Canon
Professional Services

MOSCA-LCM

MyOwnSmartCertificateAuthority
Life Cycle Management

V4.4.0

Content

Introduction 3
Requirements 4
- General 4
- NTP-Onboarding 4
- Webservice 4
- External Signing 4
- Subject Alternative Names 5

Preparation checklist 7
First Start 8
Licensing 8
Overview of the user interface 10
- Menus 10
- DeviceList 10
- Edit Devices section 10
- Edit Files section 10
- Scheduled Task & Service section 10
- Credentials section 10
- List Control section 11

The Device List 11
- Description 11
- Special Functions 13
- Context Menu 13
- The Tools Menu 14
- Adding/Importing devices 18
- Edit devices 20
- Remove devices 20
- Export devices 20

Base Configuration....21
Credentials....21
MOSCA_LCM_Service settings....23
General settings....24
Onboarding Settings....26
EMC Import settings....27
EMCOnboarding settings....27
SAN settings....27
WS2 settings....29
Command Files....30
MOSCA LCM Commands (*.mlc)....30
External Signing Commands using RPC....37
External Signing Commands using CEWS....38
External Signing Commands using an API....39
Sample 1 – SignAPI.txt....39
Sample 2 – SignAPI_PS.txt....40
Sample 3 – SignAPI_Venafi_PS.txt....41
List of API parameters....42
External Signing using NDES SCEP....43
External Commands....44
EMCOnboarding....45
Scheduled Task & Service....46
List Control....49
NTP Onboarding....49
Description....49
Configure Canon Onboarding Service....51
Install Canon Onboarding Service....52
Uninstall Canon Onboarding Service....52
Webservice....53
Configuration of the Webservice....53
Install Webservice....54
Uninstall Webservice....54
Use the Webservice....55
Monitoring....57
SysLog....57
Mail Client....58
Web-API....61
Export a CSV-file....63
Devices in uniFLOW Online....66
Account without MFA....66
Existing Account with MFA....66
First login to a new account enabling MFA....69
Update MOSCA-LCM....71
Manual Update....71
In place Update....73
Appendix....74
List of supported devices....74
Used Ports and Protocols....76
How to use MOSCA-LCM with non-supported devices....77
SCEP Requirements....79

Introduction

MOSCA-LCM is a tool for reviewing, requesting, generating and deploying security certificates to Canon print and copy devices. Once configured, parts of MOSCA-LCM run in background triggered by a scheduled task and as a Windows service.
MOSCA-LCM reads all known Canon devices periodically, checks the validity of the activated certificates and (if necessary) requests/generates and deploys new certificates.
MOSCA-LCM supports direct signing, resigning to add Subject Alternative Names and signing after approval (approval workflow).

”MOSCA_LCM.exe” is the program to configure the most settings and to monitor the certificate states of the devices.

“MOSCA_LCM_Sevice.exe” is the program that is doing the work when checking devices and during the enrollment of certificates. It can be run manually each time wanted or installed as a scheduled task during configuration. When the “MOSCA-LCM-Service” has finished all actual tasks to do, it will automatically end. With this behavior you have full control when MOSCA-LCM is working and when not.

New devices can be added manually or in a bulk import by file. In addition to this MOSCA-LCM offers an Auto-Onboarding feature via NTPS requests. With this feature activated, new devices will be automatically added to the MOSCA-LCM database and certificates will be enrolled just after the devices entered the network for the first time.

MOSCA-LCM supports TLS/SSL certificates as well as 802.1X certificates. For the highest security, certificate requests can be generated on the Canon devices and either signed via connection to your Certificate Authority or in MOSCA itself via OpenSSL.

With MOSCA-LCM you always have a complete overview of your network security certificate states. Certificates will be automatically replaced before they reach their date of expiry.

MOSCA-LCM comes with Code Integrity Verification (CIV). This means, that all executables and dynamic link libraries (DLL’s) are checked during runtime to ensure that they belong to the MOSCA-LCM package. Binary Planting and DLL Hijacking will be prevented.
The webservice of MOSCA-LCM is secured with credentials and different access roles. It is also secured against several attack methods like Parameter Injection and Cross-Site-Scripting.

