DCLI User's Guide

Data Center Command-Line Interface 2.10.2

vmware®

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

If you have comments about this documentation, submit your feedback to

docfeedback@vmware.com

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

About This Book

DCLI User's Guide gives an overview of DCLI (Data Center CLI) commands, syntax, and options. This guide also includes instructions for setting up DCLI and provides examples.

Intended Audience

This guide is for experienced system administrators who are familiar with data center operations.

Introduction to DCLI

1

DCLI (Data Center CLI) is a simplified command-line interface that you can use to automate tasks in your VMware Cloud on AWS, NSX-T, and vCenter Server environments.

Note DCLI supports running NSX-T commands only against NSX-T in VMware Cloud on AWS environments.

This chapter includes the following topics:

Using DCLI

You can use DCLI to make quick calls to VMware Cloud on AWS, NSX-T, or vCenter Server API commands through the shell.

DCLI is designed to enable the completion of simple API tasks by administrators and end users. You can get a quick overview of the VMware Cloud on AWS, NSX-T, or vCenter Server API by using the drop-down menu and auto-completion features of DCLI.

Supported Platforms

DCLI is available on the PyPI repository and you can install it by using the `pip` command. DCLI can run on all systems that support Python 2.7 or later.

Install DCLI

You can install DCLI by running the `pip` command which downloads the latest version of the software from the PyPI repository.

You should not run the `pip` command in conjunction with the `sudo` command. If you do not have write permissions, you should first try installing DCLI by using the `––user` option.

Prerequisites

- Verify that Python 2.7 or later is installed on your system.

- Verify that your system has Internet access.

Procedure

- Install DCLI by running the `pip` command.
  - If you have root write permissions, run the following command.

    ```
    pip install dcli
    ```

  - If you do not have root write permissions, run the following command.

    ```
    pip install --user dcli
    ```

    Note This command installs DCLI to the Python user install directory of your platform. The default path is `%APPDATA%\Python` on Windows and `~/.local/` on other platforms.

Running DCLI Commands

2

You can run DCLI commands on Windows, Linux, and Mac OS.

DCLI is compatible with the built-in command-line applications of each operating system. On Windows, you can use the command prompt. On Linux and Mac OS, you can use Bash. You can also run DCLI commands by using custom command-line applications on each platform.

This chapter includes the following topics:

Overview of Running DCLI Commands

You can run DCLI commands interactively or in scripts.

DCLI supports running commands in either interactive or non-interactive mode.

- Interactive mode allows for running commands quickly.
- Non-interactive mode is suitable for scripting purposes.

DCLI Syntax

Each DCLI command uses the same syntax.

The command name can be followed by DCLI connection and formatting options, each preceded by a plus (+) sign. You also specify the namespace, the command, and the command options. Namespaces are nested.

Note The order in which DCLI options are provided on the command line is not important. However, you must specify DCLI options with a plus (+) and command-specific options with two minus signs (––).

The syntax of a DCLI command is the following.

```
dcli +[DCLI options] <namespace> [<namespace> ...] <cmd> ––[cmd option] [option value]
```

The following table describes the DCLI syntax elements.

Syntax Element	Description
`DCLI options`	Predefined options for connection information and formatting options. Always preceded by a plus (+) sign.
`namespace`	Groups DCLI commands. Namespaces are nested.
`command`	Reports on or modifies the state of the system.
`option` and `value`	Command option and value pairs preceded by two minus signs (––).

Examples

```
dcli +vmc-server com vmware vmc orgs sddcs list ––org orgID
```

```
dcli +nsx-server <mynsxtaddress> com vmware nsx policy api v1 infra services list
```

```
dcli +server myvc +username user42 com vmware cis tagging tag list
```

DCLI Options

You can run each DCLI command with connection or formatting options preceded by a + sign.

For many of the options, you can instead use variables. See Using DCLI with Variables.

```
dcli [+vmc-server]
     [+nsx-server NSX_T_SERVER_ADDRESS]
     [+server SERVER_IP]
     [+interactive]
     [+prompt PROMPT]
     [+skip-server-verification]
     [+cacert-file CACERT_FILE]
     [+username USERNAME]
     [+password PASSWORD]
```

```
      [+filter FILTER]
      [+formatter {simple,table,xml,json,html,csv}]
      [+verbose]
      [+loglevel {debug, info, warning, error}]
      [+credstore-file CREDSTORE_FILE]
      [+credstore-add | +credstore-list | +credstore-remove]
      [+session-manager SESSION_MANAGER]
      [+configuration-file CONFIGURATION_FILE]
      [+more]
      [args [args ...]]
```

With these options, you can provide the following information. If you are entering options in DCLI interactive mode, tab completion is supported on Linux, Windows, and Mac OS systems. In all cases, you can specify a partial option if the option is not ambiguous. For example, `+i` indicates interactive, but you have to specify, at least, `+credstore-a` to disambiguate `option +credstore-add`.

