DCE 1.2.2 Command Reference

OSF® DCE Product Documentation

The Open Group

OSF® DCE Product Documentation:

DCE 1.2.2 Command Reference

ISBN 1–85912–138–1

Document Number F212

Published in the U.K. by The Open Group, 1997.

Any comments relating to the material contained in this document may be submitted to:

The Open Group
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:
OGPubs@opengroup.org

OTHER NOTICES

Contents

Preface vii

The Open Group vii
The Development of Product Standards viii
Open Group Publications ix
Versions and Issues of Specifications xi
Corrigenda xi
Ordering Information xi
This Book xii
Audience xii
Applicability xii
Purpose xii
Document Usage xii
Related Documents xiii
Typographic and Keying Conventions xiv
Problem Reporting xv
Pathnames of Directories and Files in DCE Documentation xv

Trademarks xv

Chapter 1. DCE Commands 1
sams 2
svcdumplog 12
dce_intro 14
account 17
acl 35

attrlist 53
aud 59
audevents 66
audfilter 70
audtrail 78
cds 82
cdsalias 87
cdscache 92
cdsclient 100
cell 105
cellalias 114
clearinghouse 118
clock 131
csrc 139
dce_config 145
dcecp 162
dced 186
directory 190
dts 210
endpoint 227
getcellname 239
getip 240
group 241
host 255
hostdata 266
hostvar 276
keytab 281
link 293
log 301
name 307
object 312
organization 320
principal 337
registry 349
rpcentry 377
rpcgroup 389
rpcprofile 398
secval 412
server 418
user 435
utc 448
uuid 454
xattrschema 458

Chapter 2. Remote Procedure Call Commands 473
rpc_intro 474
idl 476
uuidgen 485
rpc_intro 489
rpccp 491
add element 504
add entry 508
add mapping 510
add member 514
export 516
help 521
import 524
remove element 528
remove entry 531
remove group 533
remove mapping 535
remove member 538
remove profile 540
show entry 542
show group 545
show mapping 548
show profile 552
show server 556
unexport 559

Chapter 3. Cell Directory Service Commands 563
cds_intro 564
add directory 566
add object 568
cdsadv 570
cdsbrowser 572
cdsclerk 573
cdscp 575
cdsd 585
clear cached server 587
clear clearinghouse 589
create child 591
create clearinghouse 593
create directory 595
create link 597
create object 599
create replica 601

define cached server 603
delete child 605
delete clearinghouse 607
delete directory 609
delete link 611
delete object 613
delete replica 615
disable clerk 617
disable server 619
dump clerk cache 620
gdad 621
list child 623
list clearinghouse 625
list directory 627
list link 629
list object 631
remove directory 633
remove link 635
remove object 637
set cdscp confidence 639
set cdscp preferred clearinghouse 641
set directory 643
set directory to new epoch 646
set directory to skulk 648
set link 650
set object 652
show cached clearinghouse 654
show cached server 656
show cdscp confidence 658
show cdscp preferred clearinghouse 660
show cell 662
show child 664
show clearinghouse 667
show clerk 671
show directory 674
show link 679
show object 682
show replica 684
show server 689

Chapter 4. Distributed Time Service Commands 693
dts_intro 694
advertise 696

change 698
create 700
delete 702
disable 704
dtscp 706
dtsd 710
dtsdate 713
enable 716
exit 718
help 719
quit 721
set 722
show 728
synchronize 740
unadvertise 742
update 743

Chapter 5. Security Service Files and Commands 745
sec_intro 746
aud_audit_events 748
dts_audit_events 751
event_class 756
group_override 758
passwd_override 761
sec_audit_events 766
v5srvtab 787
sec_intro 788
acl_edit 790
auditd 801
chpass 804
dce_login 805
k5dcelogin 807
k5login 808
kdestroy 809
kinit 810
klist 813
passwd_export 815
passwd_import 817
pwd_strengthd 822
rgy_edit 824
rlogin 841
rlogind 845
rsh 848

rshd 852
sec_admin 855
sec_create_db 868
sec_salvage_db 871
secd 892
su 897

Index Index–1

Preface

The Open Group

The Open Group is the leading vendor-neutral, international consortium for buyers and suppliers of technology. Its mission is to cause the development of a viable global information infrastructure that is ubiquitous, trusted, reliable, and as easy-to-use as the telephone. The essential functionality embedded in this infrastructure is what we term the IT DialTone. The Open Group creates an environment where all elements involved in technology development can cooperate to deliver less costly and more flexible IT solutions.

Formed in 1996 by the merger of the X/Open Company Ltd. (founded in 1984) and the Open Software Foundation (founded in 1988), The Open Group is supported by most of the world's largest user organizations, information systems vendors, and software suppliers. By combining the strengths of open systems specifications and a proven branding scheme with collaborative technology development and advanced research, The Open Group is well positioned to meet its new mission, as well as to assist user organizations, vendors, and suppliers in the development and implementation of products supporting the adoption and proliferation of systems which conform to standard specifications.

With more than 200 member companies, The Open Group helps the IT industry to advance technologically while managing the change caused by innovation. It does this by:

- consolidating, prioritizing, and communicating customer requirements to vendors
- conducting research and development with industry, academia, and government agencies to deliver innovation and economy through projects associated with its Research Institute
- managing cost-effective development efforts that accelerate consistent multi-vendor deployment of technology in response to customer requirements
- adopting, integrating, and publishing industry standard specifications that provide an essential set of blueprints for building open information systems and integrating new technology as it becomes available
- licensing and promoting the Open Brand, represented by the "X" mark, that designates vendor products which conform to Open Group Product Standards
- promoting the benefits of IT DialTone to customers, vendors, and the public.

The Open Group operates in all phases of the open systems technology lifecycle including innovation, market adoption, product development, and proliferation. Presently, it focuses on seven strategic areas: open systems application platform development, architecture, distributed systems management, interoperability, distributed computing environment, security, and the information superhighway. The Open Group is also responsible for the management of the UNIX trademark on behalf of the industry.

The Development of Product Standards

This process includes the identification of requirements for open systems and, now, the IT DialTone, development of CAE and Preliminary Specifications through an industry consensus review and adoption procedure (in parallel with formal standards work), and the development of tests and conformance criteria.

This leads to the preparation of a Product Standard which is the name used for the documentation that records the conformance requirements (and other information) to which a vendor may register a product. There are currently two forms of Product

Standard, namely the Profile Definition and the Component Definition, although these will eventually be merged into one.

The “X” mark is used by vendors to demonstrate that their products conform to the relevant Product Standard. By use of the Open Brand they guarantee, through the X/Open Trade Mark License Agreement (TMLA), to maintain their products in conformance with the Product Standard so that the product works, will continue to work, and that any problems will be fixed by the vendor.

Open Group Publications

The Open Group publishes a wide range of technical documentation, the main part of which is focused on specification development and product documentation, but which also includes Guides, Snapshots, Technical Studies, Branding and Testing documentation, industry surveys, and business titles.

There are several types of specification:

CAE Specifications

CAE (Common Applications Environment) Specifications are the stable specifications that form the basis for our Product Standards, which are used to develop X/Open branded systems. These specifications are intended to be used widely within the industry for product development and procurement purposes.

Anyone developing products that implement a CAE Specification can enjoy the benefits of a single, widely supported industry standard. Where appropriate, they can demonstrate product compliance through the Open Brand. CAE Specifications are published as soon as they are developed, so enabling vendors to proceed with development of conformant products without delay.

Preliminary Specifications

Preliminary Specifications usually address an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations. They are published for the purpose of validation through implementation of products. A Preliminary Specification is not a draft specification; rather, it is as

stable as can be achieved, through applying The Open Group's rigorous development and review procedures.

Preliminary Specifications are analogous to the trial-use standards issued by formal standards organizations, and developers are encouraged to develop products on the basis of them. However, experience through implementation work may result in significant (possibly upwardly incompatible) changes before its progression to becoming a CAE Specification. While the intent is to progress Preliminary Specifications to corresponding CAE Specifications, the ability to do so depends on consensus among Open Group members.

Consortium and Technology Specifications

The Open Group publishes specifications on behalf of industry consortia. For example, it publishes the NMF SPIRIT procurement specifications on behalf of the Network Management Forum. It also publishes Technology Specifications relating to OSF/1, DCE, OSF/Motif, and CDE.

Technology Specifications (formerly AES Specifications) are often candidates for consensus review, and may be adopted as CAE Specifications, in which case the relevant Technology Specification is superseded by a CAE Specification.

In addition, The Open Group publishes:

Product Documentation

This includes product documentation—programmer's guides, user manuals, and so on—relating to the Prestructured Technology Projects (PSTs), such as DCE and CDE. It also includes the Single UNIX Documentation, designed for use as common product documentation for the whole industry.

Guides

These provide information that is useful in the evaluation, procurement, development, or management of open systems, particularly those that relate to the CAE Specifications. The Open Group Guides are advisory, not normative, and should not be referenced for purposes of specifying or claiming conformance to a Product Standard.

Technical Studies

Technical Studies present results of analyses performed on subjects of interest in areas relevant to The Open Group's Technical Program. They

are intended to communicate the findings to the outside world so as to stimulate discussion and activity in other bodies and the industry in general.

Versions and Issues of Specifications

As with all live documents, CAE Specifications require revision to align with new developments and associated international standards. To distinguish between revised specifications which are fully backwards compatible and those which are not:

- A new Version indicates there is no change to the definitive information contained in the previous publication of that title, but additions/extensions are included. As such, it replaces the previous publication.
- A new Issue indicates there is substantive change to the definitive information contained in the previous publication of that title, and there may also be additions/extensions. As such, both previous and new documents are maintained as current publications.

Corrigenda

Readers should note that Corrigenda may apply to any publication. Corrigenda information is published on the World-Wide Web at *http://www.opengroup.org/public/pubs*.

Ordering Information

Full catalogue and ordering information on all Open Group publications is available on the World-Wide Web at *http://www.opengroup.org/public/pubs*.

This Book

The *DCE 1.2.2 Command Reference* provides complete and detailed reference information to help system and network administrators use the correct syntax for OSF® 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 OSF DCE Version 1.2.2 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.

- For DCE cross-component commands, see Chapter 1.
- For DCE remote procedure call (RPC) commands, see Chapter 2.
- For DCE Cell Directory Service (CDS) commands, see Chapter 3.
- For DCE Distributed Time Service (DTS) commands, see Chapter 4.

- For DCE Security Service commands, see Chapter 5.

Related Documents

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

- *DCE 1.2.2 Introduction to OSF DCE*
  Document Number F201, ISBN 1–85912–182–9
- *DCE 1.2.2 Application Development—Introduction and Style Guide*
  Document Number F202, ISBN 1–85912–187–7
- *DCE 1.2.2 Application Development Reference*
  Document Number F205A, ISBN 1–85912–103–9 (Volume 1)
  Document Number F205B, ISBN 1–85912–108–X (Volume 2)
  Document Number F205C, ISBN 1–85912–159–4 (Volume 3)
- *DCE 1.2.2 Administration Guide—Introduction*
  Document Number F207, ISBN 1–85912–113–6
- *DCE 1.2.2 Administration Guide—Core Components*
  Document Number F208, ISBN 1–85912–118–7
- *DCE 1.2.2 DFS Administration Guide and Reference*
  Document Number F209A, ISBN 1–85912–123–3 (Volume 1)
  Document Number F209B, ISBN 1–85912–128–4 (Volume 2)
- *DCE 1.2.2 GDS Administration Guide and Reference*
  Document Number F211, ISBN 1–85912–133–0
- *DCE 1.2.2 File-Access Administration Guide and Reference*
  Document Number F216, ISBN 1–85912–158–6
- *DCE 1.2.2 File-Access User's Guide*
  Document Number F217, ISBN 1–85912–163–3
- *DCE 1.2.2 Problem Determination Guide*
  Document Number F213A, ISBN 1–85912–143–8 (Volume 1)
  Document Number F213B, ISBN 1–85912–148–9 (Volume 2)
- *DCE 1.2.2 Testing Guide*
  Document Number F215, ISBN 1–85912–153–5

- *DCE 1.2.2 File-Access FVT User's Guide*
  Document Number F210, ISBN 1–85912–189–6
- *DCE 1.2.2 Release Notes*
  Document Number F218, ISBN 1–85912–168–3

Typographic and Keying Conventions

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.

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 vendor-supplied documentation, contact your software vendor's customer service department. Comments relating to this Open Group document, however, should be sent to the addresses provided on the copyright page.

Pathnames of Directories and Files in DCE Documentation

For a list of the pathnames for directories and files referred to in this guide, see the *DCE 1.2.2 Administration Guide—Introduction* and *DCE 1.2.2 Testing Guide*.

Trademarks

Motif®, OSF/1®, and UNIX® are registered trademarks and the IT DialTone™, The Open Group™, and the "X Device"™ are trademarks of The Open Group.

DEC, DIGITAL, and ULTRIX are registered trademarks of Digital Equipment Corporation.

DECstation 3100 and DECnet are trademarks of Digital Equipment Corporation.

HP, Hewlett-Packard, and LaserJet are trademarks of Hewlett-Packard Company.

Network Computing System and PasswdEtc are registered trademarks of Hewlett-Packard Company.

AFS, Episode, and Transarc are registered trademarks of the Transarc Corporation.

DFS is a trademark of the Transarc Corporation.

Episode is a registered trademark of the Transarc Corporation.

Ethernet is a registered trademark of Xerox Corporation.

Chapter 1

DCE Commands

sams

Purpose

Builds DCE message system files

Synopsis

sams [**-d** *dest_dir*] [**-f**][**-g** *gencat_command*] [**-I** *interface*] [**-m**][**-n** *output_name*] [**-o** *output_files*] [**-s** *style*] [**-t** *table*] [**-x**]*input_file*

Options

-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 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** will automatically insert them when generating the catalog.

ARGUMENTS

input_file Specifies the message input file.

DESCRIPTION

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 technology and component fields are assigned by OSF; the message code is assigned by **sams** or specified in the input file.

The technology and component determine the name 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.

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:

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

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 for the DCE or DFS technology provided by OSF. 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 (that is, reference pages or the *DCE 1.2.2 Problem Determination Guide*).

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

reserved Same action as **obsolete**.

Each flag may be preceeded by the word **not** or a **!** (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 provided by OSF for which the messages are being generated. This header may 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 may be omitted; the default value is 1.

table *varname*
The name of the in-core message table that is created. This header may 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:

```
serviceability table dst_svc_table handle dts_svc_handle
start
    subcomponent dts_s_general  "general"  dts_i_svc_general
    subcomponent dts_s_provider "provider" dts_i_svc_provider
end
```

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:

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

start !intable undocumented
code dts_i_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:

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 may 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 catalog.

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 may 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.

administrator response *text* **operator response** *text* **programmer response** \
text **severity** *text* **system response** *text* **user response** *text* **vendor** *name text*

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 serviceabilty 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.

Errors

A representative list of errors that might be returned is not shown here. Refer to the *DCE 1.2.2 Problem Determination Guide* for complete descriptions of all error messages.

Related Information

Commands: **gencat(1)**.

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*

Options

-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 for a component's messages, 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.

ERRORS

A representative list of errors that might be returned is not shown here. Refer to the *DCE 1.2.2 Testing 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: **svcroute(5dce)**.

Books: *DCE 1.2.2 Porting and Testing Guide*, *DCE 1.2.2 Application Development Guide*.

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 Manipulates 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

dce_config Installs, configures, and starts up DCE

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
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 *DCE 1.2.2 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)**, **cdsalias(8dce)**, **cdscache(8dce)**, **cdsclient(8dce)**, **cell(8dce)**, **cellalias(8dce)**, **clearinghouse(8dce)**, **clock(8dce)**, **csrc(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)**.

Books: *DCE 1.2.2 Administration Guide*, *DCE 1.2.2 Application Development Guide*.

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:

- Fully qualified account names in the form **/...**/*cell_name*/*account_name* or **/.:**/*account_name*
- 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.

operation 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:

- Account attributes may or may not have default values. They assume a default value or a value set by administrators.
- 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.
- 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.

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 *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**.

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 *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. Its value is either **yes** or **no**. The default is **yes**.

usertouser {yes | no}
For server principals, a flag set to determine whether the server must use user-to-user authentication. Its value is either **yes** (must use user-to-user authentication) or **no** (uses server-key-based 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 ([-]*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. To specify the time, use 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 **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 256 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 256 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 ecryption. 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 ecryption. 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 | pkss}
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*.

The **pkss** argument indicates the public key information will be stored by the Private Key Storage Server.

See the *DCE 1.2.2 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 *DCE 1.2.2 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*]

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***}...{***attribute value***}}**

-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:

- **gmau** (**groups**, **mgmt_info**, **auth_info**, and **user_info**) permissions to the principal named in the account

- **rtM** (**read**, **test**, **Member_list**) permissions to the organization named in the account
- **tM** (**test**, **Member_list**) permissions to the group named in the account
- **r** (**read**) permission on the registry policy object.

Examples

```
dcecp> principal create John_Hunter
dcecp>
dcecp> group add users -member John_Hunter
dcecp>
dcecp> organization add users -member John_Hunter
dcecp>
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 *DCE 1.2.2 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

```
dcecp> account generate john_hunter
dcecp>
```

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.
dcecp>
```

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***}}**

-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 pkss \
> -generatekey 485 -newpassphrase pokey
dcecp>
```

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-----}
{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**.

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.

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 may 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 may 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 hierarchical (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.

Data Structures

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.)

ndr.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 object's real or effective owner.

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

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.

foreign_other_delegate
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.

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

Key

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:

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.

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 *DCE 1.2.2 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 *DCE 1.2.2 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}
rwdtcia
dcecp>

dcecp> acl check /.:/hosts
rwdtcia
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 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*]
[**-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.

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

acl 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 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 **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:

acl replace *acl_name_list* [**-ic** | **-io** | **-entry**] [**-type** *manager_type_name*]
-acl *acl_entry_list* [**-cell** *new_default_cellname*] [**-local**]

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.

-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.

-acl *acl_entry_list*
Specifies ACL entries and their new values.

-cell *new_default_cellname*
Specifies a new default cell for all of the ACLs named in *acl_entry_list*. The **-cell** option is always applied before the other options.

-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 **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 **-acl** option is a list of ACL entries. The **-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 **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 replace /.:/hosts -acl {group dce rwdtcia}
dcecp>
```

acl show

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

acl show *acl_name_list* [**-ic** | **-io | -entry**] [**-type** *manager_type_name*] [**-cell** | **-managers**] [**-local**]

acl show [**-ic** | **-io**] [**-type**] [**-cell** | **-managers**] [**-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 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 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 **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** may 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 may 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}
{group subsys/dce/cds-admin rwdtcia}
{group subsys/dce/cds-server rwdtcia}
{any_other r--t---}
dcecp>
```

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 *DCE 1.2.2 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*

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
dcecp>
```

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

```
dcecp> attrlist getvalues [dir show /.:/hosts] -type CDS_UTS
1994-07-01-10:29:59.265-05:00I0.000/00-00-c0-f7-de-56
dcecp>
```

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

```
dcecp> at g [dir show /.:/hosts] -t CDS_UTS
1994-07-01-10:29:59.265-05:00I0.000/00-00-c0-f7-de-56
dcecp>
```

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
dcecp>
```

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*] {**-change** *attribute_list* | **-attribute** *value*}

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 to operate on. 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 to daemon acts when the file system storage for its audit trails is full, and 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 as follows:

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 *DCE 1.2.2 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 *DCE 1.2.2 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 enable

Enables an audit daemon. The syntax is as follows:

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

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

aud help

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

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

```
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 [*remote_audit_daemon_name*] {**-change** *attribute_list* | **-***attribute value*}

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. 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 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

```
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 [*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 performance of the command.

Privileges Required

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

Examples

```
dcecp> aud show
{stostrategy 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 show *event_class_list*

audevents catalog

Arguments

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 *DCE 1.2.2 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

```
dcecp> audevents catalog
dce_audit_admin_modify
dce_audit_admin_query
dce_audit_filter_modify
dce_audit_filter_query
dce_dts_mgt_modify
dce_dts_mgt_query
dce_dts_synch
dce_dts_time_provider
dce_sec_authent
dce_sec_control
dce_sec_modify
dce_sec_query
dce_sec_server
dcecp>
```

audevents help

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

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

```
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 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 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.)

Privileges Required

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

Examples

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

audfilter

Purpose

A dcecp 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:

Type	Key
principal	The key is a *principal_name*.
foreign_principal	The key is a */.../cellname/principal_name*.
group	The key is a *group_name*.
foreign_group	The key is a */.../cellname/group_name*.
cell	The key is a *cellname*.

cell_overridable
The key is a *cellname*.

world This type has no key.

world_overridable
This type has no key.

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.

Data Structures

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}**.

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 *DCE 1.2.2 Problem Determination Guide* for complete descriptions of all error messages.

Operations

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>
```

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
dcecp>
```

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)**.

audtrail

Purpose

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

Synopsis

audtrail help [*operation* | **-verbose**]

audtrail operations

audtrail show *audit_trail_file_name_list* [**-to** *filename*]

Arguments

audit_trail_file_name_list
: A list of one or more names of audit trail files. The names are not the full pathnames, but only the residual file name.

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 *DCE 1.2.2 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:

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

audtrail show *audit_trail_file_name_list* [**-to** *filename*]

Options

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

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
--- Start of an event record --- Event Number: 275
Event Name: LOGIN_getinfo
Event Outcome: success
Server: /.../stp.gburg.ibm.com/hosts/dceos2
Client: /.../stp.gburg.ibm.com/hosts/drinkernisti/self
```

```
Number of groups: 0
Authorization Status: Authorized with a pac
Date and Time recorded: 1994-12-19-19:02:27.037-05:00I-----
1 Event(s) specific:
   - item number 1             hosts/drinkernisti/self
--- End of an event record ---

--- Start of an event record --- Event Number: 275
Event Name: LOGIN_getinfo
Event Outcome: success
Server: /.../stp.gburg.ibm.com/hosts/dceos2
Client: Unknown client and cell uuids
Number of groups: 0
Authorization Status: Authorized with a pac
Date and Time recorded: 1994-12-19-19:02:28.819-05:00I-----
1 Event(s) specific:
   - item number 1             dce-rgy
--- End of an event record ---

dcecp>
```

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)**.

cds

Purpose

A dcecp object that represents a Cell Directory Service server

Synopsis

cds disable *server_name*

cds help [*operation* | **-verbose**]

cds operations

cds show *server_name*

Arguments

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

server_name The name of one CDS server running somewhere in the local cell. Specify the server name in one of the following formats:

/.../*cell_name*/**hosts**/*host_name*/**cds-server**

/.:/hosts/*host_name*/**cds-server**

Description

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_Replicas
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 since it started.

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 *DCE 1.2.2 Problem Determination Guide* for complete descriptions of all error messages.

Operations

cds disable

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

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 **dwc** (**delete**, **write**, **create**) permissions on the namespace entry of the server.

Example

```
dcecp> cds disable /.:/hosts/blech/cds-server
dcecp>
```

cds help

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

cds help [*operation* | **-verbose**]

Options

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

Used without an argument or option, the **cds help** command returns brief information about each **cds** 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 **cds** object itself.

Privileges Required

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

Examples

```
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:

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

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

cds show

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

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

```
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}
{Skulks_Initiated 278}
{Skulks_Completed 278}
{Times_Lookup_Paths_Broken 0}
{Crucial_Replicas 0}
{Child_Update_Failures 0}
{Security_Failures 0}
{Known_Clearinghouses /.../gumby1/blech_ch}
dcecp>
```

Related Information

Commands: **cdsd(8dce), dcecp(8dce), cdsclient(8dce).**

cdsalias

Purpose

A dcecp object that lets you manipulate cell names in CDS

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 may 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 *DCE 1.2.2 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

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

Examples

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

cdsalias delete

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

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

```
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:

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

```
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.
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.
dcecp>
```

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

Commands: **dcecp(8dce)**, **account(8dce)**, **cellalias(8dce)**, **directory(8dce)**, **hostdata(8dce)**.

cdscache

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

operation 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 (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 *DCE 1.2.2 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 may 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 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 client system, **/.:/hosts/***hostname***/cds-clerk**.

Examples

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

```
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 client system. No CDS permissions are required.

Examples

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

```
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 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

```
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:

cdscache show *server_name* {**-server** | **-clearinghouse**}

Options

-clearinghouse

This option 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 client on the cached clearinghouse

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

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

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

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

Name The directory cell name

Towers The protocol sequence and network address of the server node

The **show** operation displays information about clearinghouses or servers stored in the cache. The required *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 **-clearinghouse** or **-server** options to select the information you want to display.

Privileges Required

You must have **r** (**read**) permission to the CDS client.

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**/*host_name*/**cds-clerk**

/.:/hosts/*host_name*/**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 *DCE 1.2.2 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.

Privileges Required

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

Examples

```
dcecp> cdsclient 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.
```

```
dcecp>
```

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

```
decp> cdsclient 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> cdsclient show /.:/hosts/blech/cds-server
{Creation_Time 1995-10-11-15:09:45.187-04:00I-----}
{Protocol_Errors 0}
```

```
{Authentication_Failures 0}
{Read_Operations 78935}
{Cache_Hits 55007}
{Cache_Bypasses 23726}
{Write_Operations 50}
{Miscellaneous_Operations 53}
dcecp>
```

Related Information

Commands: **cdsadv(8cds)**, **cds(8dce)**, **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 *DCE 1.2.2 Administration Guide* for more information about attributes.

Errors

A representative list of errors that might be returned is not shown here. Refer to the *DCE 1.2.2 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. Put the DCE daemon into partial service mode by sending the **dced** process the correct signal:

```
# kill -SIGUSR1 pid_of_dced
#
```

2. Invoke **dcecp** with the **-local** option:

```
# dcecp -local
dcecp>
```

3. 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 -riI} -local
dcecp> acl modify srvrconf -add {group subsys/dce/dced-admin -riI} -local
dcecp> acl modify srvrconf -add {group subsys/dce/dced-admin -d-rwx} -io -local
dcecp>
```

4. Put the DCE daemon back into full service mode with the following command:

```
# kill -SIGUSR1 pid_of_dced
#
```

5. 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 {principal r m r D}
dcecp>
```

6. 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}
dcecp>
```

7. 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}
dcecp>
```

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

```
dcecp> cell backup
dcecp>
```

cell catalog

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

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

```
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:

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 **-verbose** option for more detailed information about the **cell** task object itself.

Privileges Required

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

Examples

```
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>
```

cell operations

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

cell 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 **cell operations** command.

Examples

```
dcecp> cell operations
backup catalog ping show help operations
dcecp>
```

cell ping

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

cell ping [*cell_name*] [**-clients**] [**-replicas**]

Options

-clients This option causes the command to ping every machine in the cell. It does this by looping though **/.:/hosts** and doing a **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 **DCE clients available**.

-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 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

-simplename
Returns the cell information without prepending the cell name.

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 *DCE 1.2.2 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 **/.:/susbys/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>
```

Related Information

Commands: **dcecp(8dec0**, **directory(8dce)**, **host(8dce)**, **server(8dce)**.

cellalias

Purpose

A dcecp task object that manages cell name aliases

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 **dced** 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 *DCE 1.2.2 Problem Determination Guide* for complete descriptions of all error messages.

Operations

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)**.

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 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 may 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 may 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.

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 *DCE 1.2.2 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 *DCE 1.2.2 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 cellname.

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/***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/***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:

```
dcecp> clearinghouse disable /.:/Paris2_CH
dcecp>
```

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. The syntax is as follows:

clearinghouse initiate *clearinghouse_name_list* **-checkpoint**

Options

-checkpoint Forces the clearinghouse to checkpont 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.

Privileges Required

You need **w** (**write**) permission on the clearinghouse server and **A** (**admin**) permission on the cell root directory. The server principal (/**hosts/***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
dcecp>
```

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
dcecp>
```

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/***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 1994-06-18-20:16:22.150-05:00I0.000/00-00-c0-f7-de-56}
{CDS_UTS 1994-06-19-17:17:43.911-05:00I0.000/00-00-c0-f7-de-56}
{CDS_ObjectUUID 0066ccea-d978-1db3-8259-0000c0f7de56}
{CDS_AllUpTo 1994-07-01-21:30:18.948-05:00I0.000/00-00-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}}
{CDS_CHState on}
{CDS_CHDirectories
 {{Dir_UUID 00146037-d97b-1db3-8259-0000c0f7de56}
  {Dir_Name /.../terrapin}}
 {{Dir_UUID 0043797a-d991-1db3-8259-0000c0f7de56}
  {Dir_Name /.../terrapin/subsys}}
 {{Dir_UUID 004faa42-d992-1db3-8259-0000c0f7de56}
  {Dir_Name /.../terrapin/subsys/HP}}
 {{Dir_UUID 004fa65a-d993-1db3-8259-0000c0f7de56}
  {Dir_Name /.../terrapin/subsys/HP/sample-apps}}
 {{Dir_UUID 004b1130-d994-1db3-8259-0000c0f7de56}
  {Dir_Name /.../terrapin/subsys/dce}}
 {{Dir_UUID 00498a0e-d995-1db3-8259-0000c0f7de56}
  {Dir_Name /.../terrapin/subsys/dce/sec}}
 {{Dir_UUID 003ed80c-d996-1db3-8259-0000c0f7de56}
  {Dir_Name /.../terrapin/subsys/dce/dfs}}
 {{Dir_UUID 003d4d8e-d997-1db3-8259-0000c0f7de56}
  {Dir_Name /.../terrapin/hosts}}
 {{Dir_UUID 003bc522-d998-1db3-8259-0000c0f7de56}
  {Dir_Name /.../terrapin/hosts/drkstr}}
 {{Dir_UUID 0089ee8c-44e0-1dbe-929b-0000c0f7de56}
  {Dir_Name /.../terrapin/help}}
 {{Dir_UUID 001c6cea-00fb-1dc5-929b-0000c0f7de56}
  {Dir_Name /.../terrapin/test_1}}
```

```
 {{Dir_UUID 00440fe8-02a1-1dc5-929b-0000c0f7de56}
  {Dir_Name /.../terrapin/dirmod}}}
{CDS_ReplicaVersion 3.0}
{CDS_NSCellname /.../terrapin}
dcecp>
```

```
dcecp> clearinghouse show /.:/Chicago1_CH -counters
{corruptions 0}
{disables 0}
{enables 1}
{failedupgrades 0}
{missingentries 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. 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. 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/***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

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 **-initd** options, **dts_entity** can be either of the following:

- The name of a **dtsd** server, which can be on a remote host, in the format:

  /.../*cellname*/**hosts**/*hostname*/**dts-entity**

- 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.

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 *DCE 1.2.2 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 may 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

```
dcecp> 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.