Requirements

General

MOSCA-LCM is portable and needs no installation. In most cases MOSCA-LCM needs to be started once with the “Run as administrator” option for the base configuration. After that a normal start in a user context will be sufficient.
MOSCA-LCM needs the MS .Net Framework 4.8 or its runtime to be installed. The software runs on MS Windows 10 up to MS Windows 11 or MS Windows Server 2012R2 up to MS Windows Server 2025 with at least 8 GB of RAM installed.
Since MOSCA-LCM uses the Remote User Interface (RUI) of the Canon devices, it requires web access to the devices. MOSCA-LCM requires Microsoft Edge Browser installed and comes in addition with the Chromium Headless Shell browser in its package.

NTP-Onboarding

When using the NTP-Onboarding feature, incoming UDP port 123 has to be opened in the firewall on the server MOSCA-LCM is running on. Other NTP services on the same server, like Windows Server’s “Windows Time” service have to be stopped and set to disabled state. It is recommended, to unregister the “Windows Time” service with the command “w32tm /unregister”. Furthermore, SNMP v1 or v3 has to be configured on the Canon devices.

Webservice

If you want to use the included webservice to use MOSCA-LCM via its One Page Application in a web browser, you have to ensure that none of the ports configured in MOSCA-LCM are in use on the same system. An IIS installation is not needed but will also not conflict with the MOSCA-LCM webservice if using different ports.
Please refer to the section “Configuration of the Webservice” in this document for more information.

External Signing

If you want to let MOSCA-LCM sign certificates on your Certificate Authority (CA), then one of four channels have to be enabled to do that:

1. When singing via Remote Procedure Call (RPC), the RPC service has to run on the CA server and the communication port 135 has to be opened.
2. When signing with the MS Certificate Enrollment Webservices (CEWS), the CEP/CES service has to be configured on your CA and port 443 has to be opened.
3. When signing against a CA providing an API for signing, the API call has to be compatible to CURL or Invoke-WebRequest (PowerShell). In addition, port 443 has to be opened.
4. When signing against an NDES SCEP interface from Microsoft, NDES SCEP has to be configured on your CA and port 443 has to be opened.

In addition to that, the user context in which the scheduled task for the MOSCA_LCM_Service runs, has to have access to the servers RPC / CEP/CES-

URL / API-URL / NDES SCEP-URL. When signing a certificate, a certificate template is used. The MOSCA_LCM_Service user, a special configured user account in MOSCA-LCM or the machine MOSCA-LCM is running on, has to have the access rights to use this template.
Since MOSCA-LCM supports direct signing, resigning and signing after approval, up to 3 shell commands have to be configured and tested. Please refer to “External Signing Commands” to get more information about these commands.

For NDES SCEP support, the MOSCA_LCM_Service user needs to be in the local group “IIS_IUSRS” of the CA/Sub-CA. You can use a fixed challenge password, or to let MOSCA_LCM request dynamic challenge passwords, the service user also needs access to the “/certsrv/mscep_admin“-page of the CA/Sub-CA. Find detailed information about SECP requirements under „Appendix / SCEP Requirements“.

Subject Alternative Names

Some Canon devices are not able to set Subject Alternative Names (SANs) in their Certificate Signing Requests (CSRs). If you only have devices supporting SANs in the CSR and only want to have the CommonName as SAN entry, you might skip this chapter. MOSCA will automatically fill the SAN field in the device CSR.

If you want to have multiple SAN entries (e.g. CN, FQDN, IP) or you have devices not supporting a SAN entry in the CSR, MOSCA can also help you. MOSCA-LCM can add SANs in the external signing process against an MS Certificate Authority. There are two by MOSCA-LCM supported methods to do that:

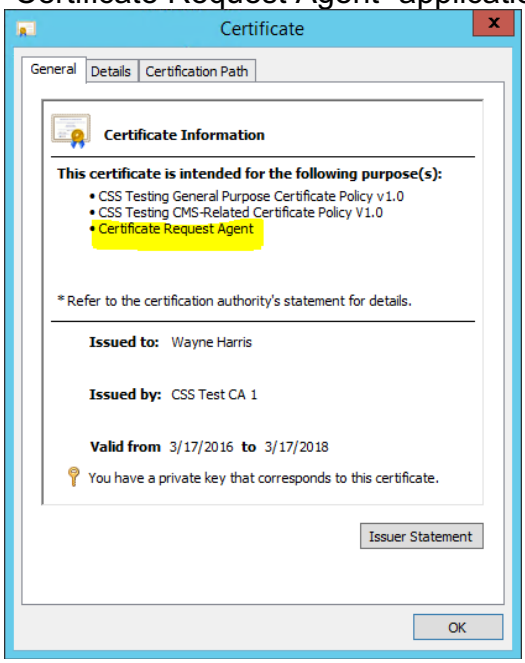
Method 1 – During Signing:
To enable this method please refer to the section “Base Configuration / MOSCA_LCM_Service”.
The feature has also to be enabled on the MS CA. This can be done by setting the policies on the CA Server to allow adding SANs in the signing process with the following shell command:
certutil -setreg policy\EditFlags +EDITF_ATTRIBUTESUBJECTALTNAME2

You have to restart the MS Certificate Services to let the changes take effect, e.g. with these shell commands:
net stop certsvc
net start certsvc

NOTE: Changing policies on the CA Server to allow adding SANs (method 1) is associated with a high security risk and should only be used in test environments.

Method 2 – Resigning CSR using EA certificate:
This method uses an enrollment agent certificate to resign the original certificate request and add the needed SANs.

You have to acquire an enrollment agent certificate. This is a certificate based off the Enrollment Agent default template. The resulting EA certificate must contain the “Certificate Request Agent” application policy extension. (1.3.6.1.4.1.311.20.2.1.)



This certificate (and associated private key) should be located on the workstation of MOSCA-LCM. It is this certificate that MOSCA-LCM will use during the re-signing process to authorize itself against the CA.
To enable this method please refer to the section “Base Configuration / MOSCA_LCM_Service”.

Preparation checklist

- Access via ports 80, 8000, 443, and 8443 must be possible from the MOSCA server to all target machines.

- The Microsoft Edge browser needs to be installed.

- A shared certificate template with Server Authentication (when used for SSL) and/or Client Authentication (when used for 802.1x) must be available.

- A service user account under which the MOSCA service runs must be created. Since this service is started as a scheduled task, the account must also be granted the **“Log on as a batch job”** policy right.

- The certificate template must be accessible for this service account. Alternatively, a special account for external signing can be configured in MOSCA-LCM.

- If Subject Alternative Names (SANs) are to be included in the certificate, in some cases an Enrollment Agent certificate is required on the machine where MOSCA is running. This enables MOSCA to add SAN entries to the certificate requests of the Canon devices.

- To enable communication with the CA/Sub-CA, the RPC service (Port 135) must be running on the CA/Sub-CA and be accessible from the MOSCA server. Alternatively, communication via CEWS, an external API and NDES SCEP is also possible. Therefore Port 443 needs to be accessible on the CA or its subsystem.

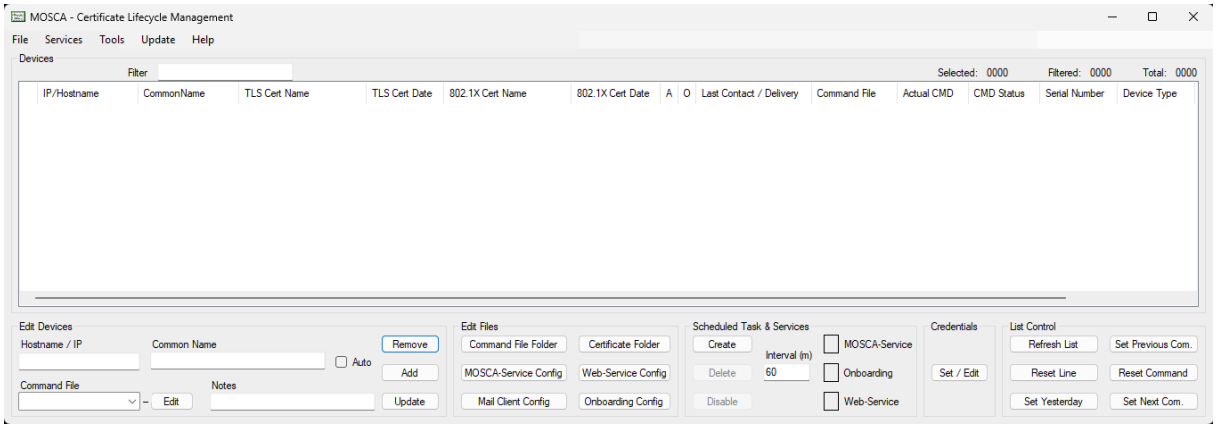
  Sample RPS call:
  certreq -submit -binary -rpc -q -f -config "DC1\DC1-CA" -attrib "CertificateTemplate:Computer802" …

