Enabling Python Scripting on Windows to Manage XClarity Controller

Planning / Implementation

Python is a high-level, general-purpose programming language designed to be readable, flexible, and efficient. In the context of data centers, Python becomes a powerful automation engine. It can execute scripts that interact with servers, network devices, storage systems, and cloud platforms. Through libraries such as Ansible modules, and REST API clients, Python can remotely connect to servers, issue commands, transfer files, gather metrics, and orchestrate multi-step workflows. These features enables administrators to automate repetitive tasks, enforce configuration consistency, and manage thousands of machines with predictable, repeatable processes.

Python's strengths become even more pronounced when paired with Lenovo's data-center ecosystem, especially the Lenovo XClarity Controller (XCC) and Lenovo XClarity Administrator (LXCA). These platforms expose rich, well-structured REST APIs that Python can interact with seamlessly, turning routine server management into fully automated workflows. Lenovo offers a large library of scripts that can be used to interact with the XCC. The Lenovo scripts in the GitHub repository section at the end of this document has a complete list of the Lenovo Python scripts.

Trying to get all the items needed to use Python scripts under Windows is not always straightforward, especially if you are unfamiliar with the process. This guide will assist you with installing the correct tools for the first time.

This document is focused on Python in a Windows environment. The main site for where the scripts are located is the following:
https://github.com/lenovo/python-redfish-lenovo

> **Note**: Lenovo supports other methods to access the XCC, including Redfish, PowerShell, or a web browser, however we only cover Python in this paper.

The the following are the latest releases, as of January 9, 2026:

- Python for Windows 3.14.2 (64-bit)
- PIP 25.3
- Redfish 3.3.4
- Configparser 7.2.0
- PIP requests, various versions
- GitHub Desktop app 3.5.4 (64-bit)

By following this document you will perform the following tasks:

- Install and configure Python and the PIP applications in Windows
- Install Redfish, configparser, requests
- Install the Lenovo .py files to access the XCC
- Edit the default configuration file to talk to the XCC
- Explore communicating to the XCC

Throughout the document, there are commands listed for each step. You can copy and paste those in your command prompt shell.

Installing Python and libraries

This section describes how to install Python as well as the libraries that are needed to run the Lenovo scripts.

Install Python as follows:

1. Download Python from https://www.python.org/downloads/windows/

   > **Install from Microsoft Store**: If you prefer, you can install Python using the Python Install Manager from the Microsoft store.

2. Run the installer. The following figure shows the key installation dialog.

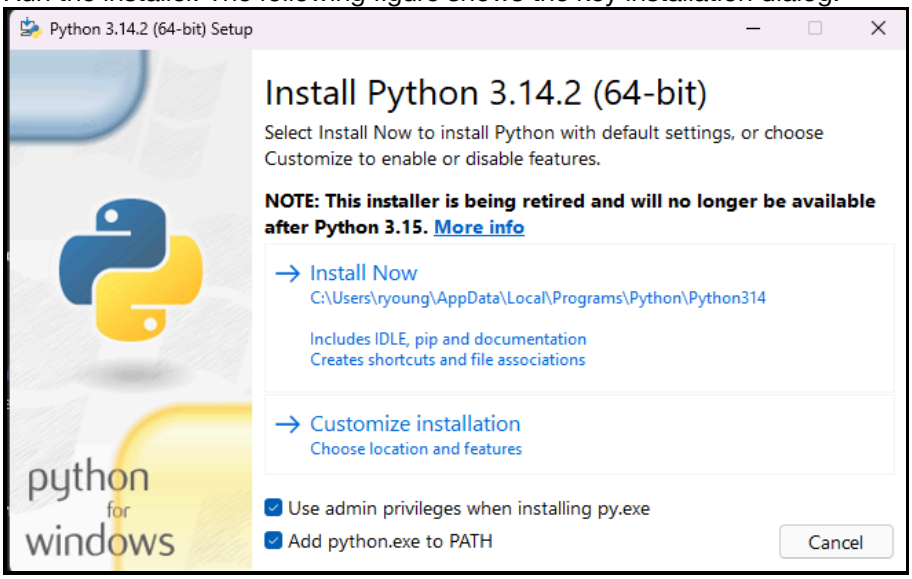

Figure 1. First Python install window

3. Follow instructions for specific version.
4. Select **Add python.exe to PATH** in the Setup wizard when promted
5. Depending on your environment, you might need to run with Administrator permissions by right-clicking on the installer and selecting **Run as Administrator**.
6. At the end of the installation, you'll see the following windows. Optionally click **Disable path length limit** if you want to run Python from a folder other than the installation directory.