Examples

```
dcecp> clock operations
compare set show synchronize help operations
dcecp>
```

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 system clock directly. This operation returns an empty string on success.

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. See the g for more information about DTS.

Privileges Required

You must have **w** (**write**) permission on the clock object (**/.:/hosts/***hostname***/dts-entity**) if using DTS to set the time, otherwise no special privileges are required.

Examples

```
dcecp> clock set -to 1994-07-15-16:27:28.000-04:00 -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:

- The **-dced** option specifies that **dced** services should be used instead of DTS services
- 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
1994-07-15-16:28:02.229+00:00I-----
dcecp>

dcecp> clock show oddball -dced
1994-07-16-17:29:05.321+00:00I-----
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.

-dced Use **dced** services instead of DTS as the time source.

-inetd Use **inetd** socket connections instead of DTS as the time source.

-dtsd 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)**.

csrc

Purpose Builds a DCE character and code set registry on a host

Synopsis **csrc** [**-i** *source_filename*] [**-m** *intermediate_cs_list*] [**-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 **/usr/lib/nls/csr/code_set_registry.txt**.

-m *intermediate_cs_list*
: Adds code set names to the code set registry file's intermediate code set priority list.

-o *destination_filename*
: Places the generated code set registry file in the location you specify rather than in the default location **/usr/lib/nls/csr/code_set_registry.db**.

Description

The code set registry compiler **csrc** 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 **csrc** in the following two instances:

- 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.
- By cell administrators, when they are configuring machines that are part of an internationalized DCE cell. In this instance, the cell administrator adds the local code set names of any additional code sets that the site supports to the licensee-generated code set registry source file for each different operating system platform that exists in the cell. The cell administrator can also add to each platform-specific source file any site-specific, nonOSF registered code set names and values.

Each code set record specifies one code set, and has the following form:

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. Licensee or cell administrator-defined values must be in the range 0xf5000000 through 0xffffffff.

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.

The OSF DCE source tape provides a partial version of a code set registry source file in the file **/usr/lib/nls/csr/code_set_registry.txt**. This source file contains records for all OSF-registered code sets, and assigns the text string **NONE** to **loc_name** fields intended for modification to a local code set name.

DCE licensees who port DCE to their operating system platform and who plan to support internationalized DCE RPC applications must replace the **NONE** text string with their local name for the code set, for each code set that their operating system platform supports. If their platform does not support a given code set, they must leave the **NONE** keyword in the code set record.

Cell administrators of internationalized DCE cells carry out the same procedure on the licensee-supplied, platform-specific source files that exist at their site. For each platform-specific source file, they replace the **NONE** keyword with the local code set names for any site-specific supported code sets.

DCE licensees and cell administrators can also add vendor-specific or site-specific code set values that have not been registered with OSF. These vendor or user-defined values must be in the range 0xf5000000 through 0xffffffff.

The following is an excerpt from the OSF-supplied code set registry source file:

```
start
description ISO 8859:1987; Latin Alphabet No. 1
loc_name NONE
rgy_value 0x00010001
char_values 0x0011
max_bytes 1
end

start
description ISO 8859-2:1987; Latin Alphabet No. 2
loc_name NONE
code_value 0x00010002
char_values 0x0012
max_bytes 1
end

start
description ISO 8859-3:1988; Latin Alphabet No. 3
loc_name NONE
code_value 0x00010003
char_values 0x0013
max_bytes 1
end

start
description ISO 8859-6:1987; Latin-Arabic Alphabet
loc_name NONE
code_value 0x00010006
char_values 0x0016
max_bytes 1
end

[...]
```

```
start
description ISO/IEC 10646-1:1993; UCS-2 Level 1
loc_name NONE
code_value 0x00010100
char_values 0x1000
max_bytes 2
end

[...]

start
description JIS eucJP:1993; Japanese EUC
loc_name NONE
code_value 0x00030010
char_values 0x0011:0x0080:0x0081:0x0082
max_bytes 3
end
```

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 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.

When invoked with no options, **csrc** uses the default source file **/usr/lib/nls/csr/code_set_registry.txt** and creates the default output file **/usr/lib/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.

Adding Intermediate Code Sets

Use the **-m** option to add a maximum of five intermediate code set names to the code set registry file's intermediate code set priority list. The order in which you specify intermediate code sets determines their order of precedence in the list; that is, the first intermediate code set you specify with **-m** becomes the first intermediate code set in the priority list, and thus will be the first code set used should an intermediate code set be required for client-server communication. If you do not specify intermediate code sets with **-m**, the universal code set ISO 10646 will be used as the default intermediate code set.

Restrictions

You need **w** (**write**) permission to the **/usr/lib/nls/csr** directory, which usually requires **root** privilege.

Examples

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 -m euc -m sjis
```

Errors

A representative list of errors that might be returned is not shown here. Refer to the *DCE 1.2.2 Problem Determination Guide* for complete descriptions of all error messages.

Files

/usr/lib/nls/csr/code_set_registry.txt
Default pathname for code set registry source file.

/usr/lib/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)**.

Books: *DCE 1.2.2 Administration Guide*, *DCE 1.2.2 Application Development Guide—Core Components*.

dce_config

Purpose

Installs, configures, and starts up DCE

Synopsis

dce_config [**-i**][**-e** *environment_file*] [**-c** *command_file*]

Options

-i
The **-i** option tells **dce_config** to look in the **/etc** directory of the install area (which is generally **/opt/dce1.0/etc**) for the component scripts it needs to run. After you have invoked **dce_config** once with the **-i** option, you do not need to use the option again.

-e *environment_file*
The **-e** option causes **dce_config** to source *environment_file* at startup. The *environment_file* variable represents a user-created file that sets the DCE and Distributed File Service (DFS) variables that specfy responses to the **dce_config** user prompts. Note that if you do not specify the **-e** option, **dce_config** looks for the **/etc/dce_config.conf** file and sources it if it exists. If the file does not exist, it uses shell variable settings if they are set.

-c *command_file*
The **-c** option causes **dce_config** to source *command_file* at startup. The *command_file* variable represents a user-created shell script that initiates installation and configuration processing.

Description

The **dce_config** shell command invokes a menu-driven interface that installs, configures, and starts up DCE. The **dce_config** command displays a hierarchy of menus and invokes individual installation and configuration routines, according to users' menu selections.

Installation routines store the binaries required for the server installation that is selected into **$DCELOCAL**. Binaries required for a client installation are stored on every machine. The configuration menu consists of initial cell configuration, additional server configuration, and DCE client configuration. The security server and the first Cell Directory Service (CDS) server constitute initial cell configuration.

If you specify an environment file with the **-e** option and a command file with the **-c** option, you can completely automate **dce_config** processing.

The Command File

The command file consists of **install** and **config** command lines that specify the component to install and, for DFS, the type of server.

A sample command file, **config.cmd**, is provided by OSF with the DCE source. You can copy the file and use it as supplied or you can use it as guide to creating your own environment file. The sample file is not copied to the install tree during DCE installation.

The **install** lines are in the following form:

install *component* **[***dfs_server***]**

where:

component Can be one of the following values:

sec	Master security server binaries
cds	Initial CDS server binaries
gds	GDS server binaries
dts	Distributed Time Service (DTS) server binaries
client	DCE client binaries
sec-replica	Security replica binaries
appdev	Interface Definition Language (IDL) compiler and header files for use in DCE application development
cdsbrowser	CDS browser utility
nidl_to_idl	A utility to convert files written in NIDL to files written in IDL

dfs_server Specifies the type of DFS server to install; can be one of the following values:

client	DFS client
scm	System control machine
privatefs	Private file server
fs	File server
fldb	File location database server

The **config** lines are in the following form:

config *component*
{
client |
gda |
sec {client | server | replica} |
cds {client | server | replica} |
dts {clerk | local | global | ntp-provider | null-provider}
dfs {client | scm | privatefs | fs | fldb}
}

where:

component Can be one of the following values:

client	DCE client configuration	
gda	GDA configuration	
sec	Security configuration of any one of the following:	
	client	Security client machine
	server	Security master server machine
	replica	Security replica machine
cds	CDS configuration of any one of the following:	
	client	CDS client machine
	server	CDS initial server machine

	replica	Additional CDS server machines
dts	DTS configuration of any one of the following:	
	clerk	DTS clerk machine
	local	DTS local server machine
	global	DTS global server machine
	ntp-provider	DTS NTP time-provider machine
	null-provider	DTS null time-provider
dfs	DFS configuration of any one of the following:	
	client	DFS client specify
	scm	System control machine
	privatefs	Private file server machine
	fs	File server machine
	fldb	File location database server machine

The Environment File

The environment file sets the DCE and DFS variables. The file entries are in the following form:

variable=value

To change a value, simply replace it with the new value.

A sample environment file, **config.env**, is provided by OSF with the DCE source. You can copy the file and use it as supplied or you can use it as guide to creating your own environment file. The sample file is not copied to the install tree during DCE installation.

The DCE and DFS Variables

The first of the tables that follows lists the DCE variables you can set for **dce_config** processing. The second of the tables lists the DFS variables you can set. In the tables, the term default refers to the setting assigned to the variable by OSF.

Variable	Value
CACHE_CDS_SERVER	The name of the CDS server to cache. It is not required that the cached server be the initial CDS server. Used during CDS client configuration.
CACHE_CDS_SERVER_IP	The IP address of the CDS server to cache.
CELL_ADMIN	The principal name of the initial privileged user of the registry database (known as the *registry creator*). Used during security server configuration.
CELL_ADMIN_PW	The default password assigned to the accounts created when the registry database is created, including the account for the registry creator. The default is **-dce-**.
CELL_NAME	The name of the cell (without the **.../**) on which the configuration is being performed. Used during security server configuration.
CHANGE_PW	Indicates whether or not **dce_config** displays the message **Password must be changed** on exiting when the cell administrator password (**CELL_ADMIN_PW**) is the same as the default password. The default is **n**. It is recommended that you do not change this value in order to help ensure that the cell administrator is not assigned a commonly known password. This variable is used in conjunction with the **DEFAULT_PW** variable.
CHECK_TIME	Specifies whether or not to check client and server clock synchronization; **y** indicates the time will be checked while **n** indicates it will not. The default is **y**.

Variable	Value
DC_DISPLAY_THRESHOLD	Specifies the messages to write to stdout. Possible values are **ERROR**, **WARNING**, **SUMMARY**, **DETAIL**, **VERBOSE**, and **DEBUG**. The default is **SUMMARY**.
DC_LOG_THRESHOLD	Specifies the minimum priority log messages to write to the log file, **/tmp/dce_config.log**. Possible values are **ERROR**, **WARNING**, **SUMMARY**, **DETAIL**, **VERBOSE**, and **DEBUG**. The default is **DEBUG**.
DEFAULT_MAX_ID	The highest value UNIX ID for principals. The default value is 32767, which means that only principals with UNIX IDs lower than 32767 can access the cell. It is recommended that you accept the default. Used during security server configuration.
DEFAULT_PW	Contains the default password used when the registry is created. This variable is used to determine if the cell administrator's password (CELL_ADMIN_PW) is the same as the default password. When the user exits **dce_config**, the value of **DEFAULT_PW** and **CELL_ADMIN_PW** are checked. If they are the same and if the **CHANGE_PW** variable is set **Y**, **dce_config** issues the warning message **Password must be changed**. The default for this variable is **-dce-**. If your site has a commonly used and known password, change the **DEFAULT_PW** variable to that password to help ensure that the cell administrator account is not assigned a commonly known password.

Variable	Value
DIR_REPLICATE	Controls the replication of CDS directories when an additional CDS server is being created at DCE configuration time. The value **y** will cause **dce_config** to prompt for more directories to replicate; **n** will not. The default is **n**.
DOMAIN_NAME	The name of the host domain. Used as a default in the Security client configuration for Kerberos5 compatibility if **/etc/resolv.conf** does not contain a domain name. This variable is appended to the hostname to get the fully qualified name in the format: *hostname.domain_name*. For example, if **DOMAIN_NAME** is set to **company.com** and the host name is **abc**, the fully qualified hostname will be **abc.company.com.**

Variable	Value
DO_CHECKS	Controls the display of three prompts. The first is whether or not the prompt **Press <Return> to continue, <Ctrl-C> to exit:** is returned when **dce_config** encounters a nonfatal error. This prompt forces the user to acknowledge the error and offers a way to exit **dce_config**. The second and third prompt occur during master security server configuration. They prompt for a UNIX ID number at which the security server will start assigning automatically generated group UNIX IDs and principal UNIX IDs. If this prompt is turned off, the default is the default described in the **DEFAULT_MAX_ID** and **GID_GAP** variables. For the **DO_CHECKS** variable, **y** displays the prompt; **n** does not. The default is **y**.
EXIT_ON_ERROR	An indication of whether or not **dce_config** will exit in the event of a fatal error: **y** indicates that **dce_config** exits when it encounters a fatal error; **n** indicates it will not. The default is **n**. Setting this variable to **y** or **n** can help prevent a ‘‘here’’ file from getting out of sync with **dce_config**.
GID_GAP	The increment above highest currently used GID at which the security service will start assigning automatically generated GIDs. The value of this variable is used with the **LOW_GID** variable to set the starting point for unique identifiers (UIDs) automatically assigned by the security server. Default is 100. Used in security server configuration.
HOST_NAME_IP	The IP address of node on which **dce_config** is running.

Variable	Value
KEYSEED	A character string used to seed the random key generator in order to create the master key for the master and each slave database. Each database has its own master key and thus keyseed. Used in security server configuration.
LAN_NAME	For multiple local area network (LAN) configurations, the internal name of the LAN (in the LAN profile). Used in CDS server configuration.
LOGFILE	The full pathname of the log file produced by **dce_config**. The default is **/tmp/dce_config.log**
LOW_GID	The value at which the security server will start assigning automatically generated group IDs. The default is the value of the highest group ID currently used on the machine being configured, incremented by the value of **GID_GAP**. Although there is no restriction that the value of **LOW_GID** must be higher than the machine's highest group ID, if you supply a **LOW_GID** that is less than or equal to the highest currently used group ID, **dce_config** issues a warning message and prompts the user to reenter **LOW_GID**. Used in master security server configuration.

Variable	Value
LOW_UID	The value at which the security server starts assigning automatically generated UNIX IDs. Default is value of highest UNIX ID currently used on machine being configured, incremented by value of **UID_GAP**. There is no requirement that the value of **LOW_UID** be higher than the machine's highest UNIX ID, but if you supply a **LOW_UID** less than or equal to the highest currently used UNIX ID, **dce_config** issues a warning message and prompts you to reenter **LOW_UID**. Used in master security server configuration.
MULTIPLE_LAN	An indication of whether or not to configure the node with multiple LAN capabilities: **y** indicates configure with multiple LAN capabilities, **n** indicates do not. Used in CDS configuration.
NTP_HOST	The name of the host on which the NTP time-provider server is running. Used in DTS time provider configuration.
PWD_MGMT_SVR	The default pathname to the password management server, which is **$DCELOCAL/bin/pwd_strength**. Used in password management server configuration.
PWD_MGMT_SVR_OPTIONS	The default option or options for the password management server (**pwd_strength**). The value of the variable is set to **-v** (verbose) at server configuration.

Variable	Value
REMOVE_PREV_INSTALL	An indication of whether or not to remove all remnants of previous DCE installations before performing the new install; **y** indicates remove all remnants while **n** indicates do not. Be aware that if you set this variable to O, **dce_config** will automatically remove all installed components each time you install any component, and you must reinstall them all. Used in all component installations.
REMOVE_PREV_CONFIG	An indication of whether or not to remove all remnants of previous configurations before performing the new configuration: **y** indicates remove all remnants; **n** indicates do not. Be aware that if you set this variable to **y**, **dce_config** will stop and remove all configured components each time you configure any component, and you must reconfigure them all. Used in all component configurations.
REP_CLEARINGHOUSE	The name for new clearinghouse. Used in additional CDS server configuration.
SEC_SERVER	Name of the machine on which cell's master security server runs. Used in security client configuration.
SEC_SERVER_IP	IP address for server named in SEC_SERVER.

Variable	Value
SYNC_CLOCKS	Indication of whether or not to synchronize all client clocks with the security server clock; **y** indicates that client and server clocks will be synchronized while **n** indicates they will not. If set to **n**, and clocks are out of sync by more than value specified in the **TOLERANCE_SEC** variable, user is prompted whether or not to synchronize them. Valid only if **CHECK_TIME** variable is set to **y**. For DFS machine configurations, should be set to **y**.
TIME_SERVER	Specifies the host that security client will try to synchronize its clock against. Host must have a DTS server (**dtsd**) running. Recommended choice for host is the one running the master security server (name specified in the **SEC_SERVER** variable).
TOLERANCE_SEC	Number of seconds a client system clock can differ from the security server system clock before either the user prompted to synchronize clocks or clocks are synchronized automatically. Default is 120 seconds. Both security service and CDS require there be no more than 5-minute difference between clocks on any two nodes in a cell. For a DFS file location database server, should not be set to less than 90 seconds.
UID_GAP	The increment above highest currently used UID at which security service starts assigning automatically generated UIDs. The value of this variable is used with the **LOW_UID** variable to set the starting point for UIDs automatically assigned by the security server. Default is 100. Used in security server configuration.

Variable	Value
UNCONFIG_HOST_PRESET	Name of node to be unconfigured. Used with **unconfigure** option.
AGG_FS_TYPE	The type of file system for the aggregate to be exported. Possible values are **native** meaning the native file system (such as UFS, JFS) or **episode** meaning the episode (LFS) file system.
AGG_DEV_NAME	The device name of the aggregate to be exported,
AGG_MOUNT_PATH	The mount path for the aggregate (such as **/usr/users**).
AGG_NAME	The name to be used for the aggregate to be exported (such as **user.jlw**).
AGG_ID	The unique numerical aggregate ID for the exported aggregate.
CACHE_SIZE_RAM	The number of bytes to use for an in-memory cache.
CACHE_SIZE_DISK	The number of bytes to use for a local disk cache.
CACHE_DIR_DISK	The pathname of the directory to use for a local disk cache.
CLIENT_CACHE_LOC	An indication of whether the cache is stored in memory or on disk. machine values are **mem** meaning the cache is stored in memory or **disk** meaning the cache is stored on the local disk.
CONFIG_NFS_GATEWAY	An indication of whether or not to configure the DFS client as an NFS gateway. Possible values are **y** and **n**; **n** is the default.

Variable	Value
DFS_SERVER_INSTALL	An indication of which type of DFS server to install: **SCM** for system control machine; **FS** for file server; **PRIVATEFS** for private file server; **FLDB** for file location database server.
EPI_FORMAT_PART	An indication of whether or not to format a disk partition as an episode aggregate. Possible values are **y** to format the partition or **n** to not.
EPI_FORCE_INIT	An indication of whether or not to force the initialization of a partition as an Episode aggregate, possibly losing data. Possible values are **y** or the initialization or **n** to not.
INIT_LFS	An indication of whether or not to initialize the LFS (using **epiinit**). Possible values are **y** to initialize or **n** to not.
INSTALL_OPT_SERS	An indication of whether or not to install the optional DFS servers (**bak**, **butc**, **upclient**). Possible values are **y** to install or **n** to not.
INSTALL_OPT_CLIENT	An indication of whether or not to install the optional DFS client (**cm**, **bos**, and **fts**) binaries. Possible values are **y** to install or **n** to not.
LOAD_LFS_KEXT	An indication of whether or not to load the LFS kernel extensions. Possible values are **y** to load or **n** to not.
ROOT_FILESET_NM	The name of the DFS root fileset.
SCM_NAME	The name of the system control machine to be used during configuration.

Component Scripts

The **dce_config** script calls component scripts that reside in the **/opt/dcelocal/etc** directory (or in the **etc** directory of the install area) with symbolic links to **/etc**. In a custom configuration script, you can call the component scripts directly and supply the required input via the environment variables. The names and functions of the component scripts follows:

dce_shutdown

Shuts down all DCE server processes (**auditd**, **dtsd**, **cdsadv**, **cdsd**, **secd**, **gdsd**, **gdad**), except for DFS server processes (**dfsd**). The script is executed via **dcecp** or another control program. It must be run on the machine running the daemon processes to be shut down. You must be **root** or another priviledged user to run the script. You should always run the script before reconfiguring DCE.

The **dce_shutdown** script attempts to shut down a daemon gently. If it fails to do so, it will send a kill signal to all the DCE daemons.

The **dce_shutdown** script can also be run directly if for any reason you do not want to use a control program. When the script is run with the **-f** option, it will find and kill the DCE daemons. This behavior is the same as that of the **dce.clean** script, which was included in DCE R1.0.3 and previous releases. DCE R1.1 does not include the **dce.clean** script, but provides the name as a symbolic link to the **dce_shutdown** script for the user's convenience.

dfs.clean Kills DFS server processes. This script must be run on the machine running the processes. It should be run before reconfiguring DCE. (Note that some DFS daemon processes cannot be killed by **dfs.clean**.)

dce.rm [**install**]

Removes all data and configuration files created by DCE servers after initial configuration except for data and files created by DFS servers. This script must be run on the machine running the processes. It should be run before reconfiguring DCE. If you invoke the script with the **install** parameter, the script removes the binary files added during installation.

dfs.rm [**install**]

Removes data and configuration files created by DFS servers after initial configuration. This script must be run on the machine running the processes, and **dced** must be running on that machine. The **dfs.rm** script should be run before reconfiguring DCE. If you invoke the script with

the **install** parameter, the script removes the binary files added during installation. Note that this script invokes the **dce.clean** script.

dce.unconfig *hostname*
Removes all DCE clients on *hostname* from the security and directory service databases. It should be run before reconfiguring a client machine.

dfs.unconfig *hostname*
Removes the DFS client on *hostname* from the security and directory service databases. It should be run before reconfiguring a client machine.

dce_com_env
Sets environment variables.

dce_config_env
Calls the **dce_com_env** script that sets the environment variables.

dce_com_utils
Contains common functions used by **dce_config** and **dfs_config**.

dce_config_utils
Contains internal routines used by **dce_config** scripts.

dfs_config Configures a machine as a DFS server or client.

rc.dce Starts DCE daemons. This script cannot be run remotely; it must be run on the machine on which the daemons are being started.

rc.dfs Starts DCE daemons. This script cannot be run remotely; it must be run on the machine on which the daemons are being started.

Privileges Required

You must have **root** authority to run the **dce_config** command.

Exit Values

In case of an error, this command repeats requests for correct input. The user can exit the program from any menu.

Errors

A representative list of errors that might be returned is not shown here. Refer to the *DCE 1.2.2 Problem Determination Guide* for complete descriptions of all error messages.

Related Information

Books: *DCE 1.2.2 Administration Guide*

dcecp

Purpose

Administrative interface for DCE management tasks

Synopsis

dcecp [**-s**][**-local**][*script_name* | **-c** *command*]

Options

-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

DCE 1.2.2 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:

- Interactive mode
- 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.

```
% dcecp
dcecp> directory list /.: -directories
/.:/hosts /.:/subsys
dcecp>
```

Command-Line Mode

Activate command-line mode from the system prompt by using one of the following methods:

- Enter the **dcecp** command with a filename of a script containing **dcecp** commands, other valid Tcl commands, or both, as follows:

  ```
  % dcecp myown.Tcl
  ```

- 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.

$HOME/.dcecprc
Contains user customizations.

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.

cdsclient Manages the CDS client daemon on any DCE host.

cell Performs cellwide tasks.

cellalias Performs cell aliasing and connection tasks.

clearinghouse
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.

endpoint 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 may 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 may 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.

errtext 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** comand 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:

- If the command is found in a **dcecp** script file, **dcecp** executes the command.
- 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 don not lose any context you have established.
- 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.
- 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 may 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 object* 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 **.dcecprc** 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:

```
# 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**:

acl 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 **-l** (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.

_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 ***L_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 exmaple 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 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 *DCE 1.2.2 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:

```
dcecp> principal operations
catalog create delete modify rename show help operations
dcecp>
```

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 **dcecp** list that can be used by other **dcecp** commands.

To see other information about an object, use an object's **help** operation. All **dcecp** objects have a **help** operation that offers three kinds of information.

- View brief information about an object's operations by using **help** without arguments or options. Operations are listed in alphabetical order with the **operations** and **help** operations listed last because all objects support these operations. An example is as follows:

```
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>
```

- View brief information about the options an operation supports by using **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:

```
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.
dcecp>
```

- View a short description of a **dcecp** object by using the **help** operation with the **-verbose** option. This operation returns text explaining what the object represents and how to use it. An example follows:

```
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.
dcecp>
```

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 **man** 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> man principal
```

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:

Control Sequence	Action Performed
<Ctrl-A>	Move to the beginning of the line
<Ctrl-B>	Move left (backward) [*n*]
<Ctrl-D>	Delete the next character [*n*]
<Ctrl-E>	Move to the end of the line
<Ctrl-F>	Move right (forward) [*n*]
<Ctrl-G>	Ring the bell
<Ctrl-H>	Delete the character before the cursor [*n*]

<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-Y> Yank back the last killed test

<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

<ESC ?> Show the possible completions

<ESC <> Move to the start of history

<ESC >> Move to the end of history

<ESC b> Move backward one word [*n*]

<ESC d> Delete the word under the cursor [*n*]

<ESC f> Move forward one word [*n*]

<ESC l> Make the word lowercase [*n*]

<ESC u> Make the word uppercase [*n*]

<ESC y> Yank back the last killed text

<ESC w> Make area up to mark yankable

<ESC *nn***>** Set repeat count to the number *nn*

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
dcecp>
```

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:

```
dcecp> !dir
[execution of last event beginning with  dir]
dcecp>
```

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>
dcecp> ^vreate^create
[command output]
dcecp>
```

Examples

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+00:00I-----
%
```

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 rwdtcia}
{group subsys/dce/cds-server rwdtcia}
{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_Replicas
 {{CH_UUID 03ccab5c-03d6-11cf-ad88-08000985b5a6}
  {CH_Name /.../gumby1/blech_ch}
  {Replica_Type Master}
  {Tower {ncadg_ip_udp 15.22.50.213}}
  {Tower {ncacn_ip_tcp 15.22.50.213}}}}
{CDS_AllUpTo 1995-10-23-13:06:43.560848100/08-00-09-85-b5-a6}
{CDS_Convergence medium}
{CDS_ParentPointer
 {{Parent_UUID 044a2a14-03d6-11cf-ad88-08000985b5a6}
  {Timeout
    {expiration 1994-04-19-16:39:58.049}
    {extension +1-00:00:00.000I0.000}}
    {myname /.../brain_cell.osf.org/subsys}}
```

```
{CDS_DirectoryVersion 3.0}
{CDS_ReplicaState on}
{CDS_ReplicaType Master}
{CDS_LastSkulk 1995-10-23-13:06:43.560848100/08-00-09-85-b5-a6}
{CDS_LastUpdate 1995-10-23-03:06:43.209673100/08-00-09-85-b5-a6}
{CDS_Epoch 0c3512fc-03d6-11cf-ad88-08000985b5a6}
{CDS_ReplicaVersion 3.0}
%
```

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)**.

dced

Purpose

the DCE host daemon

Synopsis

dced [**-h** | **-i**][**-cfr**][**-w** *route*] [**-b** | **-p** | **-s**][**-e** | *prot_seq*…]

Options

-h Prints **dced** usage and exits.

