

# Lenovo XClarity Integrator for Proxmox

(Version 1.01 - 20250918)

## Overview

**Lenovo XClarity Integrator for Proxmox** is a Python-based tool that enables automated firmware updates for **Proxmox Cluster** environments running on **Lenovo ThinkSystem** servers. It integrates with **Lenovo XClarity Administrator (LXCA)** to ensure that your infrastructure remains up-to-date with the latest hardware updates, minimizing downtime and reducing manual intervention.

This solution is designed for IT administrators who want to streamline update management across their Proxmox clusters.

---

## Features

- Seamless integration with **Lenovo XClarity Administrator** REST API.
  - Automatic firmware update based on policy applied via **Lenovo XClarity Administrator**
  - Cluster-aware update operations to reduce service disruption.
  - Logging and reporting of update results.
  - Customizable configuration via `config.ini`.
- 

## How it works

When you execute the program, it performs the following steps:

- Loads runtime parameters from the `config.ini` file
- Asks for the master password, if passwords are stored in the secure way (i.e. encrypted)
- Verifies that the hosts are managed by LXCA and a firmware compliance policy is associated
- Verifies if the hosts are compliant with the assigned compliance policy
  - If not, sequentially, evacuates VMs and CTs from the host, performs the firmware update by LXCA, moves back VMs to the updated host

## Requirements

- **Python:** 3.x (tested with Python 3.11 and Python 3.13 (strict SSL))
- The Python libraries listed in `requirements.txt`
- A running **Lenovo XClarity Administrator** instance with appropriate API credentials

- A **Proxmox VE cluster** or **Proxmox VE single node** running on Lenovo XClarity Administrator supported servers (tested on Lenovo ThinkSystem servers only)
  - A **Compliance Policy** assigned to each server in the cluster
  - An OS instance where to run this tool (Linux is preferred but it should work also on Windows). **ATTENTION:** Not supported/recommended to be run directly on Proxmox servers.
- 

## Installation

1. Unzip the package on your system (better if Linux based)
2. (Optional but recommended) Setup a Python Virtual Environment

2.1. Create a new virtual environment in the project directory:

```
python -m venv venv
```

2.2. Activate the virtual environment:

– On Linux/Mac:  
`source venv/bin/activate`

– On Windows:  
`venv\Scripts\activate`

3. Install the dependencies:

```
pip install -r requirements.txt
```

4. Prepare your configuration file (see **Config File** below).
- 

## Config File

1. Review the sample configuration file `config.ini.sample`. This file contains all configurable parameters, including:
  - LXCA hostname or IP
  - LXCA API credentials
  - Proxmox cluster node list
2. Copy the sample and customize it:

```
cp config.ini.sample config.ini
```

Edit `config.ini` with your preferred editor to declare environment-specific settings.

## Fields details

### *Proxmox section*

- **PROXMOX\_USER** - Mandatory - This field contains the user name authorized to access the Proxmox cluster (every node in the cluster) and to perform actions such as migrate VMs. It **MUST** be a Linux user since it is also needed to set the node in maintenance mode. - Default = root@pam (with no quotes).
- **PROXMOX\_PASS** - Mandatory - You must store here the password for the user declared in the PROXMOX\_USER field. The password can be stored, with no quotes, in plain-text (easiest, less secure) or in encrypted way (more secure). For the instruction on how to store encrypted password in the config file see the "Security section" below.
- **PROXMOX\_VERIFY\_SSL** - Mandatory - If you want to skip SSL Certificate validation you can set this field to False. Usually with self-signed certificates this is the most common behavior, if you are using self-signed certs and Python > 3.11 you **MUST** set this parameter to False. You can set this field to True following the instructions in the "CA Certificate validation" section
- **CLUSTER\_NODES** - Mandatory - This field contains the host(s) that can be updated by the tool. The field is an array where each element is a JSON-like object describing three parameters for the host:
  - xcc = the IP Address of the host XCC (**MUST** be the IP address and not the hostname)
  - pve\_ip = the IP Address of the PVE host
  - pve\_host = the hostname of the PVE host

the expected format for the host is: {"xcc":<IP\_xcc\_node>, "pve\_ip": <IP\_node> , "pve\_host": <NAME\_node>}