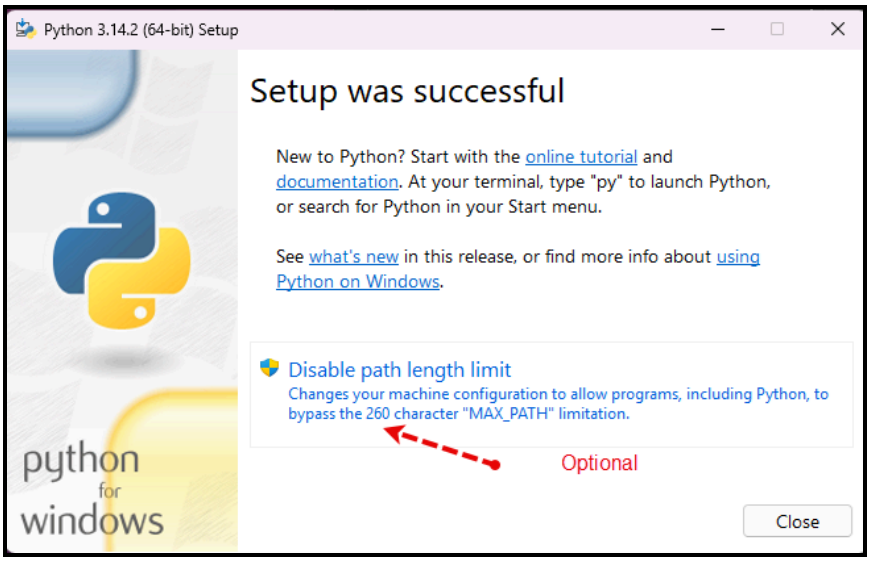

Figure 2. Install completion

7. Reboot the OS after the installation.

Verify that Python and PIP (Python package installer) were installed correctly:

1. Open a Command Prompt window.
2. Enter the command `python` without parameters. If installation was successful, you will see the following:

```
python
```

```
C:\Users\ryoung\AppData\Local\Programs\Python\Python314>python
Python 3.14.2 (tags/v3.14.2:df79316, Dec  5 2025, 17:18:21) [MSC v.1944 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 3. Issuing the python command to verify installation

If you see an error, or the Microsoft Store is launched, then this could be a path issue. To test, change to the directory that was used in the install and re-type the command. Also, verify you rebooted after the install.

3. Enter the command `py` without parameters:

```
py
```

```
C:\Users\ryoung>py
Python 3.14.2 (tags/v3.14.2:df79316, Dec  5 2025, 17:18:21) [MSC v.1944 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 4. Issuing the py command to verify installation

4. By default, the Python package installer PIP should be installed. Enter the following pip command:

```
pip --version
```

```
C:\Users\ryoung>pip --version
pip 25.3 from C:\Users\ryoung\AppData\Local\Programs\Python\Python314\Lib\site-packages\pip (python 3.14)
```

Figure 5. Check PIP version

If the pip command can't be found, verify that you rebooted after the Python install.

Install the three additional Python libraries that we will need:

1. Install the python redfish library:

```
pip install redfish
```

You will see several items being installed and should see `successfully installed` at the end.

2. Install configparser:

```
pip install configparser
```

The configparser library only supports python3 starting from configparser 5.0.0. If you are using python2.7, please specify version 4.0.2 while installing using the following command:

```
pip install configparser==4.0.2
```

A successful configparser install looks like the following:

```
C:\Users\ryoung>pip install configparser
Collecting configparser
  Downloading configparser-7.2.0-py3-none-any.whl.metadata (5.5 kB)
Downloading configparser-7.2.0-py3-none-any.whl (17 kB)
Installing collected packages: configparser
Successfully installed configparser-7.2.0
```

Figure 6. Install configparser

3. Install requests:

```
pip install requests
```

A successful installation looks like the following:

```
C:\Users\ryoung>pip install requests
Requirement already satisfied: requests in c:\users\ryoung\appdata\local\programs\python\python314\lib\site-packages (2.
32.5)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\users\ryoung\appdata\local\programs\python\python314\lib\s
ite-packages (from requests) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\ryoung\appdata\local\programs\python\python314\lib\site-packages
 (from requests) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\ryoung\appdata\local\programs\python\python314\lib\site-pa
ckages (from requests) (2.6.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ryoung\appdata\local\programs\python\python314\lib\site-pa
ckages (from requests) (2026.1.4)
```

Figure 7. Install requests

Installing Lenovo specific .py files

Now that Python and the associated libraries are installed, the next step is to install the custom Lenovo scripts.

The following steps will download and install the specific Lenovo related files to communicate with your servers. You can perform this task in one of three ways:

- Use the GitHub desktop application
- Use the GitHub web site
- Manually download the required items

We will use the GitHub desktop application, because it makes it easier to download the required files.

