OSF DCE SIG **R. Salz (OSF)**
Request For Comments: 47.1 **P. Wang (OSF)**
January 1994

DCED: THE DCE HOST DAEMON — COMMAND SET

1. INTRODUCTION

This document specifies the commands that will be available to control `dced`, the DCE Daemon. The commands will be part of `dcecp`, the DCE Control Program [RFC 42.1].

The commands defined here will be built into the `dcecp` executable and will follow the model and style presented by that program. If [RFC 42.1] and this document conflict, [RFC 42.1] is the overriding authority. Later versions of the RFC 42 series (presumably beginning with revision RFC 42.2) will incorporate the contents of this document, and this document will no longer be separately issued.

Readers should be familiar with the `dcecp` and `dced` functional specifications ([RFC 42.1], [RFC 47.0]).

2. COMMON FACILITIES

This section details the syntax and capabilities that the `dced` objects share either with each other, or with other `dcecp` objects.

2.1. Naming

Every object controlled by `dced` has a name. Some services (`hostdata` and `keymgmt`) do this by having a table that is directly available, while others (`srvrconf` and `srvrexec`) do it implicitly by having a ''name'' field in the internal state of each object.

The naming model provided by `dced` is junction-like, with only a single entry in the namespace. Full junction support is not implemented, but the appearance of a unified namespace is provided by the `sec_acl_bind` and `dce_host_bind` routines. This covers the most common cases: doing `dced`-specific operations (such as creating a new object), and modifying an object's ACL.

The full name of a `dced` object is as follows:

/.../*cell*/*hostname*/config/*service*/*name*

where:

`cell`	is the name of the cell
`hostname`	is the DCE name of the host by convention, `hosts/`*a-simple-name*
`service`	is the name of the service maintaining the object, such as `hostdata` or `srvrconf`
`name`	is the name of the object maintained within `dced`

When referring to objects in the local cell, a shorter form of the name can be used. That is *cell* is local, the *hostname* can be specified by a switch, and the *service* is known from the `dcecp` object. This means that in most cases the following can be used:

name −host *hostname*

or, if referring to the local host, the following can be used:

name

For example, the following names are equivalent if run on the DCE host `sulphur` (the first two are always equivalent):

```
/.../dced-dev/hosts/sulphur/config/keymgmt/v5srvtab
v5srvtab -host hosts/sulphur
v5srvtab
```

In this document a ‘‘name’’ should be taken to mean any of the three formats.

2.2. Common Verbs and Switches

The following verbs and switches will be implemented to have their standards `dcecp` meaning:

```
operations
help [-verbose] [operation]
```

All verbs set their subscript in the `$s` convenience variable array to the full name of the object to which they refer.

3. DATA TYPES

The DCE daemon introduces some new data types. These data types will only have a TCL syntax.

The DCE daemon maintains lists of the objects it maintains, usually in a `hostdata_t`. This will be represented as a four-element TCL list:

```
{
    /* UUID, name, description, storage-tag */
    ea145db8-0dd7-11cd-9526-08000925634b "name" "description"
        "remote file name"
}
```

In cases where an object is being created, an empty string for the UUID will mean that a UUID should be generated.

The DCE daemon supports attributes, where each attribute has a UUID, a name, and a value chosen from a small set of primitive datatypes. This will be represented using the standard `dcecp` syntax for attributes. The UUID and ‘‘type of data’’ fields can be omitted because they will be known from context.

The contents of a `hostdata` file will be represented as a `dcecp` attribute named `filecontents` whose value will be an array of strings, each string representing one line of the file.

The `service_t` datatype contains RPC binding information and a list of objects. It is represented using the following syntax:

```
{
    interface_spec interface_name annotation
    { list of endpoints }
    flags
    { list of object uuids }
    namespace_entry_name
}
```

The `server_t` datatype contains a set of fixed attributes, a list of variable attributes, and a set of services. These are described in [RFC 47]; some fields (and therefore the attributes) have not yet been specified. These are indicated below with `(TBD)`:

```
{
    /* Object UUID and name */
    5f13854a-0dd2-11cd-a6a6-08000925634b "print"
    { service list }
    { fixed attributes (TBD) }
    { /* Variable attributes */
        attribute list
    }
    false { running server fixed attributes (TBD) }
}
```

This is not a convenient data struct to enter interactively. The namespace project is working on defining a method of creating server ''templates''.

A `key` contains a principal, a version, and tagged data which is represented as a `dcecp` attribute The only accepted tag is `des` which means that the value is a TCL string or an array of bytes (if the string contains non-printable data):

```
{
    /* principal_name, version, data */
    rsalz 1 { des braedon }
}
```

4. THE SERVER OBJECT

The `server` object refers to either the static configuration data for a server or the dynamic, running, server process.

When using the long form of a name, the user may enter either `srvrconf` or `srvrexec` as the service name. It is an error if the name is not appropriate for the verb, such as trying to `delete` a `srvrexec` object. The user may also use the ambiguous term `server` and the command will internally map this to the right service. If the verb is appropriate for either service and the user uses the ambiguous name, then they must also use a switch to disambiguate.

The following verbs are supported:

(a) `server create` *`name server`*

Creates the named object from the TCL syntax of a `server_t` and returns the UUID of the created object.

(b) `server catalog [-configured | -executing] [-host` *`hostname`*`]`

Returns a list of the names of all objects on the specified host in arbitrary order. Either the `-configured` `-executing` switch must be specified if the `name` is ambiguous.

(c) `server show` *`name`* `[-configured | -executing]`

Returns the TCL syntax of the `server_t` of the named object. If `name` is a list the output is concatenated into a single list in the order specified. Either the `-configured` `-executing` switch must be specified if the `name` is ambiguous.