- **TIMEOUT\_HOST** - Optional - Specify here the maximum time (in seconds) to wait the upgrade process to complete. - Default = 3600
- **TIMEOUT\_VM** - Optional- Specify here the maximum time (in seconds) to wait for VM operations (migrate, shutdown) to complete before to return an error. - Default = 300 secs
- **VM\_LOCAL\_DISKS** - Mandatory - Allowed values are: POWEROFF to authorize the tool to shutdown the VMs with local disks (e.g. cdrom, vm disks on local storage), FAIL to stop the procedure and allow manual remediation. - Default = FAIL
- **VM\_LOCAL\_RESOURCES** - Mandatory - Allowed values are: POWEROFF to authorize the tool to shutdown the VMs with local resources (e.g. PCIe adapter), FAIL to stop the procedure and allow manual remediation. - Default = FAIL
- **VM\_LOCAL\_DISKS\_EXPERT** - Experimental - Optional - Use at your own risk!!  
- EXPERT ONLY - Do not use unless you know what you're doing. This

parameter allows the migration of VMs with local disk (no-cdrom). It's up to you verify that all the requirements are satisfied at the runtime (e.g. same storage names, enough disk space, etc). Allowed values are: `True`, the program will try to migrate the VM with local disk (no local cd-rom), or `False`, the program will follow the statement in `VM_LOCAL_DISKS`. Default = `False`

#### *LXCA session*

- **LXCA\_HOST** - Mandatory - This field contains the IP Address (or the hostname if resolved by DNS) of the Lenovo XClarity Administrator (LXCA) instance that manages the Proxmox servers.
- **LXCA\_USER** - Mandatory - This field contains the LXCA administrative username authorized to perform updates to the Proxmox managed servers.
- **LXCA\_PASS** - Mandatory - Insert here the password for the LXCA user specified in the `LXCA_USER` field. The password can be stored, with no quotes, in plain-text (easiest, less secure) or in encrypted way (more secure). For the instruction on how to store encrypted password in the config file see the "Security section" below.
- **LXCA\_VERIFY\_SSL** - Mandatory - If you want to skip SSL Certificate validation you can set this field to `False`. Usually with self-signed certificates this is the most common behavior, if you are using self-signed certs and Python > 3.11 you **MUST** set this parameter to `False`. You can set this field to `True` following the instructions in the "CA Certificate validation" section

#### *Security section*

- **SALT** - Optional - You can store the SALT here to allow encrypted passwords storing.

**Please note:** if you are worried about storing plain-text passwords in the `config.ini` file, you can use the following procedure to store them encrypted. If `SALT` option is declared in the `config.ini` file, then the program will ask for the master password to decrypt the password fields in the configuration.

---

### **How to encrypt the password in the `config.ini` (optional)**

You can run the `create_encrypted_password.py` program if you want to store encrypted password in the configuration file.

```
python create_encrypted_password.py
```

It will ask for a master password (needed later to run the integrator) and it will encrypt the PVE and LXCA passwords, printing out the lines that must be put in the config file (example):

Add the following three lines to your config.ini:

```
LXCA_PASS = gAAAAABonuj0JNenRko_DS-HTcZkFfoho_ZID6RcwLTFPNn7A2QYsNgtE3-  
wjzCE8R0_bpVquZLDGpe9ARpJdZ7tSEWoNXT0Qg==  
PROXMOX_PASS = gAAAAABonuj0aXh5IYUhClWjBa0Bh9HgcvtmwyB9rYz1vnS0-NAS51rN  
QXaBH0CjlgvF4lkwpPsXSIi2SLpozKmueTtpV_uQJQ==  
SALT = E3T1B/Ty6nk3RJq7dQq8Pg==
```

## CA Certificate validation

If you are using custom certificates in your environment, you can choose to verify the certificates of the Proxmox hosts and/or of the LXCA instance to be security compliant.

