Lenovo XClarity Administrator Python Toolkit Reference

Version 4.1

Note

Before using this information and the product it supports, read the general and legal notices in the XClarity Administrator online documentation.

Second Edition (June 2024)

Contents

Summary of changes in the Python (PyLXCA) toolkit

Lenovo XClarity Administrator supports enhancements to the PyLXCA toolkit.

This documentation includes new cmdlets and parameters that apply to the current XClarity Administrator release and later. For information about enhancements to the PyLXCA toolkit in other releases, see Summary of changes in the Python (PyLXCA) toolkit in the XClarity Administrator online documentation.

Version 4.1.0

There are no changes to the PyLXCA toolkit in this release.

Version 4.0.0

There are no changes to the PyLXCA toolkit in this release.

Chapter 1. Python (PyLXCA) toolkit

The PyLXCA toolkit provides a Python-based library of commands and APIs to automate provisioning and resource management from an OpenStack environment, such as Ansible.

The PyLXCA toolkit provides an interface to Lenovo XClarity Administrator REST APIs to automate functions such as:

- Logging in to XClarity Administrator
- Managing and unmanaging chassis, servers, storage devices, and top-of-rack switches (devices)
- Viewing inventory data for devices and components
- Deploying an operating-system image to one or more servers
- Configuring servers through the use of Configuration Patterns
- Applying firmware updates to devices

For information about installing and using the Ansible toolkit for each supported OpenStack environment, see the Ansible LXCA Client toolkit for Ansible website.

Chapter 2. Installing the Python (PyLXCA) toolkit

To use the PyLXCA commands and APIs, you must install the Lenovo XClarity Administrator Python Client Toolkit (PyLXCA) in Python.

Before you begin

Wheel is required to install the latest version of the PyLXCA toolkit. You can download Wheel from the Python Wheels webpage, and install Wheels using the following command: `pip install wheel`

Python (including the request and logging modules) is required to use to the PyLXCA toolkit. Ensure that the following requirements are met. For more information about Python, see the Python website.

- Python 3.6.x (Later versions have not been tested.)

  Note: For XClarity Administrator Python Client Toolkit v2.8 and earlier, Python v2.7.x or 3.6.x is required.
- Python requests v2.7.0 or later (Install using the following command: `pip install requests –upgrade`)
- Python logging v0.4.9.6 or later (Install using the following command: `pip install logging –upgrade`)

Procedure

To install the PyLXCA from the Internet using pip, run the following command:

```
pip install pylxca
```

To build and install from source, download the PyLXCA source code from Python LXCA Client (PyLXCA) Toolkit website, and run the following command from the source root directory:

```
python setup.py install
```

After you finish

For information about using the PyLXCA commands, see Using the Python (PyLXCA) toolkit.

You can display the current version of PyLXCA from the python prompt, for example:

```
$ python
Python 2.7.10 (default, Sep 16 2015, 14:46:04)
[GCC 4.8.4] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import pylxca
>>> print pylxca.__version__
1.0
```

Chapter 3. Using the Python (PyLXCA) toolkit

You can use the Lenovo XClarity Administrator Python (PyLXCA) toolkit from the Python shell, interactive shell, or as a Python script.

Python shell

You can run any of the PyLXCA commands from the Python shell.

To start the Python shell, run the `lxca_shell` command. When you are in the Python shell, the prompt changes to `PyLXCA >>`.

```
joe@joe_vm:~# lxca_shell
------------------------------------------------
Welcome to LXCA Command Shell
Type "help" at any time for a list of commands.
Use "lxca_shell --api" to enable Interactive Python LXCA Shell
------------------------------------------------

PyLXCA >> connect --url https://192.0.2.0 --user USER1 -noverify true
```

For information about each PyLXCA command, see PyLXCA commands reference.

Interactive shell

When you use the interactive Python XClarity Administrator shell, you can use built-in Python functions in addition to PyLXCA commands.

To start the interactive shell, run the `pyshell --api` command. When you are in the interactive shell, the prompt changes to `>>>`.

```
joe@joe_vm:~#lxca_shell --api
Interactive Python Shell for Lenovo XClarity Administrator
Type "dir()" or "help(lxca command object)" for more information.
>>>
>>> con1 = connect("https://192.0.2.0","USER1","password","true")
```

To get detailed help for the PyLXCA functions, use the python `help`*(command_name)* function. For example, `help(connect)` returns help for the `connect` function.

Python scripts

Just like the Python shell, when you create the Python scripts, you can use built-in Python functions in addition to PyLXCA APIs

Note: At the beginning of your script, you must import PyLXCA and connect to the XClarity Administrator instance, for example:

```
#!/usr/bin/env python
from pylxca. import *
con1 = connect("https://192.0.2.0","USER1","password","True")
```

Python (PyLXCA) toolkit examples

A sample script is available to help you to begin using the PyLXCA command-line interface (CLI) to manage devices quickly.

The sample script is located in the following directory:
<python_install>`\lib\site-packages\pylxca\test`

The following script is included in the directory:

- `pylxca_unittest`

  Retrieves inventory for managed devices of each supported type.

Creating custom views

A *view* (or *filter*) determines what data is returned by a PyLXCA inventory command. The default views for each command are defined in the *<python_install_dir>*/`site-packages/pylxca-1.0-py2.7.egg/pylxca/pylxca_cmd/lxca_filters.xml` file. You can add your own custom views.

In the `lxca_filters.xml` file, all default and custom views are included under the **<filters>** tag. You can also add custom views under this tag for any command that returns inventory data. In the following example, there is a filter for the `chassis` command named "default." This is the default view that is used if the **--view** parameter is not specified. The following example also has a filter for the `cmm` command named "location_view" To use this view, specify **--view location_view** when you run the command.

```
<filters>
   <chassis name="default">...</chassis>
   <chassis name="health_view">...</chassis>
   <cmm name="location_view">...</cmm>
</filters>
```

Under each command tag (such as **<chassis>**) are the fields that you want returned for that view. You can choose to return any response-body field that the inventory REST API supports for the device type. In the following example, "Name" and "ChassisLocation" are the labels that are returned by the `chassis` command, and "name" and "location" are the names of the response-body fields in the request. If the field is an object, you must specify **type="object"** in the tag and then list the fields in the object that you want to return under the tag.

```
<filters>
   <chassis name="location_view">
      <Name>name</Name>
      <ChassisLocation name="location" type="object">
         <Rack>rack</Rack>
         <Room>room</Room>
      </Chassis_Location>
      <Contact>contact</Contact>
      <Type>type</Type>
      <IPAddresses>ipAddresses</IPAddresses>
      <HostName>hostname</HostHame>
      <UUID>uuid</UUID>
   </chassis>
</filters>
```

- If you do not specify **type="object"** for objects and arrays of objects, the returned data is null.
- The response-body field names are case-sensitive.

The following example is returned if you run the chassis command using the "location_view" filter that is defined previously:

```
Name: SN#Y031BG19H006
ChassisLocation:
    Rack: []
    Room: []
Contact: None
Type: Chassis
IPAddresses: [u'192.0.2.0', u'fe80::5ef3:fcff:fe25:e1dd', u'fd55:faaf:e1ab:2021:5ef3:fcff:fe25:e1dd']
HostName: SN#Y031BG19H006
UUID: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Syntax diagram conventions

To understand the command descriptions, Review the conventions used in syntax diagrams.

The syntax diagram consists of options and arguments. *Options* consist of a dash and single letter (for example, **-v**) or two dashes and multiple letters (for example, **--view**). Options can be followed by one or more *arguments* (for example, as illustrated in [**-v** <*filter*>]).

Consider these conventions when reading syntax diagrams:

- Options that are preceded by one dash (-) are case-sensitive.
- Options that are preceded by two dashes (--) must be specified in their entirety. These options *are not* case-sensitive.
- The names of arguments that require substitution by actual values are italicized and enclosed in greater-than and less-than symbols (< >).
- Options that are enclosed in brackets ([]) are optional. Do not include these brackets in the command.
- Options that are enclosed in braces ({}) are required. Do not include these braces in the command.
- Options that are not enclosed in either brackets or braces are required.
- The pipe (|) character signifies that you choose one option or the other. For example, [a | b] indicates that you can choose either a or b, but not both. Similarly, {a | b} indicates that you must choose either a or b.
- An ellipsis (...) signifies that you can repeat the operand and option argument on the command line.

Chapter 4. PyLXCA commands reference

These are the commands that are provided by the Lenovo XClarity Administrator PyLXCA toolkit.

Virtual appliance and PyLXCA management commands

The following PyLXCA commands are available for managing the Lenovo XClarity Administrator virtual appliance and PowerShell sessions.

Note: Use the `exit` command to exit the PyLXCA interactive shell.

connect

This command creates a connection (logs in) to the Lenovo XClarity Administrator server that can be used by other commands.

Use the disconnect command to log out of the XClarity Administrator server.

Syntax

```
connect  -h

connect  -l <HTTPS_addresss> -u <user_ID> [--noverify]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-l | --url} *<HTTPS_addresss>*
Specifies the HTTPS address (IP address or hostname) for the XClarity Administrator server (for example, `https://192.0.2.0`).

{-u | --user} *<user_ID>*
Specifies the user ID for logging in to the XClarity Administrator server.

You are prompted to input the password. Password is entered in a non-verbose mode.

--noverify
Does not verify the XClarity Administrator server certificate. If this option is not specified, the server certificate is verified when the connection is made.

Examples

The following example connects to the XClarity Administrator server using the HTTPS address `192.0.2.0` and user ID `USER1`.

```
connect --url https://192.0.2.0 --user USER1 --noverify
```

Related links

- disconnect

disconnect

This command ends the connection to (logs out of) the Lenovo XClarity Administrator server.

Use the connect command to log in to the XClarity Administrator server.

Syntax

```
disconnect  -h

disconnect
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

Examples

The following example disconnects from the XClarity Administrator server.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
disconnect
```

Related links

- connect

help

This command displays a list of all available commands and a short description of each.

Syntax

```
help
```

Options

None

Examples

The following example retrieves help for all commands.

```
help
```

ostream

This command sets the output-stream of the PyLXCA interactive shell to standard output (stdout), a file, or both.

Syntax

```
ostream  -h

ostream  -l <output_stream>
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-l | --lvl} *<output_stream>*
Specifies the output stream. You can specify one of the following values:

- **0**. Quiet.
- **1**. Console.
- **2**. File.

- **3**. Console and File.

  The output-stream file is specified in the `<install_dir>\lxca_console.out` file

  Tip: You can retrieve the list of values by entering the **-l ?** option.

Examples

The following example sets the output stream to a file:

```
ostream -1 2
```

Related links

- connect

Management-server commands

The following PyLXCA commands are available for updating the management server and retrieve license compliance status.

license

This command returns the license compliance status.

Syntax

```
license  -h
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

Examples

The following example returns the license compliance status.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
license
```

managementserver

This command retrieves information about and manages management-server updates in the repository.

Syntax

```
configprofiles  -h

managementserver { query | query_fixids | acquire | apply | refresh  | delete | import }
                 <action_specific_parmeters>
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{query} *<action_specific_parmeters>*
Retrieves information about all management-server updates in the repository. For more information, see managementserver query.

{query_fixids} *<action_specific_parmeters>*

Retrieves information about, readme, or change history for a specific management-server update in the repository, see managementserver query_fixids

{acquire} *<action_specific_parmeters>*
Downloads the specified management-server updates from Lenovo XClarity Support website to the repository. For more information, see managementserver acquire.

{apply} *<action_specific_parmeters>*
Installs specific management-server updates. For more information, see managementserver apply .

{refresh} *<action_specific_parmeters>*
Retrieves information about the latest available management-server updates from the Lenovo XClarity Support website. For more information, see managementserver refresh.

{delete} *<action_specific_parmeters>*
Deletes specific management-server updates. For more information, see managementserver delete.

{import} *<action_specific_parmeters>*
Imports management-server updates from the local system to the repository. For more information, see managementserver import.

managementserver acquire

This command downloads the specified management-server updates from Lenovo XClarity Support website to the repository.

Syntax

```
managementserver acquire  -h

managementserver acquire  -f <update_list> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-f | --fixids} *<update_list>*
Specifies a list of management-server-update IDs, separated by a comma.

You can find a list of IDs by using managementserver query –k updates.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example download a specific management-server update.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
managementserver acquire --fixids ibm_fw_imm2_1aoo78j-6.20_anyos_noarch
```

Related links

- connect
- managementserver query
- managementserver refresh

managementserver apply

This command installs specific management-server updates.

Syntax

```
managementserver apply  -h

managementserver apply  -f <update_list> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-f | --fixids} *<update_list>*
Specifies a list of management-server IDs, separated by a comma.

You can find a list of IDs by using managementserver query –k updates.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example apply a specific management-server update.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
managementserver apply --fixids ibm_fw_imm2_1aoo78j-6.20_anyos_noarch -v result
```

Related links

- connect
- managementserver acquire
- managementserver query

managementserver delete

This command deletes specific management-server updates.

Syntax

```
managementserver delete -h

managementserver delete -f <update_list> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-f | --fixids} *<update_list>*
Specifies a list of firmware-update IDs, separated by a comma.

You can find a list of IDs by using managementserver query –k updates.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example deletes management-server updates from the repository.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
managementserver delete -f lnvgy_sw_lxca-fw-repository-pack_1-1.0.1_anyos_noarch
```

Related links

- connect
- managementserver query

managementserver import

This command imports management-server updates from the local system to the repository.

Syntax

```
managementserver import -h

managementserver import -f <file_list> [-j <job_ID> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-f | --files} *<file_list>*
Specifies a list of firmware-update file, separated by a comma. Include the file name and directory.

{-j | --job} *<job_ID>*
Specifies the job ID for the import operation. If specified, this command returns status information about the specified job.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example creates job for importing a management-server update and then imports the update.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
managementserver import -f /home/updates/updates/lnvgy_sw_lxca_thinksystemrepo1-1.3.2_anyos_noarch.txt
managementserver import -j -f /home/updates/updates/lnvgy_sw_lxca_thinksystemrepo1-1.3.2_anyos_noarch.txt
```

Related links

- connect
- managementserver query
- managementserver query_fixids

managementserver query

This command retrieves information about management-server updates in the repository.

Syntax

```
managementserver query -h

managementserver query -k <update_type> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-k | --keys} *<update_type>*
Returns specific information about the management-server update. You can specify one of the following values.
- **all**. (default) Returns all information.
- **currentVersion**. Returns the current version of XClarity Administrator.
- **history**. Returns the history of management-server updates.
- **importDir**. Returns the directory for the management-server updates repository.
- **size**. Returns the repository size (in bytes).
- **updates**. Returns information about all updates packages.
- **updatedDate**. Returns the date when the last update was performed.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns the repository size (in bytes).

```
connect --url https://192.0.2.0 --user ADMIN --noverify
managementserver query -k size
```

Related links

- connect
- managementserver query_fixids

managementserver query_fixids

This command retrieves information about specific management-server updates in the repository.

Syntax

```
managementserver query_fixids -h

managementserver query_fixids -f <update_list> [-k <update_type>] [-t <filte_type>]
                              [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-f | --fixids} *<update_list>*
Specifies a list of management-server update IDs, separated by a comma.

You can find a list of update IDs by using the managementserver query cmdlet.

{-k | --keys} *<update_type>*
Returns specific information about the management-server update. You can specify one of the following values.
- **all**. (default) Returns all information.
- **actions**. Returns the actions that are available for the specified update.
- **keys**. Returns the specified key values.
- **filetypes**. Returns the file types that are available for the specified update.
- **update**. Returns information about the update package.

{-t | --type} *<file_type>*
Returns the readme or change history file. This can be one of the following values.
- **changeHistory**. Returns the change-history file for the specified management-server update.

- **readme**. Returns the readme file for the specified management-server update.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns information about a specific management-server update.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
managementserver query_fixids -f ibm_fw_imm2_1aoo78j-6.20_anyos_noarch -k all
```

Related links

- connect
- managementserver query

managementserver refresh

This command retrieves information about the latest available management-server updates from the Lenovo XClarity Support website.

Syntax

```
managementserver refresh -h

managementserver refresh [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example refresh management server repository.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
managementserver refresh
```

Related links

- connect
- managementserver query
- managementserver query_fixids

Discovery and management commands

The following PyLXCA commands are available for performing discovery, manage, and unmanage functions.

discover

This command discovers manageable devices using SLP discovery.

This command starts a job to perform the discover action. If successful, the job ID is returned. You can use the job ID in the tasks -j or discover -j command to determine the status of the job.

Syntax

```
discover  -h

discover  -i <IP_address>

discover  -j <job_ID>
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-i | --ip} *<IP_address>*
Specifies the IP addresses of one or more devices to be discovered. Separate each IP address using a comma.