Following these steps to download the python-redfish-lenovo script repository from GitHub using the GitHub desktop app:

1. Download and install GitHub from: https://desktop.github.com/download/
2. You will be asked to sign into your GitHub account, create an account, or skip. Select which one is best for your situation. More details on these options can be found on the GitHub site under Help or Documentation.
3. Open the GitHub app and click **Clone a repository from the Internet** as shown below:

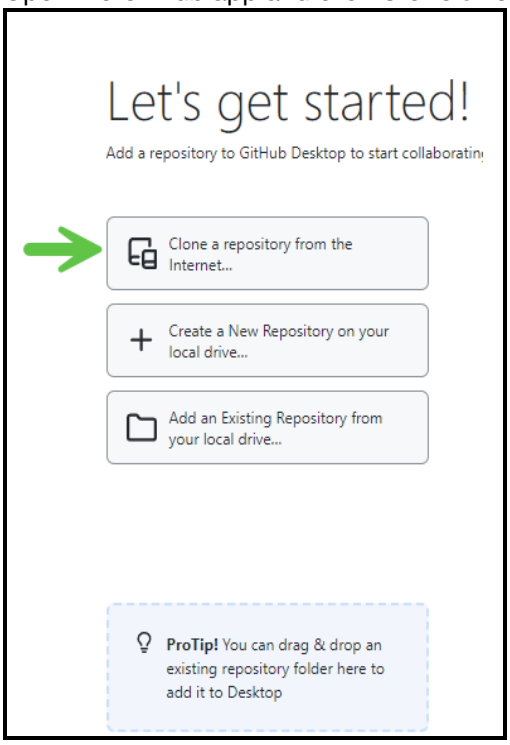

Figure 8. GitHub options

4. The following window will be displayed; Follow the steps as indicated in the figure below:
   a. Click **URL** to show that tab
   b. Enter this link under the Repository URL: `https://github.com/lenovo/python-redfish-lenovo`
   c. Change the Local Path if needed
   d. Click **Clone**