-i Initializes **dced** 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.

-c Starts **dced** so it does not require DCE privacy encryption for remote key table management. The default is to use DCE privacy encryption.

-f Starts the **dced** process in the foreground. The default is for **dced** to run in the background.

-r Starts **dced** in remote-update mode. This mode allows DCE cell administration tasks to be performed by an administrator on a remote machine. By default, **dced** prevents any remote administration to help prevent attacks by malicious administrators.

-w *route* Establishes the serviceability routing for **dced**'s messages.

-b Starts **dced** in bootstrap mode with the endpoint mapper service and access control lists (ACLs). This mode means it may need to wait for other daemons such as **secd** and **cdsd** before it can perform its own initialization.

-p Purges the existing machine context and removes the bindings file before starting.

-s Starts **dced** without the security validation service.

-e Starts **dced** without the endpoint mapper service. No protocol sequences are valid for this option.

Arguments

prot_seq Starts **dced** by using the specified remote procedure call (RPC) protocol sequence string or strings. Possible values include **ncadg_ip_udp** (for a datagram protocol) and **ncacn_ip_tcp** (for a connection-based protocol). A complete list of the protocol sequences recognized can be found in **dce/ep.idl**.

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 may 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 *DCE 1.2.2 Problem Determination Guide* for complete descriptions of all error messages.

Files

The **dced** databases are as follows:

dcelocal/**var/dced/Ep.db**	*dcelocal*/**var/dced/cell_aliases**
dcelocal/**var/dced/Hostdata.db**	*dcelocal*/**var/dced/cell_name**
dcelocal/**var/dced/Srvrconf.db**	*dcelocal*/**var/dced/host_name**
dcelocal/**var/dced/Srvrexec.db**	*dcelocal*/**var/dced/post_processes**
dcelocal/**var/dced/Keytab.db**	*dcelocal*/**bin/dcecf_postproc**
dcelocal/**var/dced/Acl.db**	**/krb5/v5srvtab**
dcelocal/**var/dced/Xattrschema.db**	**/etc/rc.dce**
dcelocal/**dce_cf.db**	

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)**.

Books: *DCE 1.2.2 Application Development Guide*.

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 may 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 may 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 *DCE 1.2.2 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 *DCE 1.2.2 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 may 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.

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**.

-single 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

-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>

dcecp> dir list /.:/depts/radiology -simple
applications services staff users
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 *DCE 1.2.2 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:

```
    OID          LABEL           SYNTAX
1.3.22.1.3.91    myname          char
```

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:

```
    OID            LABEL          SYNTAX
1.3.22.1.3.66   dirregion       small
```

2. Enter the following command to remove the value **1** from the attribute **dirregion**:

```
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:

```
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

```
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 may 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+00:00I0.000/00-00-c0-8a-df-56}
```

```
{CDS_UTS 1994-07-08-19:36:31.719+00:00I0.000/00-00-c0-8a-df-56}
{CDS_ObjectUUID 2df03af4-9a76-11cd-8f2b-0000c08adf56}
{CDS_Replicas
 {{CH_UUID b32648c6-928d-11cd-b4b5-0000c08adf56}
  {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+00:00I0.000/00-00-c0-8a-df-56}
{CDS_Convergence medium}
{CDS_ParentPointer
 {{Parent_UUID 8eeb369a-9a4b-11cd-8f2b-0000c08adf56}
  {Timeout
   {expiration 1994-07-09-17:13:31.959}
   {extension +1-00:00:00.000I0.000}}
  {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+00:00I0.000/00-00-c0-8a-df-56}
{CDS_LastUpdate 1994-07-08-19:36:31.719+00:00I0.000/00-00-c0-8a-df-56}
{CDS_RingPointer b32648c6-928d-11cd-b4b5-0000c08adf56}
{CDS_Epoch 2f617aa6-9a76-11cd-8f2b-0000c08adf56}
{CDS_ReplicaVersion 3.0}
dcecp>
```

```
dcecp> directory show /.:/depts/radiology -schema
{RPC_ClassVersion multi}
{CDS_CTS single}
{CDS_UTS single}
{CDS_ObjectUUID single}
{CDS_Replicas 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
dcecp>
```

Related Information

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

dts

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:

- As a fully qualified name, for example:

/.../*cellname*/**hosts/**/*hostname*/**dts-entity**

- 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 **dtsd** (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 **dtsd** entity by default. Use the **dts_server** argument to operate on a remote DCE **dtsd**. This argument is a single server entry or string binding representing a **dtsd** that will be contacted for the operation. If the **_s(dts)** convenience variable is set, it is treated as the name of a **dtsd** 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 *DCE 1.2.2 Administration Guide* for more information.

Attributes

The **dts** object supports attributes and counters. Most attributes and counters pertain to **dtsd** processes in general. A subset of attributes and counters pertains only to **dtsd** 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 may not be set by a user, but is built in to **dtsd**.

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 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 **dtsd** wait for a longer interval.

maxdriftrate

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 may 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 may 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 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 may 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 may 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/***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/***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.

badtimereps
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.

providertimeouts
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 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.

See the *DCE 1.2.2 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 *DCE 1.2.2 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.

-global 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**/*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 **w** (**write**) permission on the DTS entity in order to execute the command.

Examples

The following example sets the local **dtsd** to be a global DTS server:

```
dcecp> dts configure -global
global
dcecp>
```

dts deactivate

Changes a DTS entity from an active state to an inactive state. The syntax is as follows:

dts deactivate [*dts_server*]

The **deactivate** operation changes a DTS entity from an active state to an inactive state. The **status** attribute is changed to **disabled**, which tells the DTS entity to stop synchronizing. 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

```
dcecp> dts deactivate
dcecp>
```

dts help

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

dts help [*operation* | **-verbose**]

Options

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

Used without an argument or option, the **dts help** command returns brief information about each **dts** 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 **dts** object itself.

Privileges Required

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

Examples

```
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'.
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**:

```
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.

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:

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 1000000 nsec/sec}
{clockresolution 10000000 nsec}
{version V1.0.1}
{timerep V1.0.0}
{provider no}
{autotdfchange no}
{nexttdfchange 1994-10-30-01:00:00.000-05:00I0.000}
{serverprincipal hosts/medusa/self}
{serverentry hosts/medusa/dts-entity}
{servergroup subsys/dce/dts-servers}
{status enabled}
{uuid 000013ed-000b-0000-b8ef-03a4fcdf00a4}
dcecp>
```

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
dcecp>
```

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
dcecp>
```

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)**.

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.

Data Structures

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-bd93-0000c08adf56 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 may 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 *DCE 1.2.2 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

-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 RPC string bindings.

See **Data Structures** for the format of a protocol sequence.

-object *object_uuid_list*
Declares the UUID of an object. Each **create** operation accepts a list of up to 32 object UUIDs.

See **Data Structures** for the format of the object UUID.

-annotation *annotation*
Defines an annotation string for the endpoint. The annotation string enables you to identify the purpose of the endpoint. 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.

Use quotation marks around the annotation field of endpoints to include internal spaces in an annotation, or use **dcecp** syntax.

-noreplace Use the **-noreplace** option when you want a host to run multiple instances of a server. Normally, when you add an interface-binding combination (a mapping) that already exists in an endpoint map, **dcecp** replaces the existing mapping with the new one. This behavior limits the number of server instances to one. Bypass this limitation by using the **-noreplace** option. Using this option can cause obsolete endpoints to accumulate in the endpoint map. Remove obsolete endpoints by using the **endpoint delete** command.

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 backslashs, **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\[1067\]
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 may 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
dcecp>
```

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 map, you can delete any other stale UUIDs that may 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 *DCE 1.2.2 Problem Determination Guide* for complete descriptions of all error messages.

Files

*dcelocal/***dce_cf.db**
The local DCE configuration database.

Related Information

Functions: **dce_cf_get_cell_name(3dce)**.

getip

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 may 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 *DCE 1.2.2 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

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**]

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*.

group_name The name of a 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:

- Fully qualified names in the form **/.../***cell_name*/*group_name* or **/.:/***group_name*.
- 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.

DESCRIPTION

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 *DCE 1.2.2 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 *DCE 1.2.2 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.

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>

dcecp> group add users -member chopin
dcecp>
```

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 *DCE 1.2.2 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
dcecp>
```

group help

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

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 *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 **group** object itself.

Privileges Required

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

Examples

```
dcecp> group help
add                 Adds a member to the named group.
catalog             Returns a list of all the names of groups in the registry.
create              Creates a group.
delete              Deletes a group.
list                Returns all of the members of a group.
modify              Changes the information about a group.
remove              Removes a specified member from the named group.
rename              Renames the specified group.
show                Returns the attributes of a group.
help                Prints a summary of command-line options.
operations          Returns a list of the valid operations for this command.
dcecp>
```

group list

Returns a list of the names of all members of a group. The syntax is as follows:

group list *group_name_list* [**-simplename**]

Options

-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 *cellname*/*membername*. Use the **-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-ptgt
/.../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 *DCE 1.2.2 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 show users3
{alias no}
{gid 5212}
{uuid 0000145c-9363-21cd-a601-0000c08adf56}
{inprojlist no}
{fullname {General Nursing Staff}}
dcecp>

dcecp> group modify users3 -add {test_era 101}
dcecp>

dcecp>group show users3 -all
{alias no}
{gid 5212}
{uuid 0000145c-9363-21cd-a601-0000c08adf56}
{inprojlist no}
{fullname {General Nursing Staff}
{test_era 101}}
dcecp>
```

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.

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.

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 may 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

```
dcecp> group rename users4 -to users_temporary
dcecp>
```

group show

Returns registry information for the specified groups. The syntax is as follows:

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

```
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*] [**-simplename**]

host configure *host_name* **-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 *host_name*

host show [*host_name*]

host start [*host_name*]

host stop [*host_name*] [**-force**]

host unconfigure *host_name* [**-force**]

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

/.:

host_name The name of a single host to operate on. Some host commands accept both fully qualified names (as in **/...**/*cellname*/**hosts**/*hostname*) and cell relative names (as in **hosts**/*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 may differ from the behavior of commands operating remotely, with more operations performed on the local host than may 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 *DCE 1.2.2 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. The syntax is as follows:

host configure *host_name* **-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 *host_name* 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 *host_name* argument is the name of the local host machine without the cell name prepended, as in the following:

hosts/*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**:

```
dcecp> host configure  hosts/hydra -client \
> -cell my_cell.goodco.com -password fstzkl -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.
ping                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

```
dcecp> host operations
catalog configure ping show start stop unconfigure help operations
dcecp>
```

host ping

Tests whether DCE processes are accessible from the network. The syntax is as follows:

host ping *host_name*

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 *host_name* argument is the fully qualified name of the host to ping as in the following:

/.:/hosts/*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
```

```
dcecp>
```

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 [*host_name*]

The **show** operation returns a list describing all processes that are configured to run on the specified host. The optional *host_name* 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:

- The server name as output by the **server catalog -simplename** command.
- One of the tokens **running** or **notrunning**.
- 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. The syntax is as follows:

host start [*host_name*]

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 may 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 *host_name* argument is the fully qualified or cell-relative name of the host to operate on, as in the following:

/.:/hosts/*hostname*

hosts/*hostname*

Without the *host_name* 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**:

```
dcecp> host start hosts/hydra
dcecp>
```

host stop

Stops all DCE processes on the specified host. The syntax is as follows:

host stop [*host_name*] [**-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 may not be used to stop DCE on systems that use versions prior to Version 1.1. The *host_name* argument is the fully qualified or cell-relative name of the host to operate on, as in

/.:/hosts/*hostname***.**

hosts/*hostname*

Processes are stopped as follows:

- 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 **srvrexec** objects, they will not be stopped.
- 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/srvrexec** 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. The syntax is as follows:

host unconfigure *host_name* [**-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:

- All objects, directories and links from **/.:/hosts/***hostname* including the directory itself
- All principal names beginning with **hosts/***hostname*, 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/*hostname*

This operation returns an empty string on success.

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* | **-attribute** *value*} [**-binary**][**-local**]

hostdata delete *hostdata_name_list* [**-entry**][**-local**]

hostdata help [*operation* | **-verbose**]

hostdata modify *hostdata_name_list* {**-change** *attribute_list* | **-attribute** *value*} [**-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.

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/*hostname***/config/hostdata/***name*

However, a shorthand notation referring to the local machine consisting of just *name* can be used in some circumstances.

When the **dced** on the local machine is in partial service mode, you must use the **-local** option to access the **hostdata** object. To access the **hostdata** objects when

dced is in partial service mode, specify only the residual portion of the object name. For example, specify **hostdata**, not **/.:/hosts/gumby/config/hostdata**.

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 *DCE 1.2.2 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 *DCE 1.2.2 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/***host_name*/**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**
attribute_list |
-*attribute value*} [**-binary**] [**-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 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.

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

-entry Only the configuration information that **dced** keeps is deleted, not the actual hostdata.

-local 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 *hostdata_name_list* {**-change** *attribute_list* | *-attribute value*} [**-binary**] [**-local**]

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-91b1-080009251352}
{annotation  {}}
```

```
{storage /tmp/file1}
{hostdata/data {new first line}}
dcecp>

dcecp> cat /tmp/file1
new 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
{uuid 00174f6c-6eca-1d6a-bf90-0000c09ce054}
{annotation  {Name of cell}}
{storage cell_name}
{hostdata/data /.../my_cell}
dcecp>
```

Related Information

Commands: **dcecp(8dce)**, **dced(8dce)**, **hostvar(8dce)**.

hostvar

Purpose

A dcecp task object that manages the security binary compatibility attributes

Synopsis

hostvar help [*operation* | **-verbose**]

hostvar operations

hostvar set {**-secbinarycompat** {on | off} | [**-krbccachevno** *version_number*] | [**-krbktvno** *version_number*]}

hostvar show {**-all** | [**-cellname**]| [**-hostname**]| [**-krbccachevno**]| [**-krbktvno**]| [**-secbinarycompat**]}

Arguments

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

Description

The **hostvar** object allows you to easily set the security binary compatibility attribute and the Kerberos V5 compatibility attribute for the current host. It also lets you display the local host's cellname, hostname, security binary compatibility attribute, and Kerberos V5 compatibility attributes that are stored by the **dced** in the **hostdata/dce_cf.db** object.

The cell's security server uses these compatibility attributes to determine the mode and state in which the local machine is operating.

Attributes

cellname The name of the local hosts cell.

hostname The name of the local host.

krbccachevno
The version number format of the Kerberos V5 credential cache files created by DCE applications. The version chosen should allow optimal compatibility with any Kerberos V5 applications running on the host. Valid values are the integers **1** through **3**, inclusive. See the *DCE 1.2.2 Administration Guide* for more information about Kerberos 5 attributes.

krbktvno The version number format of the Kerberos V5 keytab files created by DCE applications. The version number format chosen should allow optimal compatibility with any Kerberos V5 applications running on the host. Valid values are the integers **1** and **2**. See the *DCE 1.2.2 Administration Guide* for more information about Kerberos 5 attributes.

secbinarycompat
The security binary compatibility attribute that enables (**on**) and disables (**off**) binary compatibility for statically linked versions of DCE.

Errors

A representative list of errors that might be returned is not shown here. Refer to the *DCE 1.2.2 Problem Determination Guide* for complete descriptions of all error messages.

Operations

hostvar help

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

hostvar help [*operation* | **-verbose**]

Option

-verbose Displays information about the **hostvar** task object.

Used without an argument or option, the **hostvar help** command returns brief information about each **hostvar** 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 **hostvar** task object itself.

Privileges Required

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

Examples

```
dcecp> hostvar help
set                 Sets the value of the specified host variable.
show                Returns the value of the specified host variable.
help                Prints a summary of command-line options.
operations          Returns a list of the valid operations for this command.
dcecp>
```

hostvar operations

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

hostvar 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 **hostvar operations** command.

Examples

```
dcecp> hostvar operations
set show help operations
dcecp>
```

hostvar set

Sets the security binary compatibility and krb5 compatibility value attributes for the current host. The syntax is as follows:

hostvar set {-secbinarycompat {**on** | **off**} |
[**-krbccachevno** *version_number*] [**-krbktvno** *version_number*]}

Option

-secbinarycompat
Sets the security binary compatibility attribute **on** or **off**. This cannot be used with the **-krbccachevno** and **-krbktvno** options.

-krbccachevno
Sets the Kerberos V5 credential cache file version number format.

-krbktvno Sets the Kerberos V5 keytab file version number format.

Under normal circumstances, **secbinarycompat** is set to **on** when DCE is configured, which allows binary compatibility between Version 1.2 and Versions 1.0.3 and 1.1. Binary compatibility allows applications linked with an archived **libdce** to share login contexts and credentials without loss of data.

You can disable binary compatibility on a per-host basis to achieve minor performance gains and slightly smaller credentials files by setting **secbinarycompat** to **off**. If you enable binary compatibility after it has been disabled, you must start and stop all DCE daemons. This operation returns an empty string on success.

The **-krbccachevno** and **-krbktvno** attributes are set to **1** when DCE is configured to allow compatibility with DCE 1.0.3 based applications and older Kerberos V5 applications. Depending on certain compatibility requirements, you may choose to change the values of the attributes. See the *DCE 1.2.2 Administration Guide* for more information about Kerberos 5 attributes.

Privileges Required

No special privileges are required to use the **hostvar set** command.

Examples

```
dcecp> hostvar set -secbinarycompat on
dcecp>
```

hostvar show

Returns the values of the host attributes on the local host. The syntax is as follows:

hostvar show {**-all** | [**-cellname**] [**-hostname**] [**-krbccachevno**] [**-krbktvno**] [**-secbinarycompat**]}

Options

-all Returns the values of all attributes.

-cellname Returns the value of the *cellname* attribute.

-hostname Returns the value of the *hostname* attribute.

-krbccachevno
Returns the value of the **krbccachevno** attribute.

-krbktvno Returns the value of the **krbktvno** attribute.

-secbinarycompat
Returns the value of the **secbinarycompat** attribute.

The **show** operation makes it easy to view the attributes stored in the **dced hostdata/dce_cf.db** object. All the values returned by this command are from that object. Use the **-all** option to display all attributes; use individual options to display individual attributes.

Privileges Required

No special privileges are required.

Examples

```
dcecp> hostvar show -all
{cellname /.../gumby1}
{hostname hosts/blech}
{krbccachevno 1}
{krbktvno 1}
{secbinarycompat on}
dcecp>
```

Related Information

Commands: **dcecp(8dce)**, **dced(8dce)**, **hostdata(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 [*host_name_list*] [**-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

host_name_list

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/*hostname*
/.../*cell_name***/hosts/***hostname*
hosts/*hostname*

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 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:

/.../*cell*/**hosts**/*hostname*/**config/keytab**/*name*

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

/.../*cell_name*/**hosts**/*hostname*/**config/keytab**/*name*

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*

melman 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 may not be modified after creation.

data *key_list*
The contents of the key table. Represented as a list of keys.

See the *DCE 1.2.2 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 *DCE 1.2.2 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. 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 keytables 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 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 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:

  - **des** indicates DES encryption.
  - **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.

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 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

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

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 may 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.

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 *DCE 1.2.2 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 *DCE 1.2.2 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:

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:

{*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 1998-01-20-12:00:00:00} {extension +10-0:0:0}} }
dcecp>
```

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
dcecp>
```

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+00:00I0.000/00-00-c0-8a-df-56}
{CDS_UTS 1994-07-11-17:52:44.698+00:00I0.000/00-00-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/***host_name***/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 serviceability and debug messages. Debug routing may 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.

ERRORS

A representative list of errors that might be returned is not shown here. Refer to the *DCE 1.2.2 Problem Determination Guide* for complete descriptions of all error messages.

OPERATIONS

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

```
dcecp> 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.
dcecp>
```

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.

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 (serviceability 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 and debug messages can be written to any of the normal output destinations. You can specify routing for serviceability and debug messages in any of the following ways:

- Through the **dcecp log** object, if the server supports the remote serviceability interface
- By the contents of the *dce-local-path*/**svc/routing** routing file
- 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 debug routing settings rather than serviceability 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 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>
```

Related Information

Commands: **dcecp(8dce)**

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 *DCE 1.2.2 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*

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
dcecp>
```

name parse

Divides a name into a cell name and a residual name. The syntax is as follows:

name parse *name*

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 *DCE 1.2.2 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 *DCE 1.2.2 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 May 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***}}**

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 May 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:

```
    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**.

```
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
dcecp>
```

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

```
dcecp> object show /.:/obj
{RPC_ClassVersion
 {0200}
 {0300}}
{RPC_Group 1234}
{CDS_CTS 1994-07-01-22:06:54.990-05:00I0.000/00-00-c0-f7-de-56}
{CDS_UTS 1994-07-01-22:07:37.248-05:00I0.000/00-00-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:

- Fully qualified names in the form: **/...**/*cell_name*/*organization_name* or **/.:**/*organization_name*
- 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.

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.

- Organization attributes consist of the organization's name, Universal Unique Identifier (UUID), and organization identifier. Organization attributes may or may not have default values. They assume a default value or a value set by administrators.
- 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 *DCE 1.2.2 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 *DCE 1.2.2 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

-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 */cellname/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 *DCE 1.2.2 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**.

-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 *DCE 1.2.2 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 fullname and/or **m** (**mgmt_info**) permission to change the organization's managment information.

Examples

```
dcecp> organization modify temps -acctlife 180-00:00:00 \
> -pwdalpha yes -pwdlife 30-00:00:00 \
> -pwdexpdate 1995-12-31-23:59:59 -pwdspaces yes
dcecp>

dcecp> organization modify temps -add {test_era 101}
dcecp>
```

```
dcecp> organization show temps -all
{fullname {}}
{orgid 12}
{uuid 0000000c-03d5-21cf-9802-08000985b5a6}
{test_era 101}
{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>
```

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

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
/.../my_cell.goodcompany.com/R_Mathews
/.../my_cell.goodcompany.com/K_Doe
/.../my_cell.goodcompany.com/W_Ross
/.../my_cell.goodcompany.com/W_Williams
/.../my_cell.goodcompany.com/D_White
dcecp>
```

```
dcecp> organization rename rigel -to sirus
dcecp>

dcecp> organization list rigel
Error: Registry object not found
dcecp>

dcecp> organization list sirus
/.../my_cell.goodcompany.com/H_Lewis
/.../my_cell.goodcompany.com/R_Mathews
/.../my_cell.goodcompany.com/K_Doe
/.../my_cell.goodcompany.com/W_Ross
/.../my_cell.goodcompany.com/W_Williams
/.../my_cell.goodcompany.com/D_White
dcecp>
```

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 may 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

```
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:

- Fully qualified principal names in the form

/.../cell_name/principal_name **or** */.:/principal_name*

- 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 may 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 *DCE 1.2.2 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 *DCE 1.2.2 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
/.../small_cell.goodcompany.com/root
/.../small_cell.goodcompany.com/daemon
/.../small_cell.goodcompany.com/sys
/.../small_cell.goodcompany.com/bin
/.../small_cell.goodcompany.com/uucp
/.../small_cell.goodcompany.com/who
/.../small_cell.goodcompany.com/mail
/.../small_cell.goodcompany.com/tcb
/.../small_cell.goodcompany.com/dce-ptgt
/.../small_cell.goodcompany.com/dce-rgy
/.../small_cell.goodcompany.com/cell_admin
/.../small_cell.goodcompany.com/krbtgt/small_cell.goodcompany.com
/.../small_cell.goodcompany.com/hosts/pmin17/self
/.../small_cell.goodcompany.com/hosts/pmin17/cds-server
/.../small_cell.goodcompany.com/hosts/pmin17/gda
/.../small_cell.goodcompany.com/William_Ward
/.../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
dcecp>
```

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 *DCE 1.2.2 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

Privileges Required

You must have **r** (**read**) permission to the principal to be modified and **f** (**full name**) permission to change the principal's fullname 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 show joe -all
{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.

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>

dcecp> principal list K_Doe
Error: Registry object not found
dcecp>
```

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}
dcecp>
```

Related Information

Commands: **account(8dce)**, **dcecp(8dce)**, **group(8dce)**, **organization(8dce)**, **xattrschema(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:

- A specific cell name to bind to any replica in the named cell, such as **/.:** or **/.../gumby1**.
- The global name of a replica to bind to that specific replica in that specific cell. such as **/.../gumby1/subsys/dce/sec/oddball**.
- 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**.
- 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 may 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:

- **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.
- **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.
- **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.
- **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 indentifiers. 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 indentifiers. 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.

uninitialized
The database is a stub database that has not been initialized by the master replica or another up-to-date replica.

unknown The replica is not known to the master.

lastupdtime The localized date and time that the master received the last replica's last update.

lastupdseq 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**}.

addresses A list of the network addresses of the replica. There can be more than one for connectionless and connection-oriented protocols.

masteraddrs
The network address of the master replica as determined by the replica. The address is not necessarily correct. More than one address may exist for connectionless and connection-oriented protocols for example.

masterseqnum
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**}.

masteruuid The UUID of the master replica as determined by the replica. This UUID is not necessarily correct. The value is a UUID.

supportedversions
DCE registry version supported by the security service. Possible values at DCE Version 1.1 are **secd.dce.1.0.2** (for DCE Version 1.0.2 and DCE version 1.0.3) and **secd.dce.1.1**. Both versions may be supported (that is by a DCE Version 1.1 security server running in a cell with DCE version 1.0.3 replicas).

updseqqueue
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**}}.

Replica-Specific Attributes

name The name of the replica. It is in the form of a fully qualified CDS name.

uuid The UUID of the replica.

type Indicates if the replica is a **master** or a **slave**.

addresses A list of the network addresses of the replica. More than one address may 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 *DCE 1.2.2 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 *DCE 1.2.2 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**]

Option

-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* | *num***m** | *num***h**}] [**-now**]

Options

-at *hh:mm* Specifies the the hours and minutes of the day (in UTC time) to perform the checkpoint.

-cpi {*num* | *num***m** | *num***h}**
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:

- To specify seconds, supply only a number. For example, **-cpi 101** specifies an interval of 101 seconds.
- To specify minutes enter the number and **m**. For example, **-cpi 101m** specifies an interval of 101 minutes.
- 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 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.

Privileges Required

You must have **d** (**delete**) permission to the **replist** object.

Examples

```
dcecp> registry delete /.:/subsys/dce/sec/oddball
dcecp>
```

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:

- 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.

- 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:

- If called with the **-force** or **-master** option, it sets **_b(sec)** to the replica to which it binds.
- If called with no options, it sets **_b(sec)** to the master.

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
dcecp>
```

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
dcecp>
```

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 **registry dump** command is the same as the following script:

```
foreach i [registry catalog] {
 lappend r [registry show $i -replica]
 append r
}
return r
```

This operation sets the **_b(sec)** variable to the last replica listed in the display.

Privileges Required

You must have **A** (**admin**) permission to the **replist** object.

Examples

```
dcecp> registry dump
{name /.../dcecp.cell.osf.org/subsys/dce/sec/snow}
{type master}
{cell /.../dcecp.cell.osf.org}
{uuid a1248a5e-e1e6-11cd-aa0c-0800092734a4}
{status enabled}
{lastupdtime 1994-10-13-14:44:48.000-04:00I-----}
{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 /.../dcecp.cell.osf.org/subsys/dce/sec/ice}
{type slave}
```

```
{cell /.../dcecp.cell.osf.org}
{uuid c772f46a-e1ec-11cd-9a16-0000c0239a70}
{status enabled}
{lastupdtime 1994-10-13-14:44:48.000-04:00I-----}
{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>
```

registry enable

Enables the master registry for updates. The syntax is as follows:

registry enable [*registry_replica_name*]

The **enable** operation enables the master registry for updates. The *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.
set                 Changes which replica is the master.
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.
dcecp>
```

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 /.:/susbys/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 polices.

-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
{name /.../absolut_cell/subsys/dce/sec/ice}
```

```
{type slave}
{cell /.../absolut_cell}
{uuid 91259b6c-9415-11cd-a7b5-080009251352}
{status enabled}
{lastupdtime 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}}
dcecp>
dcecp> registry show /.../dcecp.cell.osf.org/subsys/dce/sec/snow -master
{name /.../dcecp.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 /.../dcecp.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}
{lastupdtime 1994-10-13-14:58:28.000-04:00I-----}
{lastupdseqsent 0.528}
{numupdtogo 0}
{commstate ok}
{lastcommstatus {successful completion}}
dcecp>
```

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

You must have **A** (**admin**) permission to the **replist** object.