{-j | --job} *<job_ID>*
Specifies the job ID. If specified, this command returns status information about the specified job.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example discovers devices with the specified IP addresses.

```
connect --url https://192.0.2.0 --user ADMIN
discover -i 192.0.2.11,192.0.2.22
```

The following example retrieves status information about the specified discovery job.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
discover -j 22
```

Related links

- connect
- manage
- unmanage

manage

This command manages a discovered device.

Syntax

```
manage  -h

manage   { device | job_status } <action_specific_parmeters>
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{device} *<action_specific_parmeters>*
Manages a discovered device.

{job_status} *<action_specific_parmeters>*

Retrieves device-management job status.

manage device

This command manages a discovered device.

The devices must be discovered using the discover command before running this command

This command starts a job to perform the management action. If successful, the job ID is returned. You can use the job ID in the tasks –j command or the manage job_status command to determine the status of the job.

Syntax

```
manage device -h

manage device -i <IP_address> [-f <boolean>] [-r <recovery password>]
              {-u <device_user_ID> -p <device_password> | -s <stored credential ID>]
              [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-f | --force} *<boolean>*
If specified, the devices are force managed by this XClarity Administrator server.

Notes: Use this force-management option only if you previously attempted to manage the device and management was not successful due to one of the following error conditions.

- If the managing XClarity Administrator failed and cannot be recovered.

  Note: If the replacement XClarity Administrator instance uses the same IP address as the failed XClarity Administrator, you can manage the device again using the RECOVERY_ID account and password (if applicable) and the **Force management** option.
- If the managing XClarity Administrator was taken down before the devices were unmanaged.
- If the devices were not unmanaged successfully.

Attention: Devices can be managed by only one XClarity Administrator instance at a time. Management by multiple XClarity Administrator instances is not supported. If a device is managed by one XClarity Administrator, and you want to manage it with another XClarity Administrator, you must first unmanage the device from the original XClarity Administrator, and then manage it with the new XClarity Administrator.

{-i | --ip} *<IP_address>*
Specifies the IP address of the device to be managed.

{-p | --pw} *<device_password>*
Specifies the password for the device user account.

{-r | --rpw} *<recovery password>*
Specifies the recovery-ID password for the device.

When you manage a chassis, the CMM is configured to authenticate users with XClarity Administrator. Local CMM user accounts are no longer valid. If there are issues with XClarity Administrator, you can use the `RECOVERY_ID` to access the CMM directly. Ensure that you write down the password that you specify and store it in a secure location.

When you manage a Converged, NeXtScale, or System x server, the baseboard management controller is configured to authenticate users with the XClarity Administrator server. Local management-controller user accounts are no longer valid. If there are issues with the XClarity Administrator server, you can use the `RECOVERY_ID` to access the management controller directly. Ensure that you write down the password that you specify and store it in a secure location.

{-s | --storedcredential_id} *<stored_credential_ID>*
Specifies the ID of the stored-credential account to use to manage the device.

{-u | --user} *<device_user_ID>*
Specifies a user account with **lxc-supervisor** authority for the device. If the device to be managed is a chassis, specify a user account for the CMM in that chassis.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example manages a device using a user account.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
manage device -i 192.0.2.22 -u USERID -p xxxxxxxx -r xxxxxxxx
```

The following example manages a device using a stored-credential account.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
manage device -i 192.0.2.23 -s AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA -r xxxxxxxx
```

Related links

- connect
- discover
- manage job_status
- storedcredentials
- unmanage

manage job_status

This command retrieves device-management job status.

Syntax

```
manage job_status -h

manage job_status -j <job_ID>  [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-j | --job} *<job_ID>*
Specifies the job ID. If specified, this command returns status information about the specified job.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns information about a specific management job.

```
connect --url https://192.0.2.0 --user ADMIN -noverify
manage job_status -j 4
```

Related links

- connect
- manage device

unmanage

This command unmanages a managed device.

Syntax

```
unmanage  -h

unmanage { device | job_status } <action_specific_parmeters>
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{device} *<action_specific_parmeters>*
Unmanages a device.

{job_status} *<action_specific_parmeters>*
Retrieves device-unmanagement job status.

unmanage device

This command unmanages a managed device.

This command starts a job to perform the unmanagement action. If successful, the job ID is returned. You can use the job ID in the tasks –j command or the unmanage job_status command to determine the status of the job.

Syntax

```
unmanage device -h

unmanage device -e <device_properties> [--force <Boolean>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-e | --ep} *<device_properties>*
Specifies a set of properties for one or more devices to be unmanaged.

Each set is separated by a comma, and each property in the set is separated by a semicolon. The properties include:

- One or more IP addresses, separated by a hash sign (#).
- The UUID of the device
- The type of device to be unmanaged. You can specify one of the following values:
  - Chassis
  - Rackswitch
  - Rack-Tower
  - Storage
  - ThinkServer

For example:

```
2.22.222.222#3.33.333.333;F6F5A2630C244FDD9DE5376812C55480;Chassis,
4.44.444.444;63E29269BB634AB9A610D6F8FCE2B28F;Rack-Tower Server
```

{--force} <*Boolean*>

Forces the unmanagement of a device. You can specify one of the following values:

- **true**
- **false**

{-v | --view} <*filter*>
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example unmanages a device with a specific IP address.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
unmanage device -e 192.0.2.44;AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA;Rack-Tower Server
```

Related links

- connect
- discover
- manage
- unmanage job_status

unmanage job_status

This command unmanages a managed device.

Syntax

```
unmanage  -h

unmanage job_status -j <job_ID> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-j | --job} <*job_ID*>
Specifies the job ID. If specified, this command returns status information about the specified job

{-v | --view} <*filter*>
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns information about a specific unmanagement job.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
unmanage job_status -j 15
```

Related links

- connect
- manage device

Inventory commands

The following PyLXCA commands are available for performing inventory functions.

chassis

This command retrieves inventory data about all managed or unmanaged chassis, or a specific chassis.

Syntax

```
chassis  -h

chassis  [-s <management_status>] [-u <chassis_UUID>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-s | --status} *<management_status>*
Identifies the management status of the chassis to be returned. You can specify one of the following values:

- **manage**. (default) Returns inventory data for all managed chassis.
- **unmanaged**. Returns inventory data for all discovered chassis that Lenovo XClarity Administrator does not manage.

{-u | --uuid} *<node_UUID>*
Specifies the UUID of a chassis. If a UUID is not specified, this command returns inventory data for all managed chassis unless you also specify the **-s unmanaged** option.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns inventory data for a specific chassis.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
chassis -u AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Related links

- connect
- cmms
- fans
- fanmuxes
- nodes
- powersupplies
- switches

cmms

This command retrieves inventory data about all CMMs in managed chassis, all CMMs in a specific chassis, or a specific CMM.

Syntax

```
cmms  -h

cmms  [-c <chassis_UUID> | -u <cmm_UUID>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-c | --chassis} *<chassis_UUID>*
Specifies the UUID of a managed chassis. This command returns inventory data for all CMMs in that chassis.

{-u | --uuid} *<cmm_UUID>*
Specifies the UUID of a CMM. If a UUID is not specified, this command returns inventory data for all CMMs in all managed chassis.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns inventory data for all CMMs in managed chassis.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
cmms
```

Related links

- connect
- chassis

fanmuxes

This command retrieves inventory data about all fan logic modules (called fan muxes) in managed chassis, all fan muxes in a specific chassis, or a specific fan mux.

Syntax

```
fanmuxes  -h

fanmuxes  [-u <fan_mux_UUID>] [-c <chassis_UUID>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-c | --chassis} *<chassis_UUID>*
Specifies the UUID of a managed chassis. This command returns inventory data for all fan muxes in the chassis.

{-u | --uuid} *<fan_mux_UUID>*

Specifies the UUID of a fan mux. If a UUID is not specified, this command returns inventory data for all fan muxes.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns inventory data for all fan muxes in specific chassis.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
fanmuxes -c AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Related links

- connect
- chassis

fans

This command retrieves inventory data about all fans in managed devices, all fans in a specific chassis, or a specific fan.

Syntax

```
fans  -h

fans  [-u <fan_UUID>] [-c <chassis_UUID>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-c | --chassis} *<chassis_UUID>*
Specifies the UUID of a managed chassis. This command returns inventory data for all fans in the chassis.

{-u | --uuid} *<fan_UUID>*
Specifies the UUID of a fan. If a UUID is not specified, this command returns inventory data for all fans in managed devices.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns inventory data for all fans in the specified chassis.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
fans -c AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Related links

- connect
- chassis

nodes

This command retrieves inventory data about all managed or unmanaged servers and Flex System storage devices, all servers and storage devices in a specific Flex System chassis, a specific server, or a specific Flex System storage device. This command also system properties for a specific server or Flex System chassis.

Syntax

```
nodes  -h

nodes  -s <managemnt_status> [-v <filter>]

nodes  -u <node_UUID> [-v <filter>]

nodes  -c <chassis_UUID> [-v <filter>]

nodes  -m <properties_JSON> {-u <node_UUID> | -c <chassis_UUID>} [-v <filter>]
```

Parameters

{-h | --help}
Displays the syntax and brief usage information for this command.

{-c | --chassis} <*chassis_UUID*>
Specifies the UUID of a managed chassis. This command returns inventory data for all compute nodes in the chassis.

{-m | --modify} <*properties_JSON*>
Modifies system properties, specified in JSON format, for a specific server or Flex System chassis.

Table 1. Modify node properties

Attributes	Required / Optional	Type	Description
cmmDisplayName	Optional	String	Chassis name
contact	Optional	String	The chassis contact information
hostname	Optional	String	Hostname
ipInterfaces	Optional	Array	Information about the CMM IP addresses
name	Required	String	IP Interface name
IPv4enabled	Optional	Boolean	Identifies whether IPv4 is enabled. This can be one of the following values. • **true**. IPv4 is enabled • **false**. IPv4 is disabled
IPv6enabled	Optional	Boolean	Identifies whether IPv6 is enabled. This can be one of the following values. • **true**. IPv6 is enabled • **false**. IPv6 is disabled
IPv4DHCPmode	Optional	String	IPv4 address assignment method. This can be one of the following values. • **STATIC_ONLY** • **DHCP_ONLY** • **DHCP_THEN_STATIC** • **UNKNOWN**

Table 1. Modify node properties (continued)

Attributes	Required / Optional	Type	Description
IPv6DHCPenabled	Optional	Boolean	Identifies whether IPv6 DHCP is enabled. This can be one of the following values. • **true**. IPv6 DHCP is enabled • **false**. IPv6 DHCP is disabled
IPv6statelessEnabled	Optional	Boolean	Identifies whether IPv6 stateless is enabled. This can be one of the following values. • **true**. IPv6 stateless is enabled • **false**. IPv6 stateless is disabled
IPv6staticEnabled	Optional	Boolean	Identifies whether IPv6 static is enabled. This can be one of the following values. • **true**. IPv6 static is enabled • **false**. IPv6 static is disabled
IPv4assignments	Optional	Array	Information about IPv4 assignments
id	Required	Integer	IPv4 assignment ID
subnet	Optional	String	IPv4 subnet mask
gateway	Optional	String	IPv4 gateway
address	Optional	String	IPv4 address
IPv6assignments	Optional	Array	Information about IPv6 assignments
id	Required	Integer	IPv6 assignment ID
prefix	Optional	Integer	IPv6 prefix
gateway	Optional	String	IPv6 gateway
address	Optional	String	IPv6 address
location	Optional	String	(Flex System compute nodes only) Location in the chassis **Important:** Changes made to the location of the server or storage device using this API method are not reflected in the rack view.
location	Optional	Object	(Rack and tower servers only) Information about the location in the rack **Important:** Changes made to the location of the server using this API method are not reflected in the rack view.
location	Optional	String	Location of the server
rack	Optional	String	Rack
room	Optional	String	Room
lowestRackUnit	Optional	Integer	LRU
name	Optional	String	Server name
userDescription	Optional	String	The server description

The following example modifies the hostname, location, and contact information for the target server

```
{
   "contact": "new contact",
   "hostname":"",
   "location": {
      "location":"new location"
   }
}
```

{-s | --status} *<managemnt_status>*
Identifies the management status of servers. You can specify one of the following values:

- **manage**. Returns information for all managed servers.
- **unmanaged**. Returns information for all discovered servers that are not managed by Lenovo XClarity Administrator.

{-u | --uuid} *<node_UUID>*
Specifies the UUID of a server. If a UUID is not specified, this command returns inventory data for all managed servers unless you also specify the **-s unmanaged** option.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns inventory data for all managed compute nodes in a specific chassis.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
nodes -c AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Related links

- connect
- chassis

powersupplies

This command retrieves inventory data about all power supplies in managed devices, all power supplies in a specific chassis, or a specific power supply.

Syntax

```
powersupplies  -h

powersupplies  [-u <power_supply_UUID>] [-c <chassis_UUID>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-c | --chassis} *<chassis_UUID>*
Specifies the UUID of a managed chassis. This command returns inventory data for all power supplies in the chassis.

{-u | --uuid} *<power_supply_UUID>*
Specifies the UUID of a power supply. If a UUID is not specified, this command returns inventory data for all power supplies in managed devices.

{-v | --view} *<filter>*

Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns inventory data for all fans in managed devices.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
powersupplies
```

Related links

- connect
- chassis

scalablesystem

This command retrieves inventory data about all managed or unmanaged scalable complexes, all Flex System scalable complexes, all System x and ThinkServer scalable complexes, or a specific scalable complex.

Syntax

```
scalablesystem  -h

scalablesystem  [-i <system_UUID>] [-t <node_type>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-i | --id} *<system_UUID>*
Specifies the UUID of a scalable complex. If a UUID is not specified, this command returns inventory data for all managed scalable complexes unless you also specify the **-s unmanaged** parameter or **-t** parameter.

{-t | --type} *<node_type>*
Specifies the type of scalable complex. You can specify one of the following values:

- **flex**. Returns information for one or more Flex System scalable complexes.
- **rackserver**. Returns information for one or more System x or ThinkServer scalable complexes.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns inventory data for all managed scalable complexes in a rack.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
scalablesystem -t rackserver
```

Related links

- connect
- nodes

switches

This command retrieves inventory data about all managed Flex System and RackSwitch switches, all Flex System switches in a specific chassis, or a specific Flex System or RackSwitch switch.

Syntax

```
switches  -h

switches  [-u <switch_UUID>] [-c <chassis_UUID>] [-v <filter>]

switches  [-u <switch_UUID>] --ports <port_name> [--action <action>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{--action} *<action>*
Specifies the action to take on the specified port. This can be one or none of the following values:

- **enable**. Enable the specified port.
- **disable**. Disable the specified port

If no action is specified, this command displays port information for the specified switch.

{-c | --chassis} *<chassis_UUID>*
Specifies the UUID of a managed chassis. This command returns inventory data for all Flex System switches in the chassis.

{--ports} *<port_name>*
Specifies the ports to be modified. To obtain the port name, use this command without the **—ports** option to display information about the specified switch..

{-u | --uuid} *<switch_UUID>*
Specifies the UUID of a switch. If a UUID is not specified, this command returns inventory data for all managed switches unless you also specify the **-s unmanaged** option.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns inventory data for all Flex System switches in the specified chassis.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
switches -c AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

The following example enables a port on the specified switch..

```
connect --url https://192.0.2.0 --user ADMIN --noverify
switches -u AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA --ports Ethernet1/1 --action enable
```

Related links

- connect
- chassis

Server configuration commands

The following PyLXCA commands are available for performing server configuration (Configuration Patterns) functions.

configpatterns

This command retrieves information about and manages server patterns.

Syntax

```
configpatterns -h

configpatterns {list | apply | import | status} <action_specific_parmeters>
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{list} <*action_specific_parmeters*>
Lists server configuration patterns and details. For more information, see configpatterns list.

{apply} <*action_specific_parmeters*>
Applies a configuration pattern to a specific device. For more information, see configpatterns apply.

{import} <*action_specific_parmeters*>
Imports server and category configuration pattern to XClarity Administrator. For more information, see configpatterns import.