(d) `server modify` *`name`* `[-configured | -executing]`
`-change` *`attribute_list`*

Modifies the named object, returning an empty string on success. Either the `-configured` `-executing` switch must be specified if the `name` is ambiguous.

(e) `server delete` *`name`*

Deletes the named object, returning an empty string on success. If `name` is a list all named objects are deleted.

(f) `server enable` *`name`* `-interface` *`if_id`*

Enables the specified interface for the named server, returning an empty string on success.

(g) `server disable` *`name`* `-interface` *`if_id`*

Disables the specified interface for the named server, returning an empty string on success.

(h) `server start` *`name`*

Starts a server based on its `dced srvrconf` object, returning the UUID of the running server object. If `name` is a list the specified servers are started in the order given.

(i) `server stop` *`name`* `[-method` *`stop_method`*`]`

Stops the named server, returning an empty string on success. If `name` is a list the specified servers are stopped in the order given. The `-method` switch directs how `dced` should stop the server. The value should be chosen from the following list; the default value is `rpc`.

`rpc`	Use `rpc_mgmt_server_stop_listening`
`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)

5. HOSTDATA

Note that only the `filecontents` attribute is supported in the first release.

(a) `hostdata catalog [-host` *`hostname`*`]`

Returns a list of the names of all objects on the specified host in arbitrary order.

(b) `hostdata create` *`name hostadata`* `[-entry | -data` *`attributes`*`]`

Creates a hostdata entry and the data associated with it (typically stored in a remote file), returning the UUID of the created object. If the `-entry` switch is present, only the hostdata entry is created. Either the `-entry` or the `-data` switch must be specified.

(c) `hostdata delete` *`name`* `[-entry]`

Deletes a hostdata entry and its data, returning an empty string on success. If `name` is a list all named objects are removed in the order specified. If the `-entry` switch is present, only the hostdata entry is deleted, not the ‘‘backing storage’’ specified by the storage tag.

(d) `hostdata add` *`name`* `-data` *`attributes`*

Adds the specified attribute and data to a hostdata entry replacing any existing data that might be there, returning an empty string on success. If `name` is a list the attribute and data are added to each object in the order specified.

(e) `hostdata remove` *`name attribute_name`*

Removes the data from a hostdata entry, returning an empty string on success. If `name` is a list the attribute and data is removed from each object in the order specified.

(f) `hostdata show` *`name`* `[-entry]`

Returns the contents of a hostdata item. Returns just the four fields of the entry if the `-entry` switch is present. If `name` is a list the output is concatenated into a single list in the order specified.

6. KEYTAB

The `keytab_t` datatype is the same as the `hostdata_t` datatype.

(a) `keytab create` *name keytab* `[-entry | -data` *attributes*`]`

Creates a key table, returning the UUID of the created object. Either the `-entry -data` switch must be specified.

(b) `keytab delete` *name* `[-entry]`

Deletes a key table entry and its keys, returning an empty string on success. If `name` is a list all named objects are removed in the order specified. If the `-entry` switch is present, only the keytable entry is deleted, not the ‘‘backing storage’’ specified by the storage tag.

(c) `keytab add` *name* `-principal` *principal*
`[-key` *key*`|-randomkey] [-registry]`

Adds a key for a principal in a specified key table, returning an empty string on success. If `name` is a list the key is added to each key table in the order specified; note that if the `-randomkey` is specified a single, random, key will be used for all objects. Either the `-key` or `-randomkey` switch must be specified. The `-registry` switch will update the principal’s key in the registry as well.

(d) `keytab remove` *name* `-principal` *principal* `[-version` *version*`]`

Removes a principal’s key from a specified table, returning an empty string on success. If `name` is a list the key is removed from each object in the order specified. If the version is not specified, all versions will be removed.

(e) `keytab list` *name* `[-principal` *principal*`]`

Returns all the keys in the specified table. If `name` is a list the output is concatenated into a single list in the order specified. If the `-principal` switch is used, only that principal’s keys are listed.

(f) `keytab catalog [-host` *hostname*`]`

Returns a list of the names of all objects on the specified host in arbitrary order.

(g) `keytab show` *name* `[-entry]`

Returns a list of keys, each in TCL syntax. If `name` is a list the output is concatenated into a single list in the order specified. If the `-entry` switch is used, then only the four fields of the entry are returned.

7. SECVAL

All `secval` operations return an empty string on success.

(a) `secval ping [-host` *hostname*`]`

Makes an RPC to the secval service at the specified host to complete the authentication trust chain.

(b) `secval start [-host` *hostname*`]`

Starts the security validation service on the specified host.

(c) `secval stop [-host` *hostname*`]`

Stops the security validation service on the specified host.

8. ACKNOWLEDGEMENTS

Howard Melman gave careful review to early drafts of this document to ensure that it follows the `dcecp` conventions.

REFERENCES

[RFC 42.1] H. Melman, DCE-RFC 42.1, ''DCE Shell Functional Specification'', January 1994.

[RFC 47.0] D. Mackey, R. Salz, DCE-RFC 47.0, ''DCED: The DCE Host Daemon — Functional Specification'', December 1993.

AUTHORS' ADDRESSES

Rich Salz
Open Software Foundation
11 Cambridge Center
Cambridge, MA 02142
USA
Internet email: rsalz@osf.org
Telephone: +1-617-621-7253

Ping Wang
Open Software Foundation
11 Cambridge Center
Cambridge, MA 02142
USA
Internet email: pwang@osf.org
Telephone: +1-617-621-8988