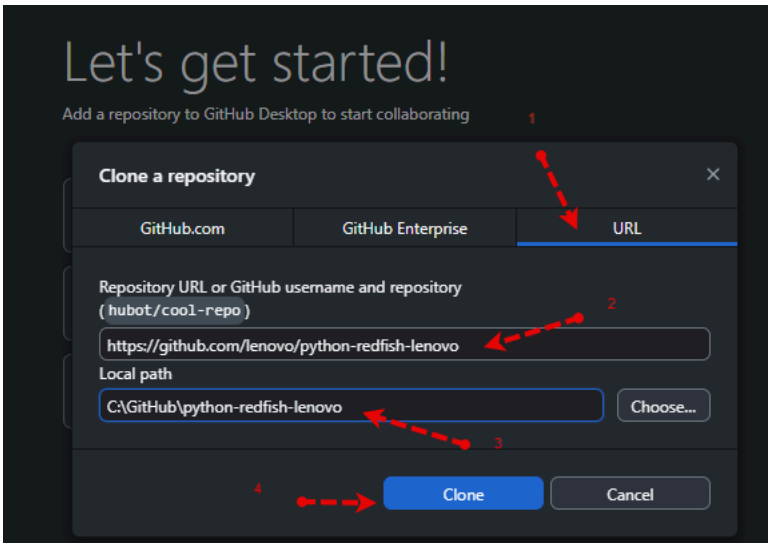

Figure 9. Repository details

5. GitHub will download the files and bring you back to the GitHub Desktop App screen. You can close the application.

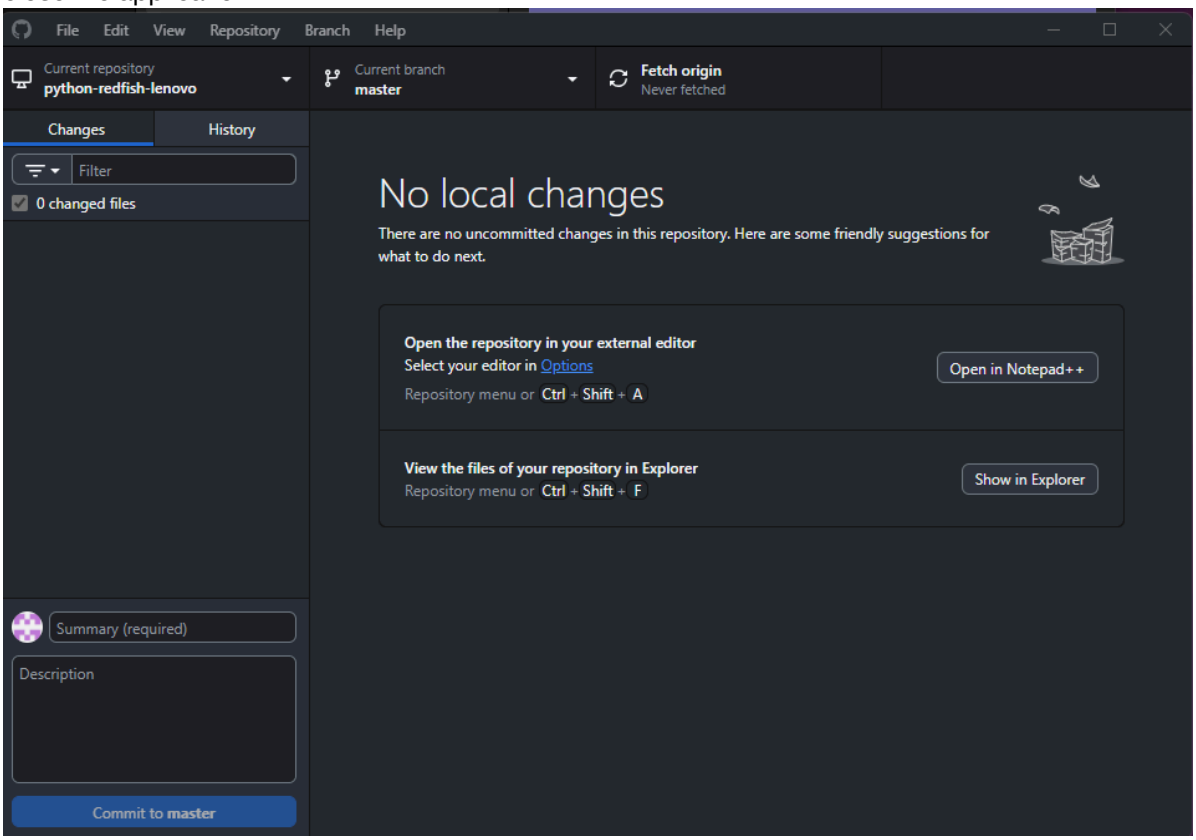

Figure 10. GitHub desktop completion

6. Reboot the OS

For a list of the Lenovo Python scripts included in the package, see the Appendix at the end of this document.

Modifying the config.ini file

To run the Python scripts against a single XCC, you will need to edit the config.ini file. By default, the file will be in the /examples folder where you previously downloaded the Python scripts. For more details review the “README.md” and “products_supported.txt” files located in the Lenovo GitHub folder.

There are 3 entries in the config.ini file that you will need to change to allow the script to communicate with the XCC. These will need to match the XCC you are trying to access.

- BmcIp - IP address of XCC
- BmcUsername - User name of the XCC user
- BmcUserpassword - Pasword of the XCC user

The following figure shows the default file.

Figure 11. default config file

The following figure shows an example of one that has been edited.

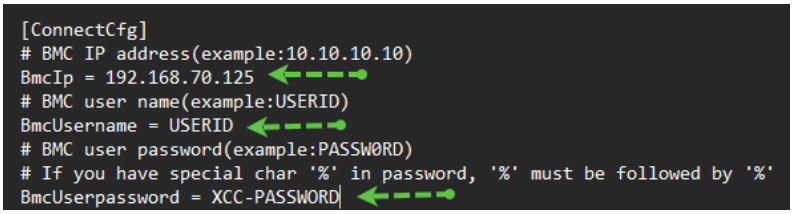

Figure 12. Example of edited config.ini file

There are additional items that can be altered in the file, including doing more with configuration files, the use of virtual media mounts, etc. This document doesn’t go into that level of detail; the three parameters above are the basic parameters needed to connect to a remote XCC.

Important security note: The config.ini file saves your XCC password in plain text. Ensure that the script is saved in a secure location and that you will be running it on a secure network

Running the Lenovo scripts

After completing the above steps, you are ready to run the various scripts. The Appendix: Lenovo scripts in the GitHub repository at the end of this document has a complete list of the Lenovo Python scripts.

This section shows a few examples and the output from each.

- Example 1. Gather all the BIOS attributes from the server
- Example 2. Gather data on the CPUs installed in the server
- Example 3. Gather data on firmware levels in the server
- Example 4. Gather and save the FFDC log file

Some commands such as get_all_bios_attributes.py return a lot of data; our screenshots only show a subset of the output.

If you want to save the output to a text file versus being displayed in the console window, add `>filename.txt` to the end of the command. You can use a filename that makes sense for you. For example:

```
py get_cpu_inventory.py >cpu.txt
```

Note: We use the `py` command to launch these scripts. You could also use the `python` command if desired.

To run the scripts, start by opening a Command Prompt window. Depending on your privileges, you may need to open the command prompte as an Administrator.

Example 1. Gather all the BIOS attributes from the server

This script will gather all the available options in uEFI and display the current status of each entery. This can be helpful to compare settings from 2 servers when you want to verify they are the same values.

```
py get_all_bios_attributes.py
```

Figure 13. Example of BIOS details

Example 2. Gather data on the CPUs installed in the server

This script will gather all the details around the processor installed in the server. In the example below, you can see it's an Intel Xeon Gold 6240M processor running at 2.60 GHz.

```
py get_cpu_inventory.py
```

Figure 14. Example of Processor details

Example 3. Gather data on firmware levels in the server

The following script will show you the various firmware levels since the last inventory was completed during the POST process. Using more advanced scripting, this can be used to check a server and then update the firmware if downlevel from your approved level(s). Note, this list can be large as it includes all known firmware for the hardware installed.