Examples

```
dcecp> registry stop /.:/subsys/dce/sec/snow
dcecp>
```

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:

```
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.

Data Structures

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-bd93-0000c08adf56 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 may 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 *DCE 1.2.2 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 on 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
dcecp>
```

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:

```
dcecp> rpcentry export /.:/subsys/applications/bbs_server \
> -interface {458ffcbe-98c1-11cd-bd93-0000c08adf56 1.0} \
> -binding {ncacn_ip_tcp 130.105.1.227} \
> -object {76030c42-98d5-11cd-88bc-0000c08adf56}
dcecp>
```

rpcentry help

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

rpcentry help [*operation* | **-verbose**]

Options

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

Used without an argument or option, the **rpcentry help** command returns brief information about each **rpcentry** 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 **rpcentry** object itself.

Privileges Required

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

Examples

```
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.
dcecp>
```

rpcentry import

Returns a string binding from the specified RPC entry. The syntax is as follows:

rpcentry import *entry_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 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}
dcecp>
```

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
dcecp>
```

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
{458ffcbe-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
{458ffcbe-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
{458ffcbe-98c1-11cd-bd93-0000c08adf56 1.0
  {ncacn_ip_tcp 130.105.1.227}}
{76030c42-98d5-11cd-88bc-0000c08adf56}
dcecp>

dcecp> rpcentry unexport /.:/subsys/applications/bbs_server \
> -interface {458ffcbe-98c1-11cd-bd93-0000c08adf56 1.0}
dcecp>
```

```
dcecp> rpcentry show /.:/subsys/applications/bbs_server
{76030c42-98d5-11cd-88bc-0000c08adf56}
dcecp>
```

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.

Data Structures

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:

-interface ec1eeb60-5943-11c9-a309-08002b102989,3.11

Leading zeros in version numbers are ignored.

Alternatively, you can use **dcecp** string syntax. For example:

-interface {458ffcbe-98c1-11cd-bd93-0000c08adf56 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 *DCE 1.2.2 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
dcecp>
```

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
dcecp>
```

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 {ec1eeb60-5943-11c9-a309-08002b102989 1.1} \
> -object 30dbeea0-fb6c-11c9-8eea-08002b0f4528
{ncadg_ip_udp 15.22.48.25}
{ncacn_ip_tcp 15.22.48.25}
```

```
dcecp>
```

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 **/.:/subsys/applications/infobases**, in the order in which they were added to the group:

```
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>
```

The following example uses the **-member** option to list a specific member of the group **/.:/subsys/applications/infobases**:

```
dcecp> rpcgroup list /.:/subsys/applications/infobases \
> -member /.:/subsys/applications/bbs_server
/.../my_cell.goodcompany.com/subsys/applications/bbs_server
dcecp>
```

rpcgroup operations

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

rpcgroup 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 **rpcgroup operations** command.

Examples

```
dcecp> rpcgroup operations
add create delete import list remove help operations
dcecp>
```

rpcgroup remove

Removes one or more members from the specified group. The syntax is as follows:

rpcgroup remove *rpcgroup_name_list* **-member** *member_name_list*

Options

-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 *rpcgroup_name_list* argument. The *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

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:

```
{d46113d0-a848-11cb-b863-08001e046aa5 2.0}
/.../my_cell.goodcompany.com/sec 0 rs_bind}
```

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**.

Data Structures

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-bd93-0000c08adf56 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 *DCE 1.2.2 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.

-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
dcecp>
```

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
dcecp>
```

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
add                 Adds members to a list of RPC profiles.
create              Creates a list of empty RPC profiles.
delete              Deletes a list of RPC profiles.
import              Returns the bindings from a list of RPC profiles.
list                Returns the names of members of a list of RPC profiles.
remove              Removes members from a list of RPC profiles.
show                Returns the attributes of a list of RPC profiles.
help                Prints a summary of command-line options.
operations          Returns a list of the valid operations for this command.
dcecp>
```

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
/.../my_cell.goodcompany.com/lan-profile
/.../my_cell.goodcompany.com/fs
/.../my_cell.goodcompany.com/subsys/dce/dfs/bak
dcecp>
```

rpcprofile operations

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

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

```
dcecp> rpcprofile operations
add create delete import list remove show help operations
dcecp>
```

rpcprofile remove

Removes one or more members from the specified profile. The syntax is as follows:

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 **dcecp** 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 /.:/subsys/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.

-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 may have only one value (that is, the value may 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
{{458ffcbe-98c1-11cd-bd93-0000c08adf56 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-0000c08adf56 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-0000c08adf56 1.0}
{{baf8c319-998f-11cd-ac7b-0000c08adf56 1.0} /.../cell.co.com/subsys/appls/infobases 0}
```

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 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/***host_name*

/*.../cell_name*/**hosts/***host_name*

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**/*host_name*/**config/secval**.

Errors

A representative list of errors that might be returned is not shown here. Refer to the *DCE 1.2.2 Problem Determination Guide* for complete descriptions of all error messages.

Operations

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 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

```
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

```
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

```
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

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**/*hostname*
/**.../***cell_name*/**hosts**/*hostname*
hosts/*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**/*host_name*/**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 **srvrexec** 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 may contain application-specific extended registry attributes (ERAs). Only the ERAs can be modified after creation; other attributes cannot.

When the **dced** on the local machine is in partial service mode, you must use the **-local** option to access the **server** object. To access the **server** object when **dced** is in this mode, specify only the residual portion of the object name. For example, specify **server/***server_name*, not **/.:/hosts/***host_name***/config/server/***server_name*.

Data Structures

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 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-bd93-0000c08adf56 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. May 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 **dced** 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 may also have ERAs attached to them. ERAs may be manipulated by the **modify** operation.

See the *DCE 1.2.2 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 *DCE 1.2.2 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

-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**/*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_mgmt** 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/srvrexec/try_tserver
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** syntax.

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/srvrexec/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_rgy_attr_list* | **-remove** *extended_rgy_attr_list* [**-types**] |
-change *attribute_list*} [**-local**]

Options

-add *extended_rgy_attr_list*
Allows you to add ERAs that may be defined for your environment. You can specify the attributes to be added as a list. See the *DCE 1.2.2 Administration Guide* for more information about ERAs.

-remove *extended_rgy_attr_list*
Allows you to remove ERAs that may be defined for your environment. You can specify the attributes to be removed as a list. See the *DCE 1.2.2 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:

{{*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 may 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:

- The name of a server entry in the namespace to be imported from. For example:

  /.../brain_cell/hosts/wallis/srvrexec/event_server

- A string binding with an object UUID specified. For example:

  {00337ea9-d979-1dd8-923f-0000c08adf56 ncacn_ip_tcp 15.121.12.72}

- A string binding with an endpoint specified. For example:

  {ncacn_ip_tcp 15.121.12.72 1075}

- An interface ID followed by a hostname, separated by commas. For example:

```
{4885772c-c6d3-11ca-84c6-08002bic8fif,oddball}
```

- 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/srvrexec/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.

-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 **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

```
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:

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

```
dcecp> server start /.:/hosts/foster/config/srvrconf/try_tserver
d90a0374-eb99-11cd-91b1-080009251352
dcecp>
```

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

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:

- Fully qualified principal names in the form

  /.../*cell_name/principal_name* **or /.:/***principal_name*

- 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:

- Changing the location of the CDS directory created for users, or remove it completely.
- Changing the default ACLs placed on the various objects.
- Setting certain attributes or policies on all newly created principals and accounts to match the site's policies.
- Setting up site specific defaults for passwords (to be changed by the user later), groups, organizations, principal directories, and so on.
- 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 may 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 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. 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 (**[-]***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*

Lets 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 *DCE 1.2.2 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 *DCE 1.2.2 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 **-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 **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 may be specified with the **-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 may not be specified, as they would probably affect more than the created user. The required group and organization names may be specified either as attributes in the **-attributes** option or via the **-group** and **-organization** options. The required **password** attribute may be provided as in the **account create** command, and the **-mypwd** option is also required.

Privileges Required

Because the **user create** command performs several operations, you need the permissions associated with each operation, as follows:

- To create the principal name, you must have **i** (**insert**) permission to the directory in which the principal is to be created.

- If the specified groups or organizations do not already exist and you use the **-force** option, you must have **i** (**insert**) permission to the directories in which the groups and organizations are to be created.
- To create the account, you must have **m** (**mgmt_info**), **a** (**auth_info**), and **u** (**user_info**) permission to the principal named in the account, **r** (**read**) permission to the organization named in the account, **r** (**read**) permission to the group named in the account, and **r** (**read**) permission on the registry policy object.
- To create the directory in CDS you must have the following permissions:
  - **r** (**read**) and **i** (**insert**) permission to the parent directory
  - **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 **K_Parsons** and adds him to a group named **users** and an organization named **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
/.../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 procedes as follows:

- Deletes the principal from the registry, which also deletes the account and removes the principal from any groups and organizations.
- 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:

- 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.
- 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.

- 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.
- 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:

```
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

```
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

```
dcecp> user operations
create delete show help operations
dcecp>
```

user show

Returns the attributes of a single DCE user. 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 *Iss.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***.***fff***Iss.***fff***

Relative times often omit fractions of seconds (the leftmost *.fff* sequence) and generally lack an inaccuracy component (*Iss.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.

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 *DCE 1.2.2 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**]

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

```
dcecp> utc compare 1994-10-18-13:22:32.816-04:00I—— \
> 1994-10-18-13:21:50.419-04:00I—— -noinaccuracy
1
dcecp>
```

utc convert

Converts a timestamp from UTC to local time. The syntax is as follows:

utc convert *absolute_timestamp* [**-gmt**]

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-----
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.
subtract        Returns the difference between two timestamps.
help            Prints a summary of command-line options.
operations      Returns a list of the valid operations for this command.
dcecp>
```

utc multiply

Multiplies 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——
1994-10-18-13:22:17.816+00:00I-----
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:

```
C069d9fb6-943e-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 *DCE 1.2.2 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

```
dcecp> uuid compare 03bb2688-943e-11cd-8bfd-0000c08adf56 \
> 069d9fb6-943e-11cd-a35c-0000c08adf56
0
dcecp>
```

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
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 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 ERAs

Synopsis

xattrschema catalog *schema_name* [**-simplename**]

xattrschema create *schema_entry_name_list* {**-attribute** *attribute_list* | **-attribute** *value*}

xattrschema delete *schema_entry_name_list*

xattrschema help [*operation* | **-verbose**]

xattrschema modify *schema_entry_name_list* {**-change** *attribute_list* | **-attribute** *value*}

xattrschema operations

xattrschema rename *schema_entry_name* **-to** *new_schema_entry_name*

xattrschema show *schema_entry_name_list*

Arguments

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

aclmgr *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**. This attribute is only advisory in DCE Version 1.1. Future versions of DCE will support this functionality.

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.

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 (OSF 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 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 *value*

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**.

This attribute is only advisory in DCE Version 1.1. Future versions of DCE will support this functionality.

multivalued {**yes** | **no**}

Indicates that ERAs of this type may be multivalued (that is, multiple instances of the same attribute type may be attached to a single registry object). The value of this attribute must be **yes** or **no**. This attribute is not modifiable after creation. The default is **no**.

reserved {**yes** | **no**}

If set, this schema entry may 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 may 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 may 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 is not modifiable after creation. The default is **none**.

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 is not modifiable after creation. The default is **no**.

This attribute is only advisory in DCE Version 1.1. Future versions of DCE will support this functionality.

uuid *uuid* The internal identifier of the ERA. The value is a UUID. This attribute is not modifiable after creation. If not specified on the **create** operation, a value is generated by the system.

See the *DCE 1.2.2 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 *DCE 1.2.2 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 catalog *schema_name* [**-simplename**]

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 *schema_entry_name_list*
{**-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 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***}}**

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 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

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

```
dcecp> xattrschema modify /.:/sec/xattrschema/test_integer \
> -aclmgr {organization r r r r}
dcecp>
```

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 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)**.

Chapter 2

Remote Procedure Call Commands

rpc_intro

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.

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.

Base Data Type Specifiers—rpc_intro(1rpc)				
Specifier				
(sign)	**(size)**	**(type)**	**Size**	**Type Macro Emitted by idl**
	small	**int**	8 bits	**idl_small_int**
	short	**int**	16 bits	**idl_short_int**
	long	**int**	32 bits	**idl_long_int**
	hyper	**int**	64 bits	**idl_hyper_int**
unsigned	**small**	**int**	8 bits	**idl_usmall_int**
unsigned	**short**	**int**	16 bits	**idl_ushort_int**
unsigned	**long**	**int**	32 bits	**idl_ulong_int**
unsigned	**hyper**	**int**	64 bits	**idl_uhyper_int**
		float	32 bits	**idl_short_float**
		double	64 bits	**idl_long_float**
		char	8 bits	**idl_char**
		Boolean	8 bits	**idl_boolean**
		byte	8 bits	**idl_byte**
		void	—	**idl_void_p_t**
		handle_t	—	**—**

Related Information

Commands: **idl(1rpc)**, **uuidgen(1rpc)**.

Books: *DCE 1.2.2 Application Development Guide*, *DCE 1.2.2 Problem Determination Guide*.

idl

Purpose

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.

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.

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. If the **-lang cxx** option is used, the source file has a **.cxx** extension.

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. If the **-lang cxx** option is used, the source file has a **.cxx** extension.

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 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 *DCE 1.2.2 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 *DCE 1.2.2 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.

-D*name*[**=***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 **-D***name* argument on the command line. This argument has no effect if you use the **-no_cpp** argument.

-U*name*
Removes (undefines) any initial definition of a symbol name as defined by **-D***name*. You can use this method to remove a symbol name instead of using **#undef** in the source code. You can use more than one **-U***name* 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 DCE 1.1.)

-stdin Takes the standard output of a previous utility as the input to the **idl** command. For example:

```
cat my_filename.idl | idl -stdin
```

-version Displays the current version of the IDL compiler.

-v Prints informational messages (verbose mode) on the screen while the compiler is running.

-no_warn Suppresses compiler warning messages.

-confirm Displays all the **idl** command arguments you chose, but does not compile the source IDL file. If you use this with the **-v** argument, informational messages about how the compiler behaves if you do not use **-confirm** are displayed but no corresponding actions are performed.

Description

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:

- An interface definition filename.
- 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 may result in unexpected behavior.

iovector.idl	**lbase.idl**	**nbase.idl**	**ncastat.idl**
ndrold.idl	**rpc.idl**	**rpcbase.idl**	**rpcpvt.idl**
rpcsts.idl	**rpctypes.idl**	**twr.idl**	**uuid.idl**

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 may occur during compilation of the stubs by the C compiler. A C compiler that is not fully ANSI compliant may 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_AUX_FILES** environment variable as follows:

```
setenv IDL_GEN_AUX_FILES 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. The server auxiliary default filename is overridden by creating a file named **test_sa.c** for the server auxiliary module.

```
idl test.idl -keep c_source -client none -sstub test_ss.c -saux test_sa.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 *DCE 1.2.2 Problem Determination Guide* for complete descriptions of all error messages.

Files

/lib/cpp C preprocessor

dceshared/**bin/idl**
Compiler

dceshared/**include**
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

Books: *DCE 1.2.2 Application Development Guide—Core Components*.

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.

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

   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 1251ace6-93al-11cd-95ad-0800097086e4

   This results in output like the following:

   ```
   = { /* 1251ace6-93al-11cd-95ad-0800097086e4 */
    0x1251ace6,
   ```

```
    0x93al,
    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 *DCE 1.2.2 Problem Determination Guide* for complete descriptions of all error messages.

Files

dceshared/**bin/uuidgen**
: Generator

dceshared/**nls/msg/***LANG*/**uuidgen.cat**
: Generator error messages

rpc_intro

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 superceded 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 endoint 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)**.

Books: *DCE 1.2.2 Administration Guide*, *DCE 1.2.2 Application Development Guide—Core Components*, *DCE 1.2.2 Application Development Reference*.

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.

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 superceded by the DCE control program (**dcecp**) in OSF DCE Version 1.1. This command may be fully replaced by the **dcecp** command in a future release of DCE, and may 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.

Function	At System Prompt	Inside rpccp
Strings within quotation marks	Supported	Not required
Wildcard substitution	Supported	Unsupported

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.

Scope	Command
All entries	**add entry** **remove entry** **show entry**
Server entry	**export** **import** **show server** **unexport**
Group	**add member** **remove group** **remove member** **show group**
Profile	**add element** **remove element** **remove profile** **show profile**
Endpoint map	**add mapping** **remove mapping** **show mapping**

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:

- A string representation of binding information (*binding string*)
- A string representation of an object or interface UUID (*string UUID*)
- An *interface identifier* (the interface UUID and version numbers)
- 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, may 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 *DCE 1.2.2 Application Development Guide—Core Components*. For a description of CDS, see the *DCE 1.2.2 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:

- 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:

  ```
  /.../C=US/O=uw/OU=MadCity
  ```

  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.

- cell-relative name

  Each name service entry requires a cell-relative name, which contains a directory pathname and a leaf name.

  — 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).

— 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 may 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 *DCE 1.2.2 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:

- To create an entry, you need **i** (**insert**) permission to the parent directory.
- To read an entry, you need **r** (**read**) permission to the CDS object entry.
- To write to an entry, you need **w** (**write**) permission to the CDS object entry.
- 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 may 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:

- interface identifier

  This is the primary search key. The interface identifier consists of the interface UUID and the interface version numbers.

- member name

  The entry name of one of the following kinds of name service entries:

— A server entry for a server offering the requested remote procedure call (RPC) interface and object

— A group corresponding to the requested RPC interface

— A profile

- 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.

- 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 may 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-a309-08002b102989,1.1 \
     -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:

- For a server entry, specify the new entry as the target entry for the **rpccp export** command.

- For a group, specify the new entry as the target group for the **rpccp add member** command.
- 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).

Notes

This command was replaced at DCE Version 1.1 by the **dcecp** command and may 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) may 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 ec1eeb60-5943-11c9-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:

```
-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:

- An interface ID, consisting of an interface UUID and versions (major and minor)
- Binding information
- An object UUID (optional)

- 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 may 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:

interface ID **ec1eeb60-5943-1169-a309-08002b102989,1.1**

string binding
ncadg_ip_udp:63.0.2.17[5347]

objects **005077d8-8022-1acb-9375-10005a4f533a 001bc29a-8041-1acb-b377-10005a4f533a**

annotation **Calendar version 1.1**

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 **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
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 **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 may have been created previously (by either the **add entry** or **add member** command). If the specified entry does not exist, the **add member** command tries to create the entry.

Privileges Required

You need both **r** (**read**) permission and **w** (**write**) permission to the Cell Directory Service (CDS) object entry (the target group 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 may not be provided in future releases of DCE.

Examples

The following command adds the member **/.:/LandS/anthro/Cal_host_3** to the group **/.:/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)**.

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*…} [**-s** *syntax*]

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 ec1eeb60-5943-11c9-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) may 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 may 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  \
  -i 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-11c9-8eea-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  \
     -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-11c9-8eea-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 Displays a list of commands or the options of a specified command

Synopsis **rpccp help** [*rpccp-command*]

Arguments

rpccp-command

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 may be replaced in future releases by the **dcecp** command, and may 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:

```
rpccp help
```

2. The following command displays the syntax of the **remove entry** command:

```
rpccp> help remove entry
```

Related Information

Commands: **rpccp_add_element(8rpc)**, **rpccp_add_entry(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(8rpc)**, **rpccp_show_entry(8rpc)**, **rpccp_show_group(8rpc)**, **rpccp_show_mapping(8rpc)**, **rpccp_show_profile(8rpc)**, **rpccp_show_server(8rpc)**, **rpccp_unexport(8rpc)**

import

Purpose

Imports binding information and an object UUID from a server entry

Synopsis

rpccp import *starting-entry-name* **-i** *if-id* [**-v** *versions*] [**-e**][**-n** [*integer*]] [**-o** *object-uuid*] [**-s** *syntax*] [**-u**]

Options

-i *if-id* Defines an interface identifier to be imported (required). You can import only one interface at a time.

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 ec1eeb60-5943-11c9-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 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.

-e Shows the name of the entry where the binding is found (optional).

-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 may return the same 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 Firectory 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.

Notes

This command was replaced at DCE Version 1.1 by the **dcecp** command and may not be provided in future releases of DCE.

Examples

The following commandd imports an interface and object:

```
rpccp> import -i ec1eeb60-5943-11c9-a309-08002b102989,1.1 \
     -o 30dbeea0-fb6c-11c9-8eea-08002b0f4528 /.:/LandS/anthro/Cal_host_3
```

Related Information

Commands: **rpccp_export(8rpc)**, **rpccp_show_server(8rpc)**, **unexport(8rpc)**.

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 ec1eeb60-5943-11c9-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. he **-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 **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 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 **add entry(8rpc)** reference page.

The **remove element** command requires the entry name of the profile. You must also specify either **-d**, or **-i** and **-m**, or **-a**.

Privileges Required

You need **r** (**read**) and **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 may not be provided in future releases of DCE.

Examples

The following 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 an element from a profile:

```
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
```

Related Information

Commands: **rpccp_add_element(8rpc)**, **rpccp_remove_profile(8rpc)**, **rpccp_show_profile(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 may 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 may 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) may 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 ec1eeb60-5943-11c9-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:

- interface ID, consisting of an interface UUID and versions (major and minor)
- binding information
- object UUID (optional)
- 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 may 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 may not be provided in future releases of DCE.

Examples

The following commands each remove a member from a group:

```
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 may 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)**.

show entry

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 ec1eeb60-5943-11c9-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.

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 may 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 may not be provided in future releases of DCE.

Examples

The following commands show the NSI attributes of name service entries:

```
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)**.

show group

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 may 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:

```
rpccp> show group   /.:/LandS/anthro/Calendar_group
```

2. The following command shows a specific member of a group:

```
rpccp show group -m /.:/LandS/anthro/Cal_host_2 \
  /.:/LandS/anthro/Calendar_group
```

Related Information

Commands: **rpccp_add_member(8rpc)**, **rpccp_remove_group(8rpc)**, **rpccp_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 ec1eeb60-5943-11c9-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 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 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:

```
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:

```
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:

*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 may 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:

```
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:

```
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 ec1eeb60-5943-11c9-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.

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.

Description

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 may not be provided in future releases of DCE.

Examples

1. The following command shows the cell **profile /.:/cell-profile** in the local cell:

```
rpccp 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
rpccp
rpccp> show profile MOLLY_O_PROFILE
```

Related Information

Commands: **rpccp_add_element(8rpc)**, **rpccp_remove_element(8rpc)**, **rpccp_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 may 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

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 Uinversal 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.

-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
```

-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. 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 may 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=ec1eeb60-5943-11c9-a309-08002b102989,1.1
export Calendar_1_1
rpccp
rpccp> unexport -i Calendar_1_1 /.:/LandS/anthro/Cal_host_2
```

Related Information

Commands: **rpccp_export(8rpc)**, **rpccp_import(8rpc)**, **rpccp_show_server(8rpc)**

Chapter 3

Cell Directory Service Commands

cds_intro

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.

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 start-up and shutdown procedures.

Related Information

Commands: **cdsadv(8cds)**, **cdsbrowser(8cds)**, **cdscp(8cds)**, **cdsd(8cds)**, **gdad(8cds)**, **dced(8dce)**.

Books: *DCE 1.2.2 Administration Guide*.

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 *DCE 1.2.2 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 may 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**:

```
cdscp add directory /.:/sales myname = ontario
```

Related Information

Commands: **remove_directory(8cds)**, **show_directory(8cds)**.

Books: *DCE 1.2.2 Administration Guide*.

add object

Purpose

Adds a value to a modifiable, set-valued attribute of an object entry

Synopsis

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 *DCE 1.2.2 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 may 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:

```
    OID          LABEL          SYNTAX
1.3.22.1.3.70   printcap        char
```

Then enter the following command to assign the value **ps** to the attribute **printcap**:

```
cdscp> add object /.:/subsys/deskprinter printcap = ps
```

Related Information

Commands: **create object(8cds)**, **delete object(8cds)**, **list object(8cds)**, **remove object(8cds)**, **set object(8cds)**, **show object(8cds)**.

Books: *DCE 1.2.2 Administration Guide*.

cdsadv

Purpose

Starts the CDS client daemon

Synopsis

cdsadv [**-c** *size*] [**-D**] [**-s**] [**-w** *route*]

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 may necessitate defining a new cached server.

-D For debugging use only. Causes the **cdsadv** process to not fork.

-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.

Example

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)**.

Books: *DCE 1.2.2 Administration Guide*.

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 withn 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**.

Related Information

Books: *DCE 1.2.2 Administration Guide*.

cdsclerk

Purpose

Manages the interface between clients and the CDS server

Synopsis

cdsclerk [**-D**] [**-w** *route*]

Options

-D	For debugging use only. Causes the **cdsadv** clerk process not to fork.
-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**:

```
dce_login hosts/orion/self smith
```

4. Enter the following command to see whether the **dced** process is already running:

```
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. Enter the following command with the appropriate arguments to start the **cdsclerk** process:

```
cdsclerk
```

Related Information

Books: *DCE 1.2.2 Administration Guide*.

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

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 may be fully replaced by the **dcecp** command in a future release of DCE, and may 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

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	Adds 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 *DCE 1.2.2 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 **>&** *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

Books: *DCE 1.2.2 Administration Guide*.

cdsd

Purpose

Starts the CDS server

Synopsis

cdsd [**-a**] [**-D**] [**-l** *principal*] [**-v** *directory version*] [**-w** *route*]

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 **cdsd** 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 **cdsd** to create new directories with the specified CDS directory version number. Currently, 4.0 is the only version supported.

-w *route* Routes serviceability messages.

Description

The **cdsd** 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 **cdsd** server fails to start

automatically after a reboot, or if you want to restart a **cdsd** 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, hierarchical cells, 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 *DCE 1.2.2 Administration Guide—Core Components*.

Example

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)**.

Books: *DCE 1.2.2 Administration Guide*.

clear cached server

Purpose Removes knowledge of a user-defined server from the local clerk's cache

Synopsis **cdscp clear cached server** *simplename*

Arguments

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 may 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

cdscp clear clearinghouse *clearinghouse-name*

Arguments

clearinghouse-name
The full name of the clearinghouse.

Description

The **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 **list clearinghouse** command, the clearinghouse will still be listed.

Before you can delete a cleared clearinghouse, you must use the **create clearinghouse** command to recreate it. After recreating the clearinghouse, you can use the **delete clearinghouse** command to remove it.

This command is part of the process of relocating a clearinghouse. See the *DCE 1.2.2 Administration Guide* for more information.

Privileges Required

You must have **w** (**write**) permission to the server on which the clearinghouse resides.

Notes

This command was replaced at DCE Version 1.1 by the **dcecp** command and may not be provided in future releases of DCE.

Examples

The following command clears the clearinghouse **/.:/Paris2_CH** before moving it to another server:

```
cdscp clear clearinghouse /.:/Paris2_CH
```

Related Information

Commands: **create_clearinghouse(8cds)**, **delete_clearinghouse(8cds)**, **list_clearinghouse(8cds)**, **set_cdscp_preferred_clearinghouse(8cds)**, **show_cdscp_preferred_clearinghouse(8cds)**, **show_clearinghouse(8cds)**.

Books: *DCE 1.2.2 Administration Guide*.

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.

Notes

This command was replaced at DCE Version 1.1 by the **dcecp** command and may 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 may 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 *DCE 1.2.2 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 may 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)**.

Books: *DCE 1.2.2 Administration Guide*.

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 may 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

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 may not be provided in future releases of DCE.

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)
```

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 may 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 may 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)**.

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 *DCE 1.2.2 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.

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 may 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
```

Related Information

Commands: **clear_cached_server(8cds)**, **dump_clerk_cache(8cds)**, **show_cached_server(8cds)**.

Books: *DCE 1.2.2 Application Development Reference*.

delete child

Purpose

Deletes a child pointer from the namespace

Synopsis

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 may 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:

```
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 **cdscp delete clearinghouse** *clearinghouse-name*

Arguments

clearinghouse-name
The full name of the clearinghouse.

Description

The **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 **create clearinghouse** command.

The **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 *DCE 1.2.2 Administration Guide* for more information about handling master replicas when deleting a clearinghouse.

Permissions 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 needs **d** (**delete**) permission to the associated clearinghouse object entry and **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 **dcecp** command and may not be provided in future releases of DCE.

Examples

The following command deletes a clearinghouse named **/.:/sales/Orion_CH** from the local server system:

```
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)**.

Books: *DCE 1.2.2 Administration Guide*.

delete directory

Purpose

Deletes a directory

Synopsis

cdscp delete directory *directory-name*

Arguments

directory-name
The full name of the directory.

Description

The **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 **delete replica** command if you need to remove read-only replicas.

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 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.

Notes

This command was replaced at DCE Version 1.1 by the **dcecp** command and may not be provided in future releases of DCE.

Examples

The following command deletes the directory **/.:/eng** from the namespace:

```
cdscp> delete directory /.:/eng
```

Related Information

Commands: **create_directory(8cds)**, **delete_replica(8cds)**, **list_directory(8cds)**, **set_directory(8cds)**, **set_directory_to_skulk(8cds)**, **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 may 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.

