If you do not enter organization_name the subcommand affects standard policy for the entire registry.

The -al option determines the account’s lifespan, the period during which accounts are valid. After this period of time passes, the accounts become invalid and must be recreated. An account’s lifespan is also controlled by the add and change subcommands -x option. If the two lifespans conflict, the shorter one is used. Enter the lifespan in the following in the following format:

weeks w days d hours h minutes m

For example, 4 weeks and 5 days is entered as 4w5d.

If you enter only a number and no weeks, days, or hours designation, the designation defaults to hours. If you end the lifespan with an number and no weeks, days, or hours designation, the number with no designation defaults to seconds. For example, 12w30 is assumed to be 12 weeks thirty seconds.

The -pl option determines the password lifespan, the period of time before account’s password expires. Generally, users must change their passwords when the passwords expire. However, the policy to handle expired passwords and the mechanism by which users change their passwords are defined for each platform, usually through the login facility.

Enter passwd_lifespan as a number indicating the number of days. If you define a password lifespan as forever, the password has an unlimited lifespan.

The -px option specifies the password expiration date in the following format:

yy/ mm/ dd/ hh. mm: ss

Generally, users must change their passwords when the passwords expire. However, the policy to handle expired passwords and the mechanism by which users change their passwords are defined for each platform, usually through the login facility.

If you define a password expiration date as none, the password has an unlimited lifespan.

The -pm, -ps, -pns, -pa, and -pna options all control the format of passwords as follows:

-pm Specifies the minimum length of passwords in characters. If you enter 0, no password minimum length is in effect.
rgy_edit(8sec)

-ps and -pns
Specify whether passwords can contain all spaces (-ps) or cannot be all spaces (-pns).

-pa and -pna
Specify whether passwords can consist of all alphanumeric characters (-pn) or must include some nonalphanumeric characters (-pna).

auth_policy
Changes and/or displays registry authentication policies.

This command prompts you for changes. Press <Return> to leave information unchanged.

defaults
Changes or displays the home directory, login shell, password valid option, account expiration date, and account valid option default values that rgy_edit uses.

This command first displays the current defaults. It then prompts you for whether or not you want to make changes. If you make changes, defaults immediately changes the defaults for the current session, and it saves the new defaults in $HOME $rgy_edittc. The newly saved defaults are used until you change them.

help [command]
Displays usage information for rgy_edit. If you do not specify a particular command, rgy_edit lists the available commands.

quit
Exit rgy_edit.

exit
Exit rgy_edit.

login [user_name]
Lets you log in to a local machine and assume a new local identity from within the rgy_edit session. The rgy_edit login command prompts for a principal name (if unspecified by user_name) and password.

scope [name]
Limits the scope of the information displayed by the view subcommand to the directory (specified by name) in the registry database.

Related Information

Commands: sec_admin(8sec), dcecp(8dce).
rmxcred

Purpose

Purges ticket caches whose tickets have all expired, according to the command line options provided. The default is to remove all caches whose ticket-granting tickets (TGTs) are expired as of the current time, except for those associated with the DCE daemons that perform their own periodic ticket refresh functions:

- cdssd
- dced
- secd

Synopsis

rmxcred [-h hours] [-d days [ -v ] [-f ] [ -p principal ] ]

Options

-h hours
  Specifies that rmxcred will purge only those ticket caches whose TGTs had expired by the specified number of hours ago. Can be used in conjunction with -d to specify a time using both days and hours.

-d days
  Specifies that rmxcred will purge only those ticket caches whose TGTs had expired by the specified number of days ago. Can be used in conjunction with -h to specify a time using both days and hours.

-v
  Specifies that rmxcred runs in verbose mode. This option prints out the global principal names associated with the caches that are removed.

-f
  Specifies a force option. This flag causes any ticket cache meeting the expiration criteria to be removed, even caches for the DCE daemons cdssd, dced, and secd.

-p principal
  Specifies that rmxcred removes only those expired ticket caches that are associated with the specified principal. This includes principals associated with the aforementioned DCE daemons. You can specify a shortcut of X for principals in the form hosts/any/X. The shortcut is useful for removing machine context or CDS server credentials.

Description

The rmxcred command purges ticket caches whose entire ticket sets have expired; that is, a cache is removed if its TGT has expired. This command will not remove individual tickets from caches that happen to contain both expired and unexpired tickets. The removals are subject to certain other criteria controlled by the command-line options and default behavior.

Only ticket cache files located in the credentials directory (dcelocal/var/security/creds) are eligible for removal. By default, all expired ticket caches whose TGTs are currently expired are purged, except for those caches that represent three DCE clients:

- hosts/dce_hostname/self
rmxcred

- **hosts/dce_hostname/cds-server**
- **dce-rgy**

This default protection is offered so that there is no chance of causing serious DCE daemon refresh problems in cdsc, dcsc, and secd in cases where these daemons might be having temporary problems refreshing their TGTs. This protection is also useful for maintaining evidence needed for system debug and analysis when daemons have ticket refresh problems.

You can control the expiration criteria by using the `-d` option, the `-h` option, or both. When set, ticket caches meeting all other criteria are purged only if the TGT in that cache has been expired for at least the specified amount of time.

The invoker of this command must have an effective UID of 0.

Finally, the `rmxcred` command purges cache files that represent failed login attempts. These are files that do not have companion `.data` files.

**Related Information**

Files:

- **dcecred XXXXXXXX**
  A ticket cache (XXXXXXXX represents 8 hexadecimal digits).
sec_admin

Purpose

Registry replica administration tool

Synopsis

sec_admin [-u[sage] | -v[ersion] | [-s[ite] name] [-nq]]

Options

-u[sage]
Causes command line options to be displayed.

-v[ersion]
Causes the OSF DCE version number, upon which this product is based, to be displayed.

-s[ite] name name
Causes sec_admin to bind to the replica specified by the name argument. If the option is not supplied, sec_admin binds randomly to any replica in the local cell. The name argument can be one of the following:
1. A specific cell_name (or /.: for the local cell), to bind to any replica in the named cell.
2. The global name of a replica, to bind to that specific replica in that specific cell.
3. The name of a replica as it appears on the replica list, to bind to that replica in the local cell.
4. A string binding to a specific replica. An example of a string binding is `ncadg_ip_udp:15.22.144.163`. This form is used primarily for debugging or if the Cell Directory Service (CDS) is not available.

-nq
Turns off queries initiated by certain sec_admin subcommands before they perform a specified operation. For example, the delrep subcommand deletes a registry replica; before the deletion, sec_admin prompts for verification. If you invoke sec_admin with the -nq option, the deletion is performed without prompting.

Description

Note:

With the exception of the following subcommand, this command is replaced at DCE Version 1.1 by the dcecp command. This command might be fully replaced by the dcecp command in a future release of DCE, and might no longer be supported at that time.

1. monitor

This control program has been superseded by dcecp. It is not designed for international use, and might give unexpected or undesirable results when used in non-English environments. If you are working with non-English data, you should use dcecp.
sec_admin(8sec)

The registry database is replicated: each instance of a registry server, secd, maintains a working copy of the database in virtual memory and on disk. One server, called the master replica, accepts updates and handles the subsequent propagation of changes to all other replicas. All other replicas are slave replicas, which accept only queries. Each cell has one master replica and numerous slave replicas.

Using the sec_admin command, you can:
1. View a list of replicas.
2. Delete a replica.
3. Reinitialize a replica.
4. Stop a replica.
5. Put the master replica into and out of the maintenance state.
6. Generate a new master key used to encrypt principal keys.
7. Turn the master registry into a slave registry and a slave registry into the master registry.

Note that sec_admin cannot add, delete, or modify information in the database, such as names and accounts. Use rgy_edit to modify registry database entries.

The Default Replica and Default Cell

Most sec_admin commands are directed to a default replica. When sec_admin is invoked, it automatically binds to a replica in the local cell. This replica becomes the default replica.

Identifying the Default Replica and the Default Cell

Use the site subcommand to change the default replica and, optionally, the default cell. When you use the site subcommand, you can supply the name of a specific replica, or you can simply supply the name of a cell. If you supply a cell name, sec_admin binds to a replica in that cell randomly. If you supply a specific replica name, sec_admin binds to that replica.

Specifically, you can supply any of the following names to the site subcommand:
1. A cell name. If you enter a cell name, the named cell becomes the default cell. The sec_admin command randomly chooses a replica to bind to in the named cell, and that replica becomes the default replica.
2. The global name given to the replica when it was created. A global name identifies a specific replica in a specific cell. That cell becomes the default cell and that replica the default replica.
3. The replica's name as it appears on the replica list (a list maintained by each security server containing the network addresses of each replica in the local cell). That replica becomes the default replica and the cell in which the replica exists becomes the default cell.
4. The network address of the host on which the replica is running. The replica on that host becomes the default replica, and the cell in which the host exists becomes the default cell.

Naming the Default Replica

As an example, assume that the following is true of a replica named subsys/dce/sec/rs_server_250_2:
1. It exists in the local cell .../dresden.com.
2. It has a global name of /.../dresden.com/subsys/dce/sec/rs_server_250_2.
3. It is named subsys/dce/sec/rs_server_250_2 on the replica list.
4. It runs on a host whose ip network address is 15.22.144.248.

This replica can be identified to the site subcommand in any of the following ways:
/.../dresden.com/subsys/dce/sec/rs_server_250_2
   The replica's full global name.
subsys/dce/sec/rs_server_250_2
   The replica's cell-relative name on the replica list.
ncadg_ip_udp:15.22.144.248
   The network address of the host on which the replica runs.

Naming the Default Cell

When a default replica is identified specifically, its cell becomes the default cell. In
the example in the previous section, the default cell is /.../dresden.com.

You can specify simply a cell name to the site subcommand. When this is done,
any replica in that cell is selected as the default replica.

For example, assume that the following are replicas in the cell /.../bayreuth.com:
/.../bayreuth.com/subsys/dce/sec/rs_server_300_1
/.../bayreuth.com/subsys/dce/sec/rs_server_300_2

If you enter site /.../bayreuth.com, then /.../bayreuth.com becomes the default cell
and one of the following becomes the default replica:
/.../bayreuth.com/subsys/dce/sec/rs_server_300_1
/.../bayreuth.com/subsys/dce/sec/rs_server_300_2

Automatic Binding to the Master

Some of the sec_admin subcommands can act only on the master registry and
thus require binding to the master registry. If you execute a subcommand that acts
only on the master and the master is not the default replica, sec_admin attempts to
bind to the master replica in the current default cell automatically. If this attempt is
successful, sec_admin displays a warning message informing you that the default
replica has been changed to the master registry. The master registry will then
remain the default replica until you change it with the site subcommand. If the
attempt to bind is not successful, sec_admin displays an error message, and the
subcommand fails.

Invoking sec_admin

When you invoke sec_admin, it displays the current default replica's full global
name and the cell in which the replica exists. Then it displays the sec_admin>
prompt.

sec_admin
   Default replica: /.../dresden.com/subsys/dce/sec/music
   Default cell: /.../dresden.com
sec_admin>

At the sec_admin> prompt, you can enter any of the sec_admin subcommands.
sec_admin(8sec)

Note: You must be authenticated to the DCE to invoke sec_admin.

Subcommands

The subcommand descriptions that follow use default_replica to indicate the default replica and other_replica to indicate a replica other than the default. The other_replica argument must identify a replica in the default cell. It is specified by its name on the cell’s replica list (that is, by its cell-relative name). Use the lrep subcommand to view the default cell’s replica list.

become [-master] [-slave]

The -master option makes the current default replica (which must be a slave) the master replica.

The -slave option makes the current default replica (which must be the master) a slave replica.

This method of changing to master or slave can cause updates to be lost. The change_master subcommand is the preferred means of designating a different master replica. However, you might find the become -master command useful if the master server is irrevocably damaged and you are unable to use change_master.

change_master -to other_replica

Makes the replica specified by other_replica the master replica. To perform this operation, other_replica must be a slave, and the current default replica must be the master. If the current default replica is not the master, sec_admin attempts to bind to the master. If the change operation is successful, the current master does the following:
1. Applies all updates to other_replica.
2. Becomes a slave.
3. Tells other_replica to become the master.

delr [ep] other_replica [-force]

Deletes the registry replica identified by other_replica. To perform this operation, the current default replica must be the master. If it is not, sec_admin attempts to bind to the master. If the delete operation is successful, the master does the following:
1. Marks other_replica as deleted.
2. Propagates the deletion to all replicas on its replica list.
3. Delivers the delete request to other_replica.
4. Removes other_replica from its replica list.

The -force option causes a more drastic deletion. It causes the master to first delete other_replica from its replica list and then to propagate the deletion to the replicas that remain on its list. Since this operation never communicates with the deleted replica, you should use -force only when the replica has died irrecoverably. If you use -force while other_replica is still running, you should then use the destroy subcommand to eliminate the deleted replica.

h [elp] [command]

Lists the sec_admin subcommands and shows their allowed abbreviations. If command is specified, displays help for the specified command. The -all option shows all the information.

info [-full]

Displays status information about the default replica. The info subcommand
contacts the default replica to obtain the appropriate information. If this information is not available, `info` prints the replica name and a message stating that the information is not available.

Without the `-full` option, `info` displays the following:
1. The default replica’s name and the name of the cell in which the replica exists.
2. Whether the replica is a master or a slave.
3. The date and time the replica was last updated and the update sequence number.
4. An indication of the replica’s state, as follows:
   - **Bad State**
     The state of the replica prohibits the requested operation.
   - **Uninitialized**
     The database is a stub database that has not been initialized by the master replica or another up-to-date replica.
   - **Initializing**
     The replica is in the process of being initialized by the master replica or another up-to-date replica.
   - **In Service**
     The replica is available for queries and propagation updates if it is a slave replica or queries and updates if it is the master replica.
   - **Copying Database**
     The replica is in the process of initializing (copying its database to) another replica.
   - **Saving Database**
     The replica is in the process of saving its database to disk.
   - **In Maintenance**
     The replica is unavailable for updates but will accept queries.
   - **Changing Master Key**
     The replica is in the process of having its master key changed.
   - **Becoming Master**
     The replica is in the process of becoming the master replica (applicable to slave replicas only).
   - **Becoming Slave**
     The master replica is in the process of becoming a slave replica (applicable to the master replicas only).
   - **Closed**
     The replica is in the process of stopping.
   - **Deleted**
     The replica is in the process of deleting itself.
   - **Duplicate Master**
     The replica is a duplicate master and should be deleted.

The master replica is available for queries when it is in the **In Service**, **Copying Database**, **In Maintenance**, **Changing Master Key**, and **Becoming Slave** states. It is available for updates only when it is in the **In Service** state.
A slave replica is available for queries when it is in the **In Service**, **Copying Database**, **Changing Master Key**, and **Becoming Master** states. It accepts updates from the master replica only when it is in the **In Service** state. It accepts a request from the master replica to initialize only when it is in the **Uninitialized** or **In Service** state.

The **-full** option displays all the above information and the following information as well:

1. The default replica’s unique identifier.
2. The replica’s network addresses.
3. The unique identifier of the cell’s master replica.
4. The network addresses of the cell’s master replica.
5. The master sequence number, which is the sequence number of the event that made the replica the master.
6. If the replica is the master, the update sequence numbers that are still in the propagation queue and have yet to be propagated.

**inir [ep]** _other_replica_

Reinitializes a replica by copying an up-to-date database to _other_replica_. The master replica initiates and guides the operation. If the operation is successful, the following actions take place:

1. The master replica does the following:
   a. Marks _other_replica_ for reinitialization.
   b. Tells _other_replica_ to reinitialize itself.
   c. Gives _other_replica_ a list of replicas with up-to-date databases.
2. The _other_replica_ picks a replica from the list and asks that replica to initialize it (that is, to copy its database to _other_replica_).

To perform this operation, _other_replica_ must be a slave, and the current default replica must be the master. If the current default replica is not the master, **secadmin** attempts to bind to the master.

This subcommand is generally not used under normal conditions.

**lr [ep] [-s[tate]] [-u[uid]] [-a[addr]] [-p[rop]] [-a[l]]**

Lists the replicas on the default replica’s replica list.

If you enter no options, the display includes the replica name and whether or not it is the master replica. In addition if the master replica’s list is being displayed, slave replicas marked for deletion are noted. With options, the display includes this information and the information described in the following paragraphs.

The **-state** option shows each replica’s current state, the date and time the replica was last updated, and the update sequence number. To obtain this information, **lrep** contacts each replica. If this information is not available from the replica, **lrep** prints the replica name and a message stating the information is not available.

The **-addr** option shows each replica’s network addresses. The **-uuid** option shows each replica’s unique identifier. The **-prop** option shows the following:

1. The date and time of the last update the master sent to each slave replica.
2. The sequence number of the last update to each slave replica.
3. The number of updates not yet applied to each slave replica.
4. The status of the master replica’s last communication with each slave replica.
5. The propagation state of each slave replica. This state illustrates how the master replica views the slave replica, can be any of the following:

   **Bad State**
   The state of the replica prohibits the requested operation.

   **Marked for Initialization**
   The replica has been marked for deletion by the master replica.

   **Initialized**
   The replica has been marked for initialization by the master replica.

   **Initializing**
   The replica is in the process of being initialized by the master replica.

   **Ready for Updates**
   The replica has been initialized by the master replica and is now available for propagation updates from the master replica.

   **Marked for Deletion**
   The replica has been marked for deletion by the master replica.

This information is obtained from the master replica; the slave replicas are not contacted for this information.

The `-prop` option is valid only for the master.

For slave replicas, the `-all` option shows all the information above except that displayed by the `-prop` option. For the master replica, the `-all` option shows all the information.

**mas [ter_key]**
Generates a new master key for the default replica and reencrypts account keys using the new key. The new master key is randomly generated. Each replica (master and slaves) maintains its own master key used to access the data in its copy of the database.