This list can be large as it includes all known firmware for the hardware installed.

```
py get_fw_inventory.py
```

```
C:\GitHub\python-redfish-lenovo\examples>py get_fw_inventory.py
[
  {
    "BMC-Primary": {
      "Description": "The information of BMC (Primary) firmware.",
      "SoftwareId": "BMC-CDI3-10",
      "Status": {
        "Health": "OK",
        "HealthRollup": "OK",
        "State": "Enabled"
      },
      "Version": "B8D-10.40"
    }
  },
  {
    "BMC-Primary-Pending": {
      "Description": "The information of BMC (Primary) Pending firmware.",
      "SoftwareId": "null",
      "Status": {
        "Health": "OK",
        "HealthRollup": "OK",
        "State": "Disabled"
      },
      "Version": null
```

Figure 15. Example of firmware levels

Additional data from above command listing the uEFI firmware level.

```
  "UEFI": {
    "Description": "The information of UEFI firmware.",
    "SoftwareId": "UEFI-IVE1-6",
    "Status": {
      "Health": "OK",
      "HealthRollup": "OK",
      "State": "Enabled"
    },
    "Version": "86P-4.30"
```

Figure 16. Example of UEFI firmware level

Example 4. Gather and save the FFDC log file

This script allows you to collect the First Failure Data Capture (FFDC) data so it can be uploaded to Lenovo Support.

```
py lenovo_export_ffdc_data.py
```

```
C:\GitHub\python-redfish-lenovo\examples>py lenovo_export_ffdc_data.py
Start downloading ffdc files and may need to wait a few minutes...
time cost: 140.75s
"The FFDC data is saved as C:\\GitHub\\python-redfish-lenovo\\examples\\7X06CTO1WW
_SN#_xcc_251014-181559.tzz. Messages: [{'@odata.type': '#Message.v1_1_2.Messa
ge', 'MessageId': 'Base.1.12.Success', 'MessageSeverity': 'OK', 'Resolution': 'Non
e', 'MessageArgs': [], 'Message': 'The request completed successfully.'}]"
```

Figure 17. Example to gather FFDC log file

Updating Python

As with every application, there could be updates released for the various items installed above. Please review the individual applications and update them as necessary. Here is an example of how you can upgrade PIP.

```
python -m pip install --upgrade pip
```

The following figure shows an example of a system that needed to be updated.

Figure 18. Upgrade PIP

Updating the Lenovo scripts

There are two different ways you can update the Lenovo scripts:

Option 1

Update using the GitHub application:

1. Go to the GitHub site listed in the Installing Lenovo specific .py files section. Verify that you have the “python-redfish-lenovo” selected in your current repository.
2. Click **Fetch origin** near the top of the screen as shown below.
3. If there are updates you can download them and update the directory.