To do this, you need to create, in the same tool directory, a file named `custom_cacert.pem` where you have to manually store the CA certificate(s) for your environment in the pem format (see example below):

```
-----BEGIN CERTIFICATE-----  
MIIFzTCCA7WgAwIBAgIUdmBnkIBixu29Fv7wAI4de2sxPQ0wDQYJKoZIhvcNAQEL  
BQAwZjEKMCIgA1UEAwwbUHVjveG1veCBWaxJ0dWfsIEVudmlyb25tZW50MS0wKwYD  
VQQLDCQzNzI4Mzd1NS1jMjvklTQwZjgtODA2NC02YWY2OTF1NWMyODMxHAdBgNV  
BAoMF1BWRSDbHVzdGVyIE1hbmFnZXIgaQ0EwHhcNMjUwMzE0MTE0NDE5WWhcNMzUw  
MzEyMTE0NDE5WjB2MSQwIgYDVQQDDbtQcm94bW94IFZpcnR1YWwgRW52aXJvbm1l  
bnQxLTArBgNVBAsMJDM3MjgzN2U1LWMyNWQtNDBmOC04MDY0LTZhZjY5MmU1YzE4  
MzEfMB0GA1UECgwWUZFIEsdxN0ZXIgaTWFuYXd1ciBDQTCcAiIwDQYJKoZIhvcN  
aQEBBQADggIPADCCAgcCggIBANR+cPxIkIEBNx9YCAaaL2+fOUWy/fBaeBMCdD+D  
7F/Bi+Mcpn2p3+gVuhYCaTcYZnP0aewHUI4xCL8YHFwqWv6J7ryMV++i/0Rbkzny  
EZvKISNxTuZ/9ZioP1S9K3Se3ZVZI23xyLQ/Wu2mqPo5d3QLttR5/ndBLVmUtMZe  
pz3JlJ2kQxvQrbG4SdHBS4GpqlbDoqktkFK+nJjGuibNVovZFkufv+7ySoKhbF9l  
I21A4Ta+QqlhG+TD6r4c61yGKgX/IozrsvIOa0BwLHX81gE0wh5BZ6G++OKblQu+  
nb7ZEyZYUQ4sT6I+HykcyA9amil4S1x5qn0ohNJwtd9QCZZiFWQhOS+cI627L0EU  
uQ7Dj60McDyrMqgNWZOnBKLqoDSi2M1ZkXI1ACa/IoUeutITlMnZucLIIdTM1VPhV  
qQIJDb2TaKQnjxnuqSXaA0DVwcSnX+VZgTUbzbV/+t1PENnQ3KcarqvwkUX1QB0  
uagYmCURylUJXOT4r+8oUJDak4zwPjOQLH+PF30dGScj4FIBYnL+1G08UmhHzixi  
a76J8eF0N0x65tBJDjie3McU7vrDJ8DrIbcM/LovFa9PCrGe9wNmHqQ0toUuxdaR  
b6co1YE340F2cdyeaEVRVYtKFFp/Oa0ZA10eBE0Fxxv4KTFi4/Y+nuMjKMY0Kmo  
XUxJAgMBAAgUzBRMB0GA1UdDgQWBQxTKcz/j1/m505HaGyrZtGqiDaFDAfBgNV  
HSMEGDAWgBQxTKcz/j1/m505HaGyrZtGqiDaFDAPBgNVHRMBAf8EBTADAQH/MA0G  
CSqGSIb3DQEBCwUAA4ICAQCcsA4imt0cpgYNbBvufvBp0bGKkiSoimd6p1fpJhjX  
PMDdtn6dVw2zbAmqDmBqdRTenK9J300WVoYmpnh5H+p8wM1uT5js248DmhtXTeUd  
xHHVyz9c2ooZT/7EGqWAa/VRpKAeN99uzEmLZxxxb0Sg7wvN8Q7KbIiJdtckqo09  
sZqDXIKFnrQL0Gyys9ZvsGZFpZoUipA+IbJVQ1ArKjWfNZkPhT14xMPdoZAMX4Zi  
45srzk4Z9sJjKkYaA5eFsirbP0i9N5KmrqmD2c1Jb4p3ScdzJ07ay5PmkYdeIq3K  
vN9MPAuMSDwmQasqB/o01gRmdcWNU2ABbsrEeALMPhe/J82meUQd7MorVC870UJ6  
mp2b97TTBdqsqsrygKmwB2YFXFS/uR5ZaYEXDjxe9LXCRODRJA1GF882PS20YoV  
iUBGwd361+nkLkJ3GaT8l/PkpHLg5mXU6UmiPEw90+mzYFNAhYzvzfIP4G4afYYo  
T+/8KBBqWVuW6ecv/vqLQIyc0ucOgnH9vhwliGTyMug/GhaTjXFtRdjdU0W7irmM  
CgkLeEpvWN1xgNTBILiQva6dc31V36hOVgSjRcIr4r0sCe2I3J6ixw7ShHXxMYAi  
od9Lz4mJrL4k1SovZEonEr6HmXY1sdV2R6/ZaoCKYN2F1zZ4IzGuSRU1iUiz5iCn  
WQ==  
-----END CERTIFICATE-----
```