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 may 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.

Privileges 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 may 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

cdscp disable clerk

Description

The **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.

Privileges Required

You must have **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 may be replaced in future releases of DCE by the **dcecp** command, and may no longer be supported at that time.

Examples

The following command stops the clerk on the local server system:

```
cdscp> disable clerk
```

Related Information

Command: **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.

Privileges Required

You must have **w** (**write**) permission to the server.

Notes

This command may be replaced in future DCE releases by the **dcecp** command, and may 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 may 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

gdad [**-D**][**-w** *route*] [**-b** | **-x**]

Options

-D	For debugging use only. Causes the **cdsadv** process not to fork.
-w *route*	Routes serviceability messages.
-b	Disables BIND name resolution.
-x	Disables X.500 name resolution.

Description

The **gdad** command starts the Global Directory Agent (GDA) daemon. The GDA serves as a connection to the global naming environment.

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 GAD 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

Books: *DCE 1.2.2 Administration Guide*.

list child

Purpose

Displays a list of all child pointers matching a specified child name

Synopsis

cdscp list child *child-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.

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 **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 **r** (**read**) permission to the directory that stores the child pointer. 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 child pointers.

Notes

This command was replaced at DCE Version 1.1 by the **dcecp** command and may not be provided in future releases of DCE.

Examples

The following command displays a list of all the child pointers named in the **/.:/sales** directory:

```
cdscp> list child /.:/sales/*
                          LIST
                         CHILD   /.../abc.com/sales
                            AT   1991-10-15-15:56:00
Q1
Q2
Q3
Q4
```

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 may 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
```

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 may 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
                            AT   1991-10-15-15:43:58
atlanta
austin
boston
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 may 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
                          AT   1991-10-15-15:54:38
lnk01
lnk02
lnk03
lnk04
lnk05
lnk06
```

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 may 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
                            AT   1991-10-15-15:53:06
juno
test_stats
work_disk1
work_disk2
```

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 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 *DCE 1.2.2 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 may 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:

```
    OID           LABEL         SYNTAX
1.3.22.1.3.66   dirregion       small
```

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)**.

Books: *DCE 1.2.2 Administration Guide*.

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 may 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 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 *DCE 1.2.2 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 may not be provided in future releases of DCE.

Examples

To remove the value **ps** from the attribute **printcap** of an object entry named **/.:/mlh/deskprinter**, follow these steps:

1. Read the **cds_attributes** file to check that the attribute **printcap** is listed, as shown in the following:

```
    OID          LABEL           SYNTAX
1.3.22.1.3.50    printcap        char
```

2. Enter the following command to remove the value **ps** from the attribute **printcap**:

```
cdscp> remove object /.:/mlh/deskprinter printcap = ps
```

Related Information

Commands: **add_object(8cds)**, **list_object(8cds)**, **set_object(8cds)**, **show_object(8cds)**.

Books: *DCE 1.2.2 Administration Guide*.

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 may be replaced in future DCE releases by the **dcecp** command, and may 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 may 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 *DCE 1.2.2 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 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 may not be provided in future releases of DCE.

Examples

1. The following sets a low convergence value on the **/.:/mfg** directory:

```
cdscp> set directory /.:/mfg CDS_Convergence = low
```

2. The following commands upgrade the directory version on the **/.:/host** directory:

```
dcecp> directory modify /.:/host -add {CDS_UpgradeTO 1.2} -single
dcecp> directory synchronize /.:/host
```

Related Information

Commands: **create_directory(8cds)**, **delete_directory(8cds)**, **list_directory(8cds)**, **remove_directory(8cds)**, **set_directory_to_skulk(8cds)**, **show_directory(8cds)**.

Books: *DCE 1.2.2 Administration Guide*.

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 may be replaced in future DCE releases by the **dcecp** command, and may no longer be supported at that time.

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 may 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 may not be provided in future releases of DCE.

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**:

```
cdscp> 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

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 *DCE 1.2.2 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 may 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
```

Related Information

Commands: **add_object(8cds)**, **create_object(8cds)**, **delete_object(8cds)**, **list_object(8cds)**, **remove_object(8cds)**, **show_object(8cds)**.

Books: *DCE 1.2.2 Administration Guide*.

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 may 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
        CACHED CLEARINGHOUSE   /.../abc.com/Paris2_CH
                           AT   1991-10-15-15:58:09
                 Creation Time = 1991-10-01-17:03:32.32
              Read Operations = 412
             Write Operations = 618
    Miscellaneous Operations = 278
```

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 may 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
           CACHED NAMESERVER   emv_udp
                            AT   1991-10-15-15:56:56
                          Name = /.../emv.abc.com
                         Tower = ncadg_ip_udp:14.20.14.32
                         Tower = ncacn_ip_tcp:14.20.14.32
                          SHOW
           CACHED NAMESERVER   emv_tcp
                            AT   1991-10-15-15:56:57
                          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 may be replaced in future DCE releases by the **dcecp** command, and may 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

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 may 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 *DCE 1.2.2 Administration Guide*.

Privileges Required

You must have **r** (**read**) permission to the cell root directory.

Notes

This command may be replaced in future DCE releases by the **dcecp** command, and may 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
                          AT   1991-10-15-15:58:25
              Namespace Uuid = 2d2d50ad-8b1a-11ba-8983-08002b0f79aa
          Clearinghouse Uuid = 2ab024a8-8b1a-11ba-8983-08002b0f79aa
          Clearinghouse Name = /.../abc.com/NY_CH
                Replica Type = Master
                    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
```

Related Information

Books: *DCE 1.2.2 Administration Guide*.

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.

Notes

This command was replaced at DCE Version 1.1 by the **dcecp** command and may 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
                      AT   1991-10-15-15:56:01
                 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.

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.

Notes

This command was replaced at DCE Version 1.1 by the **dcecp** command and may 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 = 1991-10-21-13:04:04.000000009/08-00-2b-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 may be replaced in future DCE releases by the **dcecp** command, and may no longer be supported at that time.

Examples

The following command displays the attributes of the clerk on the local system:

```
cdscp> show clerk
                    SHOW
                   CLERK
                      AT   1991-10-15-15:56:50
           Creation Time = 1991-10-15-15:38:19.000000051-04:00I0.000000000
 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.

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.

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 may 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
                      AT   1991-10-15-15:43:59
       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_Replicas = :
   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/Paris_CH
           CDS_AllUpTo = 1991-10-17-08:51:18.000000032/08-00-2b-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 = 1991-10-17-08:51:18.000000032/08-00-2b-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 may 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
                      AT   1991-10-15-15:54:40
                 CDS_CTS = 1991-10-15-19:54:35.00000003/08-00-2b-1c-8f-1f
                 CDS_UTS = 1991-10-15-19:54:35.00000006/08-00-2b-1c-8f-1f
          CDS_LinkTarget = /.../abc.com/sales/service

                    SHOW
                SOFTLINK   /.../abc.com/sales/region1
                      AT   1991-10-15-15:54:41
                 CDS_CTS = 1991-10-15-19:54:36.00000077/08-00-2b-1c-8f-1f
                 CDS_UTS = 1991-10-15-19:54:36.00000009/08-00-2b-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

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.

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 may 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
     AT   1991-10-15-15:53:07
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
```

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**.

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 may 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
                      AT   1991-10-15-15:55:29
       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_Replicas = :
   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_Replicas = :
   Clearinghouse's UUID = 49757f28-8b1a-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 = :
             Expiration = 1991-10-15-19:55:18.711
              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
```

Related Information

Commands: **create_replica(8cds)**, **delete_replica(8cds)**.

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 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 may be replaced in future releases by the **dcecp** command, and may 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
                            AT   1991-10-15-15:56:47
                 Creation Time = 1991-10-15-15:39:35.35
              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)**.

Chapter 4
Distributed Time Service Commands

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)**.

Books: *DCE 1.2.2 Administration Guide*, *DCE 1.2.2 Command Reference*.

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 may 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 may 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 1990-11-30-10:58:00.000-05:00I0.000
```

Related Information

Commands: **dtscp(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 may 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 **dce_config** 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 may not be provided in future releases of DCE.

Examples

```
dtscp> delete
```

Related Information

Commands: **disable(8dts)**, **dtscp(8dts**, **dce_config(8dce)**.

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 may 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 may be fully replaced by the **dcecp** command in a future release of DCE, and may no longer be supported at that time.

- **exit**
- **help**
- **quit**

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 *DCE 1.2.2 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.
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
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 may 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
       /     /           /
 Command  [Argument]  Variable
           --------
          [Attribute]
```

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:

```
1993-03-16-14:29:47.52000-05:00I000.003
```

The timestamp uses the DTS format as explained in the *DCE 1.2.2 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)**.

Books: *DCE 1.2.2 Administration Guide*.

dtsd

Purpose

Restarts the DTS daemon

Synopsis

dtsd [**-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 **dtsd** as a server. Default is backup, courier, local server

-g Runs **dtsd** as a global server.

-k courier | **noncourier**
Runs **dtsd** as a courier or a noncourier.

-g Runs **dtsd** 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 **dtsd** as a clerk.

Description

The **dtsd** 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, **rc.dce**.

You can enter the command manually under the following conditions:

- If a DTS daemon fails to start automatically upon reboot
- If you want to restart a daemon that you shut down to perform a backup or do diagnostic work

In normal rebooting, the **rc.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 **dtsd**.

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 **rc.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)**.

Books: *DCE 1.2.2 Administration Guide*.

dtsdate

Purpose

Sets a local clock from a remote DTS daemon server host

Synopsis

dtsdate [**-q**][**-s**][**-u**]*remote_host* [*nsecs*]

Options

-q Queries the difference in time between the local host and the remote host, but does not change the local clock. The returned result (2 if the time would have been reset, 1 if there was an error, and 0 otherwise) can be used by a script to determine what action to take.

-s Causes **dtsdate** to work silently, without showing the time.

-u Shows the time in Universal Time Coordinated (UTC) format, rather than in the current time zone.

Arguments

remote_host The name or the IP address of a remote host that has a **dtsd** server.

nsecs An integer giving the number of seconds by which the remote and local host times can differ, without the local host's clock being reset. If *nsecs* is 0, or if it is not specified, it is treated as if it were extremely large, and no resetting occurs.

Description

The **dtsdate** command sets the local clock of a system to be the same as the host *remote_host*, running a **dtsd** 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.

Notes

The remote host must be running as a Distributed Time Service (DTS) server. This means that the **dtsd** 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 tolerence."
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

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 may 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 may 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 may be replaced in future DCE releases by the **dcecp** command, and may 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

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:

- **advertise**
- **change**
- **create**
- **delete**
- **disable**
- **enable**
- **set**
- **show**
- **synchronize**
- **unadvertise**
- **update**

Description

The **help** command displays information about **dtscp** commands.

Notes

This command may be replaced in future DCE releases by the **dcecp** command, and may 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 may be replaced in future DCE releases by the **dcecp** command, and may 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)**.

set

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 charactistics you specify for the Distributed Time Service (DTS) entity. 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 attemps** 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; *hostname* represents the name of the system or node that is the server's client. The default setting is the recommended value.

Default: **/.:/hosts/***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 [*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/***hostname***/self**

servers required [*integer*]
Specifies the minimum number of servers required for a synchronization. Settings of 1 or 2 may cause unreliable computed times.

Default: **1** (clerks) **3** (servers)

Value: **1**–**10**

synchronization hold down [*relative-time*]
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: **0-00:10:00.000**

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 may 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)**.

show

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:

- **all**
- **all characteristics**
- **all counters**
- **all status**
- **global servers**
- **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 deepending 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**

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 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 may 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 may 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+00:00I0.809
       Last Observed Time    = 1993-10-15-21:03:09.041+00:00I—–
       Last Observed Skew          = +0-00:01:22.917I—–
       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+00:00I0.809
       Last Observed Time    = 1993-10-15-21:01:46.143+00:00I0.817
       Last Observed Skew          = +0-00:00:00.019I1.625
       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+00:00I0.767
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+00:00I-----
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

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 may 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

dtscp unadvertise

Description

The **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 **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 may not be provided in future releases of DCE.

Examples

```
dtscp> unadvertise
```

Related Information

Commands: **dtscp(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 may 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)**.

Chapter 5
Security Service Files and Commands

sec_intro

Purpose

Introduction to the DCE Security Service administrative files

Description

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.

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)**.

Books: *DCE 1.2.2 Administration Guide*, *DCE 1.2.2 Application Development Guide*, *DCE 1.2.2 Application Development Reference*.

aud_audit_events

Purpose

Auditable events for the audit services

Description

The DCE Security Service supports the auditing of audit service-significant events. Among these events are:

- Administrative operations

  These are subdivided into **modify** and **query** operations.

- 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 storage strategy. 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.

EVT_MODIFY_STATE (0x306, dce_audit_admin_modify)
Event-specific information: None.

EVT_MODIFY_SSTRATEGY (0x305, dce_audit_admin_modify)
Event-specific information: None.

EVT_REWIND (0x307, dce_audit_admin_modify)
Event-specific information: None.

EVT_STOP (0x308, dce_audit_admin_modify)
Event-specific information: None.

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)
Event-specific information: None.

EVT_SHOW_STATE (0x30a, dce_audit_admin_query)
Event-specific information: None.

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)
Event-specific information: None.

EVT_DELETE_FILTER (0x300, dce_audit_filter_modify)
Event-specific information: None.

EVT_REMOVE_FILTER (0x304, dce_audit_filter_modify)
Event-specific information: None.

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)
Event-specific information: None.

EVT_SHOW_FILTER (0x301, dce_audit_filter_query)
Event-specific information:

aud_c_evt_info_long_int esl_type
aud_c_evt_info_char_string subject_name

Related Information

Commands: **dcecp(8dce)**.

Files: **event_class(5sec)**.

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:

- Time service processes
- Clock readings
- Global-set membership (in the cell service profile)
- 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)
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)
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)
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)
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)
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)
Event-specific information:

signed32 old_epoch
signed32 new_epoch
utc_t old_time
utc_t new_time

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)
Event-specific information:

signed32 setClock

The **AdvertiseCm( )** 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)
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)
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)
Event-specific information:

byte useDefault
signed32 attribute

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_modif)
Event-specific information:

signed32 attribute
signed32 attribute_type

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)
Event-specific information:

signed32 attribute
signed32 attribute_type

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)
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)
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)
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)
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 (0x210, dce_dts_mgt_query)
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 may 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)
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)
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
0x00000100
# TGS_TicketReq
0x00000101
# TGS_RenewReq
0x00000102
# TGS_ValidateReq
0x00000103
```

Related Information

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

group_override

Purpose

Registry database group override file

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 may 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.

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 enter either the word **OMIT** or another invalid password string (such as an asterisk or **NO GOOD**) in the *passwd* field for a particular group, users will not be able to issue a **newgrp** command specifying this group on the local machine. If you specify **OMIT** and run **passwd_export** with the **-x** option, the named group will not appear in the **/etc/group** file produced by **passwd_export**. You should be aware that if you have omitted groups from the **/etc/group** file, information about those groups will not be available to any programs that use the group file. For example, the **ls -lg** command reads the group file to obtain further information about a group; if no group entry exists in the group file, no information is be available. For this reason you should use **OMIT** to omit groups from the **/etc/group** file only if your user community is very large and either of the following conditions occur:

- The **group** file is taking up too much space.

- Group-ID-to-name mapping is too slow (during **ls -lg**, for example).

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
```

3. To prevent users from invoking **newgrp** to change their primary group identification to the group **typists**, and to omit this group from the local group file, put**OMIT** in the *passwd* field as follows:

```
typists:OMIT::
```

   Then run the **passwd_export** command with the **-x** option to omit the group **typists** from the **/etc/passwd** file, as follows:

```
dcelocal/etc/passwd_export -x
```

Related Information

Commands: **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 may find **passwd_override** especially useful for excluding users from logging into certain machines, establishing local root passwords, or tailoring local user environments.

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 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

An encrypted principal UNIX 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 log-in 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 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:

- The **passwd** file is taking up too much space.
- User-ID-to-name mapping is too slow (during **ls -l**, for example).

Notes

Principals can update their entries in the override file for the local host by using **chpass**. The **chpass** command is platform-specific; consult your local operating system documentation for information on how to use your version of the command.

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: **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:

- Attempts to invoke authentication server/ticket-granting server/privilege server (AS/TGS/PS) operations.
- Deletion of security server objects, including the following:
  - access control lists (ACLs)
  - accounts
  - pgo items
  - registry properties
  - registry/organization policies
  - registry master key
- Attempts to invoke an operation that modifies security server objects or updates an ACL.
- Attempts to invoke operations that involve access control.
- Failed client responses to the server's challenge, detected replays and invalid ticket requests.
- The use of cryptographic keys in the remote procedure call (RPC) runtime.
- 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**, **rdaclif**, **rdacliftmp**, **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.

Event Type (Event Number, Event Classes)
AS_Request (0x101, dce_sec_authent)

Event-Specific Information
None

Event Type (Event Number, Event Classes)
TGS_TicketReq (0x102, dce_sec_authent)

Event-Specific Information
None

Event Type (Event Number, Event Classes)
TGS_RenewReq (0x103, dce_sec_authent)

Event-Specific Information
None

Event Type (Event Number, Event Classes)
TGS_ValidateReq (0x104, dce_sec_authent)

Event-Specific Information
None

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)

Event-Specific Information

```
char          *component_name
uuid_t         manager_type
sec_acl_type_t acl_type
```

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)

Event-Specific Information

```
char           *component_name
uuid_t          manager_type
sec_acl_type_t    acl_type
sec_acl_list_t  old_acl_list
sec_acl_list_t  new_acl_list
```

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)

Event-Specific Information

```
char           *component_name
uuid_t          manager_type
```

sec_acl_permset_t net_rights

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)

Event-Specific Information

char *component_name
uuid_t q manager_type
sec_acl_permset_t desired_permset

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)

Event-Specific Information

char *component_name
sec_acl_type_t acl_type

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 may 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)

Event-Specific Information

```
char            *component_name
uuid_t           manager_type
sec_acl_type_t   sec_acl_type
```

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 may 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)

Event-Specific Information

```
char  *string client_address
unsigned16      num_groups /* Number of local groups in PAC */
uuid_t          groups  /* num_groups local groups in 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)

Event-Specific Information

```
char                 *login_name
sec_rgy_acct_key_t   key_parts
```

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)

Event-Specific Information

```
char    *login_name
```

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**).

Event Type (Event Number, Event Classes)

ACCT_Rename (0x10F, dce_sec_control, dce_sec_modify)

Event-Specific Information

```
char    *old_login_name
char    *new_login_name
```

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)

Event-Specific Information

```
char    *login_name
```

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)

Event-Specific Information

```
char          *login_name
unsigned32    key_parts
```

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

```
char    login_name
```

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)

Event-Specific Information

```
char    *login_name
```

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)

Event-Specific Information

```
sec_rgy_domain_t      name_domain
char                  *pgo_name
```

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)

Event-Specific Information

```
sec_rgy_domain_t      name_domain
char                  *pgo_name
```

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)

Event-Specific Information

```
sec_rgy_domain_t        name_domain
char                    *pgo_name
```

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)

```
sec_rgy_domain_t        name_domain
char                    *old_name
char                    *new_name
```

The **rs_pgo_get( )** operation returns the name and data for a PGO item. The desired item is identified by a query key, which can be a **name**, a **uuid**, a **unix_num**, or a **sequential-search** flag. The caller needs to have the **r** (**read**) permission on the PGO item to be viewed.

Event Type (Event Number, Event Classes)

PGO_Get (0x118, dce_sec_control, dce_sec_query)

Event-Specific Information

```
sec_rgy_domain_t    name_domain
rs_pgo_query_key_t key      /* The query key and one of the    */
                            /* following depending on the      */
                            /* query key specified:            */
  case (key == rs_pgo_query_name)
       char *name           /* Name of the item being searched */
  case (key == rs_pgo_query_id)
       uuid_t id_key        /* uuid of the item being searched */
```

```
case (key == rs_pgo_query_unix_num)
    unsigned32 unix_num     /* unix_num of item being searched */
case (key == rs_pgo_query_nex)
    char *scope             /* Scope of item being searched    */
```

The **rs_pgo_key_transfer( )** operation performs a specified key transfer between the **uuid**, **unix_num**, and **name** of a PGO item. The caller needs to have some permission on the PGO item for **id->name** and **unix_num->name** transfers.

Event Type (Event Number, Event Classes)

PGO_KeyTransfer (0x119, dce_sec_control)

Event-Specific Information

```
sec_rgy_domain_t name_domain
rs_pgo_query_key_t   key     /* The query key                  */
                             /* One of the following, depending  */
                             /* on the query key specified:      */
  case (key == rs_pgo_query_name)
     char *name              /* Name of the item being searched  */
  case (key == rs_pgo_query_id)
     uuid_t id_key  /* uuid of the item being searched  */
  case (key == rs_pgo_query_unix_num)
     unsigned32 unix_num /* unix_num of item  being searched */
     unsigned32  requested_result_type
```

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)

Event-Specific Information

```
sec_rgy_domain_t name_domain
char    *person_name /* Principal's name */
char    *go_name     /* Group or org's name */
```

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)

Event-Specific Information

```
sec_rgy_domain_t name_domain
char    *person_name /* Principal's name */
char    *go_name /* Group or org's name */
```

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)

Event-Specific Information

```
sec_rgy_domain_t name_domain
char    *person_name /* Principal's name */
char    *go_name /* Group or org's name */
```

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)

Event-Specific Information

```
sec_rgy_domain_t name_domain
char    *go_name /* PGO's uuid */
```

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)

Event-Specific Information
None

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)

Event-Specific Information
None

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)

Event-Specific Information

char *organization

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)

Event-Specific Information

char *organization

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)

Event-Specific Information

char *account

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)

Event-Specific Information

char *account

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)

Event-Specific Information
None

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)

Event-Specific Information

boolean in_maintenance

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)

Event-Specific Information
None

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)

Event-Specific Information
None

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)

Event-Specific Information

```
char  *rep_id_str
```

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)

Event-Specific Information

```
char  * component_name
unsigned32 int num_to_write
uuid  in_attrs[num_to_write].attr_id
```

The **rs_attr_delete( )** operation deletes a specified attribute(s). The caller must have **delete_permset** permission for each attribute specified.

Event Type (Event Classes)

ERA_Delete (0x12C, dce_sec_control, dce_sec_modify)

Event-Specific Information

```
char   * component_name
unsigned32   num_to_delete
uuid    attrs[num_to_delete].attr_id
```

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)

Event-Specific Information

```
char   * component_name
unsigned32   int num_attr_keys
uuid    attr_keys[num_attr_keys].attr_id
```

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, **q** (**query_permset**) permission on the registry object specified.

Event Type (Event Classes)

ERA_LookupNoExpand (0x12F, dce_sec_control)

Event-Specific Information

```
char   * component_name
unsigned32   int num_attr_keys
uuid    attr_keys[num_attr_keys].attr_id
```

The **rs_attr_lookup_by_name( )** operation performs a lookup of an attribute by name. The caller must have, for the attribute specified, **q** (**query_permset**) permission on the registry object specified.

Event Type (Event Classes)

ERA_LookupByName (0x130, dce_sec_control)

Event-Specific Information

char * component_name
char * attr_name

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)

Event-Specific Information

char * schema_name
char * schema_entry->attr_name
uuid schema_entry->attr_id

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)

Event-Specific Information

char *schema_name
uuid attr_id

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)

Event-Specific Information

char * schema_name
uuid schema_entry->attr_id

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)

Event-Specific Information

char * schema_name
uuid attr_id

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)

Event-Specific Information

char * schema_name
char * attr_name

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)

Event-Specific Information

```
char * request_location  /* "LOCAL" or "INTERCELL" */

if "LOCAL" request:
uuid req_princ_id->uuid; /* requested local principal uuid */
uuid req_group_id->uuid; /* requested local primary group uuid */
unsigned short int num_groups /* number of valid local groups */
uuid = groups[num_groups].uuid /* valid local groups' uuids */

if "INTERCELL" request:
unsigned short int num_epacs /* number of epacs in delegation chain */
uuid [num_epacs].pa.realm.uuid /* privilege attribute realm uuid */
uuid [num_epacs].pa.principal.uuid /* privilege attribute principal uuid */
uuid [num_epacs].pa.num_groups /* number of groups in privilege attribute */
uuid [num_epacs].pa.groups[([epac_set.num_epacs].pa.num_groups)].uuid
   /* uuids for groups in privilege attribute */
```

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)

Event-Specific Information

```
uuid req_princ_id->uuid;  /* requested local principal uuid */
uuid req_group_id->uuid;  /* requested local primary group uuid */
unsigned short int num_groups /* number of valid local groups */
uuid = groups[num_groups].uuid /* valid local groups' uuids */
unsigned short int num_epacs /* number of epacs in delegation chain */
uuid [num_epacs].pa.realm.uuid /* privilege attribute realm uuid */
uuid [num_epacs].pa.principal.uuid /* privilege attribute principal uuid */
uuid [num_epacs].pa.num_groups /* number of groups in privilege attribute */
uuid [num_epacs].pa.groups[([epac_set.num_epacs].pa.num_groups)].uuid
   /* uuids for groups in privilege attribute */
```

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 initator'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)

Event-Specific Information

```
uuid req_princ_id->uuid;  /* requested local principal uuid */
uuid req_group_id->uuid; /* requested local primary group uuid */
unsigned short int num_groups /* number of valid local groups */
uuid = groups[num_groups].uuid /* valid local groups' uuids */
unsigned short int num_epacs /* number of epacs in delegation chain */
uuid [num_epacs].pa.realm.uuid /* privilege attribute realm uuid */
uuid [num_epacs].pa.principal.uuid /* privilege attribute principal uuid */
uuid [num_epacs].pa.num_groups /* number of groups in privilege attribute */
uuid [num_epacs].pa.groups[([epac_set.num_epacs].pa.num_groups)].uuid
   /* uuids for groups in privilege attribute */
```

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 *DCE 1.2.2 Application Development Guide—Core Components* and the **sec_key_mgmt_*(3sec)** reference pages.

Related Information

Commands: **rgy_edit(8sec)**.

Books: *DCE 1.2.2 Application Development Guide—Core Components*, *DCE 1.2.2 Application Development Reference*.

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.

chpass Changes user information, such as login name, password, home directory, password and account expiration dates, and login shell. The implementation of this utility is platform-specific. Use the **chpass** utility supplied by your platform vendor for changing user information.

dce_login Validates a principal's identity and obtains a principal's network credentials. This command is used primarily during DCE configuration. Use the **login** utility supplied by your platform vendor for user login.

kdestroy Destroys your login context and credentials.

kinit Obtains and caches a ticket granting ticket.

klist Lists cached tickets.

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.

su
: Allows you to assume another user's identity. The implementation of this utility is platform-specific. Use the **su** utility supplied by your platform vendor.

See each command's reference page for further information.

Related Information

Commands: **acl_edit(8sec)**, **auditd(8sec)**, **chpass(8sec)**, **dce_login(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)**, **su(8sec)**.

Books: *DCE 1.2.2 Administration Guide*, *DCE 1.2.2 Application Development Guide*, *DCE 1.2.2 Application Development Reference*.

acl_edit

Purpose

Edits or lists an object's ACLs

Synopsis

acl_edit {[**-e** *pathname*] | [**-addr** *string_binding component_name*]} [**-ic** | **-io**][**-n** | **-c**][*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.

-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 wheter 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.

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 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 may be fully replaced by the **dcecp** command in a future release of DCE, and may no longer be supported at that time.

- **abort**
- **commit**
- **exit**
- **help**
- **test access**

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 *DCE 1.2.2 Application Development Reference*.

The **acl_edit** command automatically binds to the server of the object specified, and then communicates (through the standard DCE ACL interface) with that server's ACL manager in response to user input.

Exactly what the object specified is depends partly on whether or not the **−e** option is specified. Specifying **−e** means that you want to operate on the directory service ACL—in other words, you want **acl_edit** to bind to the Cell Directory Service (CDS) server and allow you to operate on the ACL maintained by that server on the object's directory entry. If, on the hand, you want to access the ACL on the object to which the directory entry refers, then simply omit the **−e** option. The result will be that **acl_edit** will bind to that object's server (the server must, of course, implement an ACL manager), giving you access to the object's ACL.

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* Sets the cell name to be associated with the ACL. 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[**mmit**] Saves all changes to the ACL without exiting.

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** displays a list of all commands and their functions. If you enter **help** and a command (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[**rge**] 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.

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.

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_byte***.***data*

where:

uuid Identifies the type extended ACL entry. (This Unierversal Unique Identifier, or 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.

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.

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, and 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**[**ermissions**] 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:rwx
```

```
sec_acl_edit> m other_obj:rwx
```

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
   ```

   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 **l**[**ist**] 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

auditd [**-t** *trail_file*] [**-a**][**-s** *size*] [**-wrap**][**-w** *svc_route*] [**-d** *debug_level*]

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 the audit daemon's control interface access.

-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 1,esl.9
```

Related Information

Commands: **aud(8dce)**, **audevents(8dce)**, **audfilter(8dce)**, **audtrail(8dce)**, **dcecp(8dce)**.