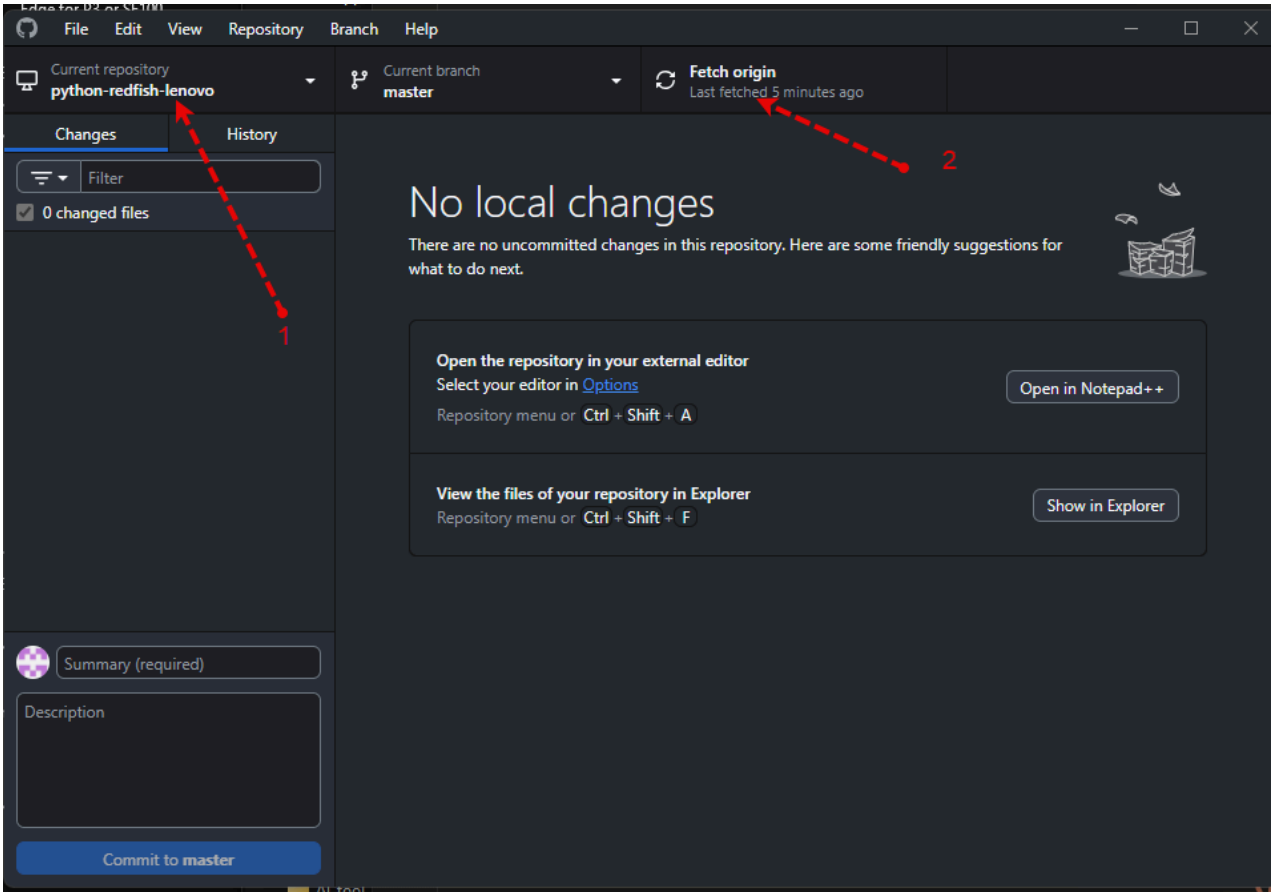

Figure 19. Update using GitHub desktop application

Option 2

Download the zip file containing all the script files manually, unzip the downloaded file.

1. Go to the following web site in your browser: https://github.com/lenovo/python-redfish-lenovo
2. Click the green **Code** button
3. In the dropdown menu, click **Download ZIP**.

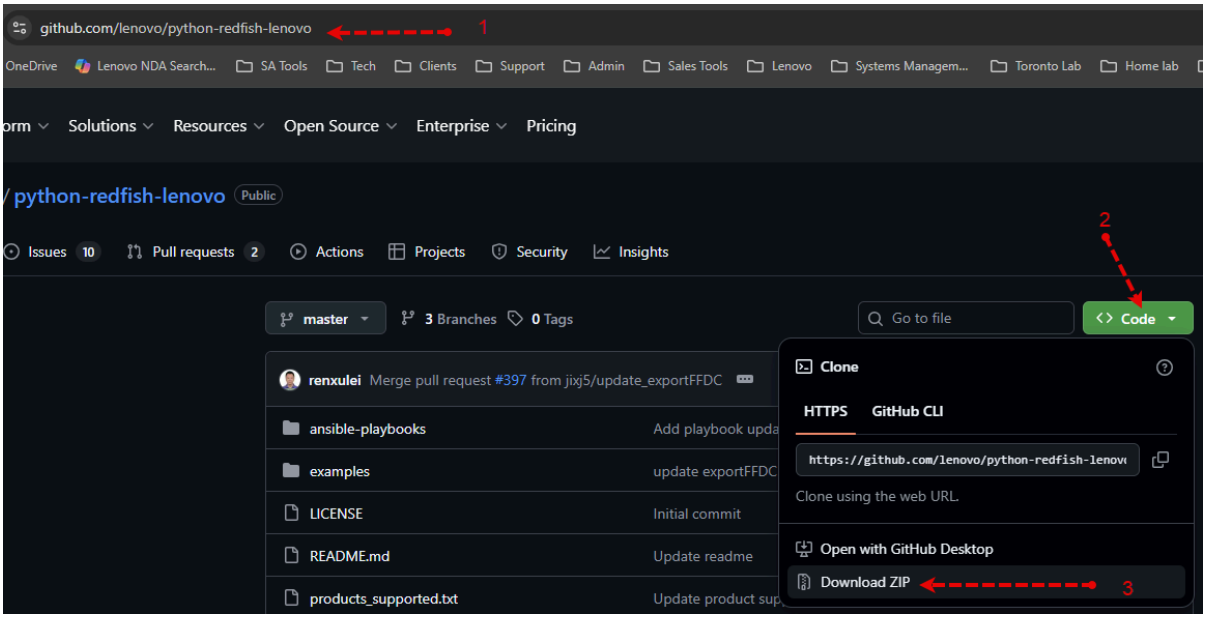

Figure 20. Update scripts via website

Once downloaded, you can exact the files to the directory you want to use.

Tip: If you overwrite the exiting folder, make sure to make the necessary changes to the config.ini file. Consider creating a new folder, edit the config.ini file there, in case you need to revert back to the previous version.

Final thoughts

The above are just a few examples to help you get started. If you want to explore more ways to use Python and interact with your Lenovo servers, please see the additional links below.

Additional links to various documentation sites that can be of help when interacting with the XCC:

- XCC Rest API Pubs site:
  https://pubs.lenovo.com/xcc-restapi/tools_for_redfish
- Main GitHub site:
  https://github.com/lenovo/
- XCC Users Guide
  https://pubs.lenovo.com/xcc/
- XCC2 Users Guide
  https://pubs.lenovo.com/xcc2/
- XCC3 Users Guide
  https://pubs.lenovo.com/xcc3/

Appendix: Lenovo scripts in the GitHub repository

The following is the list of all the scripts included in the python-redfish-lenovo script repository, at the time of writing.

