

DCLI User's Guide

Data Center Command-Line Interface 3.0.0



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

Copyright © 2017-2020 VMware, Inc. All rights reserved. [Copyright and trademark information.](#)

Contents

About This Book 4

1 Introduction to DCLI 5

Using DCLI 5

Supported Platforms 5

Install DCLI 5

2 Running DCLI Commands 7

Overview of Running DCLI Commands 7

DCLI Syntax 8

DCLI Options 8

DCLI Option Scopes 12

Run DCLI Commands in Interactive Mode 13

Run DCLI Commands in Non-Interactive Mode 14

Displaying Help Information for DCLI Commands 14

DCLI Authentication 15

VMware Cloud on AWS and NSX-T Authentication 15

vCenter Server Authentication 16

Using DCLI with a Credential Store File 17

Order of Precedence for DCLI Authentication to vCenter Server 18

Reusing Session ID and Authentication Tokens 18

Connect to an NSX-T Endpoint 18

Multi-Server Support in DCLI 19

Filtering DCLI Command Output 19

Using the DCLI Configuration Store 20

Internal DCLI Commands 21

Using Default Options 22

Input, Output, and Return Codes 23

Working with JSON Input in DCLI 23

Using DCLI with Variables 24

DCLI Security 25

SSL Communication 25

DCLI Secret Input 25

Enable FIPS Mode 26

Setting the DCLI Log File 27

DCLI History File 27

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 against environments with NSX-T 2.2 or later.

This chapter includes the following topics:

- [Using DCLI](#)
- [Supported Platforms](#)
- [Install DCLI](#)

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](#)
- [DCLI Authentication](#)
- [Connect to an NSX-T Endpoint](#)
- [Multi-Server Support in DCLI](#)
- [Filtering DCLI Command Output](#)
- [Using the DCLI Configuration Store](#)
- [Internal DCLI Commands](#)
- [Using Default Options](#)
- [Input, Output, and Return Codes](#)
- [Working with JSON Input in DCLI](#)
- [Using DCLI with Variables](#)
- [DCLI Security](#)
- [Setting the DCLI Log File](#)
- [DCLI History File](#)

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 option] ... <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 com vmware vmc orgs sddcs get --org orgID --sddc sddcID
```

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