**Please note:** Starting from Python 3.13 some strict defaults are implemented in the SSL management. These make the underlying OpenSSL implementation behave more like a conforming implementation of RFC 5280, in exchange for a small amount of incompatibility with older X.509 certificates. The SSL context now uses VERIFY\_X509\_PARTIAL\_CHAIN and VERIFY\_X509\_STRICT in its default verify flags. This means that the CA certificate must comply with all the strict rules (e.g. X509v3 Key Usage, X509v3 Basic Constraints = Critical, etc). \*\*\*

## Usage

```
python lxca_proxmox_integrator.py
```

Whit no parameter passed, you'll get this usage help:

```
usage: lxca_proxmox_integrator.py [-h] (-d | -x) [-v] [-c] [-V]
```

LXCA Proxmox integrator command line parameters

options:

-h, --help	show this help message and exit
-d, --dry-run	Simulate the upgrade
-x, --execute	Perform the upgrade
-v, --verbose	Add verbosity to the ouput
-c, --clearlog	Move current log to backup and start a clean log file
-V, --version	Show program version

You **MUST** specify either the -d | --dry-run or the -x | --execute parameter.

The dry-run parameters (-d | --dry-run) allow you to simulate the full cycle without performing any action on the cluster.

During the dry-run round a simulation will be performed, listing all the findings in the enviroment (better used with -v option).

It is strongly recommended to **always** perform a dry-run (-d | --dry-run) before running with -x | --execute parameter, as it will run the firmware updates.

The -v | --verbose parameter add verbosity to the output (both console & log).

The -c | --clearlog parameter will create a backup of the current log and initializes a new empty log file.

The tool will (main workflow):

- Ask for the master password if needed (optional)
- Connect to your LXCA instance.
- Check your ThinkSystem nodes for available updates.
- Orchestrate firmware update across the cluster/nodes.

**Please note:** At present, online migration of the CT container is unsupported; therefore, migration will be performed by shutting down the container and instantly restarting it on the target host.

---

## Logging

All operations are logged to the console and also to a log file (`lxca_proxmox_integrator.log`). Review logs to verify update success and troubleshoot any issues.

---

## Disclaimers

- As best practice, we warmly suggest to always test firmware and driver updates in a staging or non-production environment before rolling them out cluster-wide. You can also leverage the DRY\_RUN execution to perform a simulation for the other operations but not applying firmware updates.
- Any errors encountered during the procedure will result in the tool being stopped, allowing the IT admin to verify and fix the issue. The only exception is related to the “power on” of the VMs/CTS that will be showed in the log but the procedure will continue. Manual recovery could be required to restore the correct state.
- Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## License

Licensed under the Apache License, Version 2.0 (the “License”):  
<http://www.apache.org/licenses/LICENSE-2.0>

## Author

**Mario Sebastiani** is part of the Lenovo ISG Technical Sales team in Italy, where he focuses on HPC/AI and SMB businesses. He’s a Linux enthusiast with over 30 years of IT experience, most of which was spent on IBM delivery services and developing solution architectures, with a focus on security in recent years. Mario joined Lenovo in 2022 and has recently been deepening his knowledge of AI solutions.

A special thanks to **Matthias Marx** and **Silvio Erdenberger** from Lenovo EMEA IC and to **Gianfranco Bauco** from Lenovo ISG Italy Technical Sales team for their contribution and support developing this tool.