Files: **svcroute(5dce)**.

chpass

Purpose

Changes user database information

Platform Specific

This command is platform-specific. Consult your local operating system documentation for information on how to use your version of the **chpass** command.

dce_login

Purpose

Validates a principal's identity and obtains the principal's network credentials

Synopsis

dce_login [*principal_name*] [*password*] [**-c** | **-k** *keytable*] [**-r**][**-e** [xec] *cmd_string*]

Options

-c Causes the principal's identity to be certified. If you do not specify **–c**, the principal's identity is validated only.

-k *keytable*
Retrieves the authentication key from *keytable*.

-r Refreshes and validates the current login ID.

-e[xec] *cmd_string*
Executes the command supplied as *cmd_string***.**

Arguments

principal_name
The name of the principal to log in as.

password The password for *principal_name*.

Description

The **dce_login** command is supplied for use in DCE configuration. It validates a principal's identity and obtains the principal's network credentials.

If the **–c** option is supplied, the command also certifies the principal's identity, and, if the principal is able to be certified, creates an entry for the principal in the machine's local registry. If the principal is not able to be certified, the command attempts to log the principal in via the local registry.

The **-e** (or **-exec**) option executes the command specified by *cmd_string* after login. If *cmd_string* is specified without a full pathname, the path prefix is obtained by searching the directories according to the **PATH** variable.

If you do not supply the name of the principal to validate, either on the command line with the *principal_name* argument or through the **-r** option that retrieves the principal name from the current login context, **dce_login** prompts for the principal name. If you do not supply the principal's authentication key either on the command line with *password* argument or through the **-k** option that retrieves the principal authentication key from the specified keytable, **dce_login** prompts for the password.

If you supply the principal name and password on the command line, you must specify the principal name first, followed by the password. If you supply the *principal_name* argument and the **-r** option, the named principal must be the principal of the current network identity.

The **dce_login** command executes the shell specified in the **SHELL** environment variable.

Note that if the clocks on the security server and client machines are not synchronized to within 2 or 3 minutes of each other, you may receive a password validation error and be unable to be validated.

k5dcelogin

Purpose Promotes a principal's Kerberos V5 credentials to DCE credentials

Synopsis **k5dcelogin** *username cmd* [*cmd_parameter*]

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.

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

rlogind(8sec), **rshd(8sec)**.

k5login

Purpose

Contains names of Kerberos principals allowed to access the host with the user ID of the **.k5login** file owner.

Description

The **.k5login** file, which resides in a user's home directory, contains a list of Kerberos principals. Any of the listed principals with valid Kerberos tickets are allowed host access with the user ID of the user in whose home directory the file resides. One common use of the **.k5login** file is to grant system administrators remote root access to the host via Kerberos by placing the file in root's home directory.

Examples

Suppose the user **janedoe** has a **.k5login** file that contains the following lines in her home directory:

```
johndoe@FUBAR.ORG
```

This line allows the user **johndoe@FUBAR.ORG** to use Kerberos network applications, such as **rlogin(8sec)** and **rsh(8sec)**, and to access **janedoe**'s account using his own Kerberos tickets. Note that because **johndoe** retains his own Kerberos tickets, he does not have any privileges that require **janedoe**'s tickets, such as root access to any of her site's hosts, or the ability to change her password.

Suppose **janedoe** and **joeadmin** are system administrators. If they are listed in root's **.k5login** file on each host, they can log in to the hosts using their Kerberos tickets instead of having to type the root password.

See Also

rlogin(8sec), **rlogind(8sec)**, **rsh(8sec)**, **rshd(8sec)**.

kdestroy

Purpose

Destroys a principal's login context and associated credentials

Synopsis

kdestroy [**-c** *cache_name*]

Options

-c *cache_name*
: Specifies that the login context and associated credentials for *cache_name* should be destroyed instead of the default cache.

Description

The **kdestroy** command destroys a principal's login context and the principal's credentials. Until the credentials are reestablished by executing either the **dce_login** command or the **kinit** command, the principal and any processes created by the principal will be limited to unauthenticated access.

Files

/tmp/krb5cc_*unix_id*
: The default credentials cache if the **KRB5CCNAME** environment variable is set, where *unix_id* is the decimal UNIX ID of the user.

Related Information

Commands: **kinit(8sec)**, **klist(8sec)**.

kinit

Purpose

granting ticket

Synopsis

kinit [**-c** *cachename*] [**-f**][**-l** *lifetime*] [**-p**][**-r** *lifetime*] [**-v**][*principal*]

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 may 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.

-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*.

-v Specifies verbose mode.

Arguments

principal The *principal* argument 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.

Description

The **kinit** command can be used to refresh a DCE credentials cache. When you invoke **kinit**, it prompts for your password.

The ticket lifetime and renewable lifetime are specified by concatenating pairs of integers and unit specifiers, as required. The number of units precedes the unit specifier. The unit specifiers are as follows:

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

Files

/tmp/krb5cc_*unix_id*

If the **KRB5CCNAME** environment variable is not set, the name of the file is in the form shown, where *unix_id* is the decimal UNIX ID of

the user. If the **KRB5CCNAME** environment variable is set, its setting determines the name of the file.

Related Information

Commands: **kdestroy(8sec)**, **klist(8sec)**.

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.

-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, or in the cache identified by *cachename* if the **–c** option is used.

Files

/tmp/krb5cc_*unix_id*

If the **KRB5CCNAME** environment variable is not set, the name of the default credentials cache is in the form shown, where *unix_id* is the decimal UNIX ID of the user. If the **KRB5CCNAME** environment variable is set, its value is used to name the default cache

Related Information

Commands: **kdestroy(8sec)**, **kinit(8sec)**.

passwd_export

Purpose

Creates local password and group files

Synopsis

passwd_export [**-d** *dir_name*] [**-h**][**-m** *max_entries*] [**-n**][**-s**][**-v**][**-x**]

Options

-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.

-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.

Description

The *dceshared*/**bin/passwd_export** command creates local password and group files from registry data. These files are used when the network registry is unavailable and by programs that use the original UNIX passwd and group interfaces instead of the DCE interfaces. 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**. 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.

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.

Related Information

Commands: **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

passwd_import [**-c**]**-d** *path* [**-h**][**-i**][**-o** *org*] [**-p** *password*] [**-u** *username*] [**-v**]

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. 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:

- Attach DCE node(s) to an existing UNIX network
- Attach UNIX node(s) to a DCE network
- 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.

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 addtional 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:

- 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**.
- 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**.
- 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 may be more efficient to manually edit either the registry or the group and password files to resolve some of them before you run **import_passwd**.

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: **rgy_edit(8sec)**, **sec_admin(8sec)**, **secd(8sec)**.

pwd_strengthd

Purpose

The sample password management server

Synopsis

pwd_strengthd [+/–**all**] [+/–**alp**] [**-c** *size*] [**-d**] [**-m** *pwd_min_len*] [**-t** *minutes*] [**-v**]

Options

+all Allows passwords to be all spaces. If this option is not set, the effective registry policy is used.

-all Prevents passwords from being all spaces. If this option is not set, the effective registry policy is used.

+alp Allows passwords to consist of only alphanumeric characters. If this option is not set, the effective registry policy is used.

-alp Prevents passwords from consisting of only alphanumeric characters. If this option is not set, the effective registry policy is used.

-c *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.

-d Runs **pwd_strengthd** in the foreground. Log messages are written to standard output.

-m *pwd_min_len*
Specifies the minimum length of a password. If this option is not set, the effective registry policy is used.

-t *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.

-v Runs in verbose mode. More detailed messages are sent to the log file *dcelocal*/**var/security/pwd_strengthd.log**. (Use of this option is recommended.)

Description

The **pwd_strengthd** command is a sample password management server. It exports the **rsec_pwd_mgmt** application programming interface.

The **pwd_strengthd** command generates passwords and 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 **+/–alp**, **+/–all**, and **-m**.

Administrators can subject principals to **password-strength** and **-generation** policies by attaching the following extended registry attributes (ERAs):

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.

See the *DCE 1.2.2 Administration Guide—Core Components* for more information and examples.

Notes

You may want to enhance **pwd_strengthd** to support your site's policies for password strength and generation.

Related Information

Commands: **passwd_export(8sec)**, **passwd_import(8sec)**.

rgy_edit

Purpose

Edits the registry database

Synopsis

rgy_edit [[[**-a** | **-p** | **-g** | **-o**][**-s** *name*] [**-up**][**-v** [**-f**][*name* | *unix__num*] [**-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 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_num* 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 may be fully replaced by the **dcecp** command in a future release of DCE, and may no longer be supported at that time.

- **defaults**
- **delete**
- **domain**
- **exit**
- **help**
- **purge**
- **quit**
- **scope**
- **view**

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 may 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 dashs 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).

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.

Use the **-al** option to create an alias for an existing principal or group. No two principals or groups can have the same UNIX number, but a principal or group and all its aliases share the same UNIX number. The **-al** option creates an alias name for a

principal or group and assigns the alias name the same UNIX number as the principal or group.

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.

c[**hange**] [*principal_name* [**-n** *name*] [**-f** *fullname*] [**-al** | **-pr**] [**-q** *quota*]]
c[**hange**] [*group_name* [**-n** *name*] [**-f** *fullname*] [**-nl** | **-l**]] [**-al** | **-pr**]
c[**hange**] [*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.

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.

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**] *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 principals.

adopt *uuid principal_name* [**-u** *unix_number*] [**-f** *fullname*] [**-q** *quota*]
adopt *uuid group_name* [**-f** *fullname*] [**-nl**]
adopt *uuid organization_name* [**-f** *fullname*]

Creates a principal, group, or organization for the specified Universal Unique Identifier (UUID).

The principal, group, or organization is created to adopt an orphan object. Orphans are registry objects that cannot be accessed because 1) they are owned by UUIDs that are not associated with a principal or group and 2) no other principal, group, or organization has access rights to the orphaned object. UUIDs are associated with all registry objects when the object is created. When the registry object is deleted, the association between the object and the UUID is also deleted.

The *principal_name*, *group_name*, or *organization_name* you specify must be unique in the registry as it must be when you create a principal, group, or organization using the **add** subcommand. Except for the manner in which it is created, the principal,

group, or organization created by the **adopt** subcommand is no different from any other principal, group, or organization.

The *uuid* option specifies the UUID number to be assigned to the principal, group, or organization. The UUID supplied must be the one that owns the orphaned object. Specify the *uuid* in RPC print string format as 8 hexadecimal digits, a dash; 4 hexadecimal digits, a dash; 4 hexadecimal digits, a dash; 4 hexadecimal digits, a dash; and 12 hexadecimal digits. The format is as follows:

nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn

For cell principals only, the **-u** option specifies the UNIX number to be associated with the cell name. If you do not enter this option, the next sequential UNIX number is supplied as a default. For all principals other than cells, the UNIX number is extracted from information embedded in the principal's UUID and cannot be specified here.

For principals, the **-q** option specifies the principal's object creation quota. If you do not enter the option, the object creation quota is set to **unlimited**.

For groups, the **-nl** option turns off the project list inclusion property so that groups are not included in project lists. If you do not enter this option, the group is included in project lists.

For principals, groups, and organizations, the **-f** option supplies the object's full name. If you do not enter the **-f** option, fullname defaults to blank.

An error occurs if you specify a name or UNIX number that is already defined within the same domain of the database.

Note: In the current implementation of DCE, UNIX numbers are embedded in UUID numbers. If you try to create a group or organization to adopt an orphaned object and fail, it could be because the embedded UNIX number is invalid because it does not fall within the range of valid UNIX numbers set for the cell as a registry property. If this is the case, you must reset the range of valid UNIX numbers to include the UNIX number embedded in the UUID and then try again to adopt the object.

Account Subcommands

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:

- Principal, group, and organization name
- Encrypted password
- Miscellaneous information
- Home directory
- Login shell

With **-f**, **view** displays the full entry, including the administrative fields as well as the user fields. Administrative information includes:

- Who created the account
- When the account was created
- Who last changed the account
- When the account was last changed
- When the account expires
- Whether the account is valid
- Whether the account principal's password is valid
- When the account principal's password was last changed

a[*dd*] [*pname* [**-g** *gname* **-o** *oname* **-mp** *password* {**-rp | -pw** *password*} [**-m** *misc*] [**-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 *pname* (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 **-mp**, 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.

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.

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 in the following format:

yy/mm/dd/hh/mm/ss

The default is **none**, meaning that 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:

- The **c**[**lient**] option, if enabled, allows the principal to act as 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.
- The **s**[**erver**] 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.
- The **po**[**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.
- The **f[orwardable**] 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.
- The **pr**[**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.
- The **T**[**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.
- The **r**[**enewable**] option turns on the Kerberos V5 renewable ticket feature. This feature is not currently used by DCE; any use of this option is unsupported at the present time.
- The **dup**[**_session_key**] option allows tickets issued to the account's principal to have duplicate keys. The default is disabled.

The **-gs** (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.

c[**hange**] [**-p** *pname*] [**-g** *gname*] [**-o** *oname*] [**-np** *pname*] [**-ng** *gname*] [**-no** *oname*]
[{**-rp** | **-pw** *password*} **-mp** *password*] [**-m** *misc*] [**-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*]

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.

If you do not specify all three **-p**, **-g**, and **-o** options, wildcard updates can occur. For example, if you specify only the **-g** option, the changes affect all accounts that are associated with the named group. Note that you cannot use wildcarding to change passwords. To change a password, you must enter the **-p**, **-g**, and **-o** options.

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.

del[**ete**] **-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, **rgy_edit** prompts individually for whether to delete each account associated with the group or organization.

ce[**ll**] *cellname* [**-ul** *unix_num*] [**-uf** *unix_num*] [**-gl** *gname*] [**-ol** *oname*]
[**-gf** *gname*] [**-of** *oname*] [**-mp** *passwd*] [**-fa** *name*] **[-fp** *passwd*]
[**-q** *quota*] [**-x** *account_expiration_date* | **none**]

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.

The *cellname* argument specifies the full pathname of the foreign cell with which you will establish the cross-cell authentication account. This name is stripped of the path qualifier and prefixed with **krbtgt**. The resulting name is used as the primary name for the cross-cell authentication account. For example, if you enter **/.../dresden.com**, the principal name is **krbtgt/dresden.com**.

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 total number of objects that can be created in your cell's registry by all foreign users who use the cross-cell authentication account to access your cell. The object creation quota defaults to 0 (zero), meaning that principals in the foreign cell cannot create objects in the local cell. The object creation quota set for your cell's account in the foreign cell places the same restriction on the number of objects that your cell's principals can create in the foreign cell's registry.

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.

kta[*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. Note that you must be **root** to add entries in the default keytab file.

ktl[**ist**] [**-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.

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[**elete**] **-p** *principal_name* **-v** *version_number* [**-f** *keyfile*]

Deletes a sever 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**[**pdate**]]

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 **-update** option indicates you want to talk to an update site in the specified cell.

prop[**erties**]

Changes or displays registry properties.

This command prompts you for changes. Press **<Return>** to leave information unchanged.

po[**licy**] [*organization_name*] [**-al** *lifespan* | **forever**]
[**-pl** *passwd_lifespan* | **forever**] [**-px** *passwd_exp_date* | none]
[**-pm** *passwd_min_length*] [**-pa** | **-pna**] [**-ps** | **-pns**]

Changes or displays registry standard policy or the policy for an organization.

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 lifepan 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.

-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**).

au[**th_policy**]

Changes and/or displays registry authentication policies.

This command prompts you for changes. Press **<Return>** to leave information unchanged.

def[**aults**]

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 **~/.rgy_editrc**. The newly saved defaults are used until you change them.

h[**elp**] [*command*

Displays usage information for **rgy_edit**. If you do not specify a particular command, **rgy_edit** lists the available commands.

q[**uit**]

Exit **rgy_edit**.

e[**xit**]

Exit **rgy_edit**.

l[**ogin**]

Lets you establish a new network identity for use during the **rgy_edit** session.

The **rgy_edit** login command prompts for a principal name and password.

sc[**ope**] [*name*]

Limits the scope of the information displayed by the **view** subcommand to the directory (specified by *name*) in the registry database.

Commands for the Local Registry

To edit or view the local registry, invoke **rgy_edit** with the **-l** option while you are logged into the machine whose local registry you want to maintain. This section lists the commands that are valid for editing or viewing the local registry. When you invoke **rgy_edit** with the **-l** option, only the subcommands and options listed here can be used.

v[**iew**]

Displays local registry entries.

del[**ete**] *principal_name*

Deletes the account and credential information for *principal_name* from the local registry.

pu[**rge**]

Purges expired local registry entries.

This command has no options or arguments.

The time limit, or lifespan, for which an entry in the local registry is valid is set as a property of the local registry with the **properties** subcommand. When the **purge** subcommand is run, it deletes all expired entries. The lifespan begins when an entry for the principal is added to the local registry (that is, the beginning of the lifespan is the last time the principal logged in to the local machine.) The lifespan ends after the time limit set as a local registry property.

pr[**operties**]

Changes and/or displays local registry properties and policies.

This command displays the current properties and then prompts for whether you want to make changes to them. You can change the following properties of the local registry:

Capacity
: A number representing the total number of entries the local registry can contain at any one time. When the capacity is reached, subsequent new entries overwrite the oldest entries.

Account lifespan
: The time in which an account in the local registry is valid, 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 lifepan 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.

Related Information

Commands: **sec_admin(8sec)**, **dcecp(8dce)**.

rlogin

Purpose

Performs a remote login

Synopsis

rlogin [**-8EFKLdfx**][**-e** *char*] [**-l** *username*] *host*

Options

-8 Allows an 8-bit input data path at all times. Without the **-8** option, parity bits are not stripped if the remote side stop and start characters are **^S/^Q**; if they are not **^S/^Q**, parity bits are stripped.

-E Stops any character from being recognized as an escape character. When used with the **-8** option, this provides a completely transparent connection.

-F Forwards the local credentials to the remote system, and marks the remote credentials as **Forwardable**, allowing them to be forwarded from there to another remote system.

-K Turns off all Kerberos authentication. If you specify this option, the command prompts for a password. The entered password is sent across the network in cleartext.

-L Allows the **rlogin** session to be run in **litout** mode. (See **tty(4)**.)

-d Turns on socket debugging for the TCP sockets used to communicate with the remote host. (See **setsockopt(2)**.)

-e Allows users to specify the escape character. The escape character can be specified as literal or as an octal value in the form *nnn*.

-f Forwards the local credentials to the remote system, and marks the remote credentials as **non-forwardable**. The credentials cannot be forwarded from there to another remote system.

-l Allows the remote username to be specified. By default, the remote username is the same as the local username.

-x Turns on DES encryption for all data passed via the **rlogin** session. DES encryption may impact response time and CPU utilization, but it provides increased security. This option is subject to export control.

DESCRIPTION

The **rlogin** command starts a terminal session on the remote host. The command first attempts to use the Kerberos V5 protocol to authenticate to the remote host. If the authentication is successful, user authorization is performed as described in **Kerberos Authorization** below. After a successful authentication, the user is not required to enter a password and, therefore, the password is *not* sent over the network in cleartext. If the remote host does not support Kerberos, the command uses the standard Berkeley **rhosts** authorization mechanism.

Escape Characters

Unless another escape character is specified with the **-e** option, the **~** (tilde) is the escape character. Normally **control-Y** (**^Y**) is the delayed-suspend character. Use the escape character to

- Disconnect from the remote host by entering a line in the form *e***.** (where *e* indicates the escape character)
- Suspend the **rlogin** session by entering a line in the form *e***^Z** (where *e* indicates the escape character)
- Suspend the send portion of the **rlogin** session, but allow output from the remote system by entering a line in the form *e delayed-suspend character* (where *e* indicates the escape character)

Echoing

All echoing takes place at the remote site, so that (except for delays) the **rlogin** is transparent. Flow control via **^S/^Q** and flushing of input and output on interrupts are handled properly.

Kerberos Authorization

In order for Kerberos V5 authorization to succeed, the remote account must exist in the remote systems password file. If the local principal is using the **-l** option to log into a remote account with a name that differs from the principal's local account name, either one of the following conditions must be true:

- A **.k5login** file containing the local principal's name, in the form *principal@realm*, must exist in the remote account's home directory. The remote account user must be the owner of the **.k5login** file and the only name granted write permission. In other words, the file's permissions must be **-rw-r–r–**.
- A Kerberos V5 authorization name database file must exist on the remote system. This file must contain the name of the local principal and map the principal to an account on the remote system.

If the **-l** option is not being used, that is, the local principal is logging into a remote account (in the same realm) with the same name as the principal's local name, neither of the above conditions are required to be met.

If Kerberos authentication fails, a warning message is printed, and the standard Berkeley **rlogin** is used.

Environment

The following environment variable is utilized by **rlogin**:

Term Determines the user's terminal type.

Diagnostices

Diagnostics can occur from both the local and remote hosts. Those that occur on the local host before the connection is completely established are written to standard error. Once the connection is established, any error messages from the remote host are written to standard output, like any other data.

login/tcp: Unknown service

warning, can't get entry for *servicename*/**tcp service**
The **rlogin** command could not find the login service listed in the **/etc/services** database file.

unknown user id
The **rlogin** command could not find your user ID in the password file.

system call:...
An error occurred when **rlogin** attempted the indicated system call. See the appropriate manual entry for information about the error.

kcmd to host *hostname* **failed -** *error message*
An error occurred during Kerberos authentication. The Kerberos-specific error message will be displayed.

warning, using standard rlogin: can't provide Kerberos auth data.
Kerberos authentication failed and the host is retrying using the standard Berkeley **rhosts** authorization mechanism.

Related Information

Commands: **rsh(8sec)**, **rlogind(8sec)** Files: **.k5login**

rlogind

Purpose

Remote login server

Synopsis

rlogind [**-aknx**]

OPTIONS

-k Allow Kerberos V5 with the **.k5login** access control file to be trusted. If this authentication system is used by the client and the authorization check passes, the user is allowed to log in.

-n Disable keep-alive messages.

-x Create an encrypted session. This option is subject to export control.

DESCRIPTION

The **rlogind** server is the server for the **rlogin(8sec)** program. It is based on **rlogind(8sec)**, but uses Kerberos authentication. **rlogind** is configured by command line arguments passed by **inetd**.

The **rlogind** server is invoked by **inetd(8c)** when **inetd** receives a connection on the port indicated in **/etc/inetd.conf**. A typical **/etc/inetd.conf** configuration line for **rlogind** might be as follows:

```
klogin stream tcp  nowait root /opt/dcelocal/etc/rlogind rlogind -k
```

To prevent non-secure access, comment out the entry for **login** in **/etc/inetd.conf** to deny non-Kerberos access.

When **rlogind** receives a service request, it first checks Kerberos authentication and then checks authorization via the access-control file **.k5login** in the user's home directory. If the authentication and authorization succeeds, **rlogind**

- Allocates a pseudo terminal (see **pty(4)**)
- Manipulates file descriptors so that the slave half of the pseudo terminal becomes the **stdin**, **stdout**, and **stderr** for a **login** process
- Invokes **login(1)** program with the **-f** option

If automatic authentication fails, the user is prompted to log in as if on a standard terminal line.

If ticket forwarding is requested by the **rlogin** client, **k5dcelogin(8sec)** is invoked by **rlogind** to promote the forwarded Kerberos credentials to DCE credentials. The **login** process is then invoked by **k5dcelogin**.

The parent of the **login** process manipulates the master side of the pseudo terminal, operating as an intermediary between the login process and the client instance of the **rlogin** program.

In normal operation, the packet protocol described in **pty(4)** is invoked to provide **^S/^Q** type facilities and to propagate interrupt signals to the remote programs. The **login** process propagates the client terminal baud rate and terminal type (found in the environment variable, **TERM**). See **environ(7)**. The screen or window size of the terminal is requested from the client, and the window size changes from the client are propagated to the pseudo terminal.

Transport-level keepalive messages are enabled unless the **-n** option is specified. The use of keepalive messages allows sessions to be timed out if the client crashes or becomes unreachable.

DIAGNOSTICS

All initial diagnostic messages are indicated by a leading byte with a value of 1, after which any network connections are closed. If there are no errors before **login** is invoked, a null byte is returned to indicate success.

Try again A fork by the server failed.

fork: No more processes

The server was unable to fork a process to handle the incoming connection.

Wait a period of time and try again. If this message persists, the server's host may have runaway processes that are using all the entries in the process table.

Out of ptys The server was unable to obtain a pseudo-terminal for use with the login process. Either all pseudo-terminals were in use or the pty driver has not been properly set up.

Check the **pty** configuration of the host where **rlogind executes**.

Permission denied
The server denied access because the client was not using a reserved port. This should only happen to interlopers trying to break into the system.

*system call***:** *cause_of_failure*
An error in executing a system call. A message specifying the cause of the failure is appended to this error.

/usr/bin/login: *reason*
The login program could not be started via **exec(2)** for the reason indicated.

Try to correct the condition causing the problem. If this message persists, contact your system administrator.

rcmd: connect : *hostname***: Connection refused.**
This generic message could be due to a number of reasons. One of the reasons is that the entry for **login** is not present in **/etc/inetd.conf**. The entry may have been removed or commented out to prevent non-secure access.

Kerberos authentication failed
An error occurred during Kerberos authentication. The Kerberos-specific error message will be appended to the error message.

User *remote username* **is not authorized to login to account** *local username*
An error occurred during Kerberos authorization. The Kerberos-specific error message will be appended to the error message.

RELATED INFORMATION

Commands: **rlogind(8sec)**, **rlogin(8sec)**. Files: **.k5login**

rsh

Purpose

Executes a remote command

Synopsis

rsh [**-FKdfnx**][**-l** *username*] *host* [*command*]

Arguments

host The name of the host on which to execute the remote command.

command The command to execute remotely. If no command is specified, you will be logged in on the remote host using **rlogin(8sec)**

Options

-F Forwards the local credentials to the remote system for use by the specified command and marks the remote credentials as **Forwardable**, allowing them to be forwarded from there to another remote system.

-K Turns off all Kerberos authentication.

-d Turns on socket debugging (using **setsockopt(2)**) for the TCP sockets used to communicate with the remote host.

-f Forwards the local credentials to the remote system for use by the specified command and marks the remote credentials as **Non-Forwardable**. The credentials cannot be forwarded from there to another remote system.

-l Allows the remote username to be specified. By default, the remote username is the same as the local username.

-n Redirects input from the special device **/dev/null**. See **Bugs** for more information.

-x Turns on DES encryption for all data exchange. DES encryption may introduce a significant delay in response time. This option is subject to export control.

Description

The **rsh** command executes commands on the remote host. To accomplish this, the **rsh** command copies its standard input to the remote command, the standard output of the remote command to its standard output, and the standard error of the remote command to its standard error. Interrupt, quit, and terminate signals are propagated to the remote command. The **rsh** command normally terminates when the remote command does.

The **rsh** command first attempts to use Kerberos V5 protocol to authenticate to a remote host. If the authentication is successful, user authorization is performed as described in **Kerberos Authorization** below. If the remote host does not support Kerberos, the command uses the standard Berkeley **rhosts** authorization mechanism.

Shell metacharacters that are enclosed in quotation marks are interpreted on the remote machine. Shell metacharacters that are not enclosed in quotation marks are interpreted on the local machine.

For example, the following command appends the remote file **remotefile** to the local file **localfile**:

```
rsh otherhost cat remotefile >> localfile
```

The following command appends the remote file **remotefile** to the remote file **other_remotefile**.

```
rsh otherhost cat remotefile ">>" other_remotefile
```

Kerberos Authorization

In order for Kerberos V5 authorization to succeed, the remote account must exist in the remote system's password file. If the local principal is using the **-l** option to log into a remote account with a name that differs from the principal's local account name, either one of the following conditions must be true:

- A **.k5login** file containing the local principal's name, in the form *principal@realm*, must exist in the remote account's home directory. The remote account name must be the owner of the **.k5login** file and the only name granted write permission. In other words, the file's permissions must be **-rw-r–r–**.
- A Kerberos V5 authorization name database file must exist on the remote system. This file must contain the name of the local principal and map the principal to an account on the remote system.

If **-l** option is not being used, that is, the local principal is logging into a remote account (in the same realm) with the same name as the principal's local name, neither of the above conditions are required to be met.

If Kerberos authentication fails, a warning message is printed, and the standard Berkeley **rlogin** is used.

Bugs

If you are using **csh(1)** and put a **rsh(8sec)** in the background without redirecting its input away from the terminal, it will block even if no reads are posted by the remote command. For no input, redirect the input of **rsh** to **/dev/null** using the **-n** option.

You cannot run an interactive command (like **rogue(6)** or **vi(1)**) with **rsh**. Instead, use **rlogin(8sec)**.