The following table describes the DCLI options.

Option	Description	Default	Available in Interactive Mode
`server`	The vCenter Server system to which DCLI connects.	`localhost`	Yes
`vmc-server`	The VMware Cloud on AWS server to which DCLI connects.	https://vmc.vmware.com	Yes
`nsx-server`	The NSX-T server to which DCLI connects.		Yes
`interactive`	Runs DCLI in interactive shell mode, which supports tab completion of commands, options, and some option values. It also supports saving the command history across DCLI sessions. Interactive mode is faster because DCLI caches the list of commands available on a vCenter Server system.		No
`prompt`	Prompt that the interactive shell uses.	`dcli>`	No
`skip-server-verification`	Skips the server SSL verification process.	`False`	Yes
`cacert-file`	Specifies the certificate authority certificates for validating SSL connections.		Yes

Option	Description	Default	Available in Interactive Mode
`username`	If you run from the local shell, most DCLI commands do not require the user name. If you are running vCLI commands, the user you specify must be able to authenticate to the vCenter Server system. The user you specify must have the privileges to perform the task, as specified through vCenter Server roles. You are prompted for the password. The password is not echoed to screen. **Note** Only available when DCLI connects to a vCenter Server system.		Yes
`password`	You can use this option to provide your password explicitly instead of waiting for the DCLI password prompt. **Important** Providing the value explicitly presents a security risk. **Note** Only available when DCLI connects to a vCenter Server system.		Yes
`filter`	Provides JMESPath expressions to filter command output.		Yes
`formatter`	Output formatter, which has one of the following possible values. ■ simple ■ table ■ xml ■ json ■ html ■ csv	Default is `table` for lists of structures and `simple` for all other output including `json`.	Yes
`verbose`	Prints verbose output.	`False`	Yes
`loglevel`	The log level, which has one of the following possible values. ■ debug ■ info ■ warning ■ error	`info`	Yes

Option	Description	Default	Available in Interactive Mode
`credstore-file`	Path to the credential store file to use for credential store operations or for reading login credentials. Use this option only if the default credential store filename does not work in your environment. By default, the credential store file is in the `.dcli/.dcli_credstore` directory inside the home directory.	`$HOME/.dcli/.dcli_credstore`	Yes
`credstore-add`	Adds login credentials entered for a command to the DCLI credential store file. If the provided credentials are valid, this option stores the server IP address, session manager, user name, and password for the command being run. If an entry exists, the command updates the entry.		Yes
`credstore-list`	Lists all entries stored in the DCLI credential store file. Each entry includes the server IP address, session manager, and user name.		Yes
`credstore-remove`	Removes an entry from the DCLI credential store file. This option removes the entry for a specified server IP address and user name if only one session manager is present for a target server and user. In rare cases, information about multiple session manager entries is present. You must provide the session manager with the `session-manager` option.		Yes
`session-manager`	Use this option if you use the `credstore-remove` option the same user name and password are stored through multiple session managers. Not usually required.		Yes
`configuration-file`	Path to the configuration store file to use for options and profile-related internal commands.	`$HOME/.dcli/.dcli_configuration`	Yes
`more`	Displays page-wise output.		Yes

DCLI Option Scopes

DCLI option values are preserved based on their scope.

There are three categories of logical scopes.

- Interactive session scope
- Connection session scope
- Command call scope

You create an interactive session when you enter interactive mode. The interactive session scope keeps the values of all DCLI options you pass to the console. If you do not pass values explicitly, the interactive session scope keeps the default option values. All commands that you run in this session use the DCLI option values preserved in the interactive session scope.

You create a connection session when you connect to a server. If the connection is established while entering interactive mode, both the interactive and connection sessions use the same values for the DCLI options. However, if the connection is established after entering interactive mode and the provided DCLI options are different than the interactive session values, all commands within that connection use the connection session values as defaults for the DCLI options.

The command call scope applies to values that you pass explicitly. Every DCLI option value that you pass explicitly to a command overrides both the connection and interactive session values.

The values of the following DCLI options are preserved in interactive and connection sessions.