  Sample CEWS call:
  certreq -submit -binary -q -f -config "https://dc2-ca/DC2-CA_CES_Kerberos/service.svc/CES" …

- For 802.1X, the root certificates of the certificate used on the RADIUS server must be provided. These certificates will be installed in the Trusted Root Store of the Canon devices to establish trust with the RADIUS server. The certificates must be in DER format.

First Start

To run MOSCA-LCM please start the “MOSCA_LCM.exe” once with the option “Run as administrator”. MOSCA-LCM will register some components, connect to the database and start up.



Licensing

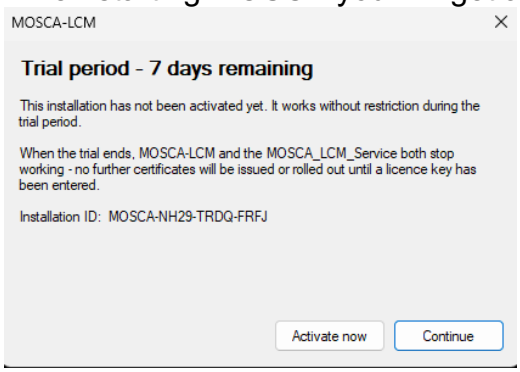
Summary

Since version 4.2.0 MOSCA needs to be activated. The licence exists only to know which customers run MOSCA.

- It is “free of charge” and there is no limit on devices, workflows or runtime.
- The license is bound to the machine MOSCA is installed on, not to a customer or a contract.
- No network connection is involved. MOSCA never contacts a licence server; the key is checked locally.
- The License is stable for a given machine. It does NOT change when MOSCA is updated, moved to another folder on the same machine, or reinstalled there.
- When using MOSCA for the first time, a 7-day trial period is started. In these 7 days no license activation is needed. MOSCA will work without any limitations. After the 7 days the MOSCA service will stop working silently and an activation is needed.

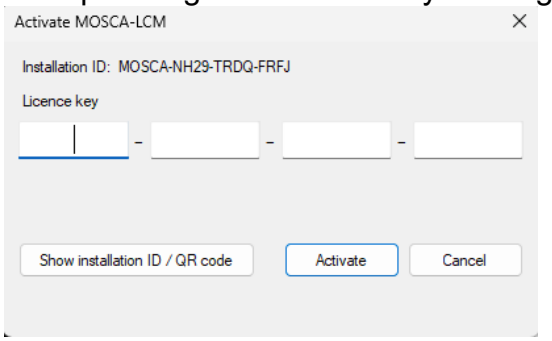
Activation process

When starting MOSCA you will get a License message.



Press “Continue” to continue with the trial period or “Activate now” to start the activation.

After pressing “Activate now” you will get this dialog.

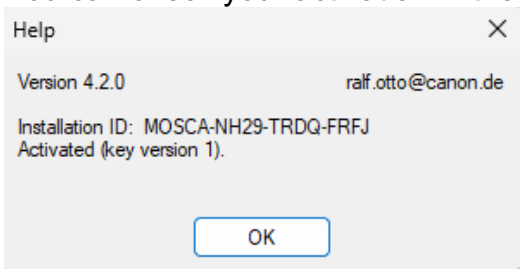


Then press “Show installation ID / QR code” to get to the Installation ID.



Press “Copy” to copy the ID. In future, there will be a license portal, by now you have to send the ID to your Canon contact for MOSCA to get a valid License code.

If you got a License code enter it in the previous form and press “Activate”.
You can check your activation in the menu “Help / About”.