**monitor [-r n]**
Periodically lists the registry replicas stored in the current default replica’s replica list. The list includes each replica’s current state, the date and time the replica was last updated and the update sequence number.

The `monitor` subcommand contacts each replica to obtain the information it displays. If this information is not available from the replica, `monitor` prints the replica name and a message stating the information is not available.

The `-r` option causes the replicas to be listed at intervals you specify. The `n` argument is a number of minutes between intervals. The default is 15 minutes.

**destroy default_replica**
Destroys the current default replica. To perform this operation, the current default replica and the default replica you name as `default_replica` must be the same. This is to confirm your desire to perform the deletion.
sec_admin(8sec)

If the operation is successful, the default replica deletes its copy of the registry database and stops running. This subcommand does not delete default_replica from the replica lists. Use the delrep -force subcommand to delete the replica from the other replica lists.

The preferred way to delete replicas is to use the delrep subcommand. However, the destroy subcommand can be used if delrep is unusable because the master is unreachable or the replica is not on the master’s replica list.

site [name [-u[update]]]
Sets or displays the default cell and the default replica.

The name argument identifies the replica to set as the default replica and, as a consequence, the default cell. It can be one of the following:

1. A specific cell_name (or /.: for the local cell) to make any replica in the named cell the default.
2. The global name of a replica to make the specified replica in the specified cell the default.
3. The name of a replica as it appears on the replica list to make the named replica (which exists in the default cell) the default replica.
4. A string binding to a specific replica. An example of a string binding is ncadg_ip_udp:15.22.144.163. This form is used primarily for debugging or if CDS is not available.

The -u option specifies that sec_admin should find the master replica. Normally you specify the name of a cell for name in conjunction with the -u option. In this case sec_admin finds the master replica in that cell. If you use a replica name for name, sec_admin queries the named replica to find the master replica in the named replica’s cell.

If you supply no arguments, sec_admin displays the current default replica and default cell.

stop
Stops the security server (secd) associated with the default replica.

sta [te] -maintenance | -service
Puts the master replica into maintenance state or takes it out of maintenance state. This subcommand is useful for performing backups of the registry database.

If the current default replica is not the master, sec_admin attempts to bind to the master.

The -maintenance flag causes the master replica to save its database to disk and refuse any updates.

The -service flag causes the master replica to return to its normal “in service” state and start accepting updates.

e [xit] or q [ui t]
Ends the sec_admin session.

Examples

1. The following example invokes sec_admin and uses the lrep subcommand to list replicas on the replica list and their states:

   /opt/dcelocal/bin/sec_admin
   Default replica: \

IBM DCE Version 3.1 for AIX and Solaris: Administration Commands Reference
sec_admin(8sec)

../../dresden.com/subsys/dce/sec/rs_server_250_2
Default cell: ../../dresden.com

sec_admin> lrep -st
Replicas in cell ../../dresden.com
(master) subsys/dce/sec/master
state: in service
Last update received at: 1993/11/16.12:46:59
Last update's seqno: 0.3bc
subsys/dce/sec/rs_server_250_2
state: in service
Last update received at: 1993/11/16.12:46:59
Last update's seqno: 0.3bc
subsys/dce/sec/rs_server_250_3
state: in service
Last update received at: 1993/11/16.12:46:59
Last update's seqno: 0.3bc

2. The following example sets the default replica to the master in the local cell:

sec_admin> site ./:. -u
Default replica: ../../dresden.com/subsys/dce/sec/master
Default cell: ../../dresden.com

Related Information

Commands: rgy_edit(8sec), dtscp(8dts).
sec_create_db

Purpose

Registry database creation utility

Synopsis

```
```

Options

The following options can be used with sec_create_db. If you specify the -slave option, you can specify only the -my[name], -k[eyseed], and -v[erbose] options.

- **-h[elp]**
  Returns help information for available options.

- **{-master | -slave}**
  Specifies whether the database for the master replica should be created (-master) or a database for a slave replica should be created (-slave). All other sec_create_db options can be used with the -master option. Only the -my[name], -keyseed, and -verbose options can be used with the -slave option.

- **-my[name]**
  Specifies the name that will be used by the Directory Service to locate the machine on which the cell’s Security Server is running.

- **-creator**
  Specifies the principal name of the initial privileged user of the registry database (known as the registry creator).

- **-cunix_id**
  Specifies the UNIX ID of the initial privileged user of the registry database. If you do not enter the UNIX ID, it is assigned dynamically.

- **-g[group_low_unix_id]**
  Specifies the starting point for UNIX IDs automatically generated by the Security Service when groups are added with the rgy_edit command.

- **-keyseed**
  Specifies a character string used to seed the random key generator in order to create the master key for the database you are creating. It should be a string that cannot be easily guessed. The master key is used to encrypt all account passwords. Each instance of a replica (master or slave) has its own master key. You can change the master key using the sec_admin command.

- **-m[ax_unix_id]**
  Specifies the highest UNIX ID that can be assigned to a principal, group, or organization.
Specifies the starting point for UNIX IDs automatically generated by the Security Service when organizations are added with the \texttt{rgy\_edit} command.

\textbf{-p[assword]}

The default password assigned to the accounts created by \texttt{sec\_create\_db}, including the account for the registry creator. If you do not specify a default password, \texttt{-dce-} is used. (Note that the \texttt{hosts /local\_host/self none none}, \texttt{krbtgt /cell\_name none none}, and \texttt{everybody none none} accounts are not assigned the default password, but instead a randomly generated password.)

\textbf{-p[erson\_low\_unix\_id]}

Specifies the starting point for UNIX IDs automatically generated by the Security Service when principals are added with the \texttt{rgy\_edit} command.

\textbf{-u[uid]}

Specifies the cell’s UUID. If you do not enter this UUID, it is assigned dynamically.

\textbf{-v[erbose]}

Specifies that \texttt{sec\_create\_db} runs in verbose mode and displays all activity.

\section*{Description}

The \texttt{sec\_create\_db} tool creates new master and slave databases in \texttt{dcelocal/var/security/rgy\_data} on the machine from which \texttt{sec\_create\_db} is run. Normally, these databases are created only once by the system configuration tool, \texttt{dce\_config}. However, you can use \texttt{sec\_create\_db} if you need to re-create the master or a slave database from scratch. You must be root to invoke \texttt{sec\_create\_db}.

The \texttt{sec\_create\_db -master} option creates the master database on the machine on which it is run. This database is initialized with names and accounts, some of them reserved. You must use the \texttt{rgy\_edit} command to populate the database with objects and accounts.

When the master registry database is created, default ACL entries for registry objects are also created. These entries give the most privileged permission set to the principal named in the \texttt{-cr[reator]} option. If the principal is not one of the reserved names and accounts, \texttt{sec\_create\_db} adds it as a new principal and adds an account for that new principal. If the \texttt{-cr} option is not used, root is the creator.

The \texttt{sec\_create\_db -slave} option creates a slave database on the machine on which it is run. This command creates a stub database on the local node in \texttt{dcelocal/var/security/rgy\_data} and adds the newly created replica to the master’s replica list. The master then marks the replica to be initialized when a Security Server is started on the slave’s node.

The \texttt{sec\_create\_db} command also creates a registry configuration file, named \texttt{dcelocal/etc/security/pe\_site}, that contains the network address of the machine on which the database is created. This file supplies the binding address of the \texttt{secd} master server if the Naming Service is not available.
sec_create_db(8sec)

Files

/dcelocal/etc/security/pe_site
The file containing the network address of the machine on which the security database is created.

/dcelocal/var/security/rgy_data
The directory in which the registry database files are stored.

Related Information

Commands: secd(8sec), sec_admin(8sec)
**secd**

**Purpose**

The DCE security server

**Synopsis**

```
secd [-b[ootstrap]] [-lockpw] [-locksm pname] [-rem[ote]] [-cpi time]
[-restore_master] [-v[erbose]] [-d [ebug]] [-t [hreads] count]
```

**Options**

- **-locksm[ith]**
  Restart the master security server in locksmith mode. Use this mode if you cannot access the registry as the principal with full registry access, because that principal's account has been inadvertently deleted or its password lost.

- **-lockpw**
  Prompt for a new locksmith password when running in locksmith mode. This option allows you to specify a new password for the locksmith account when the old one is unknown.

- **-rem[ote]**
  Allows the locksmith principal to log in remotely. If this option is not used, the principal must log in from the local machine on which secd will be started.

- **-bo[otstrap]**
  Always waits only one minute between tries to export binding information to the Cell Directory Service (CDS) during DCE configuration. If you do not specify this option, during initialization secd sleeps for 1 minute if CDS is not available when it tries to export binding information. If the export fails a second time, it sleeps for 2 minutes before it tries again. If it still fails, it sleeps for 4, 8, and 16 minutes between retries. Then, sleep time stays at 16 minutes until the binding export succeeds.

- **-cpi time**
  The checkpoint interval for the master registry database. This is the interval in seconds at which the master will read its database to disk. The default is one hour.

- **-restore_master**
  Marks all slave replicas for initialization during the master restart. Use this option only to recover from a catastrophic failure of the master security server (for example, if the database is corrupted and then restored from a backup tape).

- **-v[erbose]**
  Runs in verbose mode.

- **-d[ebug]**
  This option causes secd to run in the foreground and is normally used in conjunction with the -verbose option.

- **-t[hreads] count**
  Specifies the number of listener-threads created to service secd client requests. If this option is not specified, a default of 5 listener-threads are created. The count used must be between the range of 5 and 15, inclusive.
The secd option provides a tuning mechanism for increasing the number of requests secd can have pending at any instance. Since a 64K stack pool is reserved per listener-thread, the amount of RAM on the server machine should be taken into consideration when this option is specified.

All options start the security server on the local node.

Arguments

pname

The name of the locksmith principal. If no registry account exists for this principal, the security server creates one.

Description

The secd daemon is the security server. It manages all access to the registry database. You must have root privileges to invoke secd.

The security server can be replicated, so that several copies of the registry database exist on a network, each managed by a secd process. Only one security server, the master replica, can perform database update operations (such as adding an account). Other servers, the slave replicas, can perform only lookup operations (such as validating a login attempt).

A DCE host daemon (dced) must be running on the local node when secd is started. Typically, dced and secd are started at boot time. The secd server places itself in the background when it is ready to service requests.

Locksmith Mode

The secd -locksmith option starts secd in locksmith mode. The -locksmith option can be used only with the master replica. In locksmith mode, the principal name you specify to secd with pname becomes the locksmith principal. As the locksmith principal, you can repair malicious or accidental changes that prevent you from logging in with full registry access privileges.

If no account exists for pname, secd establishes one and prompts you for the account’s password. (Use this password when you log into the account as the locksmith principal.) If an account for pname exists, secd changes the account and policy information as described in the tables that follow. The first shows locksmith account changes; server; the second shows registry policy changes. These changes ensure that even if account or registry policy was tampered with, you will now be able to log into the locksmith account.

In locksmith mode, all principals with valid accounts can log in and operate on the registry with normal access checking. The locksmith principal, however, is granted special access to the registry: no access checking is performed for the authenticated locksmith principal. This means that, as the locksmith principal, you can operate on the registry with full access.

<table>
<thead>
<tr>
<th>If the security server finds</th>
<th>It changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password-Valid flag is set to no</td>
<td>Password-Valid flag to yes</td>
</tr>
<tr>
<td>Account Expiration date is set to less than the current time plus one hour</td>
<td>Account Expiration date to the current time plus one hour</td>
</tr>
<tr>
<td>Client flag is set to no</td>
<td>Client flag to yes</td>
</tr>
</tbody>
</table>
If the security server finds | It changes
---|---
Account-Valid flag is set to no | Account-Valid flag to yes
Good Since date is set to greater than the current time | Good Since date to the current time
Password Expiration date is set to less than current time plus one hour | Password Expiration date to the current time plus one hour
Account Lifespan is set to less than the difference between the locksmith account creation date and the current time plus one hour | Account Lifespan to the current time plus one hour minus the locksmith account creation date
Password Expiration date is set to greater than the time the password was last changed but less than the current time plus one hour | Password Expiration date to the current time plus one hour

Use the **-lockpw** option if the locksmith account exists but you do not know its password. This option causes **secd** to prompt for a new locksmith password and replace the existing password with the one entered.

Use the **-remote** option to allow the locksmith principal to log in from a remote machine.

The **secd** process normally runs in the background. When you start **secd** in locksmith mode, it runs in the foreground so that you can answer prompts.

**Examples**

All of the commands shown in the following examples must be run by **root**.

1. Start a security server after you create the database with **sec_create_db** as follows:

   ```
dcelocal/bin/secd
   ```

2. Start the security server in locksmith mode and allow the **master_admin** principal to log in on a remote machine with the following command:

   ```
dcelocal/bin/secd -locksmith
   master_admin -remote
   ```

**Related Information**

Commands: **dcecp(8dce)**, **dced(8dce)**.
secd(8sec)
Chapter 6. IBM DCE 3.1 for AIX and Solaris Configuration Commands

This chapter contains the DCE configuration commands. The commands described are:

config.dce
  Configures the DCE components.

start.dce
  Starts the DCE daemons configured on the local machine.

stop.dce
  Displays the DCE and DFS components configured on the local machine.

chpesite
  Updates the contents of the /opt/dcelocal/etc/security/pe_site file.

clean_up.dce
  Cleans the DCE databases, sockets, and cache files, creates backup log files, and removes DCE-generated core files.

dceback
  Backs up and restores DCE data.

dcesetup (Solaris only)
  Installs, uninstalls, configures and unconfigures DCE.

kerberos.dce
  Enables existing DCE clients and servers to use Kerberos.

mkreg.dce
  Adds information about a DCE cell into the domain namespace.

mkdceweb
  Configures DCE Administration, DCE Web Secure, or both into a Netscape FastTrack 3.01 or Enterprise 3.61 web server.

ps.dce
  Display process information for partially or fully configured DCE components.

rmdceweb
  Unconfigures DCE Administration, DCE Web Secure, or both from a Netscape FastTrack 3.01 or Enterprise 3.61 web server.

rmreg.dce
  Removes information about a DCE cell from the domain namespace.

unconfig.dce
  Removes configuration of the DCE components.

Note: For information about support of user-supplied configuration commands, see IBM DCE Version 3.1 for AIX: Quick Beginnings.
Purpose

Configures the DCE components.

Synopsis

config.dce
[-admin_pwd password]
[-autostart {yes | no}]
[-cache_lifetime minutes]
[-cds_replica_list "list_of_cds_servers"]
[-cds_server cds_server]
[-cell_admin cell_admin_id]
[-cell_admin_unix UNIX_id]
[-cell_name cell_name]
[-certificate_based_login {yes | no}]
[-clean_autostart {yes | no}]
[-clr_house_server_id]
[-config_type {full | local | admin}]
[-courier_role {courier | noncourier | backup}]
[-dce_hostname dce_hostname]
[-group_rsp_path filename]
[-host_id machine_identifier]
[-kdc_profile kdc_profile]
[-kdc_ini_file kdc_ini_file]
[-kdc_passphrase kdc_passphrase]
[-lan_profile profile]
[-ldap_server ldap_server | ldap_server:port_number]
[-max_unix_id max_UNIX_id]
[-min_group_id min_group_id]
[-min_org_id min_org_id]
[-min_principal_id min_principal_id]
[-no_pesite_update]
[-nsid_pwd nsid_password]
[-num_dce_unixd number] (AIX only)
[-pesite_update_time update_time]
[-protocol {tcp udp}]
[-proxy]
[-pwdstr_arg command_line_args]
[-pwdstr_cmd server_name]
[-pwdstr_principal password_strength_principal_id]
[-rsp_file filename]
[-sec_master security_server]
[-sec_server_name security_server_name]
[-sync_clocks {yes | no}]
[-time_server server_id]
[-wrap_audit_trail {yes | no}]
[usage]
[-?] [help] [operations] [components]
Notes:

1. The command can recognize unique abbreviated option strings. For example, 
   –adm is recognized as –admin_pwd. Ensure that the abbreviated strings are unique, –min would not be recognized because there are three options
   (–min_group_id, –min_org_id, and –min_principal_id) that begin with that string. –min_g, –min_o, and –min_p, however, would be recognized because they are unique.

2. Any options that are specified and are not needed for any of the components being configured, are ignored.

Configuring Clients

To Admin Configure a Client:
```
config.dce –config_type admin –host_id machine_identifier
```

To Locally Configure a Client:
```
client_components
```

To Fully Configure a Full Client:
```
client_components
```

To Configure a Slim Client:
```
config.dce [–config_type {full | local}] [–cell_name cell_name] [–dce_hostname dce_hostname] [–sec_master master_security_server] [–cds_server cds_server] [wrap_audit_trail {yes | no}]
```

Note: The default configuration type is full. For further information on the different configuration types, see the DCE Administration Commands Reference.

Configuring Servers

To Configure a Master Security Server:
```
```
To Configure a Security Replica:

```
sec_srv
```

Note: The `config.dce` command deliberately replicates the `/subsys/dce/sec` directory when it configures a secondary CDS server. During the configuration of a Security Replica, entries are created in this directory but they might not be immediately propagated to the CDS secondary servers. Since these entries are referenced during subsequent pieces of the Security Replica configuration, failures can occur. To prevent this type of failure, stop all `cdsd` daemons that are running on secondary CDS servers before configuring a Security Replica into the cell. After the successful configuration of the security replica, restart the `cdsd` daemons.