- `username`
- `password`
- `cacert-file`
- `credstore-file`
- `credstore-add`
- `configuration-file`
- `more`
- `formatter`
- `verbose`
- `loglevel`

The `skip-server-verification` option applies only to the connection session scope. The value of this option is not preserved in the interactive session scope for security reasons.

The values of the following DCLI options are preserved in the interactive session scope only.

- `verbose`
- `loglevel`

Run DCLI Commands in Interactive Mode

DCLI supports interactive shell mode which you can activate by using `dcli +interactive`.

Interactive mode supports drop-down autocompletion of namespaces, commands, command options, and option values in case of enumeration values. With DCLI interactive mode, you can also pass a short command if it is uniquely resolvable. For example, `dcli> com vmware vmc vm list` can also be run as `dcli> vm list`.

Interactive mode is also a quicker way to browse various namespaces and commands, as DCLI caches the list of namespaces and commands available on the server for faster access. DCLI interactive mode provides specific shell commands which can be accessed by running `dcli> help`.

You can change the prompt for DCLI interactive mode by using `dcli +interactive +prompt <user-prompt>` when entering interactive mode.

Procedure

1. From the command line, navigate to the location of the DCLI binary.
2. Enable interactive mode.

   ```
   dcli +vmc +i
   ```

   DCLI connects to the VMware Cloud on AWS server in interactive mode.
3. List all SDDCs in a specified organization.
   - Provide the full interactive mode command.

     ```
     dcli> com vmware vmc orgs sddcs list --org orgID
     ```

   - Provide the short interactive mode command.

     ```
     dcli> orgs sddcs list --org orgID
     ```

   Note Both commands should return the same result.

Run DCLI Commands in Non-Interactive Mode

When you run DCLI in non-interactive mode, you must provide the full command, including the DCLI options.

Procedure

1. From the command line, navigate to the location of the DCLI binary.
2. List all SDDCs in a specified organization.

   ```
   dcli +vmc com vmware vmc orgs sddcs list --org orgID
   ```

Displaying Help Information for DCLI Commands

You can display help for each namespace and command by using the `--help` command-line option.

Because the available commands depend entirely on the services that are available in the vCenter environment that you are targeting, you must include the server for accurate help information. For VMware Cloud on AWS and NSX-T, the available commands depend on the services that are available in the REST API that you are targeting.

Help returns the following information for a command.

- Each input option
- Whether the option is required
- Input type

Example

```
dcli> com vmware vmc orgs sddcs list --help
usage: com vmware vmc orgs sddcs list [-h] [--org ORG]

Lists all SDDCs of an organization

Input Arguments:
  -h, --help  show this help message and exit
  --org ORG   Organization identifier (required) (string)
```

DCLI Authentication

Most DCLI commands require authentication. VMware Cloud on AWS, NSX-T, and vCenter Server use different authentication mechanisms.

VMware Cloud on AWS and NSX-T use a refresh token for authentication. vCenter Server requires credentials, which you can provide in different ways.

VMware Cloud on AWS and NSX-T Authentication

You can authenticate to VMware Cloud on AWS or NSX-T by using a refresh token.

You can obtain a refresh token from your VMware Cloud on AWS user profile under the Access Tokens area, or from the VMware Cloud on AWS administrators in your organization. After you receive a refresh token, you can provide it to DCLI when prompted. DCLI prompts for a refresh token when you try to access the VMware Cloud on AWS or NSX-T API.

The following example shows syntax for VMware Cloud on AWS.

```
dcli +vmc +i
Refresh Token:
```

The following example shows syntax for NSX-T.

```
dcli +nsx <mynsxtaddress> +i
Refresh Token:
```

The refresh token is provided in a secure way. Be careful that you are using the copy/paste functionality of your terminal correctly to avoid pasting the refresh token more than once. After you provide the refresh token, DCLI asks whether you want to save it to the credential store. If you choose to save the refresh token, you will not need to provide it each time you connect to VMware Cloud on AWS or NSX-T.

Updating the VMware Cloud on AWS Refresh Token

The following example illustrates how you can update your VMware Cloud on AWS refresh token.

1 Remove the old refresh token.

```
dcli +vmc +credstore_remove
```

2 Connect to VMware Cloud on AWS and provide the new refresh token when prompted.

```
dcli +vmc
Refresh Token:
```

Updating the NSX-T Refresh Token

The following example illustrates how you can update your NSX-T refresh token.