Caution when making changes: The add, delete and set commands change XCC settings, and providing the wrong parameters may result in unexpected results. Example, entering the wrong IPV4 address might stop you from communicating to the XCC. We recommend you use the get command to retrieve the current value of the setting first. This will then ensure you use the correct parameter when making the change.

Some of the scripts below may work on a subset of server models / XCC versions. To see how these commands are grouped and what commands are supported by each server family, review the "products_supported.txt" file located in the main directory where you stored the files. The following figure shows a portion of this file.

Category	Script Name	ThinkSystem AMD 1P	ThinkSystem AMD 2P	ThinkSystem Intel (Purley)	ThinkSystem V2 Intel (Whitley)	ThinkSystem V3 Intel (EGS)	ThinkSystem V3 AMD (Genoa)	ThinkSystem V4 Intel (BHS)
Inventory	get_bmc_inventory.py	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	get_cpu_inventory.py	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	get_fan_inventory.py	Yes	Yes	Yes *3	Yes *3	Yes *3	Yes *3	Yes
	get_memory_inventory.py	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	get_nic_inventory.py	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	get_pci_inventory.py	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	get_psu_inventory.py	Yes	Yes	Yes *3	Yes *3	Yes *3	Yes *3	Yes
	get_power_redundancy.py	Yes	Yes	Yes *3	Yes *3	Yes *3	Yes *3	Yes
	get_storage_inventory.py	Yes	Yes	Yes	Yes	Yes	Yes	NO
	get_system_inventory.py	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	get_chassis_inventory.py	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	get_temperatures_inventory.py	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	lenovo_get_cpu_inventory.py	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	get_volt_inventory.py	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	manage_inventory.py	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	get_chassis_inventory.py	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Power Control	get_power_state.py	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	get_system_reset_types.py	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	set_power_state.py	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	restart_bmc.py	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Figure 21. Sample of the products_supported.txt file showing grouping and supported servers

The following is the list of scripts currently available:

- Inventory
  - get_bmc_inventory.py
  - get_cpu_inventory.py
  - get_fan_inventory.py
  - get_memory_inventory.py
  - get_nic_inventory.py
  - get_pci_inventory.py
  - get_psu_inventory.py
  - get_power_redundancy.py
  - get_storage_inventory.py
  - get_system_inventory.py
  - get_chassis_inventory.py
  - get_temperatures_inventory.py -
  - lenovo_get_cpu_inventory.py
  - get_volt_inventory.py
  - manage_inventory.py
  - get_chassis_inventory.py
- Power Control
  - get_power_state.py
  - get_system_reset_types.py
  - set_power_state.py
  - restart_bmc.py
- Event Log

    - get_system_log.py
    - clear_system_log.py
    - lenovo_export_ffdc_data.py
- Event subscription
    - get_event_subscriptions.py
    - add_event_subscriptions.py
    - del_event_subscriptions.py
    - send_test_event.py
    - lenovo_add_alert_recipient.py
    - lenovo_del_alert_recipient.py
    - lenovo_get_alert_recipients.py
- BIOS settings
    - get_all_bios_attributes.py
    - get_bios_attribute.py
    - set_bios_attribute.py
    - get_bios_attribute_metadata.py
    - reset_bios_default.py
    - get_bios_bootmode.py
    - set_bios_bootmode_legacy.py
    - set_bios_bootmode_uefi.py
    - lenovo_get_bios_boot_order.py
    - lenovo_set_bios_boot_order.py
    - get_server_boot_once.py
    - get_server_boot_once_types.py
    - set_server_boot_once.py
    - get_secure_boot_status.py
    - disable_secure_boot.py
    - enable_secure_boot.py
    - reset_secure_boot.py
    - set_bios_password.py
    - lenovo_set_amt.py
- User Management
    - lenovo_get_bmc_user_accounts.py
    - lenovo_create_bmc_user.py
    - lenovo_delete_bmc_user.py
    - update_bmc_user_password.py
    - lenovo_get_bmc_user_global.py
    - lenovo_set_bmc_user_global.py
    - disable_bmc_user.py
    - enable_bmc_user.py
- Light Path
    - get_chassis_indicator_led.py
    - set_chassis_indicator_led.py
- Power Management
    - get_power_limit.py
    - set_power_limit.py
    - get_power_metrics.py
- BMC Configuration
    - get_bmc_ntp.py
    - set_bmc_ntp.py
    - lenovo_set_bmc_dns.py
    - get_networkprotocol_info.py
    - set_networkprotocol.py