{status} <*action_specific_parmeters*>
Checks the configuration status of endpoint. For more information, see configpatterns status.

configpatterns apply

This command applies (deploys) a server pattern to one or more managed servers. You can also deploy a server pattern to one or more empty bays in a chassis that is managed by Lenovo XClarity Administrator or in a placeholder chassis. Deploying a server pattern before the server is installed reserves management IP addresses, reserves virtual Ethernet or Fibre Channel addresses, and pushes the network setting to the relative switch internal ports.

Syntax

```
configpatterns apply  -h

configpatterns apply  {-i <pattern_ID> | -n <pattern_name>} -r <activation_time>
                        -e <server_UUIDs> -t <server_type> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-i | --id} <*pattern_ID*>
Specifies the ID of the server pattern. You must specify either the **–id** or **–name** parameter.

{-e | --endpoint} <*server_UUID*>
Specifies the UUIDs of one or more target servers. If a target is an empty bay, specify the location ID; otherwise, specify the server UUID.

{-n | --name} <*pattern_name*>
Specifies the name of the server pattern. You must specify either the **–id** or **–name** parameter..

{-r | --restart} *<activation_time>*
Identifies when to activate the configurations. This can be one of the following values:

- **defer**. Activate management-controller settings but do not restart the server. UEFI and server settings are activated after the next restart of the server.
- **immediate**. Activate all settings and restart the server immediately.
- **pending**. Generate a profile for the server with the settings for review, but do not activate settings on the server. To activate the settings, you must manually activate the server profile and restart the server.

{-t | --type} *<server_type>*
Type of the server. This can be one of the following values.

- **flex**.
- **node**. Flex System compute node
- **rack**. Stand-alone rack server
- **tower**. Stand-alone tower server

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example deploys a server pattern to two rack servers and immediately activates the pattern on the servers.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
configpatterns apply -i 48 -r immediate -t rack
                -e AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA,BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
```

The following example deploys a server pattern to a Flex System server but does not not activate settings on the server until the server is manually restarted.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
configpatterns apply –n pattern_C -r pending -t node -e CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

Related links

- connect
- configpatterns import
- configpatterns list
- configprofiles
- configtargets

configpatterns import

This command creates a new server and category pattern.

Syntax

```
configpatterns import  -h

configpatterns import [-p <pattern_JSON>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-p | --pattern_update_dict} *<pattern_JSON>*

Specifies the server and category pattern to be imported. This must be a correctly formatted JSON of the configuration pattern that you want to import. Use JSON from a configuration pattern of the same type of pattern that you want to import using configpatterns list.

The format of the request changes depending on the type of configuration pattern (for example, server or system information) that is being imported. For information about the format for each category pattern, see the following topics in the REST API online documentation:

- System-information pattern fields
- Management-information pattern fields
- Device and I/O ports pattern fields
- Port pattern fields
- Fibre Channel boot-target pattern fields
- Extended management-controller pattern fields
- Extended-UEFI pattern fields
- Extended-port pattern fields

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns information about a specific server pattern.

```
connect --url https://192.0.2.0 --user ADMIN -noverify
configpatterns import -p '{"template_type":"SystemInfo","template":{"contact":"contact",
                          "description":"Pattern A","location":"location",
                          "name":"Learned-System_Info-99","systemName":{"autogen":"Disable",
                          "hyphenChecked":false},"type":"SystemInfo",
                          "uri":"/config/template/61","userDefined":true}}'
```

Related links

- connect
- configpatterns apply
- configpatterns list

configpatterns list

This command retrieves information about one or more server patterns.

Syntax

```
configpatterns list  -h

configpatterns list  [-i <pattern_ID>] [--includeSettings <settings_list>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-i | --id} *<pattern_ID>*
Specifies the ID of the server pattern. If an ID is not specified, all server patterns are returned.

{--includeSettings} *<settings_list>*
Specifies a list of IDs for category-pattern settings to return. If specified, The settings for category patterns that are associated with the specified server pattern are returned.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns information about all server patterns.

```
connect --url https://192.0.2.0 --user ADMIN -noverify
configpatterns list
```

The following example returns information about a specific server pattern.

```
connect --url https://192.0.2.0 --user ADMIN -noverify
configpatterns list -i 48
```

Related links

- connect
- configprofiles
- configtargets

configpatterns status

This command checks the configuration status of specific device.

Syntax

```
configpatterns status -h

configpatterns status -e <server_UUIDs> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-e | --endpoint} *<server_UUID>*
Specifies the UUIDs of one or more target servers. If a target is an empty bay, specify the location ID; otherwise, specify the server UUID.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example deploys a server pattern to two rack servers and immediately activates the pattern on the servers.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
configpatterns status -e AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA,BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
```

Related links

- connect
- configpatterns list

configprofiles

This command retrieves information about and manages server profiles.

Syntax

```
configprofiles  -h

configprofiles  -h {list | rename | activate | unassign | delete} <action_specific_parmeters>
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{list} *<action_specific_parmeters>*
Retrieves information about one or more server profiles. For more information, see configprofiles list.

{rename} *<action_specific_parmeters>*
Modifies the name of an existing profile. For more information, see configprofiles rename.

{activate} *<action_specific_parmeters>*
Activates a server profile on a replaced, reassigned, or newly installed and managed server. For more information, see configprofiles activate.

{unassign} *<action_specific_parmeters>*
Removes a server profile from a server or chassis bay by deactivating the profile. For more information, see configprofiles unassign.

{delete} *<action_specific_parmeters>*
Deletes a deactivated server profile. For more information, see configprofiles delete.

configprofiles activate

This command activates a server profile on a replaced, reassigned, or newly installed and managed server.

Syntax

```
configprofiles activate  -h

configprofiles activate  -i <profile_ID> -e <server_UUID> -r <activation_time>
                          [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-i | --id} *<profile_ID>*
Specifies the ID of the server profile. If an ID is not specified, all server profiles are returned.

{-e | --endpoint} *<server_UUID>*
Specifies the UUIDs of one or more target servers. If a target is an empty bay, specify the location ID; otherwise, specify the server UUID.

{-r | --restart} *<activation_time>*
Identifies when to activate the configurations. This can be one of the following values:.

- **defer**. Activate management-controller settings but do not restart the server. UEFI and server settings are activated after the next restart of the server.
- **immediate**. Activate all settings and restart the server immediately.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example activates a profile on a single device and defers activation until the next reboot.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
configprofiles activate -i 121 -e "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA_bay3" -r defer
```

Related links

- connect
- configprofiles list
- configprofiles unassign

configprofiles delete

This command deletes a deactivated server profile.

Syntax

```
configprofiles delete  -h

configprofiles delete  -i <profile_ID> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-i | --id} *<profile_ID>*
Specifies the ID of the server profile. If an ID is not specified, all server profiles are returned.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example deletes a server profile.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
configprofiles delete -i 121
```

Related links

- connect
- configprofiles list
- configprofiles unassign

configprofiles list

This command retrieves information about one or more server profiles.

Syntax

```
configprofiles list  -h

configprofiles list  [-i <profile_ID>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-i | --id} *<profile_ID>*
Specifies the ID of the server profile. If an ID is not specified, all server profiles are returned.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns information about a specific server profile.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
configprofiles list -i 121 -v id
```

Related links

- connect
- configprofiles activate
- configprofiles delete

configprofiles rename

This command modifies the name of an existing profile.

Syntax

```
configprofiles rename  -h

configprofiles rename  -i <profile_ID> -n <profile_name> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-i | --id} *<profile_ID>*
Specifies the ID of the server profile.

{-n | --name} *<profile_name>*
Specifies the new server profile name.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example renames a specific server profile.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
configprofiles rename -i 121 –n myNewProfileName
```

Related links

- connect
- configprofiles list

configprofiles unassign

This command removes a server profile from a server or chassis bay by deactivating the profile.

Syntax

```
configprofiles unassign  -h

configprofiles unassign  -i <profile_ID> [-p <Boolean>] [--resetimm <Boolean>]
                         [--resetswitch <Boolean>] [-f <Boolean>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-i | --id} *<profile_ID>*
Specifies the ID of the server profile. If an ID is not specified, all server profiles are returned.

{-p | --powerdown} *<Boolean>*
Identifies whether to power off the server. This can be one of the following values.
- **true**. Powers off the server.
- **false**. (default) Does not power off the server.

{--resetswitch} *<Boolean>*
Identifies whether to reset the switch internal port settings to default values. This can be one of the following values.

- **true**. Resets the switch internal port settings.
- **false**. (default) Does not reset the switch internal port settings.

{--resetimm} *<Boolean>*
Identifies whether to reset the baseboard management controller to default settings. This can be one of the following values.

- **true**. Resets the management controller.
- **false**. (default) Does not reset the management controller.

{-f | --force} *<Boolean>*
Identifies whether to force profile deactivation. This can be one of the following values.
- **true**. Forces profile deactivation.
- **false**. (default) Does not force profile deactivation.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example unassign a profile from a specific server.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
configprofiles unassign -i 121
```

The following example unassign a profile from a specific server, resets the management controller and switch settings, and then powers off the server.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
configprofiles unassign -i 121 --resetswitch true --resetimm true –p true
```

Related links

- connect
- configprofiles list
- configprofiles activate
- configprofiles delete

configtargets

This command retrieves a list of deployable servers and server bays that are associated with the specified server pattern or profile.

Syntax

```
configtargets  -h

configtargets   [-i [<pattern_ID> | <profile_ID>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-i | --id} [<*pattern_ID*> | <*profile_ID*>**]**
Specifies the ID of the configuration pattern or server profile. If an ID is not specified, data is returned for all server patterns and profiles.

{-v | --view} <*filter*>
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example retrieves the list of deployable target devices that are associated with a specific server pattern.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
configtargets -i 48
```

Related links

- connect
- configprofiles
- configpatterns

Firmware update commands

The following PyLXCA commands are available for performing firmware updates functions.

updatecomp

This command updates firmware on a specific device, and retrieves the status and progress of firmware updates and firmware repository.

Syntax

```
updatecomp  -h

updatecomp { info | apply } <action_specific_parmeters>
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{info} <*action_specific_parmeters*>
Retrieves the status and progress of firmware updates and firmware repository.

{apply} <*action_specific_parmeters*>
Updates firmware on a specific device.

updatecomp apply

This command updates firmware on a specific device.

Syntax

```
updatecomp apply  -h

updatecomp apply  -a apply [-m <activate_mode>] [-l <device_list>]
                           [-c <CMMs>] [-s <servers>] [-t <storage_systems>] [-w <switches>]

updatecomp apply  -a applyBundle [-s <servers>]

updatecomp apply  -a cancelapply [-l <device_list>]
                [-c <CMMs>] [-s <servers>] [-t <storage_systems>] [-w <switches>]

updatecomp apply  -a power [-c <cmms>] [-s <servers>] [-w <switches>]
```

Note: You must specify one or more of the following parameters: –c, –s, –t, and –w.

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{–a | --action} <*action*>
Specifies the action to take. This can be one of the following values

- **apply**. (default) Applies the associated firmware to the specified device.

  The following set of properties must be specified for each device. Each property is separated by a semicolon, and each set of properties is separated by a comma.
  – UUID of the storage system
  – Fix (firmware-update) ID of the target package to be applied to the component.
  – Component name

  For example:
  `AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA,lnvgy_fw_storage_1.1.1,Controller A`

- **applyBundle**. Applies firmware updates to all components of specified ThinkSystem SR635 and SR655 servers that are not compliant with the assigned firmware-compliance policy using a bundled image that contain the applicable firmware update packages. The bundled image is created during the update process by collecting all firmware-update packages from the compliance policy.

  The UUID of the ThinkSystem SR635 and SR655 server must be specified.

  The updates are applied using immediate activation. During the update process, the target device might be automatically restarted multiple times until the entire process is complete. Ensure that you quiesce all applications on the target device before you proceed.

- **cancelApply**. Cancels the firmware update request on the specified device.

  The following set of properties must be specified for each device. Each property is separated by a semicolon, and each set of properties is separated by a comma.
  – UUID of the storage system
  – Fix (firmware-update) ID of the target package to be applied to the component.
  – Component name

  For example:
  `AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA,lnvgy_fw_storage_1.1.1,Controller A`

- **power**. Performs a power operation on the specified device.

  The following set of properties must be specified for each device. Each property is separated by a semicolon, and each set of properties is separated by a comma.

  - UUID of the device
  - The power state. This can be one of the following values, depending on the type of device:
    - **CMM**: reset
    - **Server**: powerOn, powerOff, powerCycleSoft, powerCycleSoftGrace, powerOffHardGrace
    - **Switch**: powerOn, powerOff, powerCycleSoft
    - **Storage**: powerOn, powerOff, powerCycleSoft

  For example:
  ```
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA;powerOn,BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB;powerOn
  ```

{-c | --cmm} <*CMMs*>
Specifies a set of properties for one or more CMMs.

{-l | --device_list} <*device_list*>
Specifies a list of devices and components, in the following JSON format.

To obtain the component names and firmware-update IDs, use the updaterepo query method.

Parameters	Required / Optional	Type	Description
DeviceList	Required	Array of objects	List of devices
CMMList	Optional	Array of objects	List of CMMs
Components	Required	Array of strings	List of components in the CMM
Component	Required	String	Component name
Fixed	Optional	String	Firmware-update ID of the target package to be applied to the component. Specify if this field if the update is not driven by the firmware-compliance policy. Do not specify if the update is driven by the policy.
UUID	Required	String	UUID of the chassis CMM
ServerList	Optional	Array of objects	List of servers
Components	Required	Array of objects	List of components in the server
Component	Required	String	Component name
Fixid	Optional	String	Firmware-update ID of the target package to be applied to the component. Specify if this field if the update is not driven by the firmware-compliance policy. Do not specify if the update is driven by the policy.

Parameters	Required / Optional	Type	Description
UUID	Required	String	UUID of the server
StorageList	Optional	Array of objects	List of storage devices
Components	Required	Array of objects	List of components in the storage device
Component	Required	String	Component name
Fixid	Optional	String	Firmware-update ID of the target package to be applied to the component. Specify if this field if the update is not driven by the firmware-compliance policy. Do not specify if the update is driven by the policy.
UUID	Required	String	WWNN of the storage device
SwitchList	Optional	Array of objects	List of switches
Components	Required	Array of objects	List of components in the switch
Component	Required	String	Component name
Fixid	Optional	String	Firmware-update ID of the target package to be applied to the component. Specify if this field if the update is not driven by the firmware-compliance policy. Do not specify if the update is driven by the policy.
UUID	Required	String	UUID of the switch

The following example applies firmware updates to multiple devices and components.

```
{
   "DeviceList": [{
      "ServerList": [{
         "UUID": "8BFBADCC33CB11E499F740F2E9903640",
         "Components": [{
            "Fixid": "lnvgy_fw_imm2_tcoo17g-3.00_anyos_noarch",
            "Component": "IMM2 (Backup)"
         },
         {
            "Fixid": "lnvgy_fw_imm2_tcoo17g-3.00_anyos_noarch",
            "Component": "IMM2 (Primary)"
         }]
      }]
   },
   {
      "CMMList": [{
         "UUID": "8BFBADCC33CB11E499F740F2E9903640",
         "Components": [{
```

```
                "Fixid": "lnvgy_fw_imm2_tcoo17g-3.00_anyos_noarch",
                "Component": "CMM")"
             }]
          }]
       },
       {
          "SwitchList": [{
             "UUID": "8BFBADCC33CB11E499F740F2E9903640",
             "Components": [{
                "Fixid": "lnvgy_fw_scsw_en4093r-8.3.9.0_anyons_noarch",
                "Component": "Main Application"
             }]
          }]
       },
       {
          "StorageList": [{
             "UUID": "8BFBADCC33CB11E499F740F2E9903640",
             "Components": [{
                "Fixid": "lnvgy_fw_storage_1.1.1",
                "Component": "Controller a"
             }]
          }]
       }]
    }
```

{-m | --mode} *<activate_mode>*
Indicates when to activate the update. This can be one of the following values.

- **immediate**. (default) During the update process, the target device might be automatically restarted multiple times until the entire process is complete. Ensure that you quiesce all applications on the target device before you proceed.
- **prioritized**. Firmware updates on the baseboard management controller are activated immediately; all other firmware updates are firmware updates are activated the next time the device is restarted. Additional restarts are then performed until the update operation completes. This rule is supported only for servers.
- **delayed**. Some but not all update operations are performed. The target device must be restarted manually to continue the update process. Additional restarts are then performed automatically until the update operation completes. This rule is supported only for servers and rack switches.

{-s | --server} *<servers>*
Specifies a set of properties for one or more servers.

{-t | --storage} *<storage_systems>*
Specifies a set of properties for one or more storage devices.

{-w | --switch} *<switches>*
Specifies a set of properties for one or more switches.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example applies a firmware update on a storage device and immediate activates the update.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
updatecomp apply -a apply -m immediate -t AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA;lnvgy_fw_storage_1.1.1;Controller A
```

The following example applies bundled firmware updates to a ThinkSystem SR635 or SR655 server and immediate activates the update.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
updatecomp apply -a applyBundle -s AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

The following example powers off a server and a storage system.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
updatecomp apply -a power –s AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA;powerOff –t BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB;powerOff
```

Related links

- connect
- updatecomp info

updatecomp info

This command retrieves the status and progress of firmware updates and firmware repository.

Syntax

```
updatecomp info -h

updatecomp info [-q <data_type>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-q | --query} *<data_type>*
The data to return. This can be one of the following values.
- **components**. Returns a list of devices and components that can be updated
- **status**. Returns the status of firmware updates that are in progress. This is the default value

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns the status and progress of firmware updates and firmware-updates repository.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
updatecomp info -q status
```

Related links

- connect
- updatecomp apply

updatepolicy

This command retrieves detailed information about firmware-compliance policies and assigns a compliance policy to a specific device.

Syntax

```
updatepolicy  -h

updatepolicy   { assign | list | query | status } <action_specific_parmeters>
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{assign } *<action_specific_parmeters>*
Assign a compliance policy to one or more devices.