```
dcli +server myvc +skip com vmware vcenter vm get --vm vm-23
```

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 [+server SERVER_IP]
    [+vmc-server]
    [+nsx-server [NSX_T_SERVER_ADDRESS]]
    [+org-id NSX_ORG_ID]
    [+sddc-id NSX_SDDC_ID]
    [+interactive]
    [+prompt PROMPT]
    [+skip-server-verification]
    [+cacert-file CACERT_FILE]
    [+username USERNAME]
```



```

[+password PASSWORD]
[+logout]
[+filter FILTER]
[+formatter {yaml, json, table, csv, html, xml, yamlc, jsonc, htmlc, xmlc}]
[+verbose]
[+log-level {debug, info, warning, error}]
[+log-file LOG_FILE]
[+generate-json-input]
[+generate-required-json-input]
[+json-input JSON_INPUT]
[+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
<code>server</code>	The vCenter Server system to which DCLI connects.	<code>localhost</code>	Yes
<code>vmc-server</code>	The VMware Cloud on AWS server to which DCLI connects.	<code>https://vmc.vmware.com</code>	Yes
<code>nsx-server</code>	The NSX-T server to which DCLI connects. If not specified, <code>+org-id</code> and <code>+sddc-id</code> must be specified.		Yes
<code>org-id</code>	Use this option along with <code>+nsx</code> , specified as a flag, and <code>+sddc-id</code> . Specifies the organization ID of the NSX-T server you want to connect to on the VMware Cloud on AWS service.		Yes
<code>sddc-id</code>	Use this option along with <code>+nsx</code> , specified as a flag, and <code>+org-id</code> . Specifies the organization SDDC of the NSX-T server you want to connect to on the VMware Cloud on AWS service.		Yes

Option	Description	Default	Available in Interactive Mode
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
username	<p>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.</p> <p>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.</p> <p>Note Only available when DCLI connects to a vCenter Server system.</p>		Yes
password	<p>You can use this option to provide your password explicitly instead of waiting for the DCLI password prompt.</p> <p>Important Providing the value explicitly presents a security risk.</p> <p>Note Only available when DCLI connects to a vCenter Server system.</p>		Yes
logout	<p>Use this option to request logout from vCenter Server and remove the session ID and authentication tokens from the credential store.</p> <p>Note It is a good practice to use this option at the end of your script to destroy the used session and prevent overflowing the server with session objects.</p>		No

Option	Description	Default	Available in Interactive Mode
filter	Provides JMESPath expressions to filter command output.		Yes
formatter	Output formatter, which has one of the following possible values. <ul style="list-style-type: none"> ■ yaml ■ json ■ table ■ csv ■ html ■ xml ■ yamlc ■ jsonc ■ htmlc ■ xmlc The yamlc, jsonc, htmlc, and xmlc formatters are colored versions of the yaml, json, html, and xml formatters.	Default is table for lists of structures and yaml for all other output including json.	Yes
verbose	Prints verbose output.	False	Yes
log-level	The log level, which has one of the following possible values. <ul style="list-style-type: none"> ■ debug ■ info ■ warning ■ error 	info	Yes
log-file	Specifies the log file to be used for detailed DCLI execution information.	\$HOME/.dcli/.dcli_log	Yes
generate-json-input	When provided and used along with a command, DCLI outputs empty JSON input for the command, which you can populate and provide back as an input by using the +json-input option.	False	Yes
generate-required-json-input	When provided and used along with a command, DCLI outputs empty JSON input for the required fields of the command, which you can populate and provide back as an input by using the +json-input option.	False	Yes
json-input	Specifies a JSON string value or a .json file that you can use as command input.		Yes

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

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
- log-level

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 Enable interactive mode.

```
dcli +vmc +i
```

DCLI connects to the VMware Cloud on AWS server in interactive mode.

- 2 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

- ◆ 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.

DCLI supports storing multiple refresh tokens in the credential store. Each refresh token stored in the credential store also keeps information about the user and organization ID associated with the token.

If you provide the `--org` parameter or connect through an NSX-T endpoint, DCLI prompts for the refresh token for that organization.

```
dcli> orgs sddcs list --org a57c766d-04e9-470f-a8ea-6846d2fe6b5a
Refresh Token (for Organization with ID a57c766d-04e9-470f-a8ea-6846d2fe6b5a):
```

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
```

Note This command deletes all VMware Cloud on AWS refresh tokens stored in the credential store.