    - lenovo_bmc_config_backup.py
    - lenovo_bmc_config_restore.py
    - get_serial_interfaces.py
    - set_serial_interfaces.py
    - lenovo_set_serial_interfaces.py
    - get_hostinterface.py
    - set_bmc_timezone.py
    - set_bmc_vlanid.py
    - lenovo_set_bmc_config_default.py
    - set_bmc_ipv4.py
    - lenovo_generate_snmp_engineid.py
    - lenovo_get_snmp_global.py
    - lenovo_set_snmp_global.py
    - update_bmc_user_snmpinfo.py
    - set_bmc_hostname.py
    - lenovo_callhome_setting.py
    - lenovo_callhome_getinfo.py
- Virtual Media
    - get_virtual_media.py
    - mount_virtual_media.py
    - umount_virtual_media.py
    - lenovo_mount_virtual_media.py
    - lenovo_umount_virtual_media.py
- RAID Configuration
    - lenovo_create_raid_volume.py
    - lenovo_delete_raid_volume.py
    - lenovo_update_raid_volume.py
    - set_raid_encryption_mode.py
- FW Update
    - get_fw_inventory.py
    - update_firmware.py
    - lenovo_update_firmware.py
    - firmware_startupdate.py
- Telemetry
    - get_metric_definition_report.py
    - send_test_metric.py
- Session
    - get_sessions.py
    - clear_sessions.py
- License
    - lenovo_bmc_license_delete.py
    - lenovo_bmc_license_export.py
    - lenovo_bmc_license_getinfo.py
    - lenovo_bmc_license_import.py
- LDAP
    - lenovo_get_bmc_external_ldap.py
    - lenovo_set_bmc_external_ldap.py
    - lenovo_ldap_certificate_disable.py
    - lenovo_ldap_certificate_enable.py
    - lenovo_get_bmc_user_ldap_policy.py
    - lenovo_set_bmc_user_ldap_policy.py
    - lenovo_ldap_certificate_add.py

- Security
  - lenovo_eklm_keyserver_config.py
  - lenovo_eklm_keyserver_getinfo.py
  - lenovo_eklm_certificate_generate_csr.py
  - lenovo_eklm_certificate_import.py
  - lenovo_ekms_localcache_setting.py
  - lenovo_ssl_certificate_getinfo.py
  - lenovo_ssl_certificate_generate_csr.py
  - lenovo_ssl_certificate_import.py
  - lenovo_get_ssh_pubkey.py
  - lenovo_import_ssh_pubkey.py
  - lenovo_delete_ssh_pubkey.py
  - lenovo_httpfs_certificate_import.py
  - lenovo_httpfs_certificate_getinfo.py
  - lenovo_httpfs_certificate_enable.py
  - lenovo_httpfs_certificate_disable.py
  - lenovo_httpfs_certificate_delete.py
- Other
  - set_server_asset_tag.py
  - get_all_tasks.py
  - del_tasks.py
  - get_schema.py
  - set_system_name.py
  - raw_command_get.py
  - raw_command_patch.py
  - raw_command_post.py

Author

Ron Young is a Senior Lenovo Solutions Architect working in the Lenovo Infrastructure Solutions Group (ISG) based in Minneapolis, Minnesota. He has more than 27 years of experience with Infrastructure solutions in his career at both IBM and Lenovo.

Thanks to the following people for reviewing and providing feedback on this document:

- Ted Tansill
- Terry Flickinger

Related product families

Product families related to this document are the following:

- Lenovo XClarity

Notices

Lenovo may not offer the products, services, or features discussed in this document in all countries. Consult your local Lenovo representative for information on the products and services currently available in your area. Any reference to a Lenovo product, program, or service is not intended to state or imply that only that Lenovo product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any Lenovo intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any other product, program, or service. Lenovo may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

Lenovo (United States), Inc.
8001 Development Drive
Morrisville, NC 27560
U.S.A.
Attention: Lenovo Director of Licensing

LENOVO PROVIDES THIS PUBLICATION ”AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Lenovo may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

The products described in this document are not intended for use in implantation or other life support applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change Lenovo product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Lenovo or third parties. All information contained in this document was obtained in specific environments and is presented as an illustration. The result obtained in other operating environments may vary. Lenovo may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any references in this publication to non-Lenovo Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this Lenovo product, and use of those Web sites is at your own risk. Any performance data contained herein was determined in a controlled environment. Therefore, the result obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

This document, LP2370, was created or updated on February 17, 2026.

Send us your comments in one of the following ways:

- Use the online Contact us review form found at:
  https://lenovopress.lenovo.com/LP2370
- Send your comments in an e-mail to:
  comments@lenovopress.com

This document is available online at https://lenovopress.lenovo.com/LP2370.

Trademarks

Lenovo and the Lenovo logo are trademarks or registered trademarks of Lenovo in the United States, other countries, or both. A current list of Lenovo trademarks is available on the Web at https://www.lenovo.com/us/en/legal/copytrade/.

The following terms are trademarks of Lenovo in the United States, other countries, or both:
Lenovo®
XClarity®

The following terms are trademarks of other companies:

Intel®, the Intel logo and Xeon® are trademarks of Intel Corporation or its subsidiaries.

Microsoft®, PowerShell, and Windows® are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.