{list} *<action_specific_parmeters>*
Retrieves basic information about firmware-compliance policies.

{query} *<action_specific_parmeters>*
Retrieves detailed information about firmware-compliance policies.

{status} *<action_specific_parmeters>*
Checks the compliance of a device against the assigned compliance policy.

updatepolicy assign

This command assigns a compliance policy to one or more devices and to return the job ID and task IDs for monitoring the status of the job and tasks.

This command starts a job to perform the update-policy action. If successful, the job ID is returned. You can use the job ID in the tasks –j or updatepolicy job_status –j command to determine the status of the job.

Syntax

```
updatepolicy assign  -h

updatepolicy assign  -p <policies> -u <device_UUID> [-t <device_type>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-p | --policy} *<policies>*
Specifies the name of the compliance policy to be assigned.

{-t | --type} *<device_type>*
Specifies the UUID of the device to which you want to assign the compliance policy.

- **CMM**. Chassis Management Module
- **IOSwitch**. Flex switch
- **RACKSWITCH**. RackSwitch switch
- **STORAGE**. Storage device
- **SERVER**. Compute node or rack server

{-u | --uuid} *<device_UUID>*
Specifies the UUID of the device to which you want to assign the compliance policy.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example assigns a compliance policy to a server.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
updatepolicy assign  -p policy_1 -t SERVER -u AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Related links

- connect
- updatepolicy list

updatepolicy list

This command retrieves basic information about all firmware-compliance policies.

Syntax

```
updatepolicy list  -h

updatepolicy list [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns information about all compliance policies.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
updatepolicy list
```

Related links

- connect
- updatepolicy query

updatepolicy query

This command retrieves detailed information about specific firmware-compliance policies.

Syntax

```
updatepolicy query -h

updatepolicy query [-i <information_type>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-i | --info} *<information_type>*
Specifies the type of information to return. This can be one of the following values.

- **firmware**. Returns information about firmware that is applicable to each managed device
- **results**. Returns persisted compare result for servers to which a compliance policy is assigned
- **namelist**. Returns information about all compliance policies.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns information about firmware updates that are applicable to each managed device.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
updatepolicy query -i firmware
```

Related links

- connect
- updatepolicy list

updatepolicy status

This command determines whether devices are compliant with the assigned compliance policy using the job or task ID that was returned when the compliance policy was assigned.

This command starts a job to perform the update-policy action. If successful, the job ID is returned. You can use the job ID in the tasks –j command to determine the status of the job.

Syntax

```
updatepolicy status  -h

updatepolicy status  -j <job_ID> -u <device_UUID> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-j | --job} *<job_ID>*
Specifies the policy job ID for the assign compliance-policy operation. If specified, this command returns status information about the specified job.

{-u | --uuid} *<device_UUID>*
Specifies UUID of the device to which the compliance policy was assigned using the specified job ID. If the value is null or empty, compliance information is returned for all devices in the job.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example retrieves the compliance status of a device.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
updatepolicy status  -j 12 -u AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Related links

- connect
- updatepolicy list

updaterepo

This command retrieves information about and manages firmware updates in the repository.

Syntax

```
updaterepo  -h

updaterepo   { query | read | acquire | refresh | delete } <action_specific_parmeters>
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{query} <*action_specific_parmeters*>
Retrieves information about firmware updates in the firmware-updates repository.

{read} <*action_specific_parmeters*>
Reloads the repository files by clearing the information in cache and retrieving the list of firmware-update file from the repository.

{acquire} <*action_specific_parmeters*>
Uploads the specified firmware updates from the Lenovo Support website, and stores the updates in the firmware-updates repository.

{refresh} <*action_specific_parmeters*>
Retrieves information about the latest available firmware updates from the Lenovo Support website.

{delete} <*action_specific_parmeters*>
Deletes the specified firmware updates from the firmware-updates repository.

updaterepo acquire

This command retrieves information about firmware updates in the firmware-updates repository.

Syntax

```
updaterepo acquire  -h

updaterepo acquire  -f <update_ids> -m <machine_types> [-s <scope>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{–f | --fixids} <*update_ids*>
Specifies a list of firmware-update IDs, separated by a comma.

You can find a list of IDs by using `updaterepro -k updates`.

{–m | --mt} <*machine_types*>
Specifies a list of machine types, separated by a comma.

{–s | --scope} <*scope*>
Specifies the scope of the operation. This can be the following value.
- **payloads**. Acquires the payload (image) file for the specified firmware update.

{-v | --view} <*filter*>
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example download firmware updates.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
updaterepo acquire -f ibm_fw_imm2_1aoo78j-6.20_anyos_noarch -m 7907
```

Related links

- connect
- updaterepo query
- updaterepo refresh

updaterepo delete

This command retrieves information about firmware updates in the firmware-updates repository.

Syntax

```
updaterepo delete  -h

updaterepo delete  -f <update_list> [-t <update_type>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{–f | --fixids} *<update_list>*
Specifies a list of firmware-update IDs, separated by a comma.

You can find a list of IDs by using updaterepo query.

{–t | --type} *<update_types>*
Specifies the update type when **–a delete** is specified. This can be one of the following values.

- **all**. Deletes selected update-package files (payload, change history, readme, and metadata files)
- **payloads**. Deletes only the selected payload (image) files

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example deletes only payload files for a specific firmware update.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
updatepolicy delete –f ibm_fw_imm2_1aoo78j-6.20_anyos_noarch
```

Related links

- connect
- updaterepo query

updaterepo query

This command retrieves information about firmware updates in the firmware-updates repository.

Syntax

```
updaterepo query -h

updaterepo query -k <update_type> [-m <machine_types>] [-s <scope>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-k | --keys} <*update_type*>
Returns specific information about the firmware update. You can specify one of the following values.
- **importDir**. Returns the import directory for the repository.
- **lastRefreshed**. Returns the timestamp of the last repository refresh.
- **publicKeys**. Returns the supported keys (actions) for this parameter.
- **size**. Returns the repository size.
- **supportedMts**. Returns a list of all device types for which the firmware-updates function is supported.
- **updates**. Returns information about all firmware updates in the repository.
- **updatesByMt**. Returns information about firmware updates organized by device type.

{–m | --mt} <*machine_types*>
Specifies a list of machine types, separated by a comma.

{–s | --scope} <*scope*>
Specifies the scope of the operation when –keys is set to **updates** or **updatesByMt**. This can be one of the following values.
- **all**. (default) Returns information about all firmware updates.
- **latest**. Returns information about only the most current firmware updates

{-v | --view} <*filter*>
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns information about all firmware updates by machine type.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
updatepolicy query –k updatesByMt
```

Related links

- connect
- updaterepo read
- updaterepo refresh

updaterepo read

This command retrieves information about firmware updates in the firmware-updates repository.

Syntax

```
updaterepo read  -h

updaterepo read [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-v | --view} <*filter*>
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns information about all firmware updates by machine type.

```
The following example read firmware updates from repository
updatepolicy read
```

Related links

- connect
- updaterepo query

updaterepo refresh

This command retrieves information about firmware updates in the firmware-updates repository.

Syntax

```
updaterepo refresh  -h

updaterepo refresh -m <machine_types> [-s <scope>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{–m | --mt} *<machine_types>*
Specifies a list of machine types, separated by a comma.

{–s | --scope} *<scope>*
Specifies the scope of the operation. This can be one of the following values.

- **all**. Retrieves information about all versions of firmware updates that are available for the specified devices.
- **latest**. (default) Retrieves information about the most current version of firmware updates for the specified devices.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example refresh firmware updates repository.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
updaterepo refresh -m 7903 -s all
```

Related links

- connect
- updaterepo query

Event, alert, and audit log commands

The following PyLXCA commands are available for performing monitoring and event functions, such as retrieving logs, retrieving event and alert status, and setting up event monitors.

log

This command sets or retrieves the current log level for the PyLXCA toolkit.

Syntax

```
log  -h

log  [-l <level>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-l | --lvl} *<level>*
Sets the log level. You can specify one of the following values:

- **DEBUG**. Displays detailed information, typically of interest only when diagnosing problems.
- **INFO**. Displays information that confirms that things are working as expected.
- **WARNING**. Displays information that indicates that something unexpected happened or that a problem might occur soon.
- **ERROR**. Displays information about serious problems, indicating that the software cannot perform a function.
- **CRITICAL**. Displays information about serious errors, indicating that the program itself might be unable to continue running.

When you choose a severity level, messages at that severity level and all lower severity levels are returned. For example, if you specify WARNING, all WARNING, INFO, and DEBUG messages are returned.

Examples

The following example retrieves the current log level.

```
connect --url https://192.0.2.0 --user ADMIN  --noverify
log
```

The following example sets the current log level to warning and lower.

```
connect --url https://192.0.2.0 --user ADMIN  -noverify
log -l WARNING
```

Related links

- connect
- lxcalog

lxcalog

This command retrieves hardware and management-server events in the events log.

Syntax

```
lxcalog  -h

lxcalog  [-f <event_filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-f | --filter} *<event_filter>*
Returns only the events that apply to the specified filters (see Filtering events).

If you do not specify an event filter, all events are returned.

Examples

The following example returns hardware and management-server events in the events log.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
lxcalog
```

Related links

- connect
- log

Filtering events

You can use the parameter **filterWith** to return a subset of all active events based on Java REGEX expressions or based on comparison operators.

You can choose to filter using one of the following methods:

- Java REGEX expressions
- Comparison operators. The following comparison operators are provided:
  - EQ (equal)
  - NOT (not equal)
  - GT (greater than)
  - GTE (greater than or equal to)
  - LT (less than)
  - LTE (less than or equal to).

Note: You cannot combine Java REGEX expressions with comparison operators.

Comparison operators

Some fields support only specific comparison operators.

Parameter	EQ	GT	GTE	LT	LTE	NOT
action	√	√	√	√	√	√
ARGS	√					√
cn	√	√	√	√	√	√
componentID	√	√	√	√	√	√
eventClass	√	√	√	√	√	√
eventDate	√	√	√	√	√	√
eventID	√					√
FAILFRUS	√					√
FAILSNS	√					√
groupUUID	√					√
localLogID	√	√	√	√	√	√
localLogSequence	√	√	√	√	√	√

Parameter	EQ	GT	GTE	LT	LTE	NOT
location	√					√
msgID	√					√
mtm	√					√
search	√	√	√	√	√	√
sequenceid	√	√	√	√	√	√
serialnum	√					√
service	√	√	√	√	√	√
severity	√	√	√	√	√	√
sourceID	√	√	√	√	√	√
sourceLogID						
sourceLogSequence	√	√	√	√	√	√
timeStamp	√	√	√	√	√	√
USERID	√					√

Filtering examples

Filtering is passed as part of the URI parameters. The filter itself is in JSON format. All filters follow the following parameter format.

Obtaining all events that have a cn (sequence ID) greater than 1:

```
https://<Server IP Address>/events?filterWith={"filterType":"FIELDNOTREGEXAND",
"fields":[{"operation":"GT","field":"cn","value":"1"}]}
```

Events can be filtered based on the following fields:

Parameter	Comparison operators example	Regex expression example

action	{
	"operation":"EQ",
	"field":"action",
	"value":"ABCDE"
	}
		{
		"field":"action",
		"value":"ABCDE"
		}

cn	{
	"operation":"EQ",
	"field":"cn",
	"value":"1"
	}
		{
		"field":"cn",
		"value":"1"
		}

componentID	{
	"operation":"EQ",
	"field":"componentID",
	"value":"FFFFF"
	}
		{
		"field":"componentID",
		"value":"FFFFF"
		}

Parameter	Comparison operators example	Regex expression example

eventClass	{
	"operation":"EQ",
	"field":"eventClass",
	"value":"200"
	}
	or
	{
	"operation":"EQ",
	"field":"eventClass",
	"value":"AUDIT"
	}
		{
		"field":"eventClass",
		"value":"200"
		}
		or
		{
		"field":"eventClass",
		"value":"AUDIT"
		}

eventDate	{
	"operation":"EQ",
	"field":"eventDate",
	"value":"2014-02-11T09:54:58Z"
	}
		{
		"field":"eventDate",
		"value":"2014-02-11T09:54:58Z"
		}

eventID	{
	"operation":"EQ",
	"field":"eventID",
	"value":"FQXHMCP5810I"
	}
		{
		"field":"eventID",
		"value":"FQXHMCP5810I"
		}

groupUUID	{
	"operation":"EQ",
	"field":"groupUUID",
	"value": [
	"FFB657408BEB4161950704AB",
	"59AFBFCF8DBB376A25D68A0A"]
	}
		{
		"field":"groupUUID",
		"value": [
		"FFB657408BEB4161950704AB",
		"59AFBFCF8DBB376A25D68A0A"]
		}

localLogID	{
	"operation":"EQ",
	"field":"localLogID",
	"value":"ABCDE"
	}
		{
		"field":"localLogID",
		"value":"ABCDE"
		}

localLogSequence	{
	"operation":"EQ",
	"field":"localLogSequence",
	"value":"1"
	}
		{
		"field":"localLogSequence",
		"value":"1"
		}

location	{
	"operation":"EQ",
	"field":"location",
	"value":"ABCDE"
	}
		{
		"field":"location",
		"value":"ABCDE"
		}

msgID	{
	"operation":"EQ",
	"field":"msgID",
	"value":"ABCDE"
	}
		{
		"field":"msgID",
		"value":"ABCDE"
		}

mtm	{
	"operation":"EQ",
	"field":"mtm",
	"value":"ABCDE"
	}
		{
		"field":"mtm",
		"value":"ABCDE"
		}

Parameter	Comparison operators example	Regex expression example

search	{
	"operation":"EQ",
	"field":"search",
	"value":"ABCDE"
	}
		{
		"field":"search",
		"value":"ABCDE"
		}

sequenceid	{
	"operation":"EQ",
	"field":"sequenceid",
	"value":"1"
	}
		{
		"field":"sequenceid",
		"value":"1"
		}

serialnum	{
	"operation":"EQ",
	"field":"serialnum",
	"value":"ABCDE"
	}
		{
		"field":"serialnum",
		"value":"ABCDE"
		}

service	{
	"operation":"EQ",
	"field":"service",
	"value":"100"
	}
	or
	{
	"operation":"EQ",
	"field":"service",
	"value":"NONE"
	}
		{
		"field":"service",
		"value":"100"
		}
		or
		{
		"field":"service",
		"value":"NONE"
		}

severity	{
	"operation":"EQ",
	"field":"severity",
	"value":"200"
	}
	or
	{
	"operation":"EQ",
	"field":"severity",
	"value":"INFORMATIONAL"
	}
		{
		"field":"severity",
		"value":"200"
		}
		or
		{
		"field":"severity",
		"value":"INFORMATIONAL"
		}

sourceID	{
	"operation":"EQ",
	"field":"sourceID",
	"value":"ABCDE"
	}
		{
		"field":"sourceID",
		"value":"ABCDE"
		}

sourceLogID	{
	"operation":"EQ",
	"field":"sourceLogID",
	"value":"ABCDE"
	}
		{
		"field":"sourceLogID",
		"value":"ABCDE"
		}

Parameter	Comparison operators example	Regex expression example

sourceLogSe-quence	{
	"operation":"EQ",
	"field":"sourceLogSequence",
	"value":"1234"
	}
		{
		"field":"sourceLogSequence",
		"value":"1234"
		}

timeStamp	{
	"operation":"EQ",
	"field":"timeStamp",
	"value":"2014-02-11T09:54:58Z"
	}
		{
		"field":"timeStamp",
		"value":"2014-02-11T09:54:58Z"
		}

Applying a filter to match a single event with a sequence ID equal to 16:

```
{
   "filterType":"FIELDNOTREGEXAND",
   "fields": [{
      "operation":"EQ",
      "field":"cn",
      "value":"16"
   }]
}

{
   "filterType":"FIELDREGEXAND",
   "fields": [{
      "field":"cn",
      "value":"16"
   }]
}
```

These two filters are equivalent; they both will match with a single event, the event that has the cn/sequenceid equal to 16.

The filtering is composed of two parts:

1. The first part is the filterType that can have only one value from the following enumeration:
   - **FIELDREGEXAND**. Regex filter of type AND
   - **FIELDREGEXOR**.Regex filter of type OR
   - **FIELDREGEXNOT**. Regex filter of type NOT
   - **FIELDNOTREGEXAND**. Non-Regex filter of type AND
   - **FIELDNOTREGEXOR**. Non-Regex filter of type OR
   - **FIELDNOTREGEXNOT**. Non-Regex filter of type NOT

   The REGEX filters accept only REGEX expressions in the "value" field. The Non-REGEX filters do not accept REGEX expressions in the "value" field. The Non-REGEX filter works with the six comparison operators (EQ, NOT, GT, GTE, LT, LTE). It also has a special field called "operation" in which to specify the comparison operation.

   The "filterType" is a mandatory field.
2. The second part is an enumeration of "fields" that define the target of the filter match. This field is required.The "fields" is a JSONArray Enumeration composed of JSON Objects. In the above example it can be seen that there is only one JSON ({"operation":"EQ" , "field":"cn" , "value":"16"}) in the entire JSONArray ([{"operation":"EQ" , "field":"cn" , "value":"16"}]).

Applying a complex filter:

```
{
   "filterType":"FIELDNOTREGEXAND",
   "fields": [
      {"operation":"GT", "field":"cn", "value":"16"},
```

```
        {"operation":"GTE", "field":"severity", "value":"400"},
        {"operation":"GTE", "field":"timeStamp", "value":"2014-02-11T09:20:35Z"}
    ]
}
```

This filter will match all events that have the cn/sequenceid greater than 16, a severity greater than or equal to 400, and a timeStamp greater than or equal to 9:20:35 Zulu - February 11, 2014.

Job commands

The following PyLXCA commands are available for performing task and job management functions.

tasks

This command retrieves information about, cancels, deletes, or updates properties for one or more tasks (jobs).

Attention: Currently, `tasks` can be used only as an API in Python scripts. It cannot be used from the command line in a Python or interactive shell.

Syntax

```
tasks  -h

tasks  [-j <task_ID_list] [-c <Boolean>]

tasks  -a cancel -j <task_ID_list>

tasks  -a create -u <properties_JSON>

tasks  -a delete -j <task_ID_list>

tasks  -a update -u <properties_JSON>

tasks  -t <template>
```

Options

If no parameters are specified, this command retrieves information about all jobs.

{-h | --help}
Displays the syntax and brief usage information for this command.

{-a | --action} <*action*>
Specifies the action to take. This can be one of the following values. If no action is specified, this command returns information about the task.
- **cancel**. Cancels the specified tasks.
- **create**. Creates a task.
- **delete**. Cancels the specified tasks.
- **update**. Updates the job state and percent complete for the specified task.

{-j | --jobUID} <*task_ID_list*>
The ID of one or more tasks, separated by a comma. If no task ID is specified, information about all tasks is returned

{-c | --children} <*Boolean*>
Indicates whether to return information about subtasks. This can be one of the following values.
- **true**. (default). Returns information about subtasks.
- **false**. Does not return information about subtasks.

{-t | --template} *<template>*
Specifies the properties for a new task, in JSON format.

Attributes	Required / Optional	Type	Description
args	Optional	Array of strings	List of arguments to include in the job title
cancelRESTBody	Required when **cancelRESTMethod** is PUT or POST	Object	JSON formatted request body that is required to cancel the job, specified as a key-value pairs
cancelRESTMethod	Required when **cancelURL** is specified	String	REST method to cancel the job. This can be one of the following values. • **GET** • **POST** • **PUT** • **DELETE**
cancelURL	Optional	String	URL to cancel the job
children	Required	Array of objects	Information about each subtask in the job (task) The attributes in this object are the same as the top-level attributes.
createdBy	Required	String	ID of the user that created the subtask. To obtain the user ID, see the GET /useraccounts in the Lenovo XClarity Administrator online documentation.
createDate	Required	String	Date and time that the job was created
customUid	Optional	String	User-defined job ID The response body associates the user-defined job ID with the job UID that is assigned by XClarity Administrator.
endDate	Optional	String	Date and time that the job completed
expirationTimeOut	Required when **cancelURL** is specified	Integer	Number of seconds after which the job expires. Use `-1` for jobs that do not expire.

Attributes	Required / Optional	Type	Description
jobCategory	Required	String	Subtask category. This can be one of the following values. • **Backup** • **Configuration** • **Custom** • **Firmware** • **Health** • **Inventory** • **Management** • **OsDeployment** • **OsDriverUpdates** • **OsImport** • **OsProfileExport** • **Power** • **RemoteAccess** • **SelfMaintenance** • **Service** • **SwitchConfiguration** • **SystemID** • **Unknown**
jobState	Required	String	State of the job. This can be one of the following values. • **Aborted** • **Blocked** • **Cancelled** • **CancelledWithError** • **CancelledWithWarning** • **Cancelling** • **Complete** • **CompleteWithError** • **CompleteWithWarning** • **Expired** • **Initializing** • **Interrupted** • **InterruptedWithError** • **InterruptedWithWarning** • **Investigating** • **Pending** • **Resolved** • **Running** • **RunningWithError** • **RunningWithWarning** • **Skipped** • **Stopped** • **StoppedWithError** • **StoppedWithWarning** • **Unknown** • **Uploading** • **Validating** • **Waiting**

Attributes	Required / Optional	Type	Description
jobSummary	Required	Object	Information about the job summary A job summary consists of the following attributes: • **Description**. Describes the problem (message) in a small phrase. • **User action**. Provides details about the problem and the steps to perform to recover from the problem. Provide as much detail as possible to help the user resolve the problem without contacting Lenovo Support. • **Severity**. The severity of the job. The job summary is optional for a job that completes successfully; however, it is good practice to set the summary, even when the severity is informational.
actionArgs	Optional	Array of strings	List of arguments (variables) to include the recovery actions
actionBundleKey	Required	String	Key for the recovery actions in the translated message.properties file
actionBundleName	Required	String	Name of the translated message.properties file that contains the recovery actions

Attributes	Required / Optional	Type	Description

actionText	Required	String	Recovery actions to use if there is no translation
			The actions can be simple text or formatted text (such as paragraphs, ordered lists, and bold text) using an array of JSON objects with the following attributes.
			• **format**. (Array of strings) List of formats for the text. This can be one of the following values.
			– **bold**. Corresponds to the <b> HTML tag.
			– **italic**. Corresponds to the <i> HTML tag.
			– **underline**. Corresponds to the <u> HTML tag.
			– **link**. Corresponds to the <a> HTML tag.
			– **newline**. Corresponds to the
			 HTML tag.
			– **paragraph**. Corresponds to the <p> HTML tag.
			– **quotation**. Corresponds to the <q> HTML tag.
			– **orderedList**. Corresponds to the HTML tag.
			– **bulletList**. Corresponds to the HTML tag.
			– **listElement**. Corresponds to the HTML tag.
			• **link**. (String) URL to be linked to
			• **text**. (String or Array of strings) Text to be formatted
			
			For example:
			`['{'`
			`"text": "A simple paragraph.",`
			`"format": []`
			`'}',`
			`'{'`
			`"text": [],`
			`"format": ["newline"]`
			`'}',`
			`'{'`
			`"text": ['{'`
			`"text": "Segment pieces to format.",`
			`"format": ["listElement"]`
			`'}',`
			`'{'`
			`"text": ['{'`
			`"text": "If the segmented text contains ",`
			`"format": []`
			`'}',`
			`'{'`
			`"text": "multiple tags",`
			`"format": ["bold"]`
			`'}',`
			`'{'`
			`"text": ", segment them as well.",`
			`"format": []`
			`'}' ],`
			`"format": ["listElement"]`
			`'}',`
			`"format": ["orderedList"]`
			`'}']`

descriptionArgs	Required	Array of strings	List of arguments (variables) to include the message description
descriptionBundleKey	Required	String	Key for the message description in the translated message.properties file

Attributes	Required / Optional	Type	Description
descriptionBundleName	Required	String	Name of the translated message.properties file that contains the message description

descriptionText	Required	String	Message description to use if there is no translation
			The description can be simple text or formatted text (such as paragraphs, ordered lists, and bold text) using an array of JSON objects with the following attributes.
			• **format**. (Array of strings) List of formats for the text. This can be one of the following values.
			– **bold**. Corresponds to the <b> HTML tag.
			– **italic**. Corresponds to the <i> HTML tag.
			– **underline**. Corresponds to the <u> HTML tag.
			– **link**. Corresponds to the <a> HTML tag.
			– **newline**. Corresponds to the
			 HTML tag.
			– **paragraph**. Corresponds to the <p> HTML tag.
			– **quotation**. Corresponds to the <q> HTML tag.
			– **orderedList**. Corresponds to the HTML tag.
			– **bulletList**. Corresponds to the HTML tag.
			– **listElement**. Corresponds to the HTML tag.
			• **link**. (String) URL to be linked to
			• **text**. (String or Array of strings) Text to be formatted
			
			For example:
			`['{'`
			`"text": "A simple paragraph.",`
			`"format": []`
			`'}',`
			`'{'`
			`"text": [],`
			`"format": ["newline"]`
			`'}',`
			`'{'`
			`"text": ['{'`
			`"text": "Segment pieces to format.",`
			`"format": ["listElement"]`
			`'}',`
			`'{'`
			`"text": ['{'`
			`"text": "If the segmented text contains ",`
			`"format": []`
			`'}',`
			`'{'`
			`"text": "multiple tags",`
			`"format": ["bold"]`
			`'}',`
			`'{'`
			`"text": ", segment them as well.",`
			`"format": []`
			`'}' ],`
			`"format": ["listElement"]`
			`'}',`
			`"format": ["orderedList"]`
			`'}']`

Attributes	Required / Optional	Type	Description
severity	Required	String	Severity of the job. This can be one of the following values. • **Unknown**. The severity is unknown. • **Informational**. The activity started or ended successfully. • **Warning**. The activity completed, but there are some problems that the user must be aware of (for example, Windows was deployed but failed to set the IP addresses successfully). The user can decide if action is needed. • **Minor**. Action is needed, but the situation is not serious at this time. • **Major**. Action is needed now. • **Critical**. The activity failed. Action is needed now and the scope is broad (perhaps an imminent outage to a critical resource will result). • **Fatal**. The activity failed. A non-recoverable error has occurred.
jobTitle	Required	String	Job title to use if there is no translation
jobTitleBundle	Required	String	Location of the message.properties file where the job title can be found if it is not located in the default task-management properties file. The key that identifies the job title in the properties file.
jobTitleKey	Required	String	Key that identifies the job title in the message. properties file
logs	Required	Array of objects	Information about each log entry
args	Optional	Array of strings	Arguments of the message
logBundleName	Required	String	Location where messages.properties can be found if it is not located in the default task-management properties file
logBundleKey	Required	String	Key that identifies the job title in the message. properties file
logDate	Optional	String	Log date
logText	Required	String	User-defined log text

Attributes	Required / Optional	Type	Description
severity	Required	String	Severity of this log. This can be one of the following values. • **Unknown**. The severity is unknown. • **Informational**. The activity started or ended successfully. • **Warning**. The activity completed, but there are some problems that the user must be aware of (for example, Windows was deployed but failed to set the IP addresses successfully). The user can decide if action is needed. • **Minor**. Action is needed, but the situation is not serious at this time. • **Major**. Action is needed now. • **Critical**. The activity failed. Action is needed now and the scope is broad (perhaps an imminent outage to a critical resource will result). • **Fatal**. The activity failed. A non-recoverable error has occurred.
percentage	Optional	String	Percentage complete of the subtask. This can be an integer from `0 - 100`.
predefinedUid	Optional	String	Job ID that is defined by the component that requested the job
rebootPersistet	Optional	Boolean	Indicates whether an active job persists after a reboot. This can be one of the following values. • **true**. (default) The state of the job is not changed after a reboot. • **false**. The job state changes to Interrupted after a reboot.
startDate	Optional	String	Date and time that the job started
stoppedBy	Optional	String	ID of the user that stopped the job
targetUUID	Required	String	ID of the device that is the target for the job.
uid	Optional	String	Job ID
weight	Optional	String	Weight of the job, which is used to update the percent complete Jobs that take a long time to complete have a higher weight than fast jobs.

The following example creates a task with no subtasks.

```
{
   "children": [],
   "createdBy": "ADMIN",
   "createDate": "2019-02-27T16:26:01Z",
   "jobCategory": "Configuration",
   "jobState": "Running",
   "jobSummary": {
      "actionBundleKey": "",
      "actionBundleName": "",
      "actionText": "",
      "descriptionArgs": [],
      "descriptionBundleKey": "",
```

```
      "descriptionBundleName": "",
      "severity": ""
   },
   "jobTitle": "",
   "jobTitleBundle": "com.lenovo.lxca.profile.mri.ConfigurationPatternsMRI",
   "jobTitleKey": "SERVER_PROFILE_JOB_NAME",
   "logs": [{
      "logBundleKey": "",
      "logBundleName": "",
      "logText": "",
      "severity": ""
   }],
   "targetUUID": "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
}
```

{-u | --updateList} <*properties_JSON*>

Specifies the properties and tasks to update, in JSON format.

Parameter	Required / Optional	Type	Description
jobUID	Required	String	Job ID
jobState	Optional	String	Job state. This can be one of the following values. • **Aborted** • **Blocked** • **Cancelled** • **CancelledWithError** • **CancelledWithWarning** • **Cancelling** • **Complete** • **CompleteWithError** • **CompleteWithWarning** • **Expired** • **Initializing** • **Interrupted** • **InterruptedWithError** • **InterruptedWithWarning** • **Investigating** • **Pending** • **Resolved** • **Running** • **RunningWithError** • **RunningWithWarning** • **Skipped** • **Stopped** • **StoppedWithError** • **StoppedWithWarning** • **Unknown** • **Uploading** • **Validating** • **Waiting**
percentage	Optional	Integer	Percentage complete

For example:

```
[{
   "jobUID": "127"
   "jobState": "Validating",
```

```
    "percentage": 95
}]
```

Examples

The following example returns information about a specific task and also its subtasks.

```
connect --url https://192.0.2.0 --user ADMIN -noverify
tasks -j 127
```

The following example returns information about a specific task but not the subtasks.

```
connect --url https://192.0.2.0 --user ADMIN -noverify
tasks -j 127 -c false
```

The following example cancels multiple tasks.

```
connect --url https://192.0.2.0 --user ADMIN -noverify
tasks -a cancel -j 127, 140
```

The following example deletes a specific task.

```
connect --url https://192.0.2.0 --user ADMIN -noverify
tasks -a cancel -j 127
```

The following example updates the job state and percent complete for multiple tasks.

```
connect --url https://192.0.2.0 --user ADMIN -noverify
tasks -a update -u '[{"jobUID":"127","jobState":"Validating","percentage":50}, {"jobUID":"140",
                     "jobState":"Complete","percentage":100}]'
```

Related links

- connect

Operating system deployment commands

The following PyLXCA commands are available for performing operating-system deployment functions.

osimages

This command retrieves information about OS images, modifies deployment settings, and deploys an operating system to a managed server.

Syntax

```
osimages  -h

osimages  -h {list | globalsettings | hostsettings | hostplatforms | import | delete
          | remotefileservers } <action_specific_parmeters>
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{list} <*action_specific_parmeters*>
Retrieves information about all OS images. For more information, see osimages list.

{globalsettings} <*action_specific_parmeters*>
Retrieves or modifies global operating-system deployment settings. Global settings serve as defaults settings when operating systems are deployed. For more information, see osimages globalsettings.