1 Remove the old refresh token.

```
dcli +nsx <mynsxtaddress> +credstore_remove
```

2 Connect to NSX-T and provide the new refresh token when prompted.

```
dcli +nsx <mynsxtaddress>
Refresh Token:
```

vCenter Server Authentication

You can authenticate to vCenter Server by providing a user name and password.

You can provide credentials in different ways.

- Specify the `+username` option.

```
dcli> vcenter vm list +username user42 +password mypass
```

Important If you skip the `+password` option, you can provide the password in a more secure way when prompted. Providing the `+password` value explicitly presents a security risk.

- Specify the *DCLI_USERNAME* environment variable.

```
export DCLI_USERNAME=user42
dcli +server <vcenter_server_url> com vmware vcenter vm list
```

- Provide credentials when prompted by DCLI.

Note You are prompted for credentials if the operation requires authentication.

```
dcli> vcenter vm list
Username: user42
Password:
Do you want to save credentials in the credstore? (y or n) [y]:
```

- Save the credentials in the credential store.

Note You must provide the command against which the authentication should be applied.

```
dcli +server <vcenter_server_url> +credstore-add +username user42 +password mypass com vmware
vcenter vm list
```

After you save the user name and password in the credential store, you will not need to provide credentials each time you connect to vCenter Server.

Using DCLI with a Credential Store File

To avoid entering the user name and password each time you run a DCLI command, you can add the current credentials or refresh token and server IP address to a credential store file by using the `credstore-add` option on the command line.

Passwords and refresh tokens are encrypted in the credential store file. If you want to remove credential store information, you can use `+credstore-remove` to do so.

By default, the credential store file is located in `$HOME/.dcli/.dcli_credstore`, but you can change the location with the `+credstore-file` option.

Examples

The following examples illustrate how you can interact with the credential store.

- Add a new credential store entry.

```
dcli com vmware cis tagging tag list +credstore-add +username user1
```

- Remove a credential store entry.

```
dcli +credstore-remove +server <vcenter_server_url> +username user1
```

- List all credential store entries.

```
dcli +credstore-list
```

Order of Precedence for DCLI Authentication to vCenter Server

When you run a DCLI command, authentication happens in order of precedence, which always applies. That means, for example, that you can override an environment variable setting from the command line.

The following table shows the DCLI authentication precedence order.

Authentication	Description
Command line	The user name and password specified on the command line take precedence, even if a credential store exists.
Environment variable	A user name specified in an environment variable takes precedence over user names in the credential store, but not over the command line.
Credential store	The user name and password retrieved from the credential store. A custom credential store file at a non-default location has precedence over a file at the default location.

Connect to an NSX-T Endpoint

To establish a connection to an NSX-T server, you must retrieve the URL of the endpoint to connect to. This endpoint address is returned from an SDDC property called `nsx_api_public_endpoint_url`.

Procedure

1. Retrieve the URL of the NSX-T endpoint.

```
dcli +vmc com vmware vmc orgs sddcs get --org <orgId> --sddc <sddcId> +filter
resource_config.nsx_api_public_endpoint_url
```

2. Establish a connection to the NSX-T server by providing the retrieved endpoint URL.

```
dcli +nsx <sddc_nsx_endpoint_url> +skip +i
```

Multi-Server Support in DCLI

DCLI supports establishing simultaneous connections to vCenter Server, VMware Cloud on AWS, and NSX-T endpoints. Currently, one instance of each connection type at the same time is supported.

You can establish multi-server connections only in interactive mode. In non-interactive mode, you can establish only one connection by using either environment variables or DCLI (+) options.

Example: Connecting to Multiple Servers Simultaneously

The following example shows how you can connect to VMware Cloud on AWS and NSX-T with a single command.

```
dcli +vmc +nsx <sddc_nsx_endpoint_url> +skip +i
```

Note When using this syntax, both connections skip certificate validations. If you want only one of the connections to skip certificate validations, you must connect to the servers sequentially.

Example: Connecting to Multiple Servers Sequentially

The following example shows how you can connect to VMware Cloud on AWS, NSX-T, and vCenter Server with separate commands. The example also includes running commands against the NSX-T and vCenter Server environments after establishing the connections.

```
dcli +vmc +i
dcli> +nsx <sddc_nsx_endpoint_url>
dcli> +server <sddc_vcenter_server_url>
dcli> nsx policy api v1 infra networks list
dcli> vcenter vm list
```

Filtering DCLI Command Output

You can filter command output to trim unnecessary information or use custom formatting.

DCLI supports output filtering by using JMESPath expressions. JMESPath is a standard query language for JSON. For more information about JMESPath, see http://jmespath.org/.