Overview of the user interface

Menus

In the “File Menu” you can import and export devices to the device list. Please refer to the sections “Adding/Importing devices” and “Export devices”.

In the “Services” menu you can install, uninstall, start and stop the two MOSCA-LCM windows services. They are the Onboarding Service and the Webservice. Please refer to the section “NTP Onboarding” and “Webservice” for more information.

In the “Tools” menu you can check the online state of devices, edit some entries in the database, create an export task and start the LiveLog tool of MOSCA-LCM. Please refer the section “The Tools Menu”.

In the “Update” menu you start the “in place update” of MOSCA-LCM. Please refer to the section “In place Update”.

In the “Help” Menu you can show the actual program version, open the documentation and the release notes.

DeviceList

In the DeviceList you have an overview of all devices. With a context menu you can call several functions per device. Please refer to the section “Context Menu” for more information.

Edit Devices section

In this section you can add or edit devices to the device list. You can also assign a command file for the chosen device and set the device to auto-mode.

Edit Files section

In this section you have quick access to several configuration files or folders.

Scheduled Task & Service section

In this section you can create, control and delete the scheduled task for the MOSCA-LCM service. In addition, you can see the state of the scheduled task and the windows services of MOSCA-LCM.

Credentials section

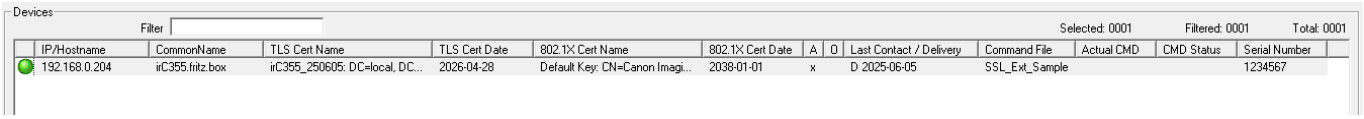
In this section you can edit all needed credentials and passwords.

List Control section

In this section you can change the workflow state of a certificate enrolment for devices in the DeviceList. You can also reload and refresh the DeviceList and the Command File list.

The Device List

Description



Icons

In the first column the actual device state will be displayed. There are 8 different states:

- State 0
  The device is new and has not been read by MOSCA-LCM.
- State 1
  The device has been checked. The found active certificates are all valid and the dates of the certificate expiry are out of the range of the MaxDaysUntilExpiry entry in the command file. In the web interface this state is shown dark green.
- State 2
  The device has been checked. One found active certificates is valid and the other one is not. In the web interface this state is shown light green.
  If “Auto” is checked and a certificate enrollment workflow for the failed certificate type is present, the certificate enrollment process will be started immediately.
- State 3
  All certificates to check are invalid. The date of the certificate expiry is in the range of the MaxDaysUntilExpiry entry in the command file or the certificate issuer is wrong. If “Auto” is checked and a certificate enrollment workflow for the failed certificate types is present, the certificate enrollment process will be started immediately.
- State 4
  The device last contact exceeded the “MaxDaysSinceLastContact” setting in the MOSCA_LCM_Service.ini
- State 5
  MOSCA found invalid certificates and started the workflow for deploying a new certificate. This workflow is now running.
- State 6
  An active certificate is invalid (see yellow dot) and the enrollment process had a final error or “Auto” is not checked.

-  State 7
  The workflow was forced to end by the workflow commands “End” or “ManualInstall”.
-  State 8
  An active certificate is invalid (see state 3, yellow dot) and the enrollment process had multiple errors. The enrollment process will be retried the next days as many times as configured with the “RetryDaysBeforeFinalError” setting in the “MOSCA_LCM_Service.ini”. Please refer to “Base Configuration / MOSCA_LCM_Service / RetryDaysBeforeFinalError” for further information.

Other columns
The columns “Hostname / IP”, “Common Name” and “Notes” are fields that have to be entered manually or imported. The “Notes” column is for your own notes / comments and will not be used by MOSCA in any way.

The column “TLS Cert Name” displays the certificate details of the active TLS/SSL certificate.

The column “TLS Cert Date” displays the TLS/SSL certificate expiry date.