{hostsettings} <*action_specific_parmeters*>
Retrieves information about network and storage settings for all servers, and creates and modifies network and storage settings for one or more servers. For more information, see osimages hostsettings.

{hostplatforms} <*action_specific_parmeters*>
Retrieves information about the host platforms, and deploys operating-system images to the host platforms as a job. For more information, see osimages hostplatforms.

{import} <*action_specific_parmeters*>
Imports OS images and custom files from a remote server to the repository. For more information, see osimages import.

{delete} <*action_specific_parmeters*>
Deletes the specified OS images from the repository. For more information, see osimages delete.

{remotefileservers} <*action_specific_parmeters*>
Retrieves information about all remote file-server profiles, and creates and modifies a remote file-server profile. For more information, see osimages remotefileservers.

osimages list

This command retrieves information about all operating-system images.

Syntax

```
osimages list  -h

osimages list [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-v | --view} <*filter*>
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns list of OS images.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
osimages list
```

Related links

- connect

osimages globalsettings

This command retrieves or modifies global operating-system deployment settings. Global settings serve as defaults settings when operating systems are deployed.

Syntax

```
osimages globalsettings  -h

osimages globalsettings  -o <settings_JSON> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-o | --osimages_dict} <*settings_JSON*>

Modifies the global operating-system deployment settings. This must be a correctly formatted JSON of the settings that you want to modify.

If not specified, this command returns the current settings.

Parameters	Required / Optional	Type	Description
activeDirectory	Required	Object	Information about Active Directory
allDomains	Required	Array of objects	Information about all Active Directory domains
domainName	Required	String	Name of the Active Directory domain
id	Required	String	ID of the Active Directory domain
OU	Required	String	Organizational unit of the Active Directory domain
defaultDomain	Required	String	Name of the default domain. This is one of the Active Directory domains in the **AllDomains** list).
credentials	Required	Array of objects	Information about user credentials
password	Required	String	Password of the operating system to be deployed For Windows 2012, this is the password for the “Administrator” user. For Linux and ESXi, this is the password for the “root” user.
passwordChanged	Required	Boolean	Indicates whether to modify the password. This can be one of the following values. • **true** • **false**
type	Required	String	Operating system type. This can be one of the following values. • **ESXi** • **LINUX** • **WINDOWS**
deploySettings	Optional	Object	Information about deployment settings
ipDuplicationCheckEnabled	Optional	Boolean	Indicates whether the duplicate-IP check runs at the beginning of an OS deployment. This can be one of the following values. • **true**. Duplicate-IP check runs • **false**. Duplicate-IP check does not run
stateTransitionTimeout	Optional	Integer	Amount of time, in seconds, to wait between status updates before failing the OS deployment
ipAssignment	Required	String	Host network setting option for operating system deployment. This can be one of the following values. • **dhcpv4** • **staticv4** • **staticv6**
isVLANMode	Required	String	Indicates whether VLAN mode is used. This can be one of the following values. • **true**. VLAN mode is used. • **false**. VLAN mode is not used.
licenseKeys	Required	Object	Information about volume-license keys

Parameters	Re-quired / Optional	Type	Description
win11	Required	Object	Information about volume-license keys for Microsoft Windows 11
enterpriseLicenseKey	Required	String	Volume-license key that is used to deploy the enterprise version of Microsoft Windows 11
workstationLicenseKey	Required	String	Volume-license key that is used to deploy the workstation version of Microsoft Windows 11
win10	Required	Object	Information about volume-license keys for Microsoft Windows 10
enterpriseLicenseKey	Required	String	Volume-license key that is used to deploy the enterprise version of Microsoft Windows 10
workstationLicenseKey	Required	String	Volume-license key that is used to deploy the workstation version of Microsoft Windows 10
win2022r1	Required	Object	Information about volume-license keys for Microsoft Windows 2022 R1
dataCenterLicenseKey	Required	String	Volume-license key that is used to deploy the data center version of Microsoft Windows 2022 R1
standardLicenseKey	Required	String	Volume-license key that is used to deploy the standard version of Microsoft Windows 2022 R1
win2019r1	Required	Object	Information about volume-license keys for Microsoft Windows 2019 R1
dataCenterLicenseKey	Required	String	Volume-license key that is used to deploy the data center version of Microsoft Windows 2019 R1
standardLicenseKey	Required	String	Volume-license key that is used to deploy the standard version of Microsoft Windows 2019 R1
win2016r1	Required	Object	Information about volume-license keys for Microsoft Windows 2016 R1
dataCenterLicenseKey	Required	String	Volume-license key that is used to deploy the data center version of Microsoft Windows 2016 R1
standardLicenseKey	Required	String	Volume-license key that is used to deploy the standard version of Microsoft Windows 2016 R1

For example,

```
{
   "activeDirectory": {
      "defaultDomain": "labs.lenovo.com",
      "allDomains": []
   },
   "credentials": [{
      "passwordChanged": true,
      "password": "Test1234",
      "type": "LINUX",
      "name": "root"
   },
   {
      "passwordChanged": false,
      "password": null,
      "type": "WINDOWS",
      "name": "Administrator"
```

```
        }],
        "ipAssignment": "dhcpv4",
        "isVLANMode": false,
        "licenseKeys": {
           "win2022r1": {
              "dataCenterLicenseKey": "",
              "standardLicenseKey": ""
           },
           "win2019r1": {
              "dataCenterLicenseKey": "",
              "standardLicenseKey": ""
           },
           "win2016r1": {
              "dataCenterLicenseKey": "",
              "standardLicenseKey": ""
           },
           "win2012r2": {
              "dataCenterLicenseKey": "",
              "standardLicenseKey": ""
           },
           "win10": {
              "enterpriseLicenseKey": "AAAA4-BBBBB-CCCCC-DDDDD-EEEEE",
              "workstationLicenseKey": "AAAA3-BBBBB-CCCCC-DDDDD-EEEEE"
           },
           "win11": {
              "enterpriseLicenseKey": "AAAA4-BBBBB-CCCCC-DDDDD-EEEEE",
              "workstationLicenseKey": "AAAA3-BBBBB-CCCCC-DDDDD-EEEEE"
           }
        }
     }
```

{-v | --view} *<filter>*

Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns the current global settings.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
osimages globalsettings -v globalsettings
```

The following example sets the default password for Linux.

```
connect --url https://192.0.2.0 --user ADMIN -noverify
osimages globalsettings -v result -o '{"activeDirectory": {"defaultDomain": "labs.lenovo.com",
          "allDomains": []}, "credentials": [{"passwordChanged": true, "password": "Test1234",
          "type": "LINUX", "name": "root"}, {"passwordChanged": false, "password": null,
          "type":"WINDOWS", "name": "Administrator"}], "ipAssignment": "dhcpv4",
          "isVLANMode": false,"licenseKeys": {"win2019r1": {"dataCenterLicenseKey": "",
          "standardLicenseKey": ""},"win2016r1": {"dataCenterLicenseKey": "",
          "standardLicenseKey": ""},"win2012r2":{"dataCenterLicenseKey": "",
          "standardLicenseKey": ""}, "standardLicenseKey": ""}}}'
```

Related links

- connect
- osimages hostsettings

osimages hostsettings

This command retrieves information about network and storage settings for all servers, and creates and modifies network and storage settings for one or more servers.

Syntax

```
osimages hostsettings  -h

osimages hostsettings  [-a <action> -o <settings_JSON>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-a | --action} *<action>*
Specifies the action to perform. This can be one of the following values.

- **create**. Creates host settings for a specific host platform.
- **update**. Modifies host settings for a specific host platform.
- **delete**. Deletes host settings for a specific host platform

If no action is specified, this command returns information about all host (global and storage) settings.

{-o | --osimages_dict} *<settings_JSON>*
Specifies the host settings for each host platform, in JSON format. You must specify this parameter to create, modify, or delete host settings

Table 2. Delete host settings

Parameters	Required / Optional	Type	Description
uuid	Required	String	UUID of the host platform

Table 3. Create or update host settings

Parameters	Required / Optional	Type	Description
hosts	Required	Array of objects	Information about the host settings for each host platform.
networkSettings	Optional	Object	Information about network settings
dns1	Optional	String	Preferred DNS server for the host server to be used after the operating system is deployed
dns2	Optional	String	Alternative DNS server for the host server to be used after the operating system is deployed
gateway	Optional	String	Gateway of the host server to be used after the operating system is deployed. This is used when the network setting is set to **static** in the Global OS deployment settings. **Tip:** To determine the IP mode, use osimages globalsettings.
hostname	Optional	String	Hostname for the host server. If a hostname is not specified, a default hostname is assigned.

Table 3. Create or update host settings (continued)

Parameters	Required / Optional	Type	Description
ipAddress	Optional	String	IP address of the host server to be used after the operating system is deployed. This is used when the network setting is set to **static** in the Global OS deployment settings.
mtu	Optional	Long	Maximum transmission unit for the host to be used after the operating system is deployed
prefixLength	Optional	String	Prefix length of the host IP address to be used after the operating system is deployed. This is used when the network setting is set to **static IPv6** in the Global OS deployment settings.
selectedMAC	Optional	String	MAC address of the host server to which the IP address is to be bound The MAC address is set to AUTO by default. This setting automatically detects the Ethernet ports that can be configured and used for deployment. The first MAC address (port) that is detected is used by default. If connectivity is detected on a different MAC address, the XClarity Administrator host is automatically restarted to use the newly detected MAC address for deployment, and selectedMAC is set to the newly detected MAC address. VLAN mode is supported only for servers that have MAC addresses in their inventory. If AUTO is the only the MAC address that is available for a server, then VLANs cannot be used to deploy operating systems to that server. **Tip:** To obtain the MAC address, use the **macaddress value** field in osimages hostplatforms.
subnetMask	Optional	String	Subnet mask of the host server to be used after the operating system is deployed. This is used when the network setting is set to static in the Global OS deployment settings. **Tip:** To determine the IP mode, use osimages globalsettings.
vlanId	Optional	String	VLAN ID for operating-system VLAN tagging This parameter is valid only if in VLAN mode is enabled (see osimages globalsettings). **Important:** Only specify a VLAN ID when a VLAN tag is required to function on the network. Using VLAN tags can affect the network routability between the host operating system and the Lenovo XClarity Administrator.
storageSettings	Optional	Object	Preferred storage location on which you want to deploy operating-system images

Table 3. Create or update host settings (continued)

Parameters	Required / Optional	Type	Description
targetDevice	Optional	String	Target device. This can be one of the following values. • **localdisk**. Local disk drive. The first enumerated local disk drive in the managed server is used. • **M.2drive**. M.2 drive. The first enumerated M.2 drive in the managed server is used. • **usbdisk**. Embedded USB Hypervisor. This location is applicable only when a VMware ESXi image is being deployed to managed servers. If two hypervisor keys are installed on the managed server, the VMware installer selects the first enumerated key for deployment. • **lunpluswwn=***LUN@WWN*. FC SAN storage (for example, `lunpluswwn=2@50:05:07:68:05:0c:09:bb`). • **lunplusiqn=***LUN@IQN*. iSCSI SAN Storage (for example, `lunplusiqn=0@iqn.1990-01.com.lenovo:tgt1`). Specifying the *IQN* is optional if only one iSCSI target is configured If the *IQN* is not specified, the first detected iSCSI target is selected for OSDN. If specified, and exact match is made. **Note:** For ThinkServer servers, this value is always “localdisk.”
uuid	Required	String	UUID of the host platform

For example,

```
{
   "hosts": [{
      "networkSettings": {
         "dns2": "",
         "dns1": "10.240.0.10",
         "gateway": "10.243.0.1",
         "hostname": "nodeundefined",
         "ipAddress": "10.243.27.27",
         "mtu": 1500,
         "prefixLength": 64,
         "selectedMAC": "AUTO",
         "subnetMask": "255.255.240.0",
         "vlanId": 0
      },
      "storageSettings": {
         "targetDevice": "localdisk"
      },
      "uuid": "A1445C6FDBAA11E6A87F86E06E3ACCCC"
   }]
}
```

{-v | --view} *<filter>*

Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns information about all host settings.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
osimages hostsettings -v hostsettings
```

The following example creates host settings for a specific host platform.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
osimages hostsetttings --action create ' -v result
         -o '{"hosts": [{"networkSettings": {"dns2": "", "dns1": "10.240.0.10",
         "gateway": "10.243.0.1", "hostname": "nodeundefined",
         "ipAddress": "10.243.27.27", "mtu": 1500, "prefixLength": 64,
         "selectedMAC": "AUTO", "subnetMask": "255.255.240.0", "vlanId": 0},
         "storageSettings": {"targetDevice": "localdisk"},
         "uuid": "A1445C6FDBAA11E6A87F86E06E3ACCCC"}]}
```

The following example modifies host settings for a specific host platform.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
osimages hostsetttings --action update --update True -v result
         -o '{{"hosts": [{"storageSettings": {"targetDevice": "localdisk"},
         "networkSettings": {"dns2": "", "dns1": "10.240.0.10", "hostname": "nodeundefined",
         "vlanId": 0, "selectedMAC": "AUTO", "gateway": "10.243.0.1",
         "subnetMask": "255.255.240.0", "mt": 1500, "prefixLength": 64,
         "ipAddress": "10.243.27.27"}, "uuid": "A1445C6FDBAA11E6A87F86E06E3ACCCC"}]}'
```

The following example deletes host settings for a host platform.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
osimages hostsettings --action delete -v result
         -o '{\"uuid\": \"A1445C6FDBAA11E6A87F86E06E3AFFFF\"}'
```

Related links

- connect
- osimages globalsettings
- osimages hostplatforms

osimages hostplatforms

This command retrieves information about the host platforms, and deploys operating-system images to the host platforms as a job.

Syntax

```
osimages hostplatforms  -h

osimages hostplatforms -o <settings_JSON> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-o | --osimages_dict} *<settings_JSON>*
Deploys an operating system image to a host platform using the specified OS deployment settings, in JSON format.

Parameters	Required / Optional	Type	Description
adusername	Optional	String	(Windows only) User name for the Active Directory domain that is specified in the **windowsDomain** field **Note:** This parameter is required if you want a Windows operating system to join an Active Directory domain.
adpassword	Optional	String	(Windows only) Password for the Active Directory user name **Note:** This parameter is required if you want a Windows operating system to join an Active Directory domain.
configFileId	Required if **selected-Image** includes a custom configuration-settings file; otherwise ignored.	String	ID of the custom configuration-settings file to use for this OS deployment
licenseKey	Optional	String	License key to be used for Microsoft Windows or VMware ESXi. If you do not have a license key, you can set this field to null.
networkSettings	Required	Object	Information about network settings
dns1	Optional	String	Preferred DNS server for the host server to be used after the operating system is deployed
dns2	Optional	String	Alternative DNS server for the host server to be used after the operating system is deployed
gateway	Required if using static IP addresses. Optional if using DHCP.	String	Gateway of the host server to be used after the operating system is deployed. This is used when the network setting is set to static in the Global OS deployment settings. **Tip:** To determine the IP mode, use osimages globalsettings.
hostname	Optional	String	Hostname for the host server. If a hostname is not specified, a default hostname is assigned.
ipAddress	Required if using static IP addresses.	String	IP address of the host server to be used after the operating system is deployed. This is used when the network setting is set to static in the Global OS deployment settings.
mtu	Optional	Long	Maximum transmission unit for the host to be used after the operating system is deployed
prefixLength	Optional	String	Prefix length of the host IP address to be used after the operating system is deployed. This is used when the network setting is set to static IPv6 in the Global OS deployment settings.