- 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
```

Note This command deletes all VMware Cloud on AWS refresh tokens stored in the credential store.

- 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.

Reusing Session ID and Authentication Tokens

DCLI preserves the session ID of a vCenter Server connection and the authentication tokens of VMware Cloud on AWS or NSX-T connections. Preservation works differently based on whether you run commands in interactive or non-interactive mode.

Interactive Mode

In interactive mode, the first command that requires authentication creates a session ID or authentication token. This information is saved in the memory. Subsequent command calls that require authentication reuse the saved session ID or authentication token.

After you close the interactive session by using the `exit` command with a vCenter Server connection, a logout request is sent to the server and the session ID is removed from the memory. With VMware Cloud on AWS or NSX-T connections, the authentication token is removed from the memory.

Non-Interactive Mode

In non-interactive mode, the first command that requires authentication prompts you for credentials. The command also prompts whether to store the credentials in the credential store. If you decide to preserve the credentials in the store, the session ID or authentication token are also saved in the credential store and are later reused by subsequent commands. If you decide not to preserve the credentials, a session ID or authentication token is created for each command.

If you want to remove a session ID or authentication token from the credential store and log out of a server, you must use the `+logout` option and provide the vCenter Server, VMware Cloud on AWS, or NSX-T connection. Using this option with a vCenter Server connection sends a logout request to the server and removes the session ID from the credential store. Using `+logout` with VMware Cloud on AWS or NSX-T connections only removes the authentication token from the credential store.

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

Internal Command	Description
<code>env style list</code>	Lists all DCLI style options.
<code>env style color enable</code>	Enables DCLI colorization. Note DCLI colorization is enabled by default.
<code>env style color disable</code>	Disables DCLI colorization.
<code>env style color input enable</code>	Enables DCLI colorization input.
<code>env style color input disable</code>	Disables DCLI colorization input.
<code>env style color output enable</code>	Enables DCLI colorization output.
<code>env style color output disable</code>	Disables DCLI colorization output.
<code>env style color theme set</code>	Sets the DCLI color theme. The possible values are <code>monokai</code> , <code>bw</code> , <code>paradiso-dark</code> , <code>autumn</code> . The default color theme is <code>bw</code> .
<code>env style color theme get</code>	Retrieves the currently set DCLI color theme.

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 provide basic types like <code>string</code> , <code>int</code> , <code>double</code> , or <code>boolean</code> 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.

The currently supported output formatter types are `yaml`, `json`, `table`, `csv`, `html`, `xml`, `yamlc`, `jsonc`, `htmlc`, and `xmlc`. 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.

Working with JSON Input in DCLI

You can provide JSON input for single options or for a whole command.

Providing JSON Input for Single Options

Input for maps and lists of structures in DCLI must be provided by using the JSON format. You can provide the input either directly to the console or by using a `.json` file.

You can use the following syntax to provide JSON command input directly to the console.

```
dcli com vmware vcenter sample command --map '{"key1":"value","key2":"value"}'
```

Note You must not use blank spaces between the JSON elements.

You can use the following syntax to provide JSON command input by using a `.json` file.

```
dcli com vmware vcenter sample command --map /path/to/json_file.json
```

Providing JSON Input for a Whole Command

DCLI provides the option to generate empty command input in JSON format. After generating the empty command input, you can populate the empty fields with custom input values. You must remove any unnecessary fields from the generated content. After you populate the empty fields, you can provide the content back as command input.

You can use the following syntax to generate empty JSON command input.

```
dcli com vmware sample command +generate-json-input
```

You can use the following syntax to generate empty JSON command input for required fields only.

```
dcli com vmware sample command +generate-required-json-input
```

You can use the following syntax to generate empty JSON command input and save it as a file, which you can populate later.

```
dcli com vmware sample command +generate-json-input > json_input.json
```

You can use the following syntax to provide JSON command input by using a .json file that you have populated.

```
dcli com vmware sample command +json-input json_input.json
```

You can use the following syntax to provide JSON command input directly to the console.