Stop signals stop the local **rsh** process only.

Diagnostices

rlogin: *reason*
: An error occured in executing **rlogin**. (The **rlogin** command is executed when the user does not specify any commands to be executed.) This error is followed by the reason the execution failed.

shell/tcp: unknown service

can't get entry for *entry_name*/tcp service
: The shell service specification is not present in the **/etc/services** file.

can't establish stderr
: The **remsh** command cannot establish secondary socket connection for **stderr**.

*system call***:** *reason*
An error occured in executing *system call*. The reason for the failure is appended to this error.

unknown user id
Check with the system administrator to see if your entry in the password file has been deleted by mistake.

rcmd: connect: *hostname***: Connection refused**
One cause for display of this generic error message could be the absence of an entry for *hostname* in **/etc/inetd.conf** on the remote system. This entry may have been removed or commented out to prevent non-secure access.

kcmd to host *hostname* **failed -** *error message*
An error occurred during Kerberos authentication. The Kerberos-specific error message is displayed.

warning, using standard rsh: *error message*
Kerberos authentication failed and the host is retrying using the standard Berkeley **rhosts** authorization mechanism.

Related Information

Commands: **rlogin(8sec)**, **rshd(8sec)**. Files: **.k5login**, **/etc/hosts**.

rshd

Purpose

Kerberized remote shell server

Synopsis

rshd [**-aknxL**]

Options

-a Verify the identify of the remote host. This option cannot be used with the **-k** option.

-k Allow Kerberos V5 authentication based on the private authorization list maintained in the **.k5login** file. If this authentication method is used by the client and the authorization check is passed, the user is allowed to log in.

-n Disable keep-alive messages.

-x Require the client to encrypt the connection. This option is subject to export control.

-L All successful accesses are logged to **syslogd(8)** as **auth.info** messages.

Description

The **rshd** server is the server for the **rcmd(3)** routine and, consequently, for the **rsh(8sec)** program. The **rshd** server provides remote execution facilities with authentication based on either privileged port numbers from trusted hosts or on the Kerberos authentication system. **rshd** is configured by command-line arguments passed by **inetd(8).**

The **rshd** server is invoked by **inetd(8c)** when **inetd(8c)** receives a connection on the port indicated in **/etc/inetd.conf**. A typical **/etc/inetd.conf** configuration line for **rshd** might be as follows:

```
kshell stream tcp  nowait root /opt/dcelocal/etc/rshd rshd -k
```

To prevent non-secure access, comment out the entry for **shell** in **/etc/inetd.conf** to deny non-Kerberos access. If non-Kerberos access is requested, the following error message is displayed:

```
rcmd: connect hostname : Connection refused
```

When **rshd** receives a service request, it initiates the following protocol:

1. Check authentication.
2. Check authorization via the access-control file **.k5login** the user's home directory.
3. Return a null byte on the initial socket and pass the command line to the normal login shell of the user. This shell inherits the network connections established by **rshd**.

If the **rsh** client requests ticket forwarding, **rsh** invokes **k5dcelogin(8sec)** to promote the forwarded Kerberos credentials to DCE credentials. **k5dcelogin** then invokes the specified command.

Transport-level keepalive messages are enabled unless the **-n** option is specified. The use of keepalive messages allows sessions to be timed out if the client crashes or becomes unreachable.

Diagnostices

All diagnostic messages are returned on the connection associated with standard error after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned on successful completion of all the steps before the command execution.)

Locuser too long.
The name of the user on the client's machine is longer than 16 characters.

Ruser too long.
The name of the user on the remote machine is longer than 16 characters.

Command too long.
The command line passed exceeds the size of the argument list (as configured into the system).

Login incorrect.
No password file entry for the user name existed.

Remote directory.
The **chdir** command to the home directory failed.

Permission denied.
The authentication procedure described above failed.

Can't make pipe.
The pipe needed for **stderr** wasn't created.

Can't fork; try again.
A fork by the server failed.

*shellname***: ...**
The user's login shell could not be started. This message is returned on the connection associated with **stderr** and is not preceded by a flag byte.

Authentication failed: *error_message*
An error occurred during Kerberos authentication. *error_message* is a Kerberos-specific error message.

Related Information

rshd(8sec), **rsh(8sec)**, **rcmd(3X)**, **k5dcelogin(8sec)**.

sec_admin

Purpose

Registry replica administration tool

Synopsis

sec_admin [**-site** *name*] [**-nq**]

Options

-site *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:

- A specific cell_name (or **/.:** for the local cell), to bind to any replica in the named cell.
- The global name of a replica, to bind to that specific replica in that specific cell.
- The name of a replica as it appears on the replica list, to bind to that replica in the local cell.
- 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 subcommands, this command is replaced at DCE Version 1.1 by the **dcecp** command. This command may be fully

replaced by the **dcecp** command in a future release of DCE, and may no longer be supported at that time.

- **monitor**
- **exit**
- **help**
- **quit**

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:

- View a list of replicas.
- Delete a replica.
- Reinitialize a replica.
- Stop a replica.
- Put the master replica into and out of the maintenance state.
- Generate a new master key used to encrypt principal keys.
- 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:

- 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.
- 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.
- 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.
- 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**:

- It exists in the local cell **/.../dresden.com**.
- It has a global name of **/.../dresden.com/subsys/dce/sec/rs_server_250_2**.
- It is named **subsys/dce/sec/rs_server_250_2** on the replica list.
- 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.

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 may 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.

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:

- The default replica's name and the name of the cell in which the replica exists.
- Whether the replica is a master or a slave.
- The date and time the replica was last updated and the update sequence number.
- 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 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:

- The default replica's unique identifier.
- The replica's network addresses.
- The unique identifier of the cell's master replica.
- The network addresses of the cell's master replica.
- The master sequence number, which is the sequence number of the event that made the replica the master.
- If the replica is the master, the update sequence numbers that are still in the propagation queue and have yet to be propagated.
- The DCE software version number.

initr[**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, **sec_admin** attempts to bind to the master.

This subcommand is generally not used under normal conditions.

lr[**ep**] [**-s**[**tate**]] [**-u**[**uid**]] [**-a**[**ddr**]] [**-p**[**rop**]] [**-al**[**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:

- The date and time of the last update the master sent to each slave replica.
- The sequence number of the last update to each slave replica.
- The number of updates not yet applied to each slave replica.
- The status of the master replica's last communication with each slave replica.
- 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 in 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** *m*]
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. Note that this is the same information as that displayed by the **info** subcommand with no options.

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 *m* 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.

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**[**pdate**]]]
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:

- A specific cell_name (or **/.:** for the local cell) to make any replica in the named cell the default.
- The global name of a replica to make the specified replica in the specified cell the default.
- 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.
- 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**[**uit**]
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: \
            /.../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

sec_create_db {-master | -slave} -my[name] *my_server_name*
[**-cr[eator]** *creator_name*] [**-cu[nix_id]** *creator_unix_id* **-g[roup_low_id]** *g_unix_id*]
[**-k[eyseed]** *keyseed*] [**-ma[x_unix_id]** *max_unix_id*] [**-o[rg_low_unix_id]** *o_unix_id*]
[**-pa[ssword]** *default_password*] [**-p[erson_low_unix_id]** *p_unix_id*]
[**-u[uid]** *cell_uuid*] [**-v[erbose]**]

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 **-myname**, **-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.

-cr[eator] Specifies the principal name of the initial privileged user of the registry database (known as the *registry creator*).

-cu[nix_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[roup_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.

k[eyseed] 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 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.

ma[x] Specifies the highest UNIX ID that can be assigned to a principal, group, or organization.

-o[rg_low_unix_id]
Specifies the starting point for UNIX IDs automatically generated by the Security Service when organizations are added with the **rgy_edit** command.

-pa[ssword] The default password assigned to the accounts created by **sec_create_db**, including the account for the registry creator. If you do not specify a default password, **-dce-** is used. (Note that the **hosts**/*local_host*/**self none none**, **krbtgt**/*cell_name* **none none**, and **nobody none none** accounts are not assigned the default password, but instead a randomly generated password.)

-p[erson_low_unix_id]
Specifies the starting point for UNIX IDs automatically generated by the Security Service when principals are added with the **rgy_edit** command.

-u[uid] Specifies the cell's UUID. If you do not enter this UUID, it is assigned dynamically.

-v[erbose] Specifies that **sec_create_db** runs in verbose mode and displays all activity.

Description

The **sec_create_db** tool creates new master and slave databases in *dcelocal*/**var/security/rgy_data** on the machine from which **sec_create_db** is run. Normally, these databases are created only once by the system configuration tool, **dce_config**. However, you can use **sec_create_db** if you need to re-create the master or a slave databse from scratch. You must be root to invoke **sec_create_db**.

The **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 **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 –**cr[eator]** option. If the principal is not one of the reserved names and accounts, **sec_create_db** adds it as a new principal and adds an account for that new principal. If the **-cr** option is not used, root is the creator.

The **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 */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 **sec_create_db** command also creates a registry configuration file, named */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 **secd** master server if the Naming Service is not available.

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)**

sec_salvage_db

Purpose

Recovers a corrupted registry database

Synopsis

sec_salvage_db -print [**-dbpath** *db_pathname*] [**-prtpath** *print_pathname*] [*print_options*] [**-verbose**]

sec_salvage_db -reconstruct [-dbpath *db_pathname*] [**-prtpath** *print_pathname*] [*reconstruct_options*] [**-verbose**]

sec_salvage_db -check [**-dbpath** *db_pathname*] [*db_options*] [**-verbose**]

sec_salvage_db -fix [**-dbpath** *db_pathname*] [*db_options*] [**-force**] [**-verbose**]

Options

-check Check the database elements specified by *db_options* for inconsistencies. This option sends a list to standard output of any detectable data inconsistencies and all bad list links, internal id references, and database keys. The **-check** option does not check fields for legal values.

db_options Specify the database elements to be acted on by the **-check** or **-fix** options. If no *db_options* are specified, all are selected. The *db_options* are

- **-princ** — Principals
- **-group** — Groups
- **-org** — Organizations
- **-acct** — Accounts
- **-acl** — ACLs
- **-policy** — Policy
- **-state** — Database State

- **-replicas** — Replicas

Note: The **.mkey.prt** file and the **princ.prt** file contain unencrypted authentication keys. Ensure that only the privileged account can access these files and that they are never transferred over a network for viewing or backup.

-fix Check the database for inconsistencies and prompt for whether to fix each inconsistency. After all inconsistencies have been processed, the option prompts for whether to save all fixes.

-force Check the database for inconsistencies and fix each one without prompting. After all inconsistencies have been processed, the option prompts for whether to save all fixes. This option is valid only when used with the **-fix** option.

-print Create files containing ASCII-formatted database records. These files are used by the **-reconstruct** option as a source for recreating the database. You can also manually edit the files to change information or fix problems. A separate file is created for each of the *print_options* specified.

By default the **-print** option stores the master key file in the current directory and the database files in the **rgy_print** directory in the current directory. The –**prtpath** option lets you specify a different directory.

print_options

Specify the database elements to be acted on by the **-print** option. If the files exist, they are overwritten. If no *print_options* are specified, all are selected. The *print_options* and the files they create are

- **-princ** — Put principal records in the file **princ.prt** and master key information in the file **.mkey.prt**.
- **-group** — Put group records in the file **group.prt**.
- **-org** — Put organization records in the file **org.prt**.
- **-policy** — Put policy records in the file **policy.prt**.
- **-state** — Put information about the state of the database in the file **rgy_state.prt**.
- **-replicas** — Put replica information in the file **replicas.prt**.

-reconstruct Reconstruct the registry database from the ASCII-formatted print files created by the –**print** option. The *reconstruct_options* specify the print files to use.

reconstruct_options
Specifies which elements of the registry database to reconstruct. If no *reconstruct_options* are specified, all are selected. The *reconstruct_options* are

- **-pgo** — Use data in the **princ.prt**, **group.prt**, **org.prt**, and **.mkey.prt** files to reconstruct:
  - — Principals, groups, organizations
  - — Principal's accounts
  - — ACLs on database objects
  - — The master key file
- **-policy** — Use data from the **policy.prt** file to reconstruct registry policies.
- **-state** — Use data from the **rgy_state.prt** file to reconstruct information about the state of the database.
- **-replicas** — Use data from the **replicas.prt** file to reconstruct the master replica list.

-dbpath *db_pathname*
For the **-print** and **-check** options, **-dbpath** specifies the directory in which the registry database and the master key file are located. For the –**reconstruct** and –**fix** options, **-dbpath** specifies the directory in which to store the reconstructed or salvaged database.

The **-print** and **-check** options expect to find the master key file, **.mkey**, in the directory above the directory that holds the database files. For example, if *db_pathname* is *dcelocal***/var/security/new_rgy**, the options look for the master key file in *dcelocal***/var/security** and the database files in *dcelocal***/var/security/new_rgy**.

If this option is not specified, the default pathname is *dcelocal***/var/security/rgy_data**.

db_pathname can be a global pathname or a cell-relative name.

-prtpath *print_pathname*

For the **print** and **-reconstruct** options only, **-prtpath** specifies the directory in which to create (**-print**) the print files, or find (**-reconstruct**) the print files from which to reconstruct the database.

By default the **-print** option creates and the **-reconstruct** option looks for the master key file in the current directory and the database files in the **rgy_print** subdirectory of the current directory. The **-prtpath** option lets you specify the directory that should be used instead of the current directory. For example, if you specify *print_pathname* as *dcelocal*/**var/security/registry**, the master key print file will be created in that directory and the database print files in *dcelocal*/**var/security/registry/rgy_print**.

If any or all of the print files exist in *print_pathname* or the default directory, their contents are overwritten.

print_pathname can be a global pathname or a cell-relative name.

Description

The **sec_salvage_db** tool is an aid to database administration and troubleshooting. Although day-to-day administration is handled by the **rgy_edit** command, **sec_salvage_db** can be useful for listing registry data, reconstructing databases, and salvaging corrupted databases.

The **sec_salvage_db** command supports two methods of operation: the check and fix method and the print and reconstruct method. These methods can be used in tandem.

Check and Fix Method

Note: The **-check** and **-fix** options are not currently available.

The check and fix method recovers data from a corrupted database, fixing corrupted data links, data retrieval keys, and other internal references. You can use it on a database so corrupted that it prevents the Security Server (**secd**) from running or registry clients from operating correctly. The check and fix method repairs the database structure so that **secd** can run. (Note that data may be lost if corrupted pointers in the registry data files irreversibly sever the links between records.) The check and fix method uses the **sec_salvage_db -check**, **-fix**, and **-force** options.

The **-check** option accesses each record in the database and reports all errors, but makes no fixes. Although you can run it to see the state of the database before you run the **-fix** option, it is not required to be run.

The **-fix** option also accesses each record in the database and reports all errors, but as it finds each error, it prompts for whether or not to fix the error. When processing is complete, **sec_salvage_db** prompts for whether or not to save the changes.

The **-force** option can only be used with the **-fix** option. If you use it, **sec_salvage_db** does not prompt for confirmation before it fixes each error it finds. The **sec_salvage_db** command will still prompt for confirmation before it saves the changes.

The Print and Reconstruct Method

The print and reconstruct method allows you to reconstruct a database. It first creates ASCII files, called print files, that contain all accessible data in the database. Then, it reads the data in these files to construct a new database. If you cannot start a Security Server on the database host machine, you cannot use the print and reconstruct method, but must use the check and fix method. (Note that before you run **sec_salvage_db** with the **–print** and **–reconstruct** options, you must stop the Security Server.)

In addition to reconstructing the database, the print and reconstruct method has other uses. You can use it to

- Make changes to the database by manually editing the print files created by the **-print** option and then reconstructing them from the changed print files. This can be especially useful for changing many user passwords, which may be necessary if the master key file is corrupted.
- Obtain a listing of database contents.
- Copy databases between different platforms.

To use the print and reconstruct method, run **sec_salvage_db** first with the **-print** option and then with the **-reconstruct** option.

The **-print** option creates the ASCII print files from the registry database files. These files can be reviewed and edited to correct faulty information, such as mismatched names and UNIX IDs or missing data, or to update existing data. The **-reconstruct** option recreates the registry database files from the print files.

Because the **-print** option creates files containing all data in the database and the **-reconstruct** option recreates the database based on these files, you can use this method to move a database to another machine or even another cell. For example, if you run **sec_salvage_db -print** on an uncorrupted database, you can then run **sec_salvage_db**

-reconstruct and specify a pathname on a different machine for where the database should be created.

Editing the Print Files

To edit the print files, your entries must be in the following format:

field_name optional_white_space=optional_white_space value

Although you can leave spaces between the field name, the equals sign, and the value, field names and values cannot contain white space.

A sample **group.prt** file follows:

```
Record_Number = 7
Object_Type = ADMIN
Name = group/admin
UUID = 000001b9-7b61-21cf-bd01-0800097086cb
Unix_ID = 441
Is_Alias_Flag = false
Is_Required_Flag = false
Projlist_Ok_Flag = true
Num_Attr_List_Entries = 0
Fullname =
Member_Name = admin_1
Member_Name = admin_2
Member_Name = admin_3
Foreign_Member_Name = /.../engobe/person1
Cell_UUID = 964dc902-7b54-11cf-b1ff-08000919bba7
Princ_UUID = 0000006a-7b61-21cf-bb00-08000919bba7
Foreign_Member_Name = /.../engobe/person10
Cell_UUID = 964dc902-7b54-11cf-b1ff-08000919bba7
Princ_UUID = 00000073-7b61-21cf-bb00-08000919bba7
Obj_Acl_Def_Cell_Name = /.../abc.com
Num_Acl_Entries = 6
Obj_Acl_Entry = any_other:r-t——
Obj_Acl_Entry = group:acct-admin:rctDnfmM
```

Obj_Acl_Entry = group_obj:r-t—-
Obj_Acl_Entry = other_obj:r-t—-
Obj_Acl_Entry = unauthenticated:r-t—-
Obj_Acl_Entry = user:cell_admin:rctDnfmM

To update existing entries, simply supply a new value. For example, to update a principal's full name, the entry in the **princ.prt** file is

Fullname = *fullname*

The *fullname* variable is the principal's full name. The **princ.prt** file contains the following entry that allows you to update a principal's password in plain text:

Plaintext_Passwd =

This field does not display the principal's password. To update the password, simply enter the new one in plain text after the equals sign. When the database is reconstructed, the password is encrypted and any keys derived from that password are regenerated and used to overwrite any existing encryption key entries.

To specify a NULL value, delete the existing value. For example, to specify a NULL value for a fullname in the **princ.prt** file, the entry is

Fullname =

Print File Fields and Values

The following lists describe the fields in the **princ.prt**, **group.prt, org.prt**, **.mkey.prt**, **policy.prt**, **rgy_state.prt**, and **replicas.prt** files. In the lists, an ***** (asterisk) indicates a segment or field that can appear multiple times in succession; a **+** (plus sign) indicates that if a stored UUID does not map to a name required for the field, the UUID is displayed.

The princ.prt File

The fields in the **princ.prt** file follow:

- *For all records:*

  Record_Number
  The sequential number of the record in the database.

  Object_Type
  An indication of the type of object: **PRINC**=principal, **DIR**=directory.

  Name Name of the object.

  UUID Unique Identifier of the object.

- *For principals:*

  Unix_ID The principal's UNIX ID.

  Is_Alias_Flag
  An indication of whether or not the principal name is an alias or a primary name: **true**=alias, **false**=primary.

  Is_Required_Flag
  An indication of whether or not the principal is reserved: **true**=principal is reserved and cannot be deleted, **false**=principal is not reserved.

  Quota The principal's object creation quota: a non-negative integer or **unlimited**.

  Fullname The principal's fullname: a text string.

  Member_Name*
  The names of the groups to which the principal belongs.

  Obj_Acl_Def_Cell_Name
  The default cell name of this principal's object ACL.

  Num_Acl_Entries
  The number of entries in the principals object ACL.

  Obj_Acl_Entry*+
  The contents of the principal's object ACL.

  Acct_Group_Name
  The account's group name.

  Acct_Org_Name
  The account's organization name.

Acct_Creator_Name
The name of principal who created this account.

Acct_Creation_Time
The date and time the account was created in *yyyy/mm/dd.hh:mm* format. The first two digits of the year, the hours, and the minutes are optional.

Acct_Changer_Name
Name of principal who last changed the account.

Acct_Change_Time
The date and time the account was last changed in *yyyy/mm/dd.hh:mm* format. (The first two digits of the year, the hours and the minutes are optional.)

Acct_Expire_Time
The date and time the account expires or **none** for no expiration date. The date and time are in *yyyy/mm/dd.hh:mm* format. (The first two digits of the year, the hours and the minutes are optional.)

Acct_Good_Since_Time
The date and time the principal's account was last known to be in an uncompromised state in *yyyy/mm/dd.hh:mm*, format or **no** for current time and date. (The first two digits of the year, the hours and the minutes are optional.)

Acct_Valid_For_Login_Flag
An indication of whether or not the account can be logged into: **true**=account is valid for login, **false**=account cannot be logged into.

Acct_Valid_As_Server_Flag
Indicates whether or not the account is a server and can engage in authenticated communication: **true**=account is a server, **false**=account is not server.

Acct_Valid_As_Client_Flag
Indicates whether or not the account is a client and can log in, acquire tickets, and be authenticated: **true**=account is a client, **false**=account is not a client.

Acct_Post_Dated_Cert_Ok_Flag
Indicates whether or not tickets with a start time some time in the future can be issued to the account's principal: **true**=postdated tickets can be issued, **false**=postdated tickets cannot be issued.

Acct_Forwardable_Cert_Ok_Flag
Indicates whether or not a new ticket-granting ticket with a network address that differs from the present ticket-granting address can be issued to the account's principal: **true**=account can get forwardable certificates, **false**=account cannot.

Acct_TGT_Auth_Cert_Ok_Flag
Indicates whether or not tickets issued to the account's principal can use the ticket-granting-ticket authentication mechanism: **true**=tickets can use the ticket-granting-ticket authentication mechanism, **false**=they cannot.

Acct_Renewable_Cert_Ok_Flag
Indicates whether or not tickets issued to the principal's ticket-granting ticket to be renewed: **true**=tickets can be renewed, **false**=tickets cannot be renewed.

Acct_Proxiable_Cert_Ok_Flag
Indicates whether or not a new ticket with a different network address than the present ticket can be issued to the account's principal: **true**=such a ticket can be issued, **false**=such a ticket cannot be issued.

Acct_Dup_Session_Key_Ok_Flag
Indicates whether or not tickets issued to the account's principal can have duplicate keys: **true**=account can have duplicate session keys, **false**=account cannot.

Unix_Key The account principal's encrypted UNIX password: ASCII string.

Plaintext_Passwd
Stores the principal's password in plain text. This field is provided to allow principal's passwords to be changed. When the **princ.prt** file is processed by the **sec_salvage_db -reconstruct** option, this password is encrypted using UNIX system encryption. This encrypted password is then stored as the principal's encrypted UNIX password in the **Unix_Key** field.

Home_Dir The account principal's home directory: text string.

Shell The account principal's login shell: text string.

Gecos The account's GECOS information: text string.

Passwd_Valid_Flag
Indicates whether or not the account principal's password is valid: **true**=password is valid, **false=**password not valid.

Passwd_Change_Time
The date and time the account principal's password was last changed in *yyyy/mm/dd.hh:mm* format or **now** for the current date and time. The first two digits of the year, the hours and the minutes are optional.

Max_Certificate_Lifetime
The number of hours before the Authentication Service must renew the account principal's service certificates: an integer indicating the time in hours or **default-policy** to use the registry default.

Max_Renewable_Lifetime
The number of hours before a session with the account principal's identity expires and the principal must log in again to reauthenticate: an integer indicating the time in hours or **default-policy** to use the registry default.

Master_Key_Version
The version of the master key used to encrypt the account principal's key.

Num_Auth_Keys
The number of the account principal's authentication keys.

Auth_Key_Version*
A list of the version numbers of the account principal's authentication key. The first version number on the list represents the current authentication key.

Auth_Key_Pepper*
The pepper algorithm used for the account principal's key: a text string or blank to use the default pepper algorithm.

Auth_Key_Len*
The length in bytes of the account principal's authentication key.

Auth_Key*
The account principal's authentication key: hex string.

Auth_Key_Expire_Time*
The date and time the account principal's authentication key expires or **none** for no expiration. Date and time are in *yyyy/mm/dd.hh:mm* format. (The first two digits of the year, the hours and the minutes are optional.)

- *For directories:*

Obj_Acl_Def_Cell_Name+
The default cell name of the directory's object ACL.

Num_Acl_Entries
The number of entries in the directory's object ACL.

Obj_Acl_Entry*+
The contents of the directory's object ACL.

Init_Obj_Acl_Def_Cell_Name+
The default cell name of the directory's initial object ACL.

Num_Acl_Entries
The number of entries in the directory's initial object ACL.

Init_Obj_Acl_Entry*+
The contents of the directory's initial object ACL.

Init_Cont_Acl_Def_Cell_Name+
The default cell name of the directory's initial container ACL.

Num_Acl_Entries
The number of entries in the directory's initial container ACL.

Init_Cont_Acl_Entry*+
The contents of the directory's initial container ACL.

The group.prt File

The fields in the **group.prt** file follow:

- *For all records:*

Record_Number
The sequential number of the record in the database.

Object_Type
An indication of the type of object: **GROUP**=group, **DIR**=directory.

Name Name of the object.

UUID Universal Unique Identifier of the object.

- *For groups:*

Unix_ID UNIX ID of the group.

Is_Alias_Flag
An indication of whether or not the group name is an alias or a primary name: **true**=alias, **false**=primary.

Is_Required_Flag
An indication of whether or not the group is reserved: **true**=group is reserved and cannot be deleted, **false**=group is not reserved.

Projlist_Ok_Flag
An indication of whether or not the group can be included in project lists: **true**=group can be included on project lists, **false**=group cannot be included.

Fullname The group's fullname: a text string.

Member_Name*
The names of the group's local members.

Foreign_Member_Name*
The names of the group's foreign members.

Cell_UUID The UUID of the cell for the principal identified in "Foreign_Member_Name."

Princ_UUID
The UUID of the principal identified in "Foreign_Member_Name."

Obj_Acl_Def_Cell_Name+
The default cell name of this group's object ACL.

Num_Acl_Entries
The number of entries in the group's object ACL.

Obj_Acl_Entry*
The contents of the group's object ACL.

- *For directories:*

  Obj_Acl_Def_Cell_Name+
  The default cell name of this directory's object ACL.

  Num_Acl_Entries
  The number of entries in the directory's object ACL.

  Obj_Acl_Entry*
  The contents of the directory's object ACL.

  Init_Obj_Acl_Def_Cell_Name+
  The default cell name of the directory's initial object ACL.

  Num_Acl_Entries
  The number of entries in the directory's initial object ACL.

  Init_Obj_Acl_Entry*+
  The contents of the directory's initial object ACL.

  Init_Cont_Acl_Def_Cell_Name+
  The default cell name of the directory's initial container ACL.

  Num_Acl_Entries
  The number of entries in the directory's initial container ACL.

  Init_Cont_Acl_Entry*+
  The contents of the directory's initial container ACL.

The org.prt File

The fields in the **org.prt** file follow:

- *For all records:*

  Record_Number
  The sequential number of the record in the database.

  Object_Type
  An indication of the type of object: **ORG**=organization, **DIR**=directory.

  Name Name of the object.

  UUID Universal Unique Identifier of the object.

- *For organizations:*

  Unix_ID UNIX ID of the organization.

Is_Alias_Flag
An indication of whether or not the organization is an alias or a primary name: **true**=alias, **false**=primary.

Is_Required_Flag
An indication of whether or not the organization is reserved: **true**=organization is reserved and cannot be deleted, **false**=organization is not reserved.

Fullname The organization's fullname: a text string.

Member_Name*
The names of the organization's members.

Obj_Acl_Def_Cell_Name
The default cell name of this organization's object ACL.

Num_Acl_Entries
The number of entries in the organization's object ACL.

Obj_Acl_Entry*+
The contents of the organization's object ACL.

- *For organizations with policy:*

Acct_Lifetime
The period during which accounts for the organization are valid: a integer number representing days or **forever**.

Passwd_Min_Len
The minimum length of the organization's password: a non-negative integer.

Passwd_Lifetime
The span in days of the lifetime of the organization's password: an integer or **forever**.

Passwd_Expire_Time
The date and time the organization's password expires in *yyyy/mm/dd.hh:mm* format. (The first two digits of the year, the hours and the minutes are optional.)

Passwd_All_Spaces_Ok
An indication of whether or not the organization's password can consist of all spaces: **true**=can consist of spaces, **false**=cannot.

Passwd_All_Alphanumeric_Ok
An indication of whether or not the organization's password can consist of all alphanumeric characters: **true**=can be all alphanumeric, **false**=cannot.

- *For directories:*

Obj_Acl_Def_Cell_Name+
The default cell name of the directory's object ACL.