Parameters	Required / Optional	Type	Description
selectedMAC	Required	String	MAC address of the host server to which the IP address is to be bound The MAC address is set to AUTO by default. This setting automatically detects the Ethernet ports that can be configured and used for deployment. The first MAC address (port) that is detected is used by default. If connectivity is detected on a different MAC address, the XClarity Administrator host is automatically restarted to use the newly detected MAC address for deployment, and selectedMAC is set to the newly detected MAC address. VLAN mode is supported only for servers that have MAC addresses in their inventory. If AUTO is the only the MAC address that is available for a server, then VLANs cannot be used to deploy operating systems to that server. **Tip:** To obtain the MAC address, use the **macaddress value** field in osimages hostplatforms.
subnetMask	Required if using static IP addresses. Optional if using DHCP.	String	Subnet mask of the host server to be used after the operating system is deployed. This is used when the network setting is set to static in the Global OS deployment settings. **Tip:** To determine the IP mode, use osimages globalsettings.
vlanId	Optional	String	VLAN ID for operating-system VLAN tagging This parameter is valid only if in VLAN mode is enabled (see osimages globalsettings). **Important:** Only specify a VLAN ID when a VLAN tag is required to function on the network. Using VLAN tags can affect the network routability between the host operating system and the Lenovo XClarity Administrator.
selectedImage	Required	String	Profile ID of the operating-system image to be deployed **Tip:** To obtain the operating-system image values, use the **availableImages value** field in osimages hostplatforms command.
storageSettings	Required	Object	Preferred storage location on which you want to deploy operating-system images

Parameters	Required / Optional	Type	Description
targetDevice	Required	String	Target device. This can be one of the following values. • **localdisk**. Local disk drive. The first enumerated local disk drive in the managed server is used. • **M.2drive**. M.2 drive. The first enumerated M.2 drive in the managed server is used. • **usbdisk**. Embedded USB Hypervisor. This location is applicable only when a VMware ESXi image is being deployed to managed servers. If two hypervisor keys are installed on the managed server, the VMware installer selects the first enumerated key for deployment. • **lunpluswwn=***LUN@WWN*. FC SAN storage (for example, `lunpluswwn=2@50:05:07:68:05:0c:09:bb`). • **lunplusiqn=***LUN@IQN*. iSCSI SAN Storage (for example, `lunplusiqn=0@iqn.1990-01.com.lenovo:tgt1`). Specifying the *IQN* is optional if only one iSCSI target is configured If the *IQN* is not specified, the first detected iSCSI target is selected for OSDN. If specified, and exact match is made. **Note:** For ThinkServer servers, this value is always “localdisk.”
unattendFileId	Required if **selected-Image** includes custom unattend files; otherwise ignored.	String	ID of the unattend file to use for this OS deployment
uuid	Required	String	UUID of the host server to which the operating system is to be deployed

Parameters	Required / Optional	Type	Description
windowsDomain	Optional	String	(Windows only) Active Directory domain that the Windows operating system is to join after the operating system is deployed successfully. If an OU is present, specify the string using the format *domain*/*ou*. If the operating system will not join a domain, you can set this field to null. **Notes:** • To join an Active Directory domain, you must specify either the **windowsDomain** or **windowsDomainBlob** parameter. If both are specified, the **windowsDomainBlob** parameter is used. • Use the **adusername** and **adpassword** parameters to specify the user name and password for the domain.
windowsDomainBlob	Optional	String	(Windows only) Active Directory Computer Account Metadata (in Base-64 encoded blob format) that can be used to join the Active Directory domain. For instructions for generating a file that contains the metadata blob data, see Integrating with Windows Active Directory in the Lenovo XClarity Administrator online documentation. **Notes:** • To join an Active Directory domain, you must specify either the **windowsDomain** or **windowsDomainBlob** parameter. If both are specified, the **windowsDomainBlob** parameter is used. • You can use metadata blob data when deploying any Windows operating system. However, this method must be used for Windows Nano Server. Specifying a domain in global settings is not supported for Windows Nano Server.

For example,

```
{
    "networkSettings": {
        "dns1": "192.0.2.255",
        "dns2": "192.0.2.254",
        "gateway": "192.0.2.200",
        "ipAddress": "192.0.2.0",
         "mtu": 1500,
        "prefixLength": 64,
      "selectedMAC": "78:9A:BC:12:34:56",
        "subnetMask": "255.255.255.0"
    },
    "selectedImage": "rhels6.4-x86_64-install-Minimal",
    "storageSettings": {
        "targetDevice": "lunpluswwn=2@50:05:07:68:05:0c:09:bb"
    },
    "uuid": "2D16B4422AC011E38A06000AF72567B0",
    "windowsDomain": null
```

```
}
```

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns a list of OS images in the repository.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
osimages hostplatforms
```

This example deploys an OS image to a specific managed server.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
osimages hostplatforms -o '{"networkSettings": {"dns1": "10.240.0.10", "dns2": "10.240.0.11",
          "gateway": "10.240.28.1", "ipAddress": "10.240.29.226", "mtu": 1500,
          "prefixLength": 64, "selectedMAC": "AUTO", "subnetMask": "255.255.252.0",
          "vlanId": 521}, "selectedImage": "rhels7.3|rhels7.3-x86_64-install-Minimal",
          "storageSettings": {"targetDevice": "localdisk"},
          "uuid": "B918EDCA1B5F11E2803EBECB82710ADE"}'
```

Related links

- connect
- osimages list
- osimages globalsettings
- osimages hostsettings

osimages import

This command imports OS images and custom files from a remote server to the repository.

To import a new file, follow these steps:

1. Start a job to import the file using osimages import -t *<image_type>*.
2. Import the file using osimages import -t *<image_type>* -o *<settings_JSON>*, where the settings JSON includes the job ID that was returned in step 1.
3. Monitor the status of the import job using tasks –j *<job_id>*.

When you import an OS image, Lenovo XClarity Administrator creates one or more OS-image profiles in the OS image repository. The profile includes both the OS image and the installation options for that image.

Syntax

```
osimages import  -h

osimages import -t <image_type> [-v <filter>]

osimages import -t <image_type> -o <settings_JSON> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-o | --osimages_dict} *<settings_JSON>*

Specifies the OS image settings, in JSON format.

If not specified, this command create a job to perform the import operation.

Parameters	Required / Optional	Type	Description
jobID	Required	Integer	ID of the job that was created to import files using the osimages import -t *<image_type>* command.
checksumType	Optional	Integer	Specify the type of checksum to be used. This can be one of the following values. • MD5 • SHA1 • SHA256 It is added as an item of the multi-part body.
checksumValue	Optional	Integer	Checksum string of the .ISO file. It is added as an item of the multi-part body.
imageName	Required	Integer	Name of the OS image to which you want to add the device driver (for example, redhat7.0) **Note:** The operating-system image must exist in the OS images repository.
imageType	Required	Integer	Type of image being imported. This can be one of the following values. • **BOOT**. Boot-options file. This is available for only customized Windows operating-system profiles. • **BUNDLE**. Bundle file (in .zip format). This is available for only customized Windows operating-system profiles. • **BUNDLESIG**. Bundle signature files (in .asc format). This is available for only customized Windows operating-system profiles. • **CONFIG**. Configuration-settings file (in JSON format) • **DUD**. Device driver. This is available for customized Windows and Linux operating-system profiles. • **OS**. (default) OS image • **OSPROFILE**. Customized OS image profile • **SCRIPT**. Installation-script file • **SOFTWARE**. Archive file (in .zip or .tar.gz format) that encapsulates the post-install software payload • **UNATTEND**. Unattend file (in kickstart .cfg, autoyast .xml, or Windows .xml format)

Parameters	Required / Optional	Type	Description
os	Required when **--imagetype** is “BOOT,” “CONFIG,”“-DUD,” “SCRIPT,” “SOFTWARE,” or “UNATTEND”	Integer	Operating system that is associated with the uploaded file. This can be one of the following values. • **esxi** • **rhels** • **sles** • **win**
path	Optional	Integer	Full path to the OS image on the remote file server **Note:** This field is applicable only when **serverId** is specified.
serverId	Optional	Integer	Profile ID for the remote file server To obtain the profile ID, use the osimages remotefileservers method.

For example,

```
{
   "description": "rhel 76 iso",
   "imageName": "RHEL76",
   "file": "/home/naval/osimage_test/RHEL-7.6-20181010.0-Server-x86_64-dvd1.iso",
   "jobId": 87,
   "os": "rhels",
   "osrelease": "7.6"
}
```

{-t | --imagetype} <*image_type*>

Imports an image of the specified type. This can be one of the following values.

- **BOOT**. Boot-options file. This is available for only customized Windows operating-system profiles.
- **BUNDLE**. Bundle file (in .zip format). This is available for only customized Windows operating-system profiles.
- **BUNDLESIG**. Bundle signature files (in .asc format). This is available for only customized Windows operating-system profiles.
- **CONFIG**. Configuration-settings file (in JSON format)
- **DUD**. Device driver. This is available for customized Windows and Linux operating-system profiles.
- **OS**. (default) OS image
- **OSPROFILE**. Customized OS image profile
- **SCRIPT**. Installation-script file
- **SOFTWARE**. Archive file (in .zip or .tar.gz format) that encapsulates the post-install software payload
- **UNATTEND**. Unattend file (in kickstart .cfg, autoyast .xml, or Windows .xml format)

Note: Unattend files and custom configuration-schema files are specific to a custom OS-image profile and are added and modified using osimages hostplatforms

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example creates a job to import an OS image and then imports the image from the remote server using the job ID.

```
connect --url https://192.0.2.0 --user ADMIN -noverify
osimages import -t OS -v import_job
osimages import -v result -t OS -o '{"imageName": "fixed", "jobId": 26, "os": "rhels",
          "path": "iso/rhel73.iso", "serverId": "1"}'
```

The following example creates a job to import an OS image and then imports the image from the remote server using the job ID.

```
connect --url https://192.0.2.0 --user ADMIN -noverify
osimages import -t OS -v import_job
osimages import -v result -t OS -o ' {"description": "rhel 76 iso", "imageName": "RHEL76",
          "file": "/home/naval/osimage_test/RHEL-7.6-20181010.0-Server-x86_64-dvd1.iso",
          "jobId": 87, "os": "rhels", "osrelease": "7.6"} '
```

The following example creates a job to import a custom script and then imports the custom script from the remote server using the job ID.

```
connect --url https://192.0.2.0 --user ADMIN -noverify
osimages import -t SCRIPT -v import_job
osimages import -v result -t OS -o '{"imageName": "Test", "file": "/home/naval/test.py",
          "jobId": 26, "os": "rhels"}'
```

Related links

- connect
- osimages remotefileservers
- osimages hostplatforms

osimages delete

This command deletes the specified OS images and custom files from the repository.

Syntax

```
osimages delete  -h

osimages delete  -i <file_id_list> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-i | --id} *<file_id_list>*
Specifies the UIDs of one or more OS images and custom files, separated by a comma

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example deletes a single custom file from the repository.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
osimages delete -i 20190131054310_trial.py
```

Related links

- connect
- configprofiles list
- configprofiles unassign

osimages remotefileservers

This command retrieves information about all remote file-server profiles, and creates and modifies a remote file-server profile.

Syntax

```
osimages remotefileservers  -h

osimages remotefileservers [-o <settings_JSON>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-o | --osimages_dict} *<settings_JSON>*

Specifies the remote file-server profile settings, in JSON format.

If this parameter is not specified, information about all remote-file-server profiles is returned.

Table 4. Return information about a specific profile

Parameters	Required / Optional	Type	Description
id	Required	String	ID of the remote-file-server profile

Table 5. Delete a profile

Parameters	Required / Optional	Type	Description
deleteid	Required	String	ID of the remote-file-server profile

Table 6. Create or modify a profile

Parameters	Required / Optional	Type	Description
putid	Optional	String	ID of the remote-file-server profile If specified, the remote-file-server profile is modified. If not specified, a remote-file-server profile is created.
address	Required	String	IP address of the remote file server
displayName	Required	String	User-defined name of the remote file server
keyComment	Optional	String	Key comment
keyPassphras	Optional	String	Key passphrase
keyType	Optional	String	Type of encryption algorithm. This can be one of the following values. • **RSA-2048** • **RSA-4096** • **ECDSA-521-secp521r1**
password	Optional	String	Password to connect to the remote file server
port	Required	Integer	Port number
protocol	Required	String	Server protocol. This can be one of the following values. • **HTTP** • **HTTPS** • **FTP** • **SFTP**
serverId	Optional	String	Profile ID for the remote file server If specified, the profile is modified. If not specified, a new profile is created.
username	Optional	String	User name to connect to the remote file server

For example,

```
{
   "address" : "192.168.1.10",
   "password" : "PasswOrd",
   "port" : 8081,
   "protocol" : "HTTPS",
   "username" : "admin"
}
```

{-v | --view} *<filter>*

Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns information about all remote-file-server profiles.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
```

```
osimages remotefileservers -v remotefileserver
```

The following example returns information about a specific remote-file-server profile.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
osimages remotefileservers -o '{"id": "1"}' -v remotefileservers
```

The following example creates a remote file server profile for an FTP server.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
oosimages remotefileservers -v remotefileservers -o '{"address": "10.211.2.243",
          "displayName": "new_ftp_207", "port": 21, "password": "password", "protocol": "FTP",
          "username": "guest"}'
```

The following example modifies a remote-file-server profile.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
oosimages remotefileservers -v result -o '{"putid": "1", "address": "10.211.2.243",
          "displayName": "new_ftp_207", "port": 21, "protocol": "FTP"}'
```

The following example deletes a remote-file-server profile.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
osimages remotefileservers -o '{"deleteid": "1"}' -v result
```

Related links

- connect
- osimages import

Resource group commands

The following PyLXCA command is available for performing resource-group functions.

resourcegroups

This command retrieves information about and manage resource groups.

Syntax

```
resourcegroups  -h

resourcegroups  -h {list | criteriaproperties | create | update| delete}
                <action_specific_parmeters>
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{list} <*action_specific_parmeters*>
Retrieves information about all resource groups or a specific group. For more information, see resourcegroups list.

{criteriaproperties} <*action_specific_parmeters*>
Retrieves inventory properties that you can use to specify criteria for dynamic resource groups. For more information, see resourcegroups criteriaproperties.

{create} <*action_specific_parmeters*>
Creates a dynamic resource group. For more information, see resourcegroups create.

{update} *<action_specific_parmeters>*
Modifies a dynamic resource group. For more information, see resourcegroups update.

{delete} *<action_specific_parmeters>*
Deletes a resource group. For more information, see resourcegroups delete.

resourcegroups list

This command retrieves information about all resource groups or a specific group.

Syntax

```
resourcegroups list  -h

resourcegroups list [-u <UUID>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-u | --uuid}
Specifies the UUID of the resource group.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns information about all resource groups.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
resourcegroups list
```

The following example returns information about a specific resource group.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
resourcegroups list -u 5C5AB42D94C6A719BEF2A375
```

Related links

- connect
- resourcegroups criteriaproperties

resourcegroups criteriaproperties

This command retrieves information about all resource groups.

Syntax

```
resourcegroups criteriaproperties  -h

resourcegroups criteriaproperties [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example retrieves inventory properties of resource groups.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
resourcegroups criteriaproperties -v criteriaproperties
```

Related links

- connect
- resourcegroups list

resourcegroups create

This command creates a dynamic resource group.

Syntax

```
resourcegroups create  -h

resourcegroups create -n <group_name> [-d <description>] -c <criteria>
                      -t <group_type> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-n | --name}
Specifies the name of the resource group.

{-d | --description}
Specifies the description of the resource group.

{-c | --criteria}
Specifies the criteria (rules) of the resource group, in JSON format.

Parameters	Required / Optional	Type	Description

criteria	Required if **type** is "dynamic"	Object	(Dynamic groups only) Information about a simple criteria object or criteria set that select which managed devices are members of the dynamic group
			Simple criteria is a query (logical rule) that compares property values. The following example selects managed devices whose contact is John@company.com.
			`{`
			`"property": "contact",`
			`"operator": "equals",`
			`"value": "John@company.com",`
			`}`
			
			A *criteria set* is the root of the tree structure that defines how the simple criteria are logically combined, using Boolean AND and OR relationships. The following example shows a criteria set that logically combines two simple criteria with an AND relationship. It selects managed devices whose contact is John@company.com and are in the Critical state.
			`{`
			`"operator": "AND",`
			`"criteria": [{`
			`"property": "contact",`
			`"operator": "equals",`
			`"value": "John@company.com"`
			`},`
			`{`
			`"property": "overallHealthState",`
			`"operator": "equals",`
			`"value": "Critical"`
			`}]`
			`}`

criteria	Required only for criteria sets	Array of objects	Nested criteria that defines the members of the dynamic group. Array elements can be a combination of simple criteria or criteria set objects.
id	Required	String	ID of the simple criteria or criteria set object
operator	Required	String	Operator For criteria, you can obtain a list of valid operator values for each property using resourcegroups criteriaproperties. For criteria sets, this can be one of the following values: • **AND**. Members must satisfy all specified values. • **OR**. Members must satisfy one or more of the specified values.
parent	Required	String	ID of the parent criteria set. This is "root" when the criteria or criteria set is not nested.
property	Required only for simple criteria	String	Inventory property. To obtain a list of properties, use resourcegroups criteriaproperties.