To Configure an Initial Cell Directory Service (CDS) Server:

```
cds_srv
```

To Configure an Additional CDS Server:

```
cds_second
```

To Configure a DTS Server:

```
dts_local | dts_global
```

To Configure a Global Directory Agent:

```
hag
```
To Configure an Event Management Server:

```bash
config.dce [–cell_name cell_name] [–cell_admin cell_admin_id]
[–sec_master security_server] [–cds_server cds_server]
[–cds_replica_list "list_of_cds_servers"] [–lan_profile profile]
[–no_pesite_update] [–pesite_update_time update_time]
[–autostart {yes | no}] [–clean_autostart {yes | no}]
[–protocol {tcp udp}] [–sync_clocks {yes | no}]
[–time_server server_id] [–group rsp_path filename] [–rsp_file filename]
[–wrap_audit_trail {yes | no}] gda_srv
```

To Configure a Simple Network Management Protocol Agent Server:

```bash
config.dce [–cell_admin cell_admin_id] [–admin_pwd password]
[–autostart {yes | no}] [–clean_autostart {yes | no}] snmp_srv
```

To Configure an Audit Server:

```bash
config.dce [–cell_name cell_name] [–sec_master security_server]
[–cds_server cds_server] [–cds_replica_list "list_of_cds_servers"]
[–lan_profile profile] [–no_pesite_update] [–pesite_update_time update_time]
[–autostart {yes | no}] [–clean_autostart {yes | no}]
[–protocol {tcp udp}]
[–sync_clocks {yes | no}] [–time_server server_id] [–group rsp_path filename]
[–rsp_file filename] [–wrap_audit_trail {yes | no}] audit
```

To Configure a Password Strength Server:

```bash
config.dce [–cell_name cell_name] [–cell_admin cell_admin_id]
[–admin_pwd password] [–sec_master security_server]
[–cds_server cds_server] [–cds_replica_list "list_of_cds_servers"]
[–lan_profile profile] [–no_pesite_update] [–pesite_update_time update_time]
[–pwdstr_arg command_line_args] [–pwdstr_cmd server_name]
[–pwdstr_principal password strength principal id]
[–pwdstr_protect_level { pktinteg | cdmf | pktprivact }]
[–autostart {yes | no}] [–clean_autostart {yes | no}]
[–protocol {tcp udp}]
[–sync_clocks {yes | no}] [–time_server server_id]
[–group rsp_path filename] [–rsp_file filename]
[–wrap_audit_trail {yes | no}] pw_strength_srv
```

To Configure an Identity Mapping Server:

The Identity Mapping server must be configured on the same machine as either a Security Master server or a Security Replica server. Use the command to configure the appropriate security server, and add the idms_srv component.

### Options

**–admin_pwd password**

Specifies the cell administrator password. Caution should be used with this option because of the security risk it poses by making this password accessible to others.

**–autostart {yes|no}**

Specifies that the configured components should be started at machine boot. (An `rc.dce` entry is placed in `/etc/inittab`.) (The appropriate links are created in the `/etc/init.d` and `/etc/rc[0-3].d` directories.)

**–cache_lifetime minutes (AIX only)**

Specifies the integrated login cache lifetime.
config.dce

--cds_replica_list "list_of cds_servers"
A quoted list of CDS server IP host names or addresses. This information is used to populate the CDS cache with the identity of other CDS servers in the cell.

--cds_server cds_server
Specifies the TCP/IP hostname or the TCP/IP address of a CDS server. If the local machine is separated from all CDS servers by a router or a gateway that does not pass broadcast packets, a CDS server must be specified using the --cds_server option or CDS cannot be configured. This option should be used for all components except rpc, the initial sec_srv, snmp_srv, and the initial cds_srv.

--cell_admin cell_admin_id
Specifies the name of the cell administrator account. When configuring the master Security server (the sec_srv component), the config.dce command gives this account privileges throughout the cell. Otherwise, the account named must have sufficient privilege to perform configuration tasks within the cell. If the --cell_admin option is not specified, the account cell_admin will be assumed. The value for cell_admin is used by all components except rpc, snmp_srv, audit, and dce_unixd.

--cell_admin_unix_id UNIX_id
The UNIX ID for the cell administrator (registry creator).

--cell_name cell_name
Specifies the name of the DCE cell into which the machine should be configured. If the --cell_name option is not specified, the config.dce command uses the cell name in the file /opt/dcelocal/etc/dce/dce_cf.db. A value for cell_name is required by all components except snmp. The value can either be in the form of /.../cellname or cellname.

--certificate_based_login {yes | no}
Enables or disables certificate based login.

--clean_autostart {yes | no}
Specifies whether to run the clean_up.dce script before auto-starting DCE.

--clr_house server id
Specifies an additional CDS server clearinghouse name.

--config_type {full | local | admin}
Allows the cell administrator to split configuration by specifying admin, local, or full configuration of clients within the DCE cell. The --config_type option has three available config_types:

admin Indicates the admin portion of client configuration. This updates the namespace and security registry with information about the new client.

The admin piece of configuring a client requires the cell administrator to run the config.dce command from a machine within the existing cell. It should not be run from the new client machine. The cell administrator does not need root user authority to run the admin portion of configuration.

Note: When config.dce is called with --config_type admin, the --host_id option is also required. The --host_id option can be in the form of a TCP/IP address or host name (with or without the domain). The --dce_hostname flag is optional. If
both flags are used and the machine identifier (–host_id) is in the form of a TCP/IP host name, host is called to get the IP address.

**local** Indicates the local portion of client configuration. This creates necessary files on the local machine and starts the daemons for the new client.

If the admin piece of **config.dce** has not yet been run, the local piece will fail when trying to contact the cell. In addition the user must have root authority on the machine, and does not need to have any authority in the DCE cell.

**Note:** When **config.dce** is called with **–config_type local**, the **–dce_hostname** option should be used with the same name that the cell administrator specified during the admin configuration. If the option is not used, the dce_hostname will be presumed to be the same as the name of the machine (including the domain, as returned from a call to the **host** command). If the name is not the same as the dce_hostname the cell administrator used when setting up the client, the configuration will fail.

**full** Indicates full configuration. This is the default. Full configuration includes both admin and local configuration steps. The DCE cell administrator must have root authority on the local machine being configured into the cell. If the **–config_type** option is not used, a full configuration will be assumed.

**–courier_role {courier | noncourier | backup}**

Specifies the interaction the server should have with the global servers in the cell when configuring a DTS server (**dts_local** or **dts_global** components). The **–courier_role** option must have one of the following values:

**courier**

The local server synchronizes with the global set of servers.

**noncourier**

The local server does not synchronize with the global set of servers.

**backup**

The local server becomes a courier if none are available on the local area network (LAN). This is the default.

**–dce_hostname**

Specifies the identifying name within the cell of the machine being configured. This can be the same as the TCP/IP hostname, but does not have to be. If the **–dce_hostname** is not used, the dce_hostname will default to the long TCP/IP hostname (hostname.domain) of the local machine. When **config.dce** is called with **–config_type local**, the **–dce_hostname** option should also be used with the same dce_hostname used by the cell administrator when **config.dce** was called with **–config_type admin** to configure the client machine. Otherwise, the configuration will fail. If the cell administrator does not use the **–dce_hostname** flag for the admin portion of configuration, the client is not required to use it either.
-group_rsp_path \textit{filename}
    Specifies a directory path to use when searching for included response files.

-host_id \textit{machine identifier}
    Specifies the TCP/IP hostname or the TCP/IP address of the client machine being admin configured. When \texttt{config.dce} is called with \texttt{--config_type admin}, the \texttt{--host_id} option must also be used. Admin configuration can be used for a machine whose TCP/IP address is not yet registered with a nameserver. In that situation, use the \texttt{--dce_hostname dce_hostname} option with the \texttt{--host_id IP_address} option.

\textbf{Note:} The \texttt{--host_id} option can be used only with the \texttt{--config_type admin} option.

-kdc_profile \textit{profile}
    Specifies the full pathname of the Entrust user's profile.

-kdc_ini_file \textit{kdc_ini_file}
    Specifies the full pathname of the Entrust initialization file.

-kdc_passphrase \textit{kdc_passphrase}
    Specifies the password associated with the Entrust profile for the Security Server.

-lan_profile \textit{profile}
    Specifies the name of the LAN profile this machine should use. If the profile does not yet exist, it is created. The default is ./\texttt{lan-profile}.

-ldap_server \textit{LDAP server}
    Specifies the TCP/IP host name or the TCP/IP address of an LDAP server.

-max_unix_id \textit{max_UNIX_id}
    Specifies the highest UNIX ID that can be assigned to principals, groups, or organizations by the Security service. The default is 2,147,483,647.

-min_group_id \textit{min_group_id}
    Specifies the starting point (minimum UNIX ID) for UNIX IDs automatically generated by the Security service when groups are added with the \texttt{dcecp} command. The default is 100.

-min_org_id \textit{min_org_id}
    Specifies the starting point for UNIX IDs automatically generated by the Security service when organizations are added with the \texttt{dcecp} command. The default is 100.

-min_princ_id \textit{min_principal_id}
    Specifies the starting point (minimum UNIX ID) for UNIX IDs automatically generated by the Security service when principals are added with the \texttt{dcecp} command. The default is 100.

-no_pesite_update
    Specifies that DCED will not update the pe_site file automatically.

-nsid_pwd
    Specifies the password for the NSI Gateway.

-num_dce_unixd
    Specifies the number of integrated login daemons.

-pesite_update_time \textit{pe_site update interval}
    Specifies the amount of time, in minutes, that DCED will wait between updates to the pe_site file.


--protocol {tcp udp}
Specifies which communication protocols to support. Valid values are tcp and udp. If you will configure any DFS components, you must use udp or tcp when you configure DCE.

--proxy
Specifies that the CDS client is to act as a CDS proxy.

--pwdstr_principal password_strength principal id
Specifies a principal ID for the password-strength server to run under. For a DCE principal ID, the password-strength server will use the credentials of the principal. Config.dce will create the server principal, keytab file, and CDS bindings based on the specified password strength principal id. If you specify a password strength principal id other than the default, pwd_strengthd, you must also specify -pwdstr_arg "s password strength principal id" along with the -pwdstr_cmd pwd_strengthd option.

--pwdstr_cmd server name
Specifies the name of the password strength server daemon. The default name is pwd_strengthd. When creating password-strength servers, it is important to remember that the server daemon should have sufficient owner and group permissions to perform its tasks. For example, if the password-strength server requires read access to /etc/security, then the user ID it runs under might need to belong to the security group.

--pwdstr_arg command line args
Specifies one or more command line arguments to be passed to the password strength server. If more than one argument is passed, double-quotation marks should be used.

--rsp_file filename
Specifies the full path name of a response file to use for configuration.

--sec_master master_security_server
Specifies the host ID of the Master Security server. You can use the TCP/IP host name or the TCP/IP address of the master Security server for this option. If the server is not specified, an attempt will be made to locate the master Security server using the Cell Directory Services (CDS). If the master Security server cannot be located, it must be specified using the --sec_master option or security cannot be configured. The --sec_master option is also needed when configuring a Security replica.

--sec_server_name security_server_name
Specifies the name to be given to the Security Server. The default name dce_hostname will be used if a Security Server is configured without specifying a name with the --sec_server_name option. Each Security Server must have a unique name within the cell. Using the default name helps ensure this uniqueness.

--sync_clocks {yes | no}
Specifies that this machine clock should be synchronized with the clock on a time server already in the cell.

--time_server server_id
Specifies the TCP/IP hostname or the TCP/IP address of a time server that can be used to synchronize clocks. If not specified, an attempt will be made to locate the DTS server using the Cell Directory Services (CDS). If a DTS server cannot be located, a DTS server must be specified using the --time_server option or the clocks cannot be synchronized.
`config.dce`

```
--wrap_audit_trail {yes | no}
   Specifies whether the audit trail should wrap.

usage Displays a help message.
--? Displays a help message.
help Displays a brief description for the passed arguments.

operations
   Lists all the options and the components.

components
   Specifies the components to be stopped.

   The Client Components are:
   all_cl   All clients (cds_cl, dts_cl, rpc, and sec_cl).
   client   Same as all_cl.
   cds_cl   CDS client.
   core     Single-machine cell components including cds_srv, sec_srv,
             cds_cl, sec_cl, and rpc.
   dts_cl   DTS client. This component and dts_local and dts_global are
            mutually exclusive.
   rpc      RPC daemon.
   sec_cl   Security client.
   slim_cl  Slim client.
   dce_unixd Integrated login (AIX only).
   nsswitch Name Service Switch (Solaris only).
   pam      Pluggable Authentication Module (Solaris only).

   The Server Components are:
   audit    Audit daemon.
   cds_second Secondary CDS server. This component and cds_srv are mutually
              exclusive.
   cds_srv  Initial CDS server for the cell. This component and cds_second are
            mutually exclusive.
   core_srv Single-machine cell components This is equivalent to including
              cds_srv, sec_srv, cds_cl, sec_cl, and rpc.
   dts_global DTS global server. This component and dts_local and dts_cl are
              mutually exclusive.
```
The `config.dce` command configures and starts the specified DCE components. This command also configures and starts any prerequisite client components. The `config.dce` command configures only the core DCE components. Use the `config.dfs` command to configure DFS components.

**Note:** If you configure the DCE cell using an X.500 style name and you are running DFS, you will not be able to access the local cell DFS file space unless GDS is also configured.

You can configure a machine into a cell in two ways:

**full configuration**
- used by the cell administrator (as root user) to complete all the configuration steps within the cell (updating the CDS namespace and the security registry) and on the local machine (creating files and starting daemons). Full configuration is specified with the `--config_type full` option. Full configuration is the default. If `--config_type` is not specified, a full configuration is performed.

**split configuration**
- breaks the configuration tasks into two distinct segments, admin and local. Admin configuration is used by the cell administrator from a machine currently configured in the cell. The administrator uses split configuration to update the CDS namespace and the security registry with necessary information about the client. Local configuration allows the root user of the new client to create files that are local to the system and to start the DCE client daemons. Split configuration is specified with the `--config_type admin` and `--config_type local` options.
Full configuration must be used for all servers. It can also be used for clients. Usually, the cell administrator does not have root user access for all the machines that are going to be configured into the cell as client machines. In this situation, split configuration is the option to use.

The admin portion must be run before the local portion can be run successfully. When \texttt{config.dce} is called with \texttt{--config_type admin}, the \texttt{--host_id} option is also needed to identify the machine to be configured as a client. The \texttt{--host_id} option can be in the form of a TCP/IP address or a TCP/IP hostname with or without the domain. The \texttt{--dce_hostname} flag is optional if the cell administrator wishes to specify the dce_hostname of the client machine. If the \texttt{--host_id IP_address} option is used without the \texttt{--dce_hostname} option, the dce_hostname will be presumed to be the same as the TCP/IP host name of the machine (including the domain as returned from a call to the \texttt{host} command).

A cell administrator might want to configure new clients into a cell before actually having the client machines available or before the host name and IP address are registered in the name server. The \texttt{config.dce --config_type admin} command, using the \texttt{--host_id IP_address --dce_hostname hostname} options will allow the namespace and security registry information to be updated without any calls to the nameserver for a machine not yet registered.

When \texttt{config.dce} is called with \texttt{--config_type local}, it is important that the client use the same dce_hostname used during the admin configuration. If the dce_hostname is not the same, the configuration will fail. (If the cell administrator did not use the \texttt{--dce_hostname} option, it is not necessary to use it for the local configuration.) If the cell name is not provided, a call to \texttt{getcellname} will determine if the local machine is already part of a cell. If it is, the assumption is made that additional client components are to be configured on this machine (for example, to add a CDS client to a machine with only a security client). If a cell name is not provided and the host is not already part of a cell, the configuration will fail.

When configuring the master Security server (the sec_srv component), \texttt{config.dce} will prompt you for the password to be assigned to the initial accounts it creates in the registry database, including that of the cell administrator. When configuring most other components, this command will prompt you for the password of the cell administrator account so it can perform configuration tasks that require DCE authentication. If the environment variable \texttt{cell_admin_pw} is set, \texttt{config.dce} uses its value for the cell administrator password without prompting you. This feature can be useful when you automate configuration tasks, but you should use it sparingly because of the security risk it poses by making this password accessible to others.

Change the cell administrator password after the tasks are completed and the \texttt{cell_admin_pw} value is unset in order to limit the security risk. If a requested component is already configured, the \texttt{config.dce} command reports this and continues configuring other components. After the command has completed running, the configured components are listed on the screen. If a requested component is already partially configured, use the \texttt{unconfig.dce} command to clean it up before using the \texttt{config.dce} command to configure that component. To reconfigure a component with different parameters, use the \texttt{unconfig.dce} command to remove the existing configuration before running the \texttt{config.dce} command to set up the new configuration.

If a machine has a component configured, and additional components are to be configured, you do not have to respecify values for the \texttt{--cell_name}, \texttt{--sec_master}, \texttt{--cds_server}, and \texttt{--lan_profile} options. For example, if you have already configured the Security client on a machine, by specifying the name of the cell.
Before configuring a machine into a cell, ensure that the machine clock is within five minutes of the cell master Security server clock. If the machine clock is skewed more than five minutes, the `config.dce` command might report authentication errors, and the configuration might fail. The `--sync_clocks` and the `--time_server server_id` options can be used to synchronize the machine clock to the specified time server.

The `--dce_hostname` option is used to specify the dce_hostname for a machine configured into a cell. The dce_hostname is completely independent of the TCP/IP host name of the machine. If the `--dce_hostname` option is not specified, the dce_hostname will default to the TCP/IP host name (including the domain; for example, `jas.austin.ibm.com`). The default clearinghouse for any cds_second servers will be `{dce_hostname}_ch`. A Security Server name will also default to the dce_hostname if the `--sec_server_name` option is not used. The recommended usage is to accept the default name.

Only one security server (either a Master Security server or a Security Replica) can run on a machine. The sec_srv component is used for the master Security server and sec_rep is used for the Security replica. The `config.dce` command will ensure that the security client (`dced`) and the CDS client (`cds_cl`, `cdsadv`, and `cdsclerk`) are running on the machine before starting the security daemon (`secd`). When configuring a Security Replica, the `--sec_master` option can be used to locate the Master Security server.