In the following example, you retrieve the NSX-T URL from an SDDC and connect to the NSX-T server in interactive mode.

```
dcli +vmc com vmware vmc orgs sddcs get --org <orgId> --sddc <sddcId> +filter
resource_config.nsx_api_public_endpoint
<sddc_nsx_endpoint_url>
dcli +nsx <sddc_nsx_endpoint_url> +skip +i
```

After establishing a connection, you can print out the names and IDs of all VMware Cloud on AWS SDDCs in an organization as a table. You can give custom names to the output columns.

```
dcli +vmc com vmware vmc orgs sddcs list --org <orgId> +filter '[].{"Organization ID": id,
"Organization Name": name}' +formatter table
|-----------------|------------------|
|Organization ID |Organization Name|
|-----------------|------------------|
|<orgId>          |<orgName>         |
|-----------------|------------------|
```

You can modify JMESPath expressions by using the `jpterm` program. To use `jpterm`, you must install the JMESPath Terminal by running the following command.

```
pip install jmespath-terminal
```

The `jpterm` program requires command output in JSON format. The following example converts the output to JSON format before sending it to `jpterm`.

```
dcli +vmc com vmware vmc orgs list +formatter json | jpterm
```

Using the DCLI Configuration Store

The configuration store is a store for key-value tuples used to set default values for command options.

The configuration store has profiles. Profiles isolate key-value pairs. DCLI uses a default profile. You can specify the default profile by using the `default_profile` key.

In the following example, the default profile is labeled `default`. Each profile first specifies key-value items for a given component. The possible component values are `vsphere`, `vmc`, and `nsx`. This example contains the `vmc` and `vsphere` components. The profile then specifies the server you want the key-value pair to be valid for, and also that this pair should be used for the default options DCLI functionality. For more information about default options, see Using Default Options.

Example: Configuration Store Syntax

```
{
    "configuration": {
        "version": "1.0",
        "profiles": {
            "default": {
                "vmc": {
                    "https://vmc.vmware.com": {
                        "default_options": {
                            "org": "myOrgId"
                        }
                    }
                },
                "vsphere": {
                    "https://myVsphereIp": {
                        "default_options": {
                            "vm": "myVMID"
                        }
                    }
                },
            }
        },
        "default_profile": "default"
    }
}
```

Configuration Store Path

The default DCLI configuration store path is `$HOME/.dcli/.dcli_configuration`. You can set a different configuration store path by using either the *DCLI_CONFIGFILE* environment variable or the `+credstore-file` option.

Internal DCLI Commands

DCLI provides specific internal commands. You can use internal commands to alter the configuration store, active profile, and default options.

Internal commands are accessible through the first level of the `env` namespace. The following table lists all internal commands.

Internal Command	Description
`env profiles default set`	Sets the default profile.
`env profiles default get`	Retrieves the current default profile.
`env profiles add`	Adds a new profile to the configuration store.
`env profiles get`	Retrieves information about the specified profile.
`env profiles list`	Lists all available profiles.
`env profiles delete`	Deletes the specified profile.
`env options set`	Sets the specified default option to a specified value.
`env options get`	Retrieves the specified default option.
`env options delete`	Deletes the specified default option.
`env options list`	Lists all default options.
`env about get`	Retrieves information about the DCLI version, build number, and Python version. You can also retrieve this information by running `dcli --version`.

Using Default Options

You can use default options to provide predefined values for specific options.

The following examples show the full commands, which contain a complex value for the `org` parameter.

```
dcli> com vmware vmc sddcs list --org <someReallyLongOrgID>
```

```
dcli> com vmware vmc orgs get --org <someReallyLongOrgID>
```

By using default options, you can set the `org` parameter as a default option and skip adding it to the command.

```
dcli> env options set --option org --value <someReallyLongOrgID>
```

Performing this operation allows you to run the same commands without specifying the parameter value and still receive the same results.

```
dcli> com vmware vmc sddcs list
```

```
dcli> com vmware vmc orgs get
```

If you want to provide a value for the `org` parameter that is different to the value set for the default option, you can specify it manually.

```
dcli> com vmware vmc sddcs list --org <anotherReallyLongOrgID>
```

Input, Output, and Return Codes

DCLI supports the following input arguments.

Basic types — You can enter basic types like string, int, double, or boolean on the command line.

List types — You can provide the same option multiple times on the command line and DCLI treats it as a list.