Parameters	Required / Optional	Type	Description
value	Required only for simple criteria	String	Value of the property
id	Required	String	ID of the simple criteria or criteria set object
operator	Required	String	Operator For criteria, you can obtain a list of valid operator values for each property using resourcegroups criteriaproperties. For criteria sets, this can be one of the following values: • **AND**. Members must satisfy all specified values. • **OR**. Members must satisfy one or more of the specified values.
parent	Required	String	ID of the parent criteria set. This is “root” when the criteria or criteria set is not nested.
property	Required only for simple criteria	String	Inventory property. To obtain a list of properties, use resourcegroups criteriaproperties.
value	Required only for simple criteria	String	Value of the property

{-t | --type}
Specifies the type of the resource group. This value is always **dynamic**.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example creates a resource group.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
resourcegroups create -n R1_GROUP -d GROUP_DESCRIPTION -t dynamic
                -c '{"parent": "root", "value": null, "criteria": [{"operator": "contains",
                "property": "hostname", "id": "1001", "value": "test",
                "parent":  lxca_customUI_resourceViews_allGroupsPage_editGroupDynamicPage_2"}],
                "operator": "AND", "property": null, "id": "root"}'
```

Related links

- connect
- resourcegroups list

resourcegroups update

This command modifies a dynamic resource group.

Syntax

```
resourcegroups update  -h

resourcegroups update -n <group_name> [-d <description>] -c <criteria>
                      -t <group_type> [-u <UUID>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-n | --name}
Specifies the name of the resource group.

{-d | --description}
Specifies the description of the resource group.

{-c | --criteria}
Specifies the criteria (rules) of the resource group, in JSON format.

Parameters	Required / Optional	Type	Description

criteria	Required if **type** is "dynamic"	Object	(Dynamic groups only) Information about a simple criteria object or criteria set that select which managed devices are members of the dynamic group
			Simple criteria is a query (logical rule) that compares property values. The following example selects managed devices whose contact is John@company.com.
			`{`
			`"property": "contact",`
			`"operator": "equals",`
			`"value": "John@company.com",`
			`}`
			
			A *criteria set* is the root of the tree structure that defines how the simple criteria are logically combined, using Boolean AND and OR relationships. The following example shows a criteria set that logically combines two simple criteria with an AND relationship. It selects managed devices whose contact is John@company.com and are in the Critical state.
			`{`
			`"operator": "AND",`
			`"criteria": [{`
			`"property": "contact",`
			`"operator": "equals",`
			`"value": "John@company.com"`
			`},`
			`{`
			`"property": "overallHealthState",`
			`"operator": "equals",`
			`"value": "Critical"`
			`}]`
			`}`

criteria	Required only for criteria sets	Array of objects	Nested criteria that defines the members of the dynamic group. Array elements can be a combination of simple criteria or criteria set objects.

Parameters	Re-quired / Optional	Type	Description
id	Re-quired	String	ID of the simple criteria or criteria set object
operator	Re-quired	String	Operator For criteria, you can obtain a list of valid operator values for each property using resourcegroups criteriaproperties. For criteria sets, this can be one of the following values: • **AND**. Members must satisfy all specified values. • **OR**. Members must satisfy one or more of the specified values.
parent	Re-quired	String	ID of the parent criteria set. This is “root” when the criteria or criteria set is not nested.
property	Required only for simple criteria	String	Inventory property. To obtain a list of properties, use resourcegroups criteriaproperties.
value	Required only for simple criteria	String	Value of the property
id	Re-quired	String	ID of the simple criteria or criteria set object
operator	Re-quired	String	Operator For criteria, you can obtain a list of valid operator values for each property using resourcegroups criteriaproperties. For criteria sets, this can be one of the following values: • **AND**. Members must satisfy all specified values. • **OR**. Members must satisfy one or more of the specified values.
parent	Re-quired	String	ID of the parent criteria set. This is “root” when the criteria or criteria set is not nested.
property	Required only for simple criteria	String	Inventory property. To obtain a list of properties, use resourcegroups criteriaproperties.
value	Required only for simple criteria	String	Value of the property

{-t | --type}
Specifies the type of the resource group. This value is always **dynamic**.

{-u | --uuid}
Specifies the UUID of the resource group to be modified.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns information about all resource groups.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
resourcegroups update
```

The following example returns information about a specific resource group.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
resourcegroups update -u 5C5AB42D94C6A719BEF2A375 -n R1_GROUP -d GROUP_DESCRIPTION -t dynamic
               -c '{"parent": "root", "value": null, "criteria": [{"operator": "contains",
               "property": "hostname", "id": "1001", "value": "test"}], "operator": "AND",
               "property": null, "id": "root",
               "parent": "lxca_customUI_resourceViews_allGroupsPage_editGroupDynamicPage_2"}'
```

Related links

- connect
- resourcegroups create
- resourcegroups list
- resourcegroups criteriaproperties
- resourcegroups delete

resourcegroups delete

This command deletes a resource group.

Syntax

```
resourcegroups delete  -h

resourcegroups delete -u <UUID> [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-u | --uuid}
Specifies the UUID of the resource group to delete.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns deletes a resource group.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
resourcegroups delete –u 5C5AB42D94C6A719BEF2A375
```

Related links

- connect
- resourcegroups list

Security commands

The following PyLXCA commands are available for performing security functions.

storedcredentials

This command creates, modifies and deletes stored credentials in Lenovo XClarity Administrator.

Syntax

```
storedcredentials -h

storedcredentials [-v <filter>]

storedcredentials  -u <user_name> -p <password> [-d <description>]

storedcredentials  -i <stored_credential_ID> [-u <user_name>] [-p <password>]
                   [-d <description>]

storedcredentials  --delete_id <stored_credential_ID>
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

--delete_id <*stored_credential_ID*>
Deletes the specified stored-credential account.

{-d | --description} <*description*>
Specifies the description of the stored-credential account

{-i | --id} <*stored_credential_ID*>
Specifies the ID of the stored-credential account to be managed. If specified, the stored credential is updated. If not specified, the stored credential is created.

{-p | --password} <*password*>
Specifies the password of the stored-credential account.

{-u | --user_name} <*user_name*>
Specifies a user name of the stored-credential account.

{-v | --view} <*filter*>
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example returns information about all stored credentials.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
storedcredentials
```

The following example creates a new stored credential.

```
connect --url https://192.0.2.0 --user ADMIN -noverify
storedcredentials -u USERID -p xxxxxxxx
```

The following example changes the user name and password for a stored credential.

```
connect --url https://192.0.2.0 --user ADMIN -noverify
storedcredentials –i AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA -u USERID2 -p xxxxxxxx
```

The following example deletes a stored credential.

```
connect --url https://192.0.2.0 --user ADMIN -noverify
storedcredentials --delete_id AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Related links

- manage

users

This command retrieves information about all local Lenovo XClarity Administrator users or a specific local user.

Syntax

```
users  -h

users [-i <user_ID>] [-v <filter>]
```

Options

{-h | --help}
Displays the syntax and brief usage information for this command.

{-i | --id} *<user_ID>*
Specifies the unique ID of the local user. If a user ID is not specified, information about all local users is retrieved.

{-v | --view} *<filter>*
Identifies the view to use for the returned data. If a filter is not specified, the default view is used.

You can also create custom views (see Creating custom views).

Examples

The following example retrieves information about all local connect users.

```
connect --url https://192.0.2.01 --user ADMIN --noverify
users
```

Related links

- connect

Service and support commands

The following PyLXCA commands are available for performing service and support functions.

ffdc

This command collects and exports first failure data capture (FFDC) data for a specific managed device (CMM, server, storage device, or switch).

Syntax

```
ffdc  -h

ffdc  -u <device_UUID>
```

Options

{-h | --help}

Displays the syntax and brief usage information for this command.

{-u | --uuid} *<device_UUID>*
Specifies the UUID of a device.

Examples

The following example returns FFDC data for a specific device.

```
connect --url https://192.0.2.0 --user ADMIN --noverify
ffdc -u AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Related links

- connect

Appendix A. Filtering events

You can use the parameter **filterWith** to return a subset of all active events based on Java REGEX expressions or based on comparison operators.

You can choose to filter using one of the following methods:

- Java REGEX expressions
- Comparison operators. The following comparison operators are provided:
  - EQ (equal)
  - NOT (not equal)
  - GT (greater than)
  - GTE (greater than or equal to)
  - LT (less than)
  - LTE (less than or equal to).

Note: You cannot combine Java REGEX expressions with comparison operators.

Comparison operators

Some fields support only specific comparison operators.

Parameter	EQ	GT	GTE	LT	LTE	NOT
action	√	√	√	√	√	√
ARGS	√					√
cn	√	√	√	√	√	√
componentID	√	√	√	√	√	√
eventClass	√	√	√	√	√	√
eventDate	√	√	√	√	√	√
eventID	√					√
FAILFRUS	√					√
FAILSNS	√					√
groupUUID	√					√
localLogID	√	√	√	√	√	√
localLogSequence	√	√	√	√	√	√
location	√					√
msgID	√					√
mtm	√					√
search	√	√	√	√	√	√
sequenceid	√	√	√	√	√	√
serialnum	√					√
service	√	√	√	√	√	√

Parameter	EQ	GT	GTE	LT	LTE	NOT
severity	√	√	√	√	√	√
sourceID	√	√	√	√	√	√
sourceLogID						
sourceLogSequence	√	√	√	√	√	√
timeStamp	√	√	√	√	√	√
USERID	√					√

Filtering examples

Filtering is passed as part of the URI parameters. The filter itself is in JSON format. All filters follow the following parameter format.

Obtaining all events that have a cn (sequence ID) greater than 1:

```
https://<Server IP Address>/events?filterWith={"filterType":"FIELDNOTREGEXAND",
"fields":[{"operation":"GT","field":"cn","value":"1"}]}
```

Events can be filtered based on the following fields:

Parameter	Comparison operators example	Regex expression example

action	{
	"operation":"EQ",
	"field":"action",
	"value":"ABCDE"
	}
		{
		"field":"action",
		"value":"ABCDE"
		}

cn	{
	"operation":"EQ",
	"field":"cn",
	"value":"1"
	}
		{
		"field":"cn",
		"value":"1"
		}

componentID	{
	"operation":"EQ",
	"field":"componentID",
	"value":"FFFFF"
	}
		{
		"field":"componentID",
		"value":"FFFFF"
		}

eventClass	{
	"operation":"EQ",
	"field":"eventClass",
	"value":"200"
	}
	or
	{
	"operation":"EQ",
	"field":"eventClass",
	"value":"AUDIT"
	}
		{
		"field":"eventClass",
		"value":"200"
		}
		or
		{
		"field":"eventClass",
		"value":"AUDIT"
		}

eventDate	{
	"operation":"EQ",
	"field":"eventDate",
	"value":"2014-02-11T09:54:58Z"
	}
		{
		"field":"eventDate",
		"value":"2014-02-11T09:54:58Z"
		}

Parameter	Comparison operators example	Regex expression example

eventID	{
	"operation":"EQ",
	"field":"eventID",
	"value":"FQXHMCP5810I"
	}
		{
		"field":"eventID",
		"value":"FQXHMCP5810I"
		}

groupUUID	{
	"operation":"EQ",
	"field":"groupUUID",
	"value": [
	"FFB657408BEB4161950704AB",
	"59AFBFCF8DBB376A25D68A0A"]
	}
		{
		"field":"groupUUID",
		"value": [
		"FFB657408BEB4161950704AB",
		"59AFBFCF8DBB376A25D68A0A"]
		}

localLogID	{
	"operation":"EQ",
	"field":"localLogID",
	"value":"ABCDE"
	}
		{
		"field":"localLogID",
		"value":"ABCDE"
		}

localLogSequence	{
	"operation":"EQ",
	"field":"localLogSequence",
	"value":"1"
	}
		{
		"field":"localLogSequence",
		"value":"1"
		}

location	{
	"operation":"EQ",
	"field":"location",
	"value":"ABCDE"
	}
		{
		"field":"location",
		"value":"ABCDE"
		}

msgID	{
	"operation":"EQ",
	"field":"msgID",
	"value":"ABCDE"
	}
		{
		"field":"msgID",
		"value":"ABCDE"
		}

mtm	{
	"operation":"EQ",
	"field":"mtm",
	"value":"ABCDE"
	}
		{
		"field":"mtm",
		"value":"ABCDE"
		}

search	{
	"operation":"EQ",
	"field":"search",
	"value":"ABCDE"
	}
		{
		"field":"search",
		"value":"ABCDE"
		}

sequenceid	{
	"operation":"EQ",
	"field":"sequenceid",
	"value":"1"
	}
		{
		"field":"sequenceid",
		"value":"1"
		}

serialnum	{
	"operation":"EQ",
	"field":"serialnum",
	"value":"ABCDE"
	}
		{
		"field":"serialnum",
		"value":"ABCDE"
		}

Parameter	Comparison operators example	Regex expression example

service	{
	"operation":"EQ",
	"field":"service",
	"value":"100"
	}
	or
	{
	"operation":"EQ",
	"field":"service",
	"value":"NONE"
	}
		{
		"field":"service",
		"value":"100"
		}
		or
		{
		"field":"service",
		"value":"NONE"
		}

severity	{
	"operation":"EQ",
	"field":"severity",
	"value":"200"
	}
	or
	{
	"operation":"EQ",
	"field":"severity",
	"value":"INFORMATIONAL"
	}
		{
		"field":"severity",
		"value":"200"
		}
		or
		{
		"field":"severity",
		"value":"INFORMATIONAL"
		}

sourceID	{
	"operation":"EQ",
	"field":"sourceID",
	"value":"ABCDE"
	}
		{
		"field":"sourceID",
		"value":"ABCDE"
		}

sourceLogID	{
	"operation":"EQ",
	"field":"sourceLogID",
	"value":"ABCDE"
	}
		{
		"field":"sourceLogID",
		"value":"ABCDE"
		}

sourceLogSe-quence	{
	"operation":"EQ",
	"field":"sourceLogSequence",
	"value":"1234"
	}
		{
		"field":"sourceLogSequence",
		"value":"1234"
		}

timeStamp	{
	"operation":"EQ",
	"field":"timeStamp",
	"value":"2014-02-11T09:54:58Z"
	}
		{
		"field":"timeStamp",
		"value":"2014-02-11T09:54:58Z"
		}

Applying a filter to match a single event with a sequence ID equal to 16:

```
{
   "filterType":"FIELDNOTREGEXAND",
   "fields": [{
      "operation":"EQ",
      "field":"cn",
      "value":"16"
   }]
}

{
   "filterType":"FIELDREGEXAND",
   "fields": [{
```

```
      "field":"cn",
      "value":"16"
   }]
}
```

These two filters are equivalent; they both will match with a single event, the event that has the cn/sequenceid equal to 16.

The filtering is composed of two parts:

1. The first part is the filterType that can have only one value from the following enumeration:
   - **FIELDREGEXAND**. Regex filter of type AND
   - **FIELDREGEXOR**.Regex filter of type OR
   - **FIELDREGEXNOT**. Regex filter of type NOT
   - **FIELDNOTREGEXAND**. Non-Regex filter of type AND
   - **FIELDNOTREGEXOR**. Non-Regex filter of type OR
   - **FIELDNOTREGEXNOT**. Non-Regex filter of type NOT

   The REGEX filters accept only REGEX expressions in the "value" field. The Non-REGEX filters do not accept REGEX expressions in the "value" field. The Non-REGEX filter works with the six comparison operators (EQ, NOT, GT, GTE, LT, LTE). It also has a special field called "operation" in which to specify the comparison operation.

   The "filterType" is a mandatory field.
2. The second part is an enumeration of "fields" that define the target of the filter match. This field is required.The "fields" is a JSONArray Enumeration composed of JSON Objects. In the above example it can be seen that there is only one JSON ({"operation":"EQ" , "field":"cn" , "value":"16"}) in the entire JSONArray ([{"operation":"EQ" , "field":"cn" , "value":"16"}]).

Applying a complex filter:

```
{
   "filterType":"FIELDNOTREGEXAND",
   "fields": [
      {"operation":"GT", "field":"cn", "value":"16"},
      {"operation":"GTE", "field":"severity", "value":"400"},
      {"operation":"GTE", "field":"timeStamp", "value":"2014-02-11T09:20:35Z"}
   ]
}
```

This filter will match all events that have the cn/sequenceid greater than 16, a severity greater than or equal to 400, and a timeStamp greater than or equal to 9:20:35 Zulu - February 11, 2014.