The `config.dce` command deliberately replicates the `/:/subsys/dce/sec` directory when it configures a secondary CDS server. During the configuration of a Security Replica, entries are created in this directory but they might not be propagated immediately to the CDS secondary servers. Since these entries are referenced during subsequent pieces of the Security Replica configuration, failures can occur. To prevent this type of failure, stop all `cdsd` daemons that are running on secondary CDS servers before configuring a Security Replica into the cell. After the successful configuration of the Security Replica, restart the `cdsd` daemons.

### Examples

When configuring a DCE cell, first configure and start the Master Security server:

```
config.dce --cell_name/.../comp.sci.cell.uw.edu sec_srv
```

This command establishes the cell name as `/.../comp.sci.cell.uw.edu`, the name specified with the `--cell_name` option.

It creates the Master Security server using the default name (cell_admin) for the cell administrator account. It also configures and starts the RPC daemon and a Security client on the same machine as the Master Security server.

To avoid UNIX ID conflicts when you merge your current `/etc/passwd` and `/etc/group` files into your new DCE registry, use the `--min_princ_id`, `--min_group_id`, `--min_org_id`, and `--max_unix_id` options to specify the starting point and maximum values for UNIX IDs assigned to principals, groups, and
config.dce

organizations when configuring the Master Security server. The range of valid IDs
should take into account the different values allowed by each of the platforms that
will be represented in the cell.

**AIX:** 100 through 2,147,483,647 (all IDs)

**Solaris:**
1. 100 through 2,147,483,647 (for all but group IDs)
2. group IDs 100 through 60,000

**NT:** 100 through 2, 147, 483, 467 (all IDs)

The **–dce_hostname** option is used to designate the dce_hostname of the
machine.

```
config.dce -cell_name/.../comp.sci.cell.uw.edu -min_princ_id\n2000 - min_group_id 2000 - min_org_id 2000 - max_unix_id\n45000 dce_hostname csadmin sec_srv
```

After configuring and starting the Master Security server on a machine with the
TCP/IP short hostname of **deptchair**, configure and start the initial CDS:

```
config.dce -cell_name/.../comp.sce.cell.uw.edu -sec_master\ndeptchair cds_srv
```

Because no **–cell_admin** option was specified, this command assumes that the
name of the cell administrator account is ”cell_admin”. This command also
configures and starts the RPC daemon, a Security client, and a CDS client on the
same machine as the initial CDS server.

To run the initial Security and CDS servers on the same machine, the previous
examples can be combined into one command:

```
config.dce -cell_name/.../comp.sci.cell.uw.edu -dce_hostname\n csadmin sec_srv cds_srv
```

To configure another machine as a DTS global courier server (in a different LAN
than the initial CDS server), which is a client to all other DCE services, enter the
following:

```
config.dce -cell_name/.../comp.sci.cell.uw.edu -dce_hostname\ntimemachine -courier_role courier -sec_master deptchair - cds_server deptchair\n-lan_profile/../lan-prof-2 dts_global cds_cl
```

The **–lan_profile** option was used to specify a user-defined LAN profile rather than
the default profile.

To specify the admin portion of configuration for a new client in the
comp.sci.cell.uw.edu cell (requires cell administrator’s password only) enter:

```
config.dce -config_type admin -host_id 129.35.6.1 all_cl
```

If the TCP/IP hostname of the machine identified with the **–host_id** flag is
**jas.austin.ibm.com**, the dce_hostname will default to **jas.austin.ibm.com**. If the
lookup at the nameserver fails, the dce_hostname will be **129.35.6.1**.

```
config.dce -config_type admin -host_id chc cds_cl
```
The `dce_hostname` will default to `chc.austin.ibm.com`.

```bash
config.dce -config_type adm -host_id
pal401.pals.austin.ibm.com -dce_hostname mikep all_cl
```

The `dce_hostname` is `mikep`. Note that it has no relationship to the TCP/IP host name. Admin configuration updates the CDS namespace and security registry with information about the new client being configured. The local configuration must subsequently be completed on the client machine.

To specify the local configuration for a new client (requires root authority only) enter:

```bash
config.dce -config_type local
-cell_name/.../comp.sci.cell.uw.edu -sec_master deptchair
[-cds_server deptchair] all_cl
```

The `dce_hostname` of this client is `jas.austin.ibm.com`, the same as its TCP/IP host name.

```bash
config.dce -config_type local -cell_name/.../comp.sci.cell.uw.edu
cds_cl
```

If local configuration is done on an existing security client, it is not necessary to use the `–sec_master` or `–cds_server` options.

```bash
config.dce -config_type local -cell_name/.../comp.sci.cell.uw.edu
-sec_master deptchair -dce_hostname mikep all_cl
```

The `dce_hostname` entered is the same one the cell administrator used during admin configuration.

Local configuration must be run after the admin configuration has been completed. To specify full configuration of a client into an existing cell (requires root authority and cell administrator password).

```bash
config.dce [-config_type full]
-cell_name/.../comp.sci.cell.uw.edu [-dce_hostname mjs]
-sec_master deptchair [-cds_server deptchair] _cl
```

The `dce_hostname` if the `dce_hostname` option is not used and client TCP/IP address is determined by `config.dce` by a call to `hostname`.

To configure the password strength server, enter:

```bash
config.dce -pw_strength_srv
```

To configure the password strength server that will run under the password strength principal `pws_id`, enter:

```bash
config.dce -pwdstr_arg "-v -s pws_id"
-pwdstr_principal pws_id
-pwdstr_cmd pwd_strengthd pw_strength_srv
```

**Related Information**

Commands: `unconfig.dce`, `start.dce`, `stop.dce`, `show.cfg`, `clean-up.dce`. 
Purpose

Starts the DCE components configured on the local machine.

Synopsis

```
start.dce
```

```
stop.dce
[all]
[usage]
[?]
[help]
[operations]
components
```

Options

- **all** Starts the configured DCE and DFS components on the local machine.
- **usage** Displays a help message.
- **?** Displays a help message.
- **help** Displays a brief description for the passed arguments.
- **operations** Lists all the options and the components.
- **components** Specifies the components to be stopped.

The **Client Components** are:

- **all** All configured components (client and server)
- **core** All configured DCE components (client and server)
- **all_cl** All clients (cds_cl, dts_cl, rpc, and sec_cl)
- **client** Same as all_cl
- **cds_cl** CDS clerk
- **dts_cl** DTS client
- **rpc** RPC daemon (rpcd)
- **sec_cl** Security client
- **slim_cl** Slim client
- **dce_unixd** Integrated login (AIX only)
- **pam** Pluggable Authentication Module (Solaris only)
- **nsswitch** Name Service Switch (Solaris only)

The **Server Components** are:
The `start.dce` command starts the currently configured component daemons on the local machine.

To start both DFS and DCE components, use `start.dce all`. Otherwise, the `start.dce` command starts only the DCE components that are configured. To display configured components use the `show.cfg` command.

**Related Information**

Commands: `config.dce`, `stop.dce`, `start.dce`, `show.cfg`, `clean-up.dce`. 
Purpose

Stops the DCE components configured on the local machine.

Synopsis

stop.dce
[all]
[usage]
[–?]  
[help]
[operations]
components

Options

all  Stops the DCE and DFS components configured on the local machine.
usage Displays a help message.
–? Displays a help message.
help Displays a brief description for the passed arguments.
operations Lists all the options and the components.
components Specifies the components to be stopped.

The Client Components are:

all    All configured components (client and server)
core   All configured DCE components (client and server)
all_cl All clients (cds_cl, dts_cl, rpc, and sec_cl)
cds_cl CDS clerk
dts_cl DTS client
rpc    RPC daemon
sec_cl Security client

The Server Components are:
stop.dce

core   All configured DCE components (client and server)

all_srv  All servers (cde_second, cds_srv, dts_global, dts_local, gda,  
               sec_srv, ems_srv, nsid, pw_strength_srv, sec_rep, snmp_srv)

core_srv  All core servers (rpc, dced, sec_srv,cds_cl, cds_srv)

audit   Audit daemon

cds_second  Additional CDS servers

cds_srv   Initial CDS server for the cell

dts_global  DTS global server

dts_local  DTS local server

ems_srv  Event Management server

gda   Global Directory Agent

idms_srv  Idenity Mapping server

nsid  Name Service Interface daemon

pw_strength_srv  Password-Strength server

sec_srv  Security server

sec_rep  Security replica

snmp_srv  SNMP Subagent

Description

The stop.dce command stops the currently configured component daemons on the local machine.

To stop both DFS and DCE components, use stop.dce all. Otherwise, the stop.dce command stops only the DCE components that are configured. To display configured components, use the show.cfg command.

Related Information

Commands: config.dce, start.dce.
Purpose

Displays the DCE and DFS components configured on the local machine.

Synopsis

`show.cfg`  
`[all]`  
`[dce]`  
`[dfs]`  
`[usage]`  
`[-?]`  
`[help]`  
`[-no_daemon_check]`  
`[operations]`

Options

- **all**: Lists both the DCE and DFS components configured on the local machine.
- **dce**: Displays the configured DCE components. This option is the default.
- **dfs**: Displays the configured DFS components.
- **usage**: Displays a help message.
- **–?**: Displays a help message.
- **help**: Displays a brief description for the passed arguments.
- **–no_daemon_check**: Specifies that the daemon running states should not be determined or displayed.
- **operations**: Lists all the options and the components.

Description

The `show.cfg` command displays the currently configured components on the local machine and returns information about the configuration state and the running state.

Possible configuration states are:

**Configured**

The component is fully configured.

**Partial**

The component is only partially configured. A previous configuration or unconfiguration attempt failed.

Possible running states are:

**Running**

The component or daemon is running, has been initialized, and is ready for normal operation.
show.cfg

Not Running
The component or daemon is not currently running.

Available
The component functions are available (there is no daemon).

Not Available
The component or daemon is running but not currently listening.

Unknown
The state is unknown. DCE/DFS daemons are started or stopped using their fully qualified paths. Unknown is displayed when the fully qualified daemon is not found, but the daemon is found with either no path or a different path.

To list both DFS and DCE components, use show.cfg all. Otherwise, the show.cfg command lists only the DCE components that are configured.

Examples

The following is an example of a component summary:

Component Summary for Host: xxxxxxx.xxxxx.xxx.xxx
Component                     Configuration State  Running State
Security Master server        Configured           Running
Security client               Configured           Running
RPC                           Configured           Running
Identity Mapping server       Partial              Not Running
Initial Directory server      Configured           Running
Directory client              Configured           Running
Password strength server      Configured           Running
  pwd_strengthd
Audit server                  Configured           Running

The component summary is complete.

Related Information

Commands: config.dce, unconfig.dce, start.dce, stop.dce.
chpesite

Purpose

Updates the contents of the /opt/dcelocal/etc/security/pe_site file.

Synopsis

chpesite [-v]

Options

-v Run in verbose mode, which allows rpccp messages to be displayed. Without this option, messages are suppressed.

Description

The chpesite command updates the contents of the /opt/dcelocal/etc/security/pe_site file. The pesite file tells the location of security servers and what protocols they support. The pe_site file is created when a security client is configure.

Using chpesite updates the pe_site file by gathering information on other machines in the cell. If security replicas exist on the other machines, the chpesite command adds the appropriate entries for each replica to the pe_site file.

The chpesite command uses rpccp to gather the information needed for the pe_site file. Normally, rpccp messages are suppressed. Use the -v option to display these messages.

Note: The cdsadv daemon must be running to use the chpesite command. If cdsadv is not running, the rpccp calls in chpesite will fail. (When chpesite fails, use the -v option to view error messages.)

Use the chpesite command in the following situations:

• After configuring or unconfiguring a security replica in the cell, update the pe_site files on all other machines in the cell. This allows each machine in the cell to recognize the addition or removal of the security server, enabling better performance.

  Note: Wait 5 to 10 minutes after configuring or unconfiguring the security replica before running chpesite on any machine in the cell. Because of an rpccp timing window.

• Run chpesite as a cron job, once a day, on every machine in the cell. This resynchronizes all machines in the cell and keeps the cell up to date.

Files

/opt/dcelocal/etc/security/pe_site
Contains information about the location of security servers and the protocols supported by each security server. The pe_site file is created when a security client is configured.
Related Information

Commands: `config.dce`, `unconfig.dce`.

For more information about configuring DCE, see *IBM DCE Version 3.1 for AIX: Quick Beginnings* and *IBM DCE Version 3.1 for AIX and Solaris: Administration Guide—Introduction*. 
### Purpose

Cleans the DCE databases, sockets, and cache files, creates backup log files, and removes DCE-generated core files.

### Synopsis

```
clean_up.dce
[–core]
[–truncate_log]
[usage]
[–?]    
[help]  
[operations]
```

### Options

- **–core** Specifies that DCE-generated core files are to be removed.
- **–truncate_log** Specifies that backup DCE-generated log files should be created.
- **usage** Displays a help message.
- **–?** Displays a help message.
- **help** Displays a brief description for the passed arguments.
- **operations** Lists all the options and the components.

### Description

The `clean_up.dce` command cleans DCE databases, sockets, and cache files, creates backup log files, and removes DCE-generated core files. If DCE problems are encountered, the `clean_up.dce` command can be used to remove possibly corrupted files. All of the files that are removed will be recreated.

Without any options, the `clean.dce` command removes DCE databases, cache files, and socket files.

With the **–core** option, the `clean.dce` command also removes core files.

When the **–truncate_log** option is used, backup files for the DCE serviceability log files are created in `/opt/dcelocal/var/svc` so that new serviceability logs can be created.

### Related Information

None.
dceback

Purpose

Backs up and restores DCE data.

Synopsis

dceback apropos -topic string [-help]
dceback help
dceback dumpsecurity {-destfile filename | -stdout} [-component {common | master}...][-help]
dceback dumpcdfs {-destfile filename | -stdout} [-help]
dceback dumpmisc {-destfile filename | -stdout} [-help]
dceback dumpdfs {-destfile filename | -stdout} [-component {admin | bakdb | config | dfstab | fldb}...][-help]
dceback restoresecurity {-sourcefile filename | -stdin} [-component {common | master}...][-destdir directory_name][-help]
dceback restorecdfs {-sourcefile filename | -stdin} [-destdir directory_name][-help]
dceback restoremisc {-sourcefile filename | -stdin} [-destdir directory_name][-help]
dceback restoreddfs {-sourcefile filename | -stdin} [-component {admin | bakdb | config | dfstab | fldb}...][-destdir directory_name][-help]

Commands

dceback dumpcdfs

The dceback dumpcdfs command backs up CDS data files on the local machine. Some of the files backed up by the command are common to every machine configured as a DCE client, so the command is useful for backing up CDS data files from any DCE client or server machine.

dceback dumpsecurity

The dceback dumpsecurity command backs up Security Service data files on the local machine. Some of the files backed up by the command are common to every machine configured as a DCE client, so the command is useful for backing up Security Service data files from any DCE client or server machine.

dceback dumpmisc

The dceback dumpmisc command backs up miscellaneous DCE data files, including some DTS data files, on the local machine. The files the command backs up from a given machine depend on the machine’s DCE-related configuration. Some files backed up by the command are unique to machines configured as DTS clerks or servers. Because the command also backs up some files common to every machine configured
as a DCE client, it is useful for backing up miscellaneous DCE data files from any DCE client or server machine.

**dceback restorecds**
The `dceback restorecds` command restores previously backed up CDS data on the local machine.

**dceback restoresecurity**
The `dceback restoresecurity` command restores previously backed up Security Service data files to the local machine. The command is useful for restoring Security Service data files to any DCE client or server machine.

**dceback restoremisc**
The `dceback restoremisc` command restores previously backed up miscellaneous DCE data files, including some DTS data files, to the local machine.

The command can restore only those files that were previously backed up from the machine. Some files the command can restore are common to every machine configured as a DCE client; other files are unique to machines configured as DTS clerks or DTS servers. A machine’s DCE related configuration determines the files that were backed up and, thus, can be restored. The `dceback restoremisc` command cannot restore a file that is absent from the source file.

**dceback dumpdfs**
The `dceback dumpdfs` command backs up DFS data files on the local machine. This option is being provided for backward compatibility with the previous release. Please refer to the documentation for the `dfsback` command in your DFS documentation.

**dceback restoredfs**
The `dceback restoredfs` command backs up DFS data files on the local machine. This option is being provided for backward compatibility with the previous release. Please refer to the documentation for the `dfsback` command in your DFS documentation.

## Options

**-destfile filename**
Names the file to which backed up data files are to be written. All data files backed up with any one invocation of a dceback command that includes this option are written to the single file specified with the option. The new backed up data overwrites the contents of an existing file of the same name. The data is written in tar format, with no compression.

**-sourcefile filename**
Names the file from which previously backed up data files are to be restored. All data files restored with any one invocation of a dceback command that includes this option are read from the single file specified with the option.

**-destdir directory_name**
Names a fully-qualified alternate root directory beneath which previously backed up data files are to be restored. The specified directory must already exist. By default, each data file is restored to the directory in which it typically resides (the directory from which it was backed up). You can use this option to specify an alternate root directory beneath which all files are to be restored. The command creates the necessary hierarchy of subdirectories if it does not already exist beneath the specified directory.
-stdout
Directs the command to write the data files it backs up to standard output instead of to a file. The command sends the data to standard output in tar format, with no compression. Use this option to reduce the amount of disk space required to store the backed up data by piping the output to the compress program.

-stdin
Directs the command to read the data files it is to restore from standard input instead of from a file. Use this option to read output piped to the command from the zcat program. This option allows you to restore data files from a compressed tar file created by piping the output of a dceback command to the compress program.

-component {common | master}
Specifies the Security Service data files to be backed up or restored.
   common
       Directs the command to back up or restore Security Service data files common to every machine configured as a DCE client.
   master
       Directs the command to back up or restore the registry database and the master key for the database. This option should only be used on the security master server.

If the -component option is not specified in conjunction with the dumpsecurity or restoresecurity dceback commands, then both common and master are assumed; this should only be done if the machine being backed up or restored is the security master server for the cell.

-component {admin | bakdb | config | dfstab | fldb}
This option is being provided for backward compatibility with the previous release. Please refer to the documentation for the dfsback command in your DFS documentation.

--help
Displays the online help for this command. All other valid options specified with this option are ignored.

-topic string
Specifies the keyword string for which to search. If it is more than a single word, surround it with " " (double quotes) or other delimiters. Enter all strings in the case you wish to match (for example, enter the names of all dceback commands in lowercase letters).

Description

The dceback command suite includes commands used to back up and restore files that contain important configuration and administrative data associated with the following DCE components:

Cell Directory Service
You can back up data files associated with the CDS server using the dceback dumpcds command. You can restore previously backed up CDS data files using the dceback restorecds command.

Security Service
You can back up data files associated with the Security Service using the dceback dumpsecurity command. You can restore previously backed up Security Service data files using the dceback restoresecurity command.
Missellaneous DCE

You can back up miscellaneous data files associated with DCE using the `dceback dumpmisc` command. You can restore previously backed up miscellaneous DCE data files using the `dceback restoremisc` command. These commands also back up and restore DTS configuration files common to all DTS clerks and servers.

Distributed File Service

You can back up data files associated with the DFS using the `dceback dumpdfs` command. You can restore previously backed up DFS data files using the `dceback restoredfs` command. The option to back up and restore DFS data is being provided for backward compatibility with the previous release. Please refer to the documentation for the `dfsback` command in your DFS documentation for more information.

File Handling

Some `dceback` commands back up data files common to all machines configured as DCE clients. Some also back up files unique only to machines configured as certain types of DCE servers or clients. The specific files backed up from a given machine with a single invocation of a command depend on the machine’s configuration. The command used to back up Security Service files includes a `-component` option that can be used to more precisely define the files to be backed up. Use all `dceback` commands necessary to dump the data files for all aspects of a machine’s DCE configuration.

All files backed up with any one invocation of a `dceback` command are written to a single destination. If the `-destfile` option is included with the command to specify the name of a file, the backed up files are written to the named file. If the `-stdout` option is included with the command, the backed up files are written to standard output.

Likewise, all files restored with any one invocation of a `dceback` command are always read from a single source. If the `-sourcefile` option is included with the command to specify the name of a file, the files to be restored are read from the named file. If the `-stdin` option is included with the command, the files to be restored are read from standard input. As with the command used to back up such files, the command used to restore Security Service files includes a `-component` option that can be used to more precisely define the files to be restored.

All files are restored with the same path name from which they were backed up. A restored file overwrites an existing file of the same name if such a file exists on the machine. All commands used to restore files include a `-destdir` option that can be used to indicate a directory beneath which all files are to be restored, thus preserving any existing files of the same name on the machine. Ensure that the directory indicated with the option has enough disk space to accommodate the files to be restored.

Commands in the `dceback` suite are robust with respect to the existence of files to be backed up or restored. If a file to be backed up does not exist on the machine, the dump command will continue without error. If a file to be restored does not exist in the specified source file, the restore command will continue without error. If a directory in the path name for a file to be restored does not exist on the machine, the command creates it.

[Table 2], [Table 3], and [Table 4 on page 745] list the path name of each file processed by `dceback`. The tables also indicate whether each file is host specific or not. A file
that is host-specific must be restored to the machine from which it was backed up because it cannot be used on a different machine. A file that is not host-specific is host-independent and can be restored to and used on any machine. All commands that back up files deal with at least one host-specific file; consequently, all commands that restore files can be used to restore files only to the machine from which they were backed up.

**Disk Space/Compression**

Backup files generated with dceback commands are uncompressed tar files. As such, they require at least as much disk space as the data files they contain. Note that the tar program can read all backup files generated by dceback commands. However, while the dceback commands can read most files generated by the tar program, the tar program can create files that dceback commands cannot read.

Before backing up any data files, make sure that the directory in which the backup file is to be written has enough disk space to accommodate the file. In general, files that contain database-like information are typically much larger than other files backed up with dceback commands; for example, the files that contain the registry database or clearinghouse are usually much larger than the other files that are backed up.

To help you minimize the disk space required to store backup files, each dceback command that backs up data files includes a -stdout option. The option directs each command to write the data to standard output; each command writes the data in tar format, with no compression, which is the format in which it creates backup files when the -destfile option is used. You can use the -stdout option and pipe the output to the compress program, which reduces the size of the resulting file.

To let you restore data files from a backup file piped to the compress program, each dceback command that restores data files includes a -stdin option. The option directs the command to accept input from standard input. You can use the zcat program to process a compressed backup file, piping the output to the appropriate dceback command to restore the data files.

**Precautions**

The **Special Instructions** section describes specific precautions that must be taken before the dceback commands can be used to back up or restore files. In many cases, specific server processes must be stopped before a command is issued and restarted when the command is finished. In such cases, the section lists the precautions and briefly describes the steps that must be taken to meet them. You should always adhere to the described precautions to ensure that your data is properly backed up and restored.

**Backup Plan**

Commands in the dceback suite are most effective when employed on a regular basis. To realize the greatest benefit from the commands, develop and follow a schedule of periodic backups. As with any approach to backing up data, instituting and following a good backup schedule ensures the availability of the most recent data possible. It can effectively minimize the effects of machine or disk failures that might otherwise cripple your DCE cell.
dceback

Security

The issuer of the dceback command must be logged in as root on the local machine.

Regardless of whether you store your backup data online or on physical media such as tapes, make sure that you protect the backup data. If you store backup files in a local directory, use local operating system protection mechanisms to restrict access to them; if you store them in a DCE LFS fileset in the DFS filespace, use access control lists (ACLs) to restrict access to them. If you store backup files on physical media, store the media in a secure location.

Special Instructions

Backing Up/Restoring CDS Data

Before using a dceback command to back up or restore CDS data files on a DCE client or CDS server machine, you must take the following precautions:

DCE Client

Before issuing the dceback dumpcds/restorecds command on a machine configured as a DCE client:

- Ensure that no one is editing the cds_attributes file on the machine.
- Use the command
  
  `stop.dce cds_cl`

  to stop the CDS client processes on the machine. When the dceback dumpcds/restorecds command is finished, restart the CDS client processes using the command
  
  `start.dce cds_cl`

CDS Server

Before issuing the dceback dumpcds/restorecds command on a machine configured as a CDS server:

- Ensure that no one is editing the cds_attributes file on the machine.
- Use the command
  
  `stop.dce cds_cl cds_srv`

  to stop the CDS processes on the machine. When the dceback dumpcds/restorecds command is finished, restart the CDS processes using the command
  
  `start.dce cds_cl cds_srv`

Backing Up/Restoring Security Data

DCE client

Before issuing the dceback dumpsecurity/restoresecurity command on a machine configured as a DCE client:

- Ensure that no one is editing any of the files shown in Table 3 on page 745 in the common section.
- Because any DCE process running on a machine can modify the keytab file on that machine, you may want to stop all DCE processes using the stop.dce command before backing up or restoring common Security...
Service data files. When you are finished backing up or restoring files, DCE can be restarted with the `start.dce` command.

Security master server

Before issuing the `dceback dumpsecurity/restoresecurity` command on a machine configured as a security master server use the `sec_admin` command to ensure that the master copy of the registry database is the default replica and to place the master copy of the database in the maintenance state. These steps cause the security master server to save its copy of the database to disk and refuse all updates. Once the files have been backed up or restored, use the `sec_admin` command to return the registry database to the service state. This step causes the security master server to resume accepting updates. For more information on the `sec_admin` command see `[sec_admin` on page 697].

When backing up the security master server, ensure that you use both the `common` and `master -component` options (the default) of `dceback dumpsecurity` command. Providing only the value `master` for the `-component` option will not back up all the required security files.

Examples

Full back up of any machine that is not a security master server:
```
stop.dce
dceback dumpcds -destfile /dce_backup/cds_data
dceback dumpsecurity -component common -destfile /dce_backup/sec_data
dceback dumpmisc -destfile /dce_backup/misc_data
start.dce
```

Full restore of any machine that is not a security master server:
```
stop.dce
dceback restorecds -sourcefile /dce_backup/cds_data
dceback restoresecurity -component common -sourcefile /dce_backup/sec_data
dceback restoremisc -sourcefile /dce_backup/misc_data
start.dce
```

Full back up of a security master server:
```
stop.dce
dceback dumpcds -destfile /dce_backup/cds_data
dceback dumpsecurity -destfile /dce_backup/sec_data
dceback dumpmisc -destfile /dce_backup/misc_data
start.dce
```

Full restore of a Security Master server:
```
stop.dce
dceback restorecds -sourcefile /dce_backup/cds_data
dceback restoresecurity -sourcefile /dce_backup/sec_data
dceback restoremisc -sourcefile /dce_backup/misc_data
start.dce
```

Back up CDS data only on a DCE client:
```
stop.dce cds cl
dceback dumpcds -destfile /dce_backup/cds_data
start.dce cds cl
```

Restore CDS data only on a CDS server machine:
dceback

stop.dce cds_cl cdssrv

dceback restorecds -sourcefile /dce_backup/cds_data

start.dce cds_cl cds_srv

Back up miscellaneous data with compression:
dceback dumpmisc -stdout | compress > /dce_backup/misc_data.Z

Restore miscellaneous data with decompression:
zcat /dce_backup/misc_data.Z | dceback restoremisc -stdin

Restore miscellaneous data to another directory:
dceback restoremisc -sourcefile /dce_backup/misc_data -destdir /tmp/dce_data
### Table 2. CDS Files

<table>
<thead>
<tr>
<th>Filename</th>
<th>Host Specific?</th>
</tr>
</thead>
<tbody>
<tr>
<td>/opt/dcelocal/etc/cds_attributes</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/etc/cds.conf</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/etc/cds_globalnames</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/etc/cds_serv_pref</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/var/adm/directory/cds/cds_cache.version</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/adm/directory/cds/cds_cache.version</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/adm/directory/cds/cds_cache.wan</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/adm/directory/cds/clerk_mgmt_acl_v1.dat</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/directory/cds/cds_files</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/var/directory/cds/gda_mgmt_acl_v1.dat</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/directory/cds/server_mgmt_acl_v1.dat</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Table 3. Security Files

<table>
<thead>
<tr>
<th>Filename</th>
<th>Host Specific?</th>
</tr>
</thead>
<tbody>
<tr>
<td>-component common</td>
<td></td>
</tr>
<tr>
<td>/krb5/krb5.conf</td>
<td></td>
</tr>
<tr>
<td>/krb5/pwd_strength_tab</td>
<td>yes</td>
</tr>
<tr>
<td>/krb5/v5srvtab</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/etc/security/pe_site</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/etc/security/sso_cell</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/var/security/lrgy_data</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/security/lrgy_tgts</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/security/pk_file/*</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/security/pkc_data/.pkc_data</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/security/pwd_strength/*</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/security/pwhist.db</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/security/sec_aud_trail.md_index</td>
<td>yes</td>
</tr>
<tr>
<td>-component master</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/var/security/.mkey</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/security/lrgy_data</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Table 4. Miscellaneous Files

<table>
<thead>
<tr>
<th>Filename</th>
<th>Host Specific?</th>
</tr>
</thead>
<tbody>
<tr>
<td>/opt/dcelocal/dce_cf.db</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/etc/cfg.dat</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/etc/cfgarg.dat</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/etc/dce_modules/local_envf</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/etc/dce_modules/local_envl</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/etc/rc.dce.conf</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/etc/setup_state</td>
<td>yes</td>
</tr>
</tbody>
</table>
### Table 4. Miscellaneous Files (continued)

<table>
<thead>
<tr>
<th>Filename</th>
<th>Host Specific?</th>
</tr>
</thead>
<tbody>
<tr>
<td>/opt/dcelocal/etc/usrstime.tcl</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/tcl/user_cmd.tcl</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/var/adm/time/configState</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/adm/time/dts_aud_trail</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/var/adm/time/dts_aud_trail.md_index</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/var/adm/time/dtsd.acl</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/var/adm/time/mgt_acl</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/adm/*</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/var/audit/adm/acl</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/var/audit/adm/central_trail</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/var/audit/adm/central_trail.md_index</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/var/dced/Acl.db</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/dced/cell_aliases</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/var/dced/cell_name</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/var/dced/dced.log</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/dced/host_name</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/dced/Hostdata.db</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/dced/Keytab.db</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/dced/Lib.db</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/dced/objectuuid.txt</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/dced/post_processors</td>
<td></td>
</tr>
<tr>
<td>/opt/dcelocal/var/dced/Svrconf.db</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/dced/Svrreexec.db</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/dced/Xattrschema.db</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/var/directory/cds/adm/nsid/nsid.binding</td>
<td>yes</td>
</tr>
<tr>
<td>/opt/dcelocal/web/etc/*</td>
<td>yes</td>
</tr>
</tbody>
</table>
**Purpose**

Installs, uninstalls, configures and unconfigures DCE.

**Synopsis**

The configuration and unconfiguration commands are supported for backward compatibility only. New daemons and options are only supported through the new commands. Refer to `config.dce` on page 714 and `unconfig.dce` on page 770 for information on configuring and unconfiguring DCE.

The installation and uninstallation commands have been enhanced to fully support this product. These commands are documented here.

```
dcesetup apropos [-topic <string>] [-help]
dcesetup help

dcesetup history [-number <number_of_commands>] [-help]
dcesetup info [-help]
dcesetup install -component {appdev | cdsserver | client | secserver | sysmgmt | priv | msgs | docs}... -dir <DCE_release_directory>
  [-mklinks <DCE_target_directory>] [-force]
  [-msg_langs {<en_US> | <it> | <es> | <ja> | <ja_BP.CPK> | <ko> | <zh> | <zh.GBK>}]...
  [-doc_langs {<en_US> | <it> | <ko> | <zh>}]... [-noman] [-help]

dcesetup recover [-help]
dcesetup uninstall {-all | -component {appdev | cdsserver | client | secserver | sysmgmt | priv | msgs | docs}}...
  [-msg_langs {<en_US> | <it> | <es> | <ja> | <ja_BP.CPK> | <ko> | <zh> | <zh.GBK>}]...
  [-doc_langs {<en_US> | <it> | <ko> | <zh>}]... [-help]

dcesetup upgrade_install -backdir <DCE_backup_directory>
  -dir <DCE_release_directory> [-mklinks <DCE_target_directory>] [-help]

dcesetup upgrade_uninstall -backdir <DCE_backup_directory> [-help]
```

**Options**

- **-backdir** `DCE_backup_directory`
  Specifies the full pathname of the directory to which dumped DCE data is to be written. Directories under `/tmp` can not be specified.

- **-component** `appdev, cdsserver, client, secserver, sysmgmt, priv, msgs, docs`
  Specifies the DCE software to be installed on the machine on which the command is issued. Specify one or more of the following arguments:

  - **appdev**
    Installs the files necessary for DCE application development on the machine.

  - **cdsserver**
    Installs the files necessary to configure the machine as a CDS Server (and a GDA server).

  - **client**
    Installs the files necessary to configure the machine as a DCE client (and DTS, Password strength, Name Service Switch and Name Service Interface servers, and DCE Web Secure).
dcesetup (Solaris only)

secserver
Installs the files necessary to configure the machine as a Security Server (and Identity Mapping server).

sysmgmt
Installs the files necessary to configure the machine as a DCE SNMP subagent and Event Management server (and DCE Web Administration).

priv
Installs all files necessary to support the Data Encryption Standard.

msgs
Installs all of the DCE message catalogs (English and non-English versions).

docs
Installs all of the HTML and PDF DCE Documentation files (English and non-English versions).

Installation of the DCE client software is a prerequisite for installation of all other DCE software; the client component must be the first component installed on any machine. You can specify multiple arguments with the -component option; the command installs the corresponding components in a predetermined order. Installation of the English Documentation is a prerequisite for installation of the non-English documentation packages.

-dir
Provides the full pathname of the directory on the DCE CD-ROM beneath which the DCE files reside. If the -mklinks option is omitted from the command, the command obtains the installation files for the DCE components to be installed on the local machine from the file system on the CD-ROM. The CD-ROM can be mounted on the machine on which the command is issued, or it can be mounted on a remote machine. The full pathname you need to specify is:

cdrom/DCE_version

where cdrom is the mount point of the DCE CD-ROM, and DCE_version indicates the version of DCE being installed (example cdrom/dce3.1).

If the -mklinks option is included with the command, you must name the directory on the DCE CD-ROM beneath which the DCE files reside, just as if the files were to be installed on the local machine. The command obtains package related instructions from the CD-ROM, but it creates links to the files in the directory specified with the -mklinks option.

-doc_langs en_US, it, ko, or zh
Specifies the languages that documentation should be installed for. You can specify multiple arguments with the -doc_langs option; installation of the English (en_US) Documentation is a prerequisite for installation of the non-English (it, ko, zh) documentation packages.

install
If the -doc_langs argument is not specified when docs was specified in the -component argument, documentation will be installed for the current locale. If there is no documentation for the current locale, then U.S. English documentation will be installed.

uninstall
If the -doc_langs argument is not specified when docs was specified in the -component argument, all DCE documentation will be uninstalled.

-force
Directs the command to re-install packages if they are already installed.
(The packages must be the same level as those already installed). By
dcesetup (Solaris only)

default, the command fails if the machine already houses files to be installed for a component specified with the -component option.

-help Displays the on-line help for this command. All other valid options specified with this option are ignored.

-mklinks DCE_target_directory
Provides the full pathname to the directory in the network that contains the installation files for the specified DCE components. The command performs a "linked installation," creating symbolic links from the local machine to the DCE files in the specified network directory. The directory must be the root of a directory structure that is fully populated with the installation files for all of the DCE components; the directory must be accessible from the machine on which the command is issued.

Omit this option to install the installation files for the specified DCE components on the local machine.

-msg_langs en_US, it, es, ja, ja_JP, PCK, ko, zh or zh.GBK
Specifies the languages that message catalogs should be installed for. You can specify multiple arguments with the -msg_langs option.

install If the -msg_langs argument is not specified when msgs was specified in the -component argument, messages will be installed for the current locale. If there are no messages for the current locale, then U.S. English messages will be installed.

uninstall
If the -msg_langs argument is not specified when msgs was specified in the -component argument, all DCE messages will be uninstalled.

-noman
Obsolete option; it is ignored.

-topic string
Specifies the keyword string for which to search. If it is more than a single word, surround it with " " (double quotes) or other delimiters. Enter all strings in the case you wish to match (for example, enter the names of all dcesetup commands in lowercase letters).

Description

The dcesetup command is an installation and configuration utility for use with the DCE product. It includes a number of subcommands that allow you to perform the following types of tasks on a machine:

- Install the DCE software necessary to perform any type of DCE configuration or conduct DCE application development on a machine.
- Configure a machine as any type of DCE server or client.
- Remove DCE configuration files from a machine, and remove configuration information for the machine from the appropriate DCE servers.
- Remove the DCE software from a machine.

All installation, uninstallation, upgrade, configuration, and unconfiguration commands can use either of two available interfaces:

Command-line interface
To use the command-line interface, supply all information necessary to
dcesetup (Solaris only)

perform the operation on the command line. The command refers to the specified command line for all parameters. You must include all required options with the command.

Interactive interface
To use the interactive interface, include no options with the command. The command prompts for all information necessary to perform the operation.

Commands

apropos

Description: The dcesetup apropos command displays the first line of the on-line help entry for any dcesetup command that contains the string specified with the -topic option in its name or short description. To see the syntax for a command, use the dcesetup help command.

Privilege Required: No privileges are required.

Output: The first line of an on-line help entry for a command lists the command and briefly describes its function. The dcesetup apropos command displays the first line for any dcesetup command for which the string specified with the -topic option is part of the command name or the first line.

Examples:

$ dcesetup apropos -topic install
info: display current installation and configuration
install: install DCE binary files
uninstall: uninstall DCE binary files
upgrade_install: continues upgrade to DCE 3.1 by installing DCE 3.1 binaries
upgrade_uninstall: initiates upgrade to DCE 3.1 by removing old DCE binaries
dcesetup succeeded.

dcesetup (Solaris only)

help

Description: The dcesetup help command displays the first line (name and short description) of the on-line help entry for every dcesetup command if the -topic option is not provided. For each command name specified with the -topic option, the output lists the entire help entry. Use the dcesetup apropos command to show each help entry that contains a specified string.

Privilege Required: No privileges are required.

Output: The on-line help entry for each dcesetup command consists of the following two lines:
- The first line names the command and briefly describes its function.
- The second line, which begins with Usage, lists the command’s options in the prescribed order.

Examples: The following command displays the on-line help entry for the dcesetup uninstall command:

$ dcesetup help -topic uninstall

    dcesetup uninstall {-all | -component {appdev | \cdsserver | client | secserver | sysmgmt | priv | msgs | docs}...}
Dcesetup (Solaris only)

History

Description: The dcesetup history command displays the dcesetup installation and configuration commands that have been issued on the local machine. The -number option can be used to restrict the number of commands displayed, as follows:

- Specify an integer with the -number option to see the most-recent number_of_commands commands; only the number of dcesetup commands specified with the option is displayed.
- Omit the -number option to display all dcesetup commands that have been issued on the machine.

Only dcesetup commands issued since the last dcesetup uninstall -all command was issued on the machine can be displayed. Issuing the dcesetup uninstall command with its -all option removes the dcesetup command history. Note that dcesetup help and dcesetup history commands are not recorded in the history of dcesetup commands.

Privilege Required: The issuer must be logged in as root on the local machine.

Output: The dcesetup history command displays the dcesetup commands in chronological order, displaying the oldest command first and the most-recent command last. For each dcesetup command, it displays the complete syntax of the command.

The general format of the command’s output is:

n  dcesetup command
n+1 dcesetup command
.
.
.
n+m  dcesetup command

Files: The dcesetup command maintains the following two ASCII files in the directory dcelocal/var/adm/messages/dcesetup on the local machine:

- history
- log

The dcesetup command creates both files when the first dcesetup install command is issued on the machine. The dcesetup command removes the files when the dcesetup uninstall -all command is issued on the machine.

Examples: The following command displays the three previous dcesetup commands issued on the local machine. The dcesetup commands were used to install the DCE client software, use the dcesetup interactive interface to configure...
dcesetup (Solaris only)

the machine as a DCE client (the complete command resulting from the interactive
configuration session was recorded), and install the DCE application development
files.

$ dcesetup history -number 3

1 /cdrom/dcesetup install -component client -dir /cdrom/dce3.1
1 /cdrom/dcesetup config_client -cellname dce.abc.com
   -secserver red.abc.com -cdsserver red.abc.com
1 /cdrom/dcesetup install -component appdev -dir /cdrom/dce3.1

info

Description: The dcesetup info command displays information about the current
DCE installation. For each installable DCE component (appdev, cdsserver, client,
secserver, sysmgmt, priv, msgs, and docs), the command examines the current
installation and reports whether that component is installed and, if installed, whether
a linked installation was used.

The command also displays information for each configured component.
Components that are fully or partially configured are listed, showing their
configuration state, and the running state of the daemon.

Privilege Required: The issuer must be logged in as root on the local machine.

Examples:

$ dcesetup info

   dcesetup version 3.1

   ## INSTALLATION ##

   INST DCE Client Services Package
   INST DCE Security Services Package
   INST DCE Cell Directory Services Package
   noinst DCE System Management Services Package
   noinst DCE Privacy Level Protection Feature
   noinst DCE Tools Package
   INST DCE U.S. English Messages Package
   noinst DCE Italian Messages Package
   noinst DCE Spanish Messages Package
   noinst DCE Japanese (EUC) Messages Package
   noinst DCE Japanese (PC Kanji) Messages Package
   noinst DCE Korean Messages Package
   noinst DCE Simplified Chinese (EUC) Messages Package
   noinst DCE Simplified Chinese (GBK) Messages Package
   noinst DCE U.S. English Documentation Package
   noinst DCE Italian Documentation Package
   noinst DCE Korean Documentation Package
   noinst DCE Simplified Chinese (EUC) Documentation Package

   ## CONFIGURATION ##

   Gathering component state information...

   Component Summary for Host: sunstroke.austin.ibm.com

   Component          Configuration State  Running State
   Security Master server        Configured   Running
   Security client              Configured   Running
   RPC                          Configured   Running
   Initial Directory server     Configured   Running
   Directory client             Configured   Running
   Password strength server     Configured   Running
   pwd_strengthd                Configured   Running
The component summary is complete.

dcesetup succeeded.

install

**Description:** The `dcesetup install` command installs DCE software on the machine on which the command is issued. The DCE software installed by a single invocation of the command depends on the arguments provided with the `-component` option when the command is issued. The first DCE component installed on any machine must be the `client` component; installation of the client software defines certain installation variables and creates installation directories required by all other DCE components. See the description of the `-component` option for information about the DCE components that can be installed on a machine.

When you initially install the `client` component on a machine, you must invoke the `dcesetup install` command by its full pathname either on the DCE CD-ROM or in a directory in the network (if such a directory has been prepared for linked installations). On the DCE CD-ROM, the `dcesetup` command is accessible from the following location:

```
cdrom/dcesetup
```

where `cdrom` is the mount point for the CD-ROM. (If a directory in the network has the CD-ROM image, you can substitute the full pathname of the network directory for `cdrom` to access the `dcesetup` command from that location.) Once the client software is installed on a machine, you can access the `dcesetup` command locally as `/etc/dcesetup`.

**The package Utility:** The dcesetup install command employs the Solaris package utility to install DCE software. When you install a DCE component, the dcesetup install command automatically invokes package to install the Solaris package that corresponds to that component. The following packages exist for the DCE components that can be installed:

- **IDCEtools**
  The package for the `appdev` component

- **IDCEcdss**
  The package for the `cdsserver` component

- **IDCEclnt**
  The package for the `client` component

- **IDCEsecs**
  The package for the `secserver` component

- **IDCEsmgmt**
  The package for the `sysmgmt` component

- **IDCEpriv**
  The package for the `priv` component

- **IDCEenUSm**
  The package for the `msgs` component and "en_US" language

- **IDCEitm**
  The package for the `msgs` component and "it" language
dcesetup (Solaris only)

IDCEesm
The package for the msgs component and "es" language

IDCEjam
The package for the msgs component and "ja" language

IDCEjaJPm
The package for the msgs component and "ja_JP.PCK" language

IDCEzhm
The package for the msgs component and "zh" language

IDCEGBKm
The package for the msgs component and "zh.GBK" language

IDCEEnUSd
The package for the docs component and "en_US" language

IDCEitd
The package for the docs component and "it" language

IDCEkod
The package for the docs component and "ko" language

IDCEzhd
The package for the docs component and "zh" language

Refer to documentation of the package utility supplied with your system for information about the pkgadd and pkгинfo commands used by the dcesetup install command.

Notes:
1. It is recommended that you use dcesetup commands to perform all operations related to DCE installation and configuration.
2. You should be aware of the following information regarding the dcesetup install command:
   - During installation of DCE files, the response of the dcesetup install command to existing files and links depends on whether the -force option is included with the command. If the -force option is omitted, the command fails if it finds files from a previous installation of a DCE component to be installed. However, if the -force option is included, the command reinstalls the files for the component if the same version of DCE is already installed. (The installation of the component files will fail if there is a different version of the code on the system already.)
   - The amount of time required for installation varies for different components. In general, installation of the client component takes longer than installation of the other components, both because it includes more files and because the library files installed with the client component take some time to install. Installation of the U.S. English documentation takes longer than installation of the other components also. This documentation is installed as tar files, some of which are quite large. The HTML files are extracted during the installation and this also takes time.

Privilege Required: The issuer must be logged in as root on the local machine.

Output: When it begins to install a component, the dcesetup install command displays a message reporting the component it is installing. The message has the following format:

## INSTALLING "COMPONENT" ##
where *COMPONENT* is the name of the component the command is installing. The command can also display additional messages in certain situations (for example, as it installs certain files), and it can display prompts asking for confirmation or direction in some situations.

If no options are provided with the command, an interactive installation session results. During an interactive installation session, the command displays many additional prompts and messages related to the installation. Regardless of the type of installation (command line or interactive), the command can also display additional messages as it performs the requested installation.

**Examples:** The following command installs version 3.1 of the DCE client software on the machine on which the command is issued. This is the first DCE component installed on the machine, so the full pathname to the `dcesetup install` command on the CD-ROM that contains the DCE software is required. The DCE CD-ROM is mounted at `/cdrom`.

```
# /cdrom/dcesetup install -component client -dir /cdrom/dce3.1
```

The following command installs the files necessary to configure a one machine DCE cell. Korean messages and documentation will be installed also. The DCE CD-ROM is mounted at `/cdrom`.

```
# /cdrom/dcesetup install -component client cdsserver seccserver \
  -msg_langs ko -doc_langs en_US ko -dir /cdrom/dce3.1
```

The following command installs the files necessary for application development. The DCE client software has already been installed on the machine, so the `dcesetup` command is installed locally and a link to the command exists in the machine’s `/etc` directory. The DCE CD-ROM is mounted at `/cdrom`.

```
# dcesetup install -component client cdsserver seccserver \
  -msg_langs ko -doc_langs en_US ko -dir /cdrom/dce3.1
```

**recover**

**Description:** The `dcesetup recover` command will restore DCE information in system files.

**Privilege Required:** The issuer must be logged in as `root` on the local machine.

**uninstall**

**Description:** The `dcesetup uninstall` command removes previously installed DCE files from the machine on which the command is issued. All DCE files can be removed with one invocation of the command, or DCE files can be removed on a component basis (which is how they are installed).

Specify the components for which installation files are to be removed as follows:

- Use the `-component` option to specify individual DCE components for which files are to be removed. The command removes only those files associated with the specified components.

- Use the `-all` option to specify that files for all DCE components are to be removed. The command removes all installation files associated with all components and also removes any configuration directories. An `unconfig.dce` must have been issued to remove the configuration files prior to issuing this command with the `-all` option, which can remove configuration directories only if they are empty.
The dcesetup uninstall command removes all of the files installed for a component by the dcesetup install command, as well as all of the links created by the command.

The dcesetup uninstall command removes only files installed by the dcesetup install command. It does not remove DCE configuration files created when the machine is configured or as users use DCE on the machine. For example, the command does not remove files associated with the registry database, which is created when a machine is initially configured as a security server.

However, if the -all option is specified, the dcesetup uninstall command attempts to remove the /opt/dce and /opt/dcelocal directories and their subdirectories. Before issuing the command with the -all option, you must use the unconfig.dce command to remove all DCE configuration files from the dcelocal directory of the machine. The dcesetup uninstall -all command fails if it finds that the machine still houses DCE configuration files. (Note that you do not need to use the unconfig.dce command prior to issuing the dcesetup uninstall command with the -component option.)

The dcesetup uninstall command uses the package utility available with Solaris to remove DCE files. Refer to the documentation of the package utility supplied with your system for information about the pkgrm and pkginfo commands used by the dcesetup uninstall command; see The package Utility on page 753 for information about the Solaris packages that exist for the DCE components.

Note: It is recommended that you use the dcesetup commands to perform all operations related to DCE installation and configuration.

Privilege Required: The issuer must be logged in as root on the local machine.

Output: When it begins to uninstall a component, the dcesetup uninstall command displays a message reporting the component it is uninstalling. The message has the following format:

## UNINSTALLING "COMPONENT" ##

where COMPONENT is the name of the component the command is uninstalling. The command can also display additional messages in certain situations (for example, as it uninstalls certain files), and it can display prompts asking for confirmation or direction in some situations.

When issued with the -all option, the command displays the following message and exits if it finds that DCE configuration files still exist on the machine:

Please back up the configuration files using "dceback" then unconfigure \ using "dcesetup unconfig" before using "dcesetup uninstall -all"

You must use the unconfig.dce command to remove all DCE configuration files from the machine before you can use the dcesetup uninstall command with the -all option. Back up the configuration files before using the unconfig.dce command to remove them. (You can use the dceback commands to back up DCE data files.)

If no options are provided with the command, an interactive uninstallation session results. During an interactive uninstallation session, the command displays many additional prompts and messages related to the uninstallation. Regardless of the type of uninstallation (command line or interactive), the command can also display additional messages as it performs the requested uninstallation.
**Cautions:** Do not directly create files in the /opt/dce or /opt/dcelocal directories or any of their subdirectories. When issued with the -all option, the **dcesetup uninstall** command attempts to remove the /opt/dce and /opt/dcelocal directory structures in their entireties. If the command succeeds, it removes these directories and their contents.

**Examples:** The following command removes the files necessary to configure a security server from the machine on which it is issued. The files were installed with the secserver component of the **dcesetup install** command.

```
# dcesetup uninstall -component secserver
```

The following command removes all DCE installation files from the machine on which it is issued. All DCE configuration files were previously removed from the machine with the **unconfig.dce** command.

```
# dcesetup uninstall -all
```

**upgrade_install**

**Description:** The **dcesetup upgrade_install** command completes an upgrade to a new version of DCE. This command performs the following functions:

- restores backed up DCE files
- installs the DCE packages (the packages that are equivalent to those previously installed)
- modifies or creates entries in system files if necessary
- restarts dce (runs **start.dce**) (this also migrates the DCE configuration data to the current level)

**Privilege Required:** The issuer must be logged in as **root** on the local machine.

**Output:** The **dcesetup upgrade_install** command can display instructive messages directing you to perform additional operations. For example, the command can direct you to reboot the machine when the command is finished executing. The command can also display warning and error messages as appropriate.

**Examples:** The following command completes the upgrade to a new version of DCE, where /my_backup_dir is the directory that contains the backed up configuration files, and the DCE CD-ROM is mounted at /cdrom.

```
# dcesetup upgrade_install -backdir /my_backup_dir -dir /cdrom/dce3.1
```

**upgrade_uninstall**

**Description:** The **dcesetup upgrade_uninstall** command completes an upgrade to a new version of DCE. This command performs the following functions:

- stops all DCE daemons
- invokes **dceback** to back up DCE files
- uninstalls the installed DCE packages

**Privilege Required:** The issuer must be logged in as **root** on the local machine.

**Output:** The **dcesetup upgrade_uninstall** command can display instructive messages directing you to perform additional operations; for example, the command will direct you to upgrade the operating system, if necessary, when the command is finished executing. The command can also display warning and error messages as appropriate.
dcesetup (Solaris only)

Examples: The following command backs up DCE configuration data and uninstalls the currently installed version of DCE, where /my_backup_dir is the directory where the back-up configuration files should be saved. This directory can not reside under /tmp.

# dcesetup upgrade_uninstall -backdir /my_backup_dir
kerberos.dce

Purpose

Enables existing DCE clients and servers to use Kerberos.

Note: Any machines, except slim clients, that are configured using the config.dce command shipped with the DCE 2.2.0.2 (DCE 3.1 for Solaris) or higher release will not need to run this command. These steps will be done automatically. This command must be run on all slim clients to enable the Kerberos function.

Synopsis

kerberos.dce

[-type {local | admin | full}]
[-ip_name ip_name]
[-cell_admin cell_admin_id]
[-admin_pwd password]
[usage]
[–?]  
[help]
[operations]

Options

-type Specifies the type of setup to be done. This option is required.

local Updates files on the local system. Requires access on the local machine, but not in the cell.

admin Updates the directory namespace. Requires cell administrator access in the cell.

full Does both the local and the admin steps.

–ip_name

Specifies the IP hostname for which to add the hosts and FTP accounts.

-cell_admin cell_admin_id

Specifies the name of the cell administrator account. When configuring the master Security server (the sec_srv component), the config.dce command gives this account privileges throughout the cell. Otherwise, the account named must have sufficient privilege to perform configuration tasks within the cell. If the –cell_admin option is not specified, the account cell_admin will be assumed. The value for cell_admin is used by all components except rpc, snmp_srv, audit, and dce_unixd.

–admin_pwd password

Specifies the cell administrator password. Caution should be used with this option because of the security risk it poses by making this password accessible to others.

usage Displays a help message.

–? Displays a help message.

help Displays a brief description for the passed arguments.
kerberos.dce

operations

Lists all the options.

Description

The kerberos.dce command creates the /etc/krb5.conf file when the type is local or full, and creates the /./hosts/ip_hostname accounts, the /./ftp/ip_hostname accounts and the keytable entries for these accounts when the type is admin or full.

1. In an existing cell, the command should be run with -type admin option to create the /./hosts/not_reg_mach_addrs. (The entry needs to be created only once.)
2. On each machine in the cell (where this function is to be used), run the command with the -type local option. This will register bindings for each IP host address in the /./hosts/not_reg_mach_addrs entry.
3. Periodically, run the command with the -type admin option to create the host and FTP accounts for each of the bindings that users registered under the /./hosts/not_reg_mach_addrs entry.
4. On each machine in the cell (where this function is to be used), run the command with the -type local option again after the FTP and host accounts are created. This will create the keytable entry for each account for this machine.

If the administrator chooses, the command can be run with the -type admin and -ip_name ip_hostname options. This will do the steps performed by 2 and 3 listed previously.

Examples

As a DCE cell administrator, to update data for clients in the cell, enter the following on a machine in the cell:

kerberos.dce -type admin -cell_admin cell_admin_id -admin_pwd password

After the administrator updates data for clients in the cell, as a DCE user in the cell, enter the following on a client machine (logged in as Root):

kerberos.dce -type local

As a DCE cell administrator, to update data for one client in the cell, enter the following on a machine in the cell:

kerberos.dce -type admin -ip_name ip_name -cell_admin cell_admin_id -admin_pwd password

Related Information

Commands: k5dcelogin.
mkreg.dce

Purpose

Adds information about a DCE cell into the domain namespace.

Synopsis

mkreg.dce
[–input_file input_file]
[–named_data_file named_data_file]
[usage]
[–?]   [help]
[operations]

Options

–input_file input_file
   Specifies the name of a file containing information about the cell you want to register. The default is /etc/input.data.

–named_data_file named_data_file
   Specifies the name of the file that contains data for the name daemon, named, when registering a DNS-style cell name. The default is /etc/named.data.

usage
   Displays a help message.

–?
   Displays a help message.

help
   Displays a brief description for the passed arguments.

operations
   Lists all the options and the components.

Description

The mkreg.dce command enters information about your DCE cell into the database maintained by your domain name server (the named daemon).

This command cannot be used to register X.500–style cell names.

If the name server machine is a member of the DCE cell you want to register, mkreg.dce will update the named data file you specify with the –named_data_file option. If the name daemon (named) is running, mkreg.dce will refresh named.

If the name server machine is not part of the DCE cell you want to register or is not configured with DCE at all, do one of the following:

• Generate the mkreg.dce input for the name server. On a machine that is part of the DCE cell you want to register, run the following two commands:

   cdscp show cell /.:as dns>input.file
   cdscp show clearinghouse/.:^*>>input.file
Take the resulting file to your domain name server and run `mkreg.dce`, using the `–input_file` option to specify the name of the input file.

- Generate the `mkreg.dce` output to add to the named data file on the name server.
  It is necessary to create a temporary data file on a machine within the cell, with the information to append to the permanent data file on the name server. To do this, run the following command to create the file and add the relevant cell data to it:

```
mkreg.dce -named_data_file output.file
```

Take the resulting output.file to your domain name server, add the contents of this file to the named data file, and refresh the named daemon in AIX (the `in.named` daemon in Solaris).

**Note:** When configuring with a DCE hostname, be sure to add the DCE hostname and the proper IP address of the machine associated with the DCE hostname to the list of host names. Remember that the DCE hostname is case sensitive.

For example, if you configure a cell with the cell name `./:/hulacell.austin.ibm.com` on the machine named `mustang1` and set the DCE hostname to be `hula.austin.ibm.com`, the following entry needs to be added to the named.data file on the DNS name server so that the machine name, or in this case the DCE hostname, can be resolved to a TCP/IP address.

```
cdsaix1.austin.ibm.com IN A 129.35.66.4
mustang1.austin.ibm.com IN A 129.35.69.52
hula.austin.ibm.com IN A 129.35.69.52
```

**Examples**

To register a cell when the name server is configured as a CDS client of the cell, enter:

```
mkreg.dce
```

To register a cell when the information about the cell and its CDS clearinghouse is contained in the file `/tmp/cell.info`, enter:

```
mkreg.dce –input_file/tmp/cell.info
```

**Related Information**

Commands: `rmreg.dce`. 
mkdceweb

Purpose

Configures DCE Administration, DCE Web Secure, or both into a Netscape FastTrack 3.01 or Enterprise 3.61 web server.

Synopsis

```bash
mkdceweb
  [-n netscape_dir]
  [-s netscape_id]
  [-i userid]
  [-u on | off]
  [-t all | secure | admin]
  [-v]
  [-?]
```

Options

- `-n netscape_dir`
  Identifies the Netscape server root directory. The default is `/usr/netscape/suitespot`.

- `-s netscape_id`
  Identifies the Netscape server identifier to configure.

- `-i userid`
  Identifies the operating system user ID under which the Netscape server will run. The Netscape server cannot run with the AIX userid of `nobody`.

- `-u on | off`
  Specifies to allow unauthenticated DFS access. Unauthenticated DFS access is not allowed if Netscape runs as `root`. The default is `off`.

- `-t all | secure | admin (AIX only)`
  Specifies which component to configure. The default is `all`.

- `-v(erbose)`
  Specifies to prompt for any values not supplied.

- `-?`
  Displays a help message.

Description

The `mkdceweb` command configures DCE Administration, DCE Web Secure, or both, into a Netscape FastTrack 3.01 or Enterprise 3.61 web server. Configuring DCE Administration also configures DCE Web Secure into the web server.

Examples

To configure both DCE Administration and DCE Web Secure into the `bullrun` web server with a Netscape root directory of `/usr/netscape`, enter:

```
mkdceweb -n /usr/netscape -s bullrun
```
**mkdceweb**

**Note:** The web server userid specified in the Netscape web server configuration file, `magnus.conf`, will not be changed. The `-t` option was not supplied because configuring all components is the default.

To configure DCE Administration into the **antietam** web server with a userid of **burnside** at the Netscape root directory of `/user/ns-home`, enter:

```
mkdceweb -s antietam -i burnside -t admin
```

**Note:** The `-n` option wasn’t specified because the Netscape root directory of `/usr/ns-home` is the default.

To configure DCE Web Secure only into the **gettysburg** web server with a Netscape root directory of `/usr/ns-home` with unauthenticated DFS access turned on, enter:

```
mkdceweb -s gettysburg -u on -t secure
```

**Note:** The web server userid specified in the Netscape web server configuration file, `magnus.conf`, will not be changed. Also the `-n` option wasn’t specified because the Netscape root directory of `/usr/ns-home` is the default.

**Related Information**

Commands: **rmdceweb**.
Purpose

Display process information for partially or fully configured DCE components.

Synopsis

ps.dce

Options

This command has no options.

Description

The ps.dce command displays the process information for partially or fully configured DCE components.

Privileges Required

No privileges are required.

Operations

None.

Output

The ps.dce command runs the “ps -u 0 -l” command to retrieve the process information for DCE daemons. Refer to the documentation supplied with your system for information on the output of the ps command.

Examples

# ps.dce
Gathering current configuration information...

DCE Daemons

<table>
<thead>
<tr>
<th>F</th>
<th>S</th>
<th>UID</th>
<th>PID</th>
<th>PPID</th>
<th>C</th>
<th>PRI</th>
<th>NI</th>
<th>ADDR</th>
<th>SZ</th>
<th>WCHAN</th>
<th>TTY</th>
<th>TIME</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>S</td>
<td>0 78</td>
<td>1 0</td>
<td>40</td>
<td>20</td>
<td>?</td>
<td>1077</td>
<td>?</td>
<td>?</td>
<td>0:18</td>
<td>dced</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>0 7898</td>
<td>1 0</td>
<td>40</td>
<td>10</td>
<td>?</td>
<td>1339</td>
<td>?</td>
<td>?</td>
<td>0:14</td>
<td>secd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>0 7912</td>
<td>1 0</td>
<td>40</td>
<td>20</td>
<td>?</td>
<td>1040</td>
<td>?</td>
<td>?</td>
<td>0:04</td>
<td>csadv</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>0 7934</td>
<td>1 0</td>
<td>51</td>
<td>20</td>
<td>?</td>
<td>1123</td>
<td>?</td>
<td>?</td>
<td>0:09</td>
<td>csdd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>0 7957 7875</td>
<td>0 44</td>
<td>20</td>
<td>?</td>
<td>934</td>
<td>?</td>
<td>?</td>
<td>0:01</td>
<td>pwd_stre</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>0 7971 7875</td>
<td>0 47</td>
<td>20</td>
<td>?</td>
<td>930</td>
<td>?</td>
<td>?</td>
<td>0:00</td>
<td>auditd</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>0 7984 7875</td>
<td>0 40</td>
<td>20</td>
<td>?</td>
<td>1038</td>
<td>?</td>
<td>?</td>
<td>0:01</td>
<td>gdad</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ps.dce completed successfully.
rmdceweb

Purpose

Unconfigures DCE Administration, DCE Web Secure, or both from a Netscape FastTrack 3.01 or Enterprise 3.61 web server.

Synopsis

rmdceweb
[-n netscape_dir]
-s netscape_id
[-t all | secure | admin]
[-c]
[-v]
[-?]  

Options

-n netscape_dir
Identifies the Netscape server root directory. The default is /usr/netscape/suitespot.

-s netscape_id
Identifies the Netscape server identifier to unconfigure.

-t all | secure | admin (AIX only)
Specifies which component to unconfigure. The default is all.

-c(lean)
Specifies to remove archived configuration data files.

-v(erbose)
Specifies to prompt for any values not supplied.

-? Displays a help message.

Description

The rmdceweb command is used to unconfigure DCE Administration, DCE Web Secure, or both, from a Netscape FastTrack 3.01 or Enterprise 3.61 web server. You must unconfigure DCE Web Secure before uninstalling it. Unconfiguring DCE Web Secure also unconfigures DCE Administration, if it was configured. Unconfiguration returns the Netscape servers to a non-DFS and a non-DCE state by removing DCE Web Secure or DCE Administration information or both.

Examples

To unconfigure both DCE Administration and DCE Web Secure from the bullrun web server with a Netscape root directory of /usr/netscape, enter:

rmdceweb -n /usr/netscape -s bullrun

To unconfigure DCE Web Secure only from the gettysburg web server, enter:

rmdceweb -s gettysburg -t secure
Related Information

Commands: mkdceweb.
rmreg.dce

Purpose

Removes information about a DCE cell from the domain namespace.

Synopsis

```bash
rmreg.dce
[–dns_cell_name dns_cell_name]
[–named_data_file named_data_file]
[usage]
[–?]  
[help]  
[operations]
```

Options

```bash
–dns_cell_name dns_cell_name
   Specifies the cell name to be unregistered. If no –dns_cell_name option is specified, the rmreg.dce command uses the cell name in the /opt/dcelocal/dce_cf.db file.

–named_data_file named_data_file
   Specifies the name of the file on the domain name server that contains the data for the named daemon. The default is /etc/named.data.

usage
   Displays a help message.

–?
   Displays a help message.

help
   Displays a brief description for the passed arguments.

operations
   Lists all the options and the components.
```

Description

The `rmreg.dce` command removes entries from the database maintained by your domain name server (the `named` daemon) that were added by the `mkreg.dce` command.

This command cannot be used to unregister X.500–style cell names.

This command must be run on the nameserver with which the cell is registered. Use the `–named_data_file` option to specify the name of the data file used by the name daemon (`named`). The cell information is removed from the specified file. In addition, `named` is refreshed if it is already running.

If the nameserver machine is not part of the DCE cell, the `–dns_cell_name` option must be used.

Examples

To unregister a cell named `/.../comp.sci.cell`, enter:
rmreg.dce

rmreg.dce –dns_cell_name /.../comp.sci.cell

Related Information

Commands: mkreg.dce.
unconfig.dce

Purpose

Removes configuration of the DCE components.

Synopsis

```
unconfig.dce
[–admin_pwd password]
[–cell_admin cell_admin_id]
[–config_type {full | local | admin}]
[–dce_hostname dce_hostname]
[–dependents]
[–force]
[–group_rsp_path filename]
[–host_id machine_identifier]
[–pwdstr_principal password_strength_principal_id]
[–rsp_file filename]
[all]
[usage]
[–?]  
[help]
[operations]
components
```

Note: The `unconfig.dce` command can be used to unconfigure both a full or a slim DCE client.

To Remove Admin Configuration:

```
unconfig.dce –config_type admin –dce_hostname dce_hostname
[–cell_admin cell_admin_id] [–host_id machine_identifier] [–dependents]
[–force] [–pwdstr_principal password_strength_principal_id] components
```

Note: When performing admin unconfiguration, unconfig.dce assumes that all components are configured on the target machine. This allows clean up of everything for that machine and is useful in instances where a previous unconfiguration attempt has failed. Because of this assumption, the -force flag should be used if specific components are to be unconfigured.

To Remove Local Configuration:

```
```

To Remove Full Configuration:

```
unconfig.dce –config_type full [–cell_admin cell_admin_id] [–dependents]
[–force] [–pwdstr_principal password_strength_principal_id] components
```

Options

`–admin_pwd password`

Specifies the cell administrator password. Caution should be used with this option because of the security risk it poses by making this password accessible to others.
unconfig.dce

-cell_admin cell_admin id
   Specifies the name of the cell administrator account. If the -cell_admin
   option is not specified, the account cell_admin will be assumed.

-config_type {full | local | admin}
   Used to specify what type of unconfiguration is to be done. The
   -config_type option has three available unconfig_types:

   admin
      Specifies that the admin portion of unconfiguration will be
      completed for the dce_host indicated by the -dce_hostname flag.
      This cleans up the CDS namespace and security registry. The user
      must have cell administrator authority within the cell.

   local
      Specifies that the local portion of unconfiguration will be completed
      for the local machine. This stops the daemons and removes the
      appropriate files. The user must have root authority on the local
      machine.

      Local unconfiguration must be selected when unconfiguring a CDS
      server whose clearinghouse contains a master replica of a directory.

   full
      Specifies full unconfiguration on the local machine. This is the
      default unconfig_type. When fully unconfiguring the local host, the
      user must be the DCE cell administrator and have root authority on
      the local machine. Full unconfiguration is the equivalent of admin
      unconfiguration and local unconfiguration combined. If the
      -config_type option is not used, a full unconfiguration will be used.

-dce_hostname dce_hostname
   Used with the -config_type option to identify the dce_host to unconfigure.
   Use -dce_hostname only when doing the admin portion of unconfiguration.

-dependents
   Unconfigures dependent components. Specifies that any components that
   depend on those listed on the command line should also be unconfigured.
   For example, on a machine with sec_cl, cds_cl, and rpc, unconfig.dce
   -dependents sec_cl will also unconfigure the cds_cl.

-force
   Forces unconfiguration of components named on the command line, even if
   other components depend on their presence. Use this option in clean-up
   situations. Use this option with extreme caution because the cell can be put
   into an unstable state.

-group_rsp_path filename
   Specifies a directory path to use when searching for included response
   files.

-host_id machine identifier
   Specifies the TCP/IP hostname or the TCP/IP address of the client machine
   being admin unconfigured. When unconfig.dce is called with -config_type
   admin, the -host_id option must also be used. Admin unconfiguration can
   be used for a machine whose TCP/IP address is not yet registered with a
   nameserver. In that situation, use the -dce_hostname dce_hostname
   option with the -host_id IP_address option.

   Note: The -host_id option can be used only with the -config_type admin
   option.
unconfig.dce

-pwdstr_principal password strength principal id
   Specifies a principal ID for the password-strength server. Because more
   than one password-strength server can be configured, the principal ID is
   used to identify a specific server.

all    Unconfigures all configured components on the local machine.

-rsp_file filename
   Specifies the full path name of a response file.

usage    Displays a help message.
-?        Displays a help message.
help      Displays a brief description for the passed arguments.

operations
   Lists all the options and the components.

components
   Specifies the components to be unconfigured.

   The Client Components are:
   all_cl    All clients (cds_cl, dts_cl, rpc, and sec_cl).
   client    Same as all_cl.
   cds_cl    CDS client.
   dts_cl    DTS client. This component and dts_local and dts_global are
             mutually exclusive.
   rpc       RPC daemon.
   sec_cl    Security client.
   slim_cl   Slim client.
   dce_unixd Integrated login (AIX only).
   pam       Password Authentication Module (Solaris only)
   nsswitch  Name Service Switch (Solaris only)

   The Server Components are:
   all_srv    All servers (cds_second, cds_srv, dts_global, dts_local, gda,
               sec_srv, ems_srv, nsid, pw_strength_srv, sec_rep, snmp_srv).
   audit      Audit daemon.
   cds_second Secondary CDS server. This component and cds_srv are mutually
               exclusive.
   cds_srv    Initial CDS server for the cell. This component and cds_second are
               mutually exclusive.
The `unconfig.dce` command stops and unconfigures the specified DCE components. The `unconfig.dce` command unconfigures only the core DCE components. Use the `unconfig.dfs` command to configure DFS components. If you are removing all DCE and DFS components, use the `unconfig.dfs` command before you use the `unconfig.dce` command.