The column “802.1X Cert Name” displays the certificate details of the active 802.1X certificate.

The column “802.1X Cert Date” displays the 802.1X certificate expiry date.

The column “A” for “Auto” shows an “x” if the certificate auto enrollment is active for this device.

The column “O” for “Online” shows an “o” if the device is online (see “Tools” to know how to start an online check).

The column “Last Contact / Delivery” normally shows the last contact to the device. This column is also used for the delivery function. If the entry in this column is set to a future date, MOSCA-LCM defines this date as the date of delivery and will not process the device until this date is reached. If a date in this column is recognized as a delivery date, MOSCA-LCM will mark the date entry with a leading “D”. In addition, delivery dates do not have a time entry.

The column “CommandFile” shows the name of the assigned command file for the enrollment process.

The column “Actual CMD” shows which command of the enrollment process is active by now and the column “CMD Status” shows the status of the active command.

The column “Serial Number” is filled automatically at readout time with the device serial number.

The column “Device Type” is filled automatically at readout time with the device type (model) of the device.

Special Functions

Access the RUI

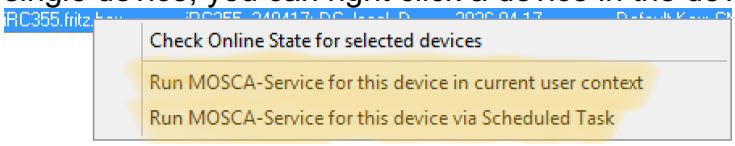
To access a devices Remote User Interface (RUI) in a web browser you can just double click the device in the device list.

Context Menu

When right clicking an entry in the device list a context menu will open up. You will have these options:

Run MOSCA-LCM-Service for a single device

If you have many devices in the device list configured for the auto enrollment of certificates and just want to run the MOSCA-LCM-Service for testing or debugging a single device, you can right click a device in the device list.



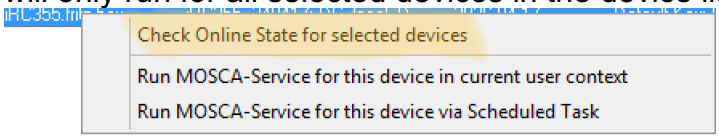
A popup menu will be displayed giving you the option to run the MOSCA-Service for just the selected device, regardless of all other devices in the device list.
With “Run MOSCA-Service for this device in current user context” the MOSCA-Service will be started for the selected device in the current user context. All debugging options are available.
With “Run MOSCA-Service for this device via Scheduled Task” the MOSCA-Service will be invoked via scheduled task for the selected device. The service will run in the context of the scheduled task user.

Please note, that the “Auto” option has to be active for the selected device and that another instance of the MOSCA-LCM-Service has to be ended when using this feature.

Check Online State for selected devices

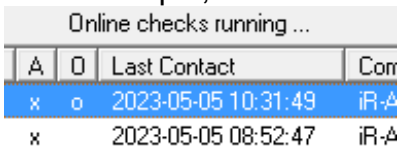
During the setup of MOSCA-LCM and the first tests it might be useful to check if devices are online before activating them for auto-enrollment.

With this option in the context menu can access the online check. The online check will only run for all selected devices in the device list.



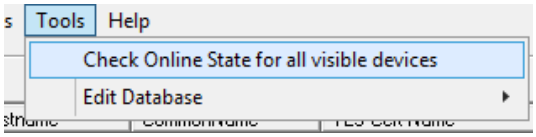
An ICMP packet (ping) will be sent to all selected devices and if a device is online, an ‘o’ will appear in the Online-Column ‘O’. The online check takes a maximum of 10 seconds for all devices. During the check the text ’Online checks running …’ will be displayed above the device list.

In this sample, the first device is online and the second is not:



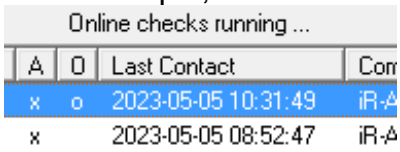
The Tools Menu

Check Online State for all visible devices
During the setup of MOSCA-LCM and the first tests it might be useful to check if devices are online before activating them for auto-enrollment.
In the tools menu you can start an online check for all shown devices in the device list.