Currently supported output formatter types are simple, xml, html, table, csv and json. You can change the output format by passing the `formatter` option to DCLI.

For scripting purposes DCLI returns a non-zero error code for an unsuccessful command. To see the last command status in interactive mode, run the `$?` command.

Using DCLI with Variables

You can predefine a set of variables in the environment where you run DCLI commands so you do not have to pass the options every time you run a command. The following environment variables are supported.

Variables Supported by DCLI

Variable	Description
DCLI_VMC_SERVER	Set this variable to pass the VMware Cloud on AWS server IP address.
DCLI_NSX_SERVER	Set this variable to pass the NSX-T server IP address. Passing the `nsx-server` option on the command line overrides this variable.
DCLI_SERVER	Set this variable to pass the vCenter Server system IP address. Passing the `server` option on the command line overrides this variable.
DCLI_CACERTFILE	Set this variable to pass the path of a CA certificate file. Passing the `cacert-file` option on the command line overrides this variable.

Variable	Description
DCLI_USERNAME	Set this variable to pass the user name required for authentication. Passing the `username` option on the command line overrides this variable.
DCLI_CREDFILE	Set this variable to point to a DCLI credential store file. Default value is `~/.dcli/.dcli_credstore`. Passing the `credstore-file` option on the command line overrides this variable.
DCLI_HISTFILE	Set this variable to point to a DCLI interactive shell history file path. Default value is `~/.dcli/.dcli_history`.
DCLI_LOGFILE	Set this variable to specify the log file for DCLI.
DCLI_CACERTS_BUNDLE	Set this variable to specify the path to a trust store. By default, DCLI includes a trust store from Certifi.
DCLI_VMC_CSP_URL	Set this variable to specify the URL to the CSP service for authentication token retrieval from a given refresh token. The default URL is https://console.cloud.vmware.com.
DCLI_VMC_METADATA_URL	Set this variable to specify metadata URL location. By default, DCLI receives metadata from VMware Cloud on AWS for API specifics such as available commands.
DCLI_VMC_METADATA_FILE	Set this variable to specify metadata file location. If set, DCLI retrieves metadata from the specified file.
DCLI_NSX_METADATA_URL	Set this variable to specify metadata URL location used for connections to NSX-T. By default, DCLI receives metadata from a location relative to the provided NSX-T URL for API specifics such as available commands.
DCLI_NSX_METADATA_FILE	Set this variable to specify metadata file location to use when connecting to NSX-T. If set, DCLI retrieves NSX-T specific metadata from the specified file.
DCLI_COLOR_THEME	Set this variable to change the color output of DCLI. The possible values are **`empty`**, **`blue`**, **`red`**, **`green`**, **`yellow`**, **`gray`**.
DCLI_CONFIGFILE	Set this variable to specify a custom configuration file.

DCLI Security

DCLI enhances security by providing SSL communication and secret input functionality.

SSL Communication

By default, DCLI verifies SSL certificates for HTTPS requests and throws an error message if it is unable to verify the certificate.

DCLI bundles certificates from `certifi.io` by using the `cacert.pem` file provided in the DCLI package. If you want to override this certification method, you have the following options.

- Use the *DCLI_CACERTS_BUNDLE* environment variable to specify a path to a custom certificates store file.
- Use the `+cacert-file` DCLI option to specify a path to a custom certificates store file.
- Use the `+skip-server-verification` DCLI option to skip certificates verification.

> **Note** Skipping certificates verification presents a security risk.

DCLI Secret Input

Some command options, more commonly when connecting to vCenter Server, are of secret type. Values for command options of secret type should be provided in a secure way.

You can provide values for secure options in two ways.

- Provide the value explicitly in the command line.

  ```
  dcli> com vmware vcenter securecommand -secureoption <securevalue>
  ```

  Important Providing the value explicitly presents a security risk.

- Provide the value when prompted after initially skipping the value.

  ```
  dcli> com vmware vcenter securecommand -secureoption
  secureoption:
  ```

Setting the DCLI Log File

You can set the DCLI log file, which can trace verbose log information, by using the *DCLI_LOGFILE* environment variable.

If you want to see additional log information in the console, you can use the `+loglevel debug` option as shown in the following example.

```
dcli +vmc +loglevel debug
```

Note DCLI does not preserve any sensitive information, such as passwords and secure input, in the log file.

DCLI History File

DCLI maintains a history file for each DCLI client that runs in interactive mode. The file stores information on a per-user basis and not on a per-client basis.

The location of the file is `$HOME/.dcli/.dcli_history`.