You can unconfigure a machine from a cell in two ways:

**full configuration**
Used by the cell administrator (as root user) to complete all the necessary steps within the cell (updating the CDS namespace and the security registry) and on the local machine (stopping daemons and deleting files). Full unconfiguration is specified with the `--config_type full` option. The `unconfig.dce` command also defaults to full unconfiguration if the `--config_type` option is not used.

If the cell administrator does not have root user access to the machine that is going to be unconfigured use split configuration.

**Note:** If you unconfigure the initial CDS server (with the master copy of the `/directory`) or master Security server in a cell, you will have to uninstall and reconfigure the entire cell.

**split configuration**
Breaks the unconfiguration tasks into two distinct segments, admin and local. Admin unconfiguration is used by the cell administrator on any
unconfig.dce

machine within the cell to update the CDS namespace and the security registry about changes in the cell. Local configuration is used by the root user on the machine being unconfigured to stop the daemons and delete the appropriate files. Split unconfiguration is specified with the –config_type admin and –config_type local options.

If the cell for which a machine is configured is inaccessible and you need to unconfigure the machine for any reason, use the –config_type local option. This option limits the unconfig.dce command to remove only the local pieces of a DCE configuration; it does not remove entries from the namespace or registry database. To remove the entries from the namespace and registry database, the cell administrator should use the –config_type admin –dce_hostname option from a machine within the cell.

If the environment variable cell_admin_pw is set, unconfig.dce uses its value for the cell administrator password without prompting you. This feature can be useful when automating unconfiguration tasks. This use should be limited because of the security risk it poses by making this password accessible to others. The cell administrator password should be changed after the tasks are completed and the cell_admin_pw value is unset in order to limit the security risk.

While the config.dce command automatically configures any client components required by the specified components, the unconfig.dce command will fail if configured components depend on the presence of those requested to be unconfigured. To unconfigure exactly those components specified on the command line, use the –force option. This option should be used with caution. In some cases, unconfiguring one component will disable other components that are dependent upon it.

To unconfigure components and all components that depend on them, use the –dependents option. When a CDS server (either cds_srv or cds_second) is requested to be unconfigured, unconfig.dce checks for several conditions. If a full configuration is specified, unconfig.dce checks to ensure that none of the clearinghouses on the server machine contain a master replica of any directory. If they do not, the unconfiguration continues. If they do, configuration exits with a message explaining what must be done before the server can be unconfigured. If it is necessary to unconfigure a CDS server with a clearinghouse that contains a master replica of a directory, unconfig.dce can be run using the –config_type local option. After that, unconfig.dce –config_type admin should be run from a machine within the cell.

When a CDS server has been unconfigured, it might not be possible to reconfigure it using the same dce_hostname until all updates have taken place in the cell. Either use a different dce_hostname or wait overnight before reconfiguring.

If you unconfigure the Master Security server in a cell, you will have to unconfigure and reconfigure the entire cell.

Examples

To remove the DTS clerk configuration from a machine when the cell administrator account name is ca, enter:

    unconfig.dce –cell_admin ca dts_cl
To remove all DCE configuration files and databases from a machine, the root user enters:

\texttt{unconfig.dce \text{-config_type local all}}

The \texttt{unconfig.dce \text{-config_type local}} option limits the \texttt{unconfig.dce} command to removing only the local pieces of a DCE configuration; it does not remove entries from the namespace or registry database. Use this command when unconfiguring the last machine in a DCE cell or when removing a CDS server whose clearinghouse contains a master replica of any directory.

To specify the admin portion of unconfiguration for a client in an existing cell, the cell administrator enters:

\begin{verbatim}
unconfig.dce \text{-config_type\textbackslash admin}\text{-dce\_hostname}\text{ chc all\_cl}
unconfig.dce \text{-config_type\textbackslash admin}\text{-dce\_hostname}\text{ jas.austin.ibm.com\text\backslash cds\_second cds\_cl sec\_cl dts\_cl}
\end{verbatim}

To specify the local portion of unconfiguration for the local machine, the root user (with no DCE authority required) enters:

\begin{verbatim}
unconfig.dce \text{-config_type local all\_cl}
unconfig.dce \text{-config_type local \text{-dependents sec\_srv}}
\end{verbatim}

To perform full unconfiguration on a client in an existing cell, the cell administrator with root user authority enters:

\begin{verbatim}
unconfig.dce all\_cl
unconfig.dce \text{-config_type full all\_cl}
\end{verbatim}

\section*{Related Information}

Commands: \texttt{config.dce}.
unconfig.dce
Appendix. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs
and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department LZKS
11400 Burnet Road
Austin, TX 78758
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM’s suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written.

These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM’s application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:
Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

- AIX
- IBM

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.
Index

Special Characters
#define 5, 371
/krb5/v5srvtab file 645
idbase.h 366
passwd_override file 610
group_override file 608

A
account
  administering 14
accounts
    importing 676
    viewing registry information 687
acl
  administering 27
ACL
    dls_audit_events 600
    acl_edit command 648
ACLs
    editing entries 648
    viewing 648
attributes
  NSI, viewing 422
attrlist
  manipulating 41
aud
  administering 46, 62
aud_audit_events 581
audevents
  administering 52
audfilter
  administering 56
audit daemon 657
audit services
  auditable events 581
audit trail file 658
auditable events
  audit services 581
  security services 613
  time services 600
auditd command 657
  privileges required to run 658
authentication services
  rpc_c_authn_dce_secret 613
auxiliary file
  client, server 368

B
binding information (RPC)
  exporting to server entries 401
  removing information 434
  viewing server entries 432
browser
  startup command 446

C
C language 369
  compiler 369
  preprocessor 370
cache
  clearing servers 464
  defining servers 475
  viewing contents 486
  viewing server addresses 519
cached clearinghouse entity 517
CDS clerks
  debugging 443
  managing interface to servers 447
  setting confidence levels 506
  solicitation daemon startup 444
  stopping 484
  viewing attributes 529
  viewing cache contents 486
cds_dbdump command 459
CDS servers
  clearing clearinghouses 465
  clearing from cache 464
  debugging 443, 461
  defining in local cache 475
  dumping 459
  restarting 457
  solicitation daemon startup 444
  stopping 485
  viewing attributes 542
  viewing cached addresses 519
cdsalias
  administering 70
cdscache
  administering 74
cdscp command 449
cdscp commands
  about 438
  add directory 439
  add object 441
  catraverse 443
cds_dbdump 459
cds_diag command 461
cdsadv 444
cdsbrowser 446
cdsclerk 447
cdsdel 460
cdssli 462
clear cached server 464
clear clearinghouse 465
create child 467
create clearinghouse 468
create directory 470
create link 471
create object 473
create replicas 474
define cached server 475
delete child 477
cdscp commands (continued)
delete clearinghouse 478
delete directory 480
delete link 481
delete replica 483
disable clerk 484
disable server 485
dump clerk cache 486
list child 491
list clearinghouse 493
list directory 495
list link 497
list object 499
remove directory 501
remove link 503
remove object 504
set cdscp confidence 506
set cdscp preferred clearinghouse 507
set directory 508
set directory to new epoch 510
set directory to skulk 512
set link 513
set object 515
show cached clearinghouse 517
show cached server 519
show cdscp preferred clearinghouse 520, 521
show cell 522
show child 524
show clearinghouse 526
show clerk 529
show directory 531
show link 534
show object 536
show replica 539
show server 542
summary 449
syntax 452
cdssd command 457
cdssdel command 460
cdsli command 462
cell
DCECP object 84
cell alias
administering 91
cell directory client
administering 80
cell directory server
administering 66
cell names
conventions 385
creating 522
cell service profile
global-set membership 600
cells
deleting 460
listing 462
CEPV 370
child entity 524
child pointers
creating 467
child pointers (continued)
deleting 477
viewing 491
viewing attributes 524
chesite command 734
clean_up.dce command 736
clearinghouse
administering 94
clearinghouses
creating 468
deleting 478
making available 468
preferred 507, 520, 521
viewing 493
viewing attributes 526
viewing cached 517
clerk entity 529
client
Audit 606
auxiliary file 368
files 368
stub 368
client entry point vector 370
clock
administering 104
clocks
adjusting 578
synchronizing 576
commands
csrc 110
dcecp command 600
for RPC programmers 366
idl 366, 368
sams 2
uuidgen 366, 377
compiler programs
idl –spmi 375
compilers
C 369
IDL 368
config.dce command 714
control programs
exiting 400
quitting 409
credentials
promoting of 672

D
DACL Management
interfaces 615
rdaclif 615
data types
IDL-to-C mappings 366
of IDL 366
DCE Audit
auditable events 580
files 580
DCE Control Program commands
dcecp 118
DCE host daemon
about 136
dce_login command 659
DCE RPC
programmer commands 366
DCE RPC entity
idl command 368
uuidgen command 377
dcecp(\)(.) commands
uuid 352
dcecp command 581, 600, 613
dcecp commands
account 14
acl 27
attrlist 41
aud 46
audfilter 56
audtrail 62
cds 66
cdsalias 70
cdsarchive 74
cdsclient 80
cellalias 91
clearinghouse 94
clock 104
directory 139
dts 154
ems 167
emsconsumer 170
emss 186
emsevents 175
emsfilters 179
emss 183
endpoint 187
group 84, 197, 207, 215, 223, 338
link 232
log 238
name 243
object 247
organization 253
principal 265
registry 274
_rpcentry 295
_rpcgroup 304
_rpcprofile 311
_secval 321
_server 326
_utc 347
attrschema 355
dcecred_* Files 662
dcesetup command 747
dceuindex command 664
delegation 633
directories 369
adding attributes (CDS) 439
changing attribute values (CDS) 508
child pointers (CDS) 467, 477
creating (CDS) 470
deleting (CDS) 480
directories 369 (continued)
removing attribute values (CDS) 501
sams command 2
updating (CDS) 512
viewing (CDS) 495
viewing attributes (CDS) 531
directory
administering 139
directory entity 531
directory pathnames
conventions 386
Domain Name Service (DNS)
defining cell names 522
dts
administering 154
dts_audit_events 600
ACL 600
DTS clerks
creating 550
deleting 551
modifying 564
starting 560
stopping 552
viewing characteristics 568
DTS control program
exiting 561, 563
invoking 553
DTS entity 601
DTS servers
advertising 548
creating 550
deleting 551
modifying 564
removing entries from profile 577
starting 560
stopping 552
viewing characteristics 568
dtsscp commands
advertise 548
change 549
create 550
delete 551
disable 552
enable 560
exit 561
help 562
quit 563
set 564
show 568
summary 546
synchronize 576
syntax 553
unadvertise 577
update 578
dtsd command 556
dtsd process
restarting 556
dtsdate command 558
E
ems
managing 167
emsconsumer
  managing 170
emsd
  starting ems services 186
emsevents
  events 175
emsefilter
  managing 179
emselog
  managing 183
endpoint
  administering 187
Endpoint Map Service 380
endpoint maps
  about 380
    add or replace server address information 395
    managing 382
    removing elements 414
    viewing elements 426
endpoints
  about 380
entities
  about 453
entry point vector 370
epochs
  changing 549
EPV 370
event class 606
  auditing execution 600
    definitions 581, 600, 613
event class file 606
  format 606
  naming convention 606
  SEP line 606
events
  audit service operations 581
  audit services 581
  auditable 581, 600, 613
  clock readings 600
dts_audit_events 600
  global-set membership 600
  security service operations 613
  security services 613
  time service attributes 600
time service processes 600
time services 600
exit command 400
extended registry attributes (ERAs) 633

F
files
  passwd_override 580
  v5srvtab 580
group_override 580
auxiliary 368
catalog 3
client 368
dts_audit_events 580
event class 606
event_class 580
header 3, 5, 369
files (continued)
  header file 3
  input 2
  keytab 645
  message 3
  output 2, 3
  problem determination 3
  registry database override 608, 610
  sams 2
  sec_audit_events 580
  security administration 580
  server 368
  serviceability table 3
  stub 368
  uuidgen command 379
functions
  rdacl_get_access() 616
  rdacl_get_manager_types() 617
  rdacl_get_referral() 617
  rdacl_lookup() 615
  rdacl_replace() 615
  rdacl_test_access() 616
  rpriv_get_ptgt() 617
  rs_acct_add() 618, 619
  rs_acct_delete() 619
  rs_acct_get_projlist() 622
  rs_acct_lookup() 620
  rs_acct_replace() 620
  rs_auth_policy_get_effective() 631
  rs_auth_policy_get_info() 630
  rs_auth_policy_set_info() 631
  rs_login_get_info() 622
  rs_pgo_add() 623
  rs_pgo_add_member() 626
  rs_pgo_delete() 623
  rs_pgo_delete_member() 626
  rs_pgo_get() 625
  rs_pgo_get_members() 627
  rs_pgo_is_member() 627
  rs_pgo_key_transfer() 625
  rs_pgo_rename() 624
  rs_pgo_replace() 624
  rs_policy_get_info() 629
  rs_policy_set_info() 629
  rs_properties_get_info() 628
  rs_properties_set_info() 628
  rs_rep_admin_maint() 632
  rs_rep_admin_mkey() 632
  rsec_krb5rpc_sendto_kdc() 613
G
gbl_time_service
time server 600
gdad command 487
gdad process 487
getcellname command 195
getip command 196
Global Directory Agent (GDA)
  starting daemon 487
Global Directory Service (GDS)
  defining cell names 522
global names
  conventions 385
global servers
  removing entries 577
group
  administering 197
groups
  adding members 686
  adding members to name service entries 398
  adding to registry 685
  changing registry information 685
  deleting 686
  naming 388
  removing from NSI entry 413
  removing members 416
  viewing members 424
  viewing registry information 684

H
  header file 5, 369
  headers 4
  host
    DCECP object 207
  hostdata
    DCECP object 215

I
  identifiers
    generator 377
    uuidgen command 377
  IDL 368
    base data types 366
    compiler 368
    file template 377
    IDL-to-C data type mappings 366
  idl_macros 366
  idl –spmi command 375
  idl command 366, 368
    options 368
  IDL compiler 366
  interface definition 372
  Interface Definition Language 368
  Interface Definition Language compiler 366

K
  k5dcelogin command 672
  kdestroy command 667
  kerberos.dce command 759
  keytab
    DCECP object 223
  keytab file 645
  kinit command 668
  klist command 671
  krb5rpc interface 613

L
  ldap_addcell
    registering cell information 489
leaf names
  conventions 386
  link
    administering 232
  link entity 534
local names
  conventions 385
  overriding CDS syntax 384
  locksmith mode 710
  log
    serviceability 238
  login
    preventing 612

M
  macros
    idl_ 366
  marshalling 371
  master keys
    creating 697
  messages
    informational 372
    strings 2
    system files 2
    warning 372, 373
  mkdceweb command 763
  mkreg.dce command 761

N
  name
    administering 243
  Name Service Interface (NSI) 384
    accessing 380
    command syntax 385
    importing binding information 406
    managing for RPC applications 382
    naming guidelines 386
    viewing NSI attributes 422

O
  object
    administering 247
  object entity 536
  objects
    adding attributes 441
    changing attribute values 515
    creating 473
    deleting 482
    removing attribute values 504
    viewing attributes 536
    viewing entries 499
  organization
    administering 253
  organizations
    adding members 686
    adding to registry 685
    changing registry information 685
    deleting 686

Index 785
organizations (continued)
  viewing registry information 684
orphans
  adopting 687

P
password strength operations 643
passwords
  backing up files 673
  managing server 691
  storing server and machine 645
preprocessor 370
principal
  administering 265
principals
  adding to registry 685
  deleting 686
  destroying login context 667
  setting security for 659
  storing ticket caches 662
  viewing registry information 684
privilege server
  interfaces 617
  rpriv 617
profiles
  adding elements 390
  naming 388
  removing elements 410
  removing from namespace 418
  viewing elements 429
programmer commands 366
protocol sequences
  rpcprotseqs 419
ps.dce command 765

Q
quit command 409

R
rdacl_get_access() function 616
rdacl_get_manager_types() function 617
rdacl_get_referral() function 617
rdacl_lookup() function 615
rdacl_replace() function 615
rdacl_test_access() function 616
rdaclif interface 613
rdaclif operations 615
rdacliftmp interface 613
registry
  administering 274
  local overrides 608, 610
registry administration 631
  interfaces 631
registry database
  creating 706
  updating 697
registry database override
  files 608, 610
registry miscellaneous operations
  interfaces 622
  rs_misc 622
registry objects
  adopting 687
registry PGO
  interfaces 623
  rs_pgo 623
registry policy
  interfaces 628
  rs_policy 628
registry server
  interfaces 618
  rs_acct 618
registry server attributes 633
replica sets
  reconstructing 510
replicas
  creating 474
  deleting 480
  deleting (CDS) 483
  viewing attributes (CDS) 539
rgy_edit subcommands
  site 692
  add 685
  authpolicy 694
  change 685
  defaults 694
  delete 686
  domain 692
  exit 694
  help 694
  ktadd 691
  ktdelete 692
  ktlist 691
  login 694
  member 686
  policy 693
  properties 692
  quit 694
  scope 694
  view 684, 687
rmdcweb command 766
rmreg.dce command 768
rmxcred command 695
RPC
  programmer commands 366
rpc_c_authn_dce_secret authentication service 613
rpc commands
  idl –spmi 375
  ldap_addcell 489
  rpcprotseqs 419
  rpcresolve 420
RPC control program
  adding entries to namespace 393
  environment variables 384
  initializing 381
  removing entries 412
RPC daemon
  about 380

786 IBM DCE Version 3.1 for AIX and Solaris: Administration Commands Reference