Num_Acl_Entries
The number of entries in the directory's object ACL.

Obj_Acl_Entry*+
The contents of the directory's object ACL.

Init_Obj_Acl_Def_Cell_Name+
The default cell name of the directory's initial object ACL.

Num_Acl_Entries
The number of entries in the directory's initial object ACL.

Init_Obj_Acl_Entry*+
The contents of the directory's initial object ACL.

Init_Cont_Acl_Def_Cell_Name+
The default cell name of the directory's initial container ACL.

Num_Acl_Entries
The number of entries in the directory's initial container ACL.

Init_Cont_Acl_Entry*+
The contents of the directory's initial container ACL.

The .mkey.prt File

The fields in the **.mkey.prt** file follow:

Master_Key_Version
The integer version of the master key.

Master_Key_Keytype
Always **des**.

Master_Key_Length
The length of the master key in bytes.

Master_Key
The master key in hex string format.

The policy.prt File

The fields in the **policy.prt** file follow:

Rgy_Policy_File_Version
An integer representing the version of the policy information.

Prop_Read_Version
A number indicating the property record's read version.

Prop_Write_Version
A number indicating the property record's write version.

Min_Certificate_Lifetime
The minimum amount of time before the principal's ticket must be renewed in *weeks***w***days***d***hours***h***minutes***m** format.

Default_Certificate_Lifetime
The default lifetime for tickets issued to principals in this cell's registry in *weeks***w***days***d***hours***h***minutes***m** format.

Low_Unix_ID_Principal
The starting point for principal UNIX IDs automatically generated by the Security Service when a principal is added: an integer, which must be less than **Max_Unix_ID**.

Low_Unix_ID_Group
The starting point for UNIX IDs automatically generated by the Security Service when a group is added: an integer, which must be less than **Max_Unix_ID**.

Low_Unix_ID_Org
The starting point for UNIX IDs automatically generated by the Security Service when an organization is added: an integer, which must be less than **Max_Unix_ID**.

Max_Unix_ID
The highest number that can be supplied as a UNIX ID when principals are created: an integer.

Rgy_Readonly_Flag
An indication of whether or not the registry is read-only: **true**=read only, **false**=updateable.

Auth_Certificate_Unbound_Flag
An indication of whether or not certificates are generated for use on any machine: **true**=yes, **false**=no.

Shadow_Passwd_Flag
Determines whether encrypted passwords are sent over the network: **true**=encrypted passwords are not sent over the network, **false**=encrypted passwords are sent over the network.

Embedded_Unix_ID_Flag
Determines if UNIX IDs are embedded in person, group, and organization UUIDs: **true**=UNIX IDs are embedded, **false**=UNIX IDs are not embedded.

Realm_Name
The name of the full global pathname of realm running the **secd**.

Realm_UUID
The UUID of the realm running the **secd**.

Unauthenticated_Quota
The quota of unauthenticated users: a number or **unlimited**.

Acct_Lifetime
The period during which accounts are valid: an integer representing days or **forever**.

Passwd_Min_Len
The minimum length of passwords: a non-negative integer.

Passwd_Lifetime
The span in days of the password lifetimes: an integer or **forever**.

Passwd_Expire_Time
The date and time the passwords expire in *yyyy/mm/dd.hh:mm* format. (The first two digits of the year, the hours and the minutes are optional.)

Passwd_All_Spaces_Ok
An indication of whether or not passwords can consist of all spaces: **true**=can consist of spaces, **false**=cannot.

Passwd_All_Alphanumeric_Ok
Am indication of whether or not passwords can consist of all alphanumeric characters: **true**=can be all alphanumeric, **false**=cannot.

Max_Certificate_Lifetime
The number of hours before the Authentication Service must renew service certificates: an integer indicating the time in hours or **default-policy** to use the registry default.

Max_Renewable_Lifetime
The number of hours before sessions expire and the session principal must log in again to reauthenticate: an integer indicating the time in hours or **default-policy** to use the registry default.

Princ_Cache_State
The timestamp of the principal cache.

Group_Cache_State
The timestamp of the group cache.

Org_Cache_State
The timestamp of the organization cache.

My_Name
The cell-relative name of the security server.

Master_Key_Version
The integer version of current master key.

Master_Key_Keytype
Always **des**.

Master_Key_Length
The length of the master key in bytes.

Master_Key
The master key in hex string format.

Old_Master_Key_Version
The version of the previous master key.

Old_Master_Key_Keytype
Always **des**.

Old_Master_Key_Length
The length of the previous master key in bytes.

Old_Master_Key
The previous master key in hex string format.

Obj_Acl_Def_Cell_Name
The default cell name of the policy object ACL.

Num_Acl_Entries
The number of entries in the policy object ACL.

Obj_Acl_Entry*+
The contents of the policy object ACL.

The rgy_state.prt File

The fields in the **rgy_state.prt** file follow:

Rgy_State_File_Version
The integer version number of the format of the **rgy_state** file.

Replica_State
The state of the master registry: **unknown_to_master**, **uninitialized**, **in_service**, **in_maintenance**, **closed**, **deleted**, or **initializing**.

Cell_UUID The UUID of cell in which the **secd** resides.

Server_UUID
The UUID of this **secd**.

Initialization_UUID
The UUID of the last initialization event.

Master_File_Version
The version number of the master replica.

Master_Known_Flag
An indication of whether or not the master replica is known to this replica: **true**=known, **false**=not known. Only if this field is **true** does the other master field contain valid information.

Master_UUID
The UUID of the master replica.

Master_Seqno:
The 2-digit sequence number of the event when the master became the master in n.*n* format.

The replicas.prt File

The fields in the **replicas.prt** file follow:

Record_Number
The sequential number of the record in the database.

Replica_UUID
The UUID listed for the replica in the replica list.

Replica_Name
The name of the replica as known to the Cell Directory Service.

Num_Towers
The number of towers.

Tower_Length*
The length of the next tower (in bytes).

Tower* The tower used to communicate with the replica (a byte stream that can be broken on word boundaries).

Propagation_Type
An indication of whether the replica is initialized, initializing, in the process of being updated, or in the process of being deleted.

Initialization_UUID
UUID of the last initialization.

Error Conditions

You receive the following error message if the default **rgy_data** directory is being used and there is an advisory lock on the **rgy_state** data file:

Registry: Error - database is locked. Put secd into maintenance mode or clear advisory lock on rgy_state file in *db_pathname*

The existence of the advisory lock implies that **secd** is in service. Use the **sec_admin** command to put **secd** in maintenance mode. If **secd** is not running, the advisory lock may be the result of an ungraceful shutdown of **secd**. To remove the advisory lock, use the **mv** command to rename the *dcelocal***/var/security/rgy_data/rgy_state** file, and then change it back to its original name. Then rerun the **sec_salvage_db** command.

secd

Purpose

The DCE security server

Synopsis

secd [**-b[ootstrap]**] [**-lockpw**] [**-locksm** *pname*] [**-rem[ote]**] [**-master_seqno** *new_master_seqno*] [**-cpi** *time*] [**-restore_master**] [**-noaudfilter**] [**-v[erbose]**]

Options

-locksm[ith] Restarts 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.

-master_seqno
Sets a new master sequence number for the master replica. This option is used only in unusual situations when a replica that you want to be the master has a master sequence number that is lower than (or equal to) another master sequence number in the system. When the master detects

that its master sequence number is lower than another one in the system, it marks itself as a duplicate master and its process exits. Each time you start the master replica, it will notice that it has been deemed a duplicate master, and its process will again exit. Use this option to assign a new master sequence number to the replica you want to be master. The new sequence number should be one digit higher than the highest master sequence number in the system. (Use the **dcecp registry show -replica** command for each replica to find the highest master sequence number.)

-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).

-noaudfilter Disables audit filtering and enables full (unfiltered) auditing. By default **secd** turns audit filtering on.

-v[erbose] Runs in verbose mode.

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.

If the security server finds	**It changes**
Password-Valid flag is set to no	Password-Valid flag to yes
Account Expiration date is set to less than the current time plus one hour	Account Expiration date to the current time plus one hour
Client flag is set to no	Client flag to yes
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

If the security server finds	It changes
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. Restart an existing replica (master or slave) as follows:

   dcelocal/**bin/secd**

3. 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)**.

su

Purpose Substitutes user ID temporarily

Platform Specific

This command is platform-specific. Consult your local operating system documentation for information on how to use your version of the **su** command.

Index

A

account
    administering, 17
accounts
    importing, 817, 822
    viewing registry information, 829
ACL
    dts_audit_events, 751
acl
    administering, 35
acl_edit command, 790
ACLs
    editing entries, 790
    viewing, 790
AGG_DEV_NAM
    dce_config environment variable, 157
AGG_FS_TYPE
    dce_config environment variable, 157
AGG_ID
    dce_config environment variable, 157
AGG_MOUNT_PATH
    dce_config environment variable, 157
AGG_NAME
    dce_config environment variable, 157
attributes
    NSI, viewing, 542
attrlist
    manipulating, 53
aud
    administering, 59, 78
aud_audit_events, 748
audevents
    administering, 66
audfilter
    administering, 70
audit daemon, 802
audit services
    auditable events, 748
audit trail file, 802
auditable events
    audit services, 748
    security services, 766
    time services, 751
auditd command, 801
    privileges required to run, 802
authentication services
    rpc_c_authn_dce_secret, 767
auxiliary file
    client, server, 476

B

binding information (RPC)
    exporting to server entries, 516
    removing information, 559
    viewing server entries, 556
browser
    startup command, 572

C

C language, 478
    compiler, 478
    preprocessor, 479
cache
    clearing servers, 587
    defining servers, 603
    viewing contents, 620
    viewing server addresses, 656
CACHE_CDS_SERVER
    dce_config environment variable, 149
CACHE_CDS_SERVER_IP
    dce_config environment variable, 149
CACHE_DIR_DISK
    dce_config environment variable, 157
CACHE_SIZE_DISK
    dce_config environment variable, 157
CACHE_SIZE_RAM
    dce_config environment variable, 157
cached clearinghouse entity, 654
CDS clerks
    managing interface to servers, 573
    setting confidence levels, 639
    solicitation daemon startup, 570
    stopping, 617
    viewing attributes, 671
    viewing cache contents, 620
CDS servers
    clearing clearinghouses, 589
    clearing from cache, 587
    defining in local cache, 603
    restarting, 585
    solicitation daemon startup, 570
    stopping, 619
    viewing attributes, 689
    viewing cached addresses, 656
cdsalias
    administering, 87
cdscache
    administering, 92
cdscp command, 575
cdscp commands
    about, 564
    add directory, 566
    add object, 568
    cdsadv, 570
    cdsbrowser, 572
    cdsclerk, 573
    clear cached server, 587
    clear clearinghouse, 589
    create child, 591
    create clearinghouse, 593
    create directory, 595
    create link, 597
    create object, 599
    create replicas, 601
    define cached server, 603
    delete child, 605
    delete clearinghouse, 607

delete directory, 609
delete link, 611
delete object, 613
delete replica, 615
disable clerk, 617
disable server, 619
dump clerk cache, 620
list child, 623
list clearinghouse, 625
list directory, 627
list link, 629
list object, 631
remove directory, 633
remove link, 635
remove object, 637
set cdscp confidence, 639
set cdscp preferred clearinghouse, 641
set directory, 643
set directory to new epoch, 646
set directory to skulk, 648
set link, 650
set object, 652
show cached clearinghouse, 654
show cached server, 656
show cdscp preferred clearinghouse, 658, 660
show cell, 662
show child, 664
show clearinghouse, 667
show clerk, 671
show directory, 674
show link, 679
show object, 682
show replica, 684
show server, 689
summary, 575
syntax, 580
cdsd command, 585
cell
DCECP object, 105
cell alias
administering, 114
cell directory client
administering, 100
cell directory server
administering, 82
cell names
conventions, 498
creating, 662
cell service profile
global-set membership, 751
CELL_ADMIN
dce_config environment variable, 149
CELL_ADMIN_PW
dce_config environment variable, 149
CELL_NAME
dce_config environment variable, 149
CEPV, 479
CHANGE_PW
dce_config environment variable, 149
CHECK_TIME
dce_config environment variable, 149
child entity, 664
child pointers
creating, 591
deleting, 605
viewing, 623
viewing attributes, 664
clearinghouse
administering, 118
clearinghouses
creating, 593
deleting, 607
making available, 593

preferred, 641, 658, 660
viewing, 625
viewing attributes, 667
viewing cached, 654
clerk entity, 671
client
Audit, 756
auxiliary file, 476
files, 476
stub, 476
client entry point vector, 479
CLIENT_CACHE_LOG
dce_config environment variable, 157
clock
administering, 131
clocks
adjusting, 743
synchronizing, 740
commands
csrc, 139
dcecp command, 751
for RPC programmers, 474
idl, 474, 476
sams, 2
uuidgen, 474, 485
compilers
C, 478, 479
IDL, 476
CONFIG_NFS_GATEWAY
dce_config environment variable, 157
configuring DCE
start up command, 145

D

DACL Management
interfaces, 768
rdaclif, 768
data types
IDL-to-C mappings, 474
of IDL, 474
DC_DISPLAY_THRESHOLD
dce_config environment variable, 150
DC_LOG_THRESHOLD
dce_config environment variable, 150
DCE Audit
auditable events, 746
files, 746
DCE Control Program commands
dcecp, 162
DCE host daemon
about, 186
DCE RPC
programmer commands, 474
DCE RPC entity
idl command, 476
uuidgen command, 485
dce.rm dce_config component script, 159
dce.unconfig dce_config component script, 160
dce_com_env dce_config component script, 160
dce_com_utils dce_config component script, 160
dce_config
component scripts, 159
dce_config command, 145
dce_config_env dce_config component script, 160

dce_config_utils dce_config component script, 160
dce_login command, 805
dce_shutdown dce_config component script, 159
dcecp command, 748, 751, 766
dcecp commands
account, 17
acl, 35
attrlist, 53
aud, 59
audevents, 66
audfilter, 70
audtrail, 78
cds, 82
cdsalias, 87
cdscache, 92
cdsclient, 100
cellalias, 114
clearinghouse, 118
clock, 131
directory, 190
dts, 210
endpoint, 227
group, 105, 241, 255, 266, 281, 435
hostvar, 276
link, 293
log, 301
name, 307
object, 312
organization, 320
principal, 337
registry, 349
rpcentry, 377
rpcgroup, 389
rpcprofile, 398
secval, 412
server, 418
utc, 448
xattrschema, 458
dcecp(\)(.) commands
uuid, 454
DEFAULT_MAX_ID
dce_config environment variable, 150
DEFAULT_PW
dce_config environment variable, 150
#define, 6, 480
delegation, 780
dfs.clean dce_config component script, 159
dfs.rm dce_config component script, 160
dfs.unconfig dce_config component script, 160
dfs_config dce_config component script, 160
DFS_SERVER_INSTALL
dce_config environment variable, 158
DIR_REPLICATE
dce_config environment variable, 151
directories, 478
adding attributes (CDS), 566
changing attribute values (CDS), 643
child pointers (CDS), 591, 605
creating (CDS), 595
deleting (CDS), 609
removing attribute values (CDS), 633
sams command, 2
updating (CDS), 648
viewing (CDS), 627
viewing attributes (CDS), 674
directory
administering, 190

directory entity, 674
directory pathnames
    conventions, 498
DO_CHECKS
    dce_config environment variable, 152
Domain Name Service (DNS)
    defining cell names, 662
DOMAIN_NAME
    dce_config environment variable, 151
dts
    administering, 210
DTS clerks
    creating, 700
    deleting, 702
    modifying, 722
    starting, 716
    stopping, 704
    viewing characteristics, 728
DTS control program
    exiting, 718, 721
    invoking, 706
DTS entity, 752
DTS servers
    advertising, 696
    creating, 700
    deleting, 702
    modifying, 722
    removing entries from profile, 742
    starting, 716
    stopping, 704
    viewing characteristics, 728
dts_audit_events, 751
    ACL, 751
dtscp commands
    advertise, 696
    change, 698
    create, 700
    delete, 702
    disable, 704
    enable, 716
    exit, 718
    help, 719
    quit, 721
    set, 722
    show, 728
    summary, 694
    synchronize, 740
    syntax, 706
    unadvertise, 742
    update, 743
dtsd command, 710
dtsd process
    restarting, 710
dtsdate command, 713

E

endpoint
    administering, 227
Endpoint Map Service, 489
endpoint maps
    about, 489
    add or replace server address information, 510
    managing, 493
    removing elements, 535
    viewing elements, 548
endpoints
    about, 489
entities
    about, 581
entry point vector, 479
EPI_FORCE_INIT

dce_config environment variable, 158
EPI_FORMAT_PAR
dce_config environment variable, 158
epochs
changing, 698
EPV, 479
event class, 756
auditing execution, 751
definitions, 748, 751, 766
event class file, 756
format, 756
naming convention, 756
SEP line, 756
events
audit service operations, 748
audit services, 748
auditable, 748, 751, 766
clock readings, 751
dts_audit_events, 751
global-set membership, 751
security service operations, 766
security services, 766
time service attributes, 751
time service processes, 751
time services, 751
EXIT_ON_ERROR
dce_config environment variable, 152
extended registry attributes (ERAs), 780

F

files
auxiliary, 476
catalog, 4
client, 476
dts_audit_events, 746
event class, 756
event_class, 746
group_override, 746
header, 4, 6, 478
header file
serviceability, 4
input
sams command, 3
keytab, 787
message, 4
output, 4
sams command, 3
passwd_override, 746
problem determination, 4
reference page, 4
registry database override, 758, 761
sams, 2
sec_audit_events, 746
security administration, 746
server, 476
serviceability table, 4
stub, 476
uuidgen command, 488
v5srvtab, 746
functions
rdacl_get_access(), 768
rdacl_get_manager_types(), 769
rdacl_get_referral(), 769
rdacl_lookup(), 768
rdacl_replace(), 768
rdacl_test_access(), 769
rpriv_get_ptgt(), 770
rs_acct_add(), 770, 771
rs_acct_delete(), 771
rs_acct_get_projlist(), 772
rs_acct_lookup(), 771

rs_acct_replace(), 772
rs_auth_policy_get_effective(), 778
rs_auth_policy_get_info(), 778
rs_auth_policy_set_info(), 778
rs_login_get_info(), 772
rs_pgo_add(), 773
rs_pgo_add_member(), 775
rs_pgo_delete(), 773
rs_pgo_delete_member(), 776
rs_pgo_get(), 774
rs_pgo_get_members(), 776
rs_pgo_is_member(), 776
rs_pgo_key_transfer(), 775
rs_pgo_rename(), 774
rs_pgo_replace(), 773
rs_policy_get_info(), 777
rs_policy_set_info(), 777
rs_properties_get_info(), 777
rs_properties_set_info(), 777
rs_rep_admin_maint(), 779
rs_rep_admin_mkey(), 779
rsec_krb5rpc_sendto_kdc(), 767

G

gbl_time_service
    time server, 751
gdad command, 621
gdad process, 621
getcellname command, 239
getip command, 240
GID_GAP
    dce_config environment variable, 152
Global Directory Agent (GDA)
    starting daemon, 621
Global Directory Service (GDS)
    defining cell names, 662
global names
    conventions, 498
global servers
    removing entries, 742
group
    administering, 241
group_override file, 758
groups
    adding members, 828
    adding members to name service entries, 514
    adding to registry, 826
    changing registry information, 827
    deleting, 828
    naming, 502
    removing from NSI entry, 533
    removing members, 538
    viewing members, 545
    viewing registry information, 826

H

header file, 6, 478
headers, 5
host
    DCECP object, 255
host attributes
    administering, 276
HOST_NAME_IP
    dce_config environment variable, 152
hostdata

DCECP object, 266

I

identifiers
generator, 485
uuidgen command, 485
IDL, 476
base data types, 474
compiler, 476
file template
uuidgen command, 485
IDL-to-C data type mappings, 474
idl command, 474, 476
options, 476
IDL compiler, 474
idl_ macros, 474
idlbase.h, 475
INIT_LFS
dce_config environment variable, 158
INSTALL_OPT_CLIEN
dce_config environment variable, 158
INSTALL_OPT_SERS
dce_config environment variable, 158
installing DCE
start up command, 145
interface definition, 481
Interface Definition Language, 476
Interface Definition Language compiler, 474

K

k5dcelogin command, 807
k5login command, 808
kdestroy command, 809
KEYSEED
dce_config environment variable, 153
keytab
DCECP object, 281
keytab file, 787
kinit command, 810
klist command, 813
/krb5/v5srvtab file, 787
krb5rpc interface, 767

L

LAN_NAME
dce_config environment variable, 153
leaf names
conventions, 499
link
administering, 293
link entity, 679
LOAD_LFS_KEXT
dce_config environment variable, 158
local names
conventions, 498
overriding CDS syntax, 496
locksmith mode, 894
log
serviceability

administering, 301
LOGFILE
dce_config environment variable, 153
login
preventing, 764
LOW_GID
dce_config environment variable, 153
LOW_UID
dce_config environment variable, 154

M

macros
idl_, 474
marshalling, 480
master keys
creating, 855
messages
informational, 481
strings
sams command, 3
system files
sams command, 2
warning, 481, 483
MULTIPLE_LAN
dce_config environment variable, 154

N

name
administering, 307
Name Service Interface (NSI), 496
accessing, 489
command syntax, 497
importing binding information, 524
managing for RPC applications, 493
naming guidelines, 499
viewing NSI attributes, 542
NTP_HOST
dce_config environment variable, 154

O

object
administering, 312
object entity, 682
objects
adding attributes, 568
changing attribute values, 652
creating, 599
deleting, 613
removing attribute values, 637
viewing attributes, 682
viewing entries, 631
organization
administering, 320
organizations
adding members, 828
adding to registry, 826

changing registry information, 827
deleting, 828
viewing registry information, 826
orphans
adopting, 828

P

passwd_override file, 761
passwords
backing up files, 815
managing server, 834
storing server and machine, 787
preprocessor, 479
principal
administering, 337
principals
adding to registry, 826
deleting, 828
destroying login context, 809
setting security for, 805, 807, 808
viewing registry information, 826
privilege server
interfaces, 770
rpriv, 770
profiles
adding elements, 504
naming, 502
removing elements, 528
removing from namespace, 540
viewing elements, 552
programmer commands, 474
PWD_MGMT_SVR
dce_config environment variable, 154
PWD_MGMT_SVR_OPTIONS
dce_config environment variable, 154

R

rc.dce dce_config component script, 160
rc.dfs dce_config component script, 160
rdacl_get_access() function, 768
rdacl_get_manager_types() function, 769
rdacl_get_referral() function, 769
rdacl_lookup() function, 768
rdacl_replace() function, 768
rdacl_test_access() function, 769
rdaclif interface, 767
rdaclif operations, 768
rdacliftmp interface, 767
registry
administering, 349
local overrides, 758, 761
maintaining local, 838
registry administration, 779
interfaces, 779
registry database
creating, 868
recovery, 871
updating, 855
registry database override
files, 758, 761
registry miscellaneous operations
interfaces, 772
rs_misc, 772

registry objects
adopting, 828
registry PGO
interfaces, 773
rs_pgo, 773
registry policy
interfaces, 777
rs_policy, 777
registry server
interfaces, 770
rs_acct, 770
registry server attributes, 780
REMOVE_PREV_CONFIG
dce_config environment variable, 155
REMOVE_PREV_INSTALL
dce_config environment variable, 155
REP_CLEARINGHOUSE
dce_config environment variable, 155
replica sets
reconstructing, 646
replicas
creating, 601
deleting, 609
deleting (CDS), 615
viewing attributes (CDS), 684
rgy_edit subcommands
add, 826
adopt, 828
authpolicy, 838
change, 827
defaults, 838
delete, 828, 839
domain, 836
exit, 838
help, 838
ktadd, 834
ktdelete, 835
ktlist, 835
login, 838
member, 828
policy, 836
properties, 836, 839
purge, 839
quit, 838
scope, 838
site, 836
view, 826, 829, 839
rlogin command, 841
rlogind command, 845
ROOT_FILESET_NM
dce_config environment variable, 158
RPC
programmer commands, 474
RPC control program
adding entries to namespace, 508
enviornment variables, 495
initializing, 491
removing entries, 531
RPC daemon
about, 489
rpc_c_authn_dce_secret authentication service, 767
rpccp commands
add element, 504
add entry, 508
add mapping, 510
add member, 514
export, 516
help, 521
import, 524
remove element, 528
remove entry, 531
remove group, 533
remove mapping, 535
remove member, 538
remove profile, 540

scope, 495
show entry, 542
show group, 545
show mapping, 548
show profile, 552
show server, 556
summary, 491
unexport, 559
rpcentry
administering, 377
rpcgroup
administering, 389
rpcprofile
administering, 398
rpriv interface, 767
rpriv operations, 770
rpriv_get_ptgt() function, 770
rs_acct interface, 767
rs_acct operations, 770
rs_acct_add() function, 770, 771
rs_acct_delete() function, 771
rs_acct_get_projlist() function, 772
rs_acct_lookup() function, 771
rs_acct_replace() function, 772
rs_auth_policy_get_effective() function, 778
rs_auth_policy_get_info() function, 778
rs_auth_policy_set_info() function, 778
rs_login_get_info() function, 772
rs_misc operations, 772
rs_pgo operations, 773
rs_pgo_add() function, 773
rs_pgo_add_member() function, 775
rs_pgo_delete() function, 773
rs_pgo_delete_member() function, 776
rs_pgo_get() function, 774
rs_pgo_get_members() function, 776
rs_pgo_is_member() function, 776
rs_pgo_key_transfer() function, 775
rs_pgo_rename() function, 774
rs_pgo_replace() function, 773
rs_policy operations, 777
rs_policy_get_info() function, 777
rs_policy_set_info() function, 777
rs_properties_get_info() function, 777
rs_properties_set_info() function, 777
rs_query interface, 767
rs_rep_admin_maint() function, 779
rs_rep_admin_mkey() function, 779
rs_rpladmn interface, 767
rs_update interface, 767
rsec_cert interface, 767
rsec_krb5rpc_sendto_kdc() function, 767
rsh command, 848
rshd command, 852

S

sams command, 2
synopsis, 2
schema
administering, 458
SCM_NAME
dce_config environment variable, 158
sec_audit_events command, 766
SEC_SERVER
dce_config environment variable, 155
SEC_SERVER_IP
dce_config environment variable, 155
secidmap interface, 767
secrurity administration
auditable events, 746
security administration

files, 746
introduction, 746
security server
interfaces
krb5rpc, 767
rdaclif, 767
rdacliftmp, 767
rpriv, 767
rs_acct, 767
rs_query, 767
rs_rpladmn, 767
rs_update, 767
rsec_cert, 767
Security Servers
administering, 855
security servers
about, 892
Security Service commands
acl_edit, 790
dce_login, 805
group_override, 758
k5dcelogin, 807
k5login, 808
kdestroy, 809
kinit, 810
klist, 813
passwd_export, 815
passwd_import, 817
passwd_override, 761
pwd_strengthd, 822
rlogin, 841
rlogind, 845
rsh, 848
rshd, 852
sec_admin, 855
sec_create_db, 868
sec_salvage_db, 871
summary, 788
security service commands
/krb5/v5srvtab, 787
secd, 892
security services
auditable events, 766
secval
administering, 412
SEP line
event class file, 756
server
administering, 418
auxiliary file, 476
files, 476
stub, 476
server entity, 689
servers
naming, 500
serviceability table
sams command, 6
skulking
startup command, 648
soft links
changing values, 650
creating, 597
deleting, 611
removing timeout value attribute, 635
viewing, 629
viewing attributes, 679
stub
client, 476
server, 476
svcdumplog command, 12
symbols
sams command, 3
SYNC_CLOCKS
dce_config environment variable, 156

T

ticket granting tickets
    obtaining and caching, 810
tickets
    viewing cached, 813
time server
    clock readings, 751
    gbl_time_service, 751
    global-set membership, 751
    time service, 751
    time service attributes, 751
    time service processes, 751
    time_control, 751
    time_provider, 751
    time_service, 751
time service
    Even-Specific Information, 751
    Event Classes, 751
    Event Types, 751
    interfaces, 751
    time server, 751
time services
    auditable events, 751
time-provider
    interfaces, 755
time_control
    time server, 751
time_provider
    time server, 751
TIME_SERVER
    dce_config environment variable, 156
time_service
    time server, 751
timestamps
    format, 709
TOLERANCE_SEC
    dce_config environment variable, 156

U

UID_GAP
    dce_config environment variable, 156
UNCONFIG_HOST_PRESET
    dce_config environment variable, 157
Universal Unique Identifier, 485
user
    DCECP object, 435
utc
    administering, 448
UUID, 485
    version number, 485
uuid
    administering, 454
uuidgen command, 474
    arguments, 485
    IDL, 485

V

variables
    in rpccp, 495
version number
    UUID generator, 485

X

administering, 458

xattrschema