```
dcli com vmware sample command +json-input '{"option":"value"}'
```

Note You must not use blank spaces between the JSON elements.

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
<i>DCLI_VMC_SERVER</i>	Set this variable to pass the VMware Cloud on AWS server IP address.
<i>DCLI_NSX_SERVER</i>	Set this variable to pass the NSX-T server IP address. Passing the <code>nsx-server</code> option on the command line overrides this variable.
<i>DCLI_SERVER</i>	Set this variable to pass the vCenter Server system IP address. Passing the <code>server</code> option on the command line overrides this variable.
<i>DCLI_ORG_ID</i>	Set this variable to specify the default value for the <code>+org-id</code> option.
<i>DCLI_SDDC_ID</i>	Set this variable to specify the default value for the <code>+sddc-id</code> option.
<i>DCLI_CACERTFILE</i>	Set this variable to pass the path of a CA certificate file. Passing the <code>cacert-file</code> option on the command line overrides this variable.
<i>DCLI_USERNAME</i>	Set this variable to pass the user name required for authentication. Passing the <code>username</code> option on the command line overrides this variable.
<i>DCLI_CREDFILE</i>	Set this variable to point to a DCLI credential store file. Default value is <code>~/.dcli/.dcli_credstore</code> . Passing the <code>credstore-file</code> option on the command line overrides this variable.
<i>DCLI_HISTFILE</i>	Set this variable to point to a DCLI interactive shell history file path. Default value is <code>~/.dcli/.dcli_history</code> .
<i>DCLI_LOGFILE</i>	Set this variable to specify the log file for DCLI.
<i>DCLI_CACERTS_BUNDLE</i>	Set this variable to specify the path to a trust store. By default, DCLI includes a trust store from Certifi.

Variable	Description
<i>DCLI_VMC_CSP_URL</i>	Set this variable to specify the URL to the CSP service for authentication token retrieval from a given refresh token. The default URL is <code>https://console.cloud.vmware.com</code> .
<i>DCLI_VMC_METADATA_URL</i>	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.
<i>DCLI_VMC_METADATA_FILE</i>	Set this variable to specify metadata file location. If set, DCLI retrieves metadata from the specified file.
<i>DCLI_NSX_METADATA_URL</i>	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.
<i>DCLI_NSX_METADATA_FILE</i>	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.
<i>DCLI_COLOR_THEME</i>	Set this variable to change the color output of DCLI. The possible values are <code>monokai</code> , <code>bw</code> , <code>paradiso-dark</code> , <code>autumn</code> . The default color theme is <code>bw</code> .
<i>DCLI_COLORS_ENABLED</i>	Set this variable to enable DCLI colorization.
<i>DCLI_COLORED_INPUT</i>	Set this variable to enable DCLI input colorization.
<i>DCLI_COLORED_OUTPUT</i>	Set this variable to enable DCLI output colorization.
<i>DCLI_CONFIGFILE</i>	Set this variable to specify a custom configuration file.

DCLI Security

DCLI enhances security by providing SSL communication and secret input functionality. You can also manually enable DCLI to run in FIPS mode.

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:
```

Enable FIPS Mode

DCLI is compliant with the Federal Information Processing Standards (FIPS), but the FIPS mode is not enabled by default. You can enable the FIPS mode by modifying the DCLI script.

For its cryptographic algorithms, DCLI relies on the OpenSSL library, which is used by default from the Python interpreter. OpenSSL has FIPS versions that provide the option to run in a mode, which prevents using algorithms that are not compliant with FIPS. However, by default Python does not include a FIPS version of OpenSSL. To turn on the FIPS mode in OpenSSL, you must have a Python interpreter build with a FIPS version of OpenSSL.

Procedure

- 1 Find the location of the DCLI script.
 - If you are using Linux or Mac OS, run the following command.

```
which dcli
```

- If you are using Windows, run the following command.

```
where dcli
```

- 2 Open the DCLI script file by using a text editor.
- 3 Locate the following line in the script.

```
from vmware.vapi.client.dcli.cli import main
```

- 4 Add the following lines after the line you located.

```
import ctypes

libcrypto = ctypes.CDLL("libcrypto.so.1.0.0")

fips_mode = libcrypto.FIPS_mode
fips_mode.argtypes = []
fips_mode.restype = ctypes.c_int

fips_mode_set = libcrypto.FIPS_mode_set
```

```
fips_mode_set.argtypes = [ctypes.c_int]
fips_mode_set.restype = ctypes.c_int

fips_mode_set(1)
```

- 5 (Optional) Verify whether the FIPS mode is enabled by printing to the console when running DCLI.

Replace the `fips_mode_set(1)` line with the following lines.

```
print("FIPS_mode(): {:d}".format(fips_mode()))
print("FIPS_mode_set(1): {:d}".format(fips_mode_set(1)))
print("FIPS_mode(): {:d}".format(fips_mode()))
```

- 6 (Optional) Verify that you receive a proper error message when calling an MD5 algorithm, which is not compliant with FIPS.

Add the following lines after the last lines you added.

```
import hashlib
print("SHA1: {:s}".format(hashlib.sha1(text).hexdigest()))
print("MD5: {:s}".format(hashlib.md5(text).hexdigest()))
```

- 7 Save the changes you made to the DCLI script.

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 `+log-level debug` option as shown in the following example.

```
dcli +vmc +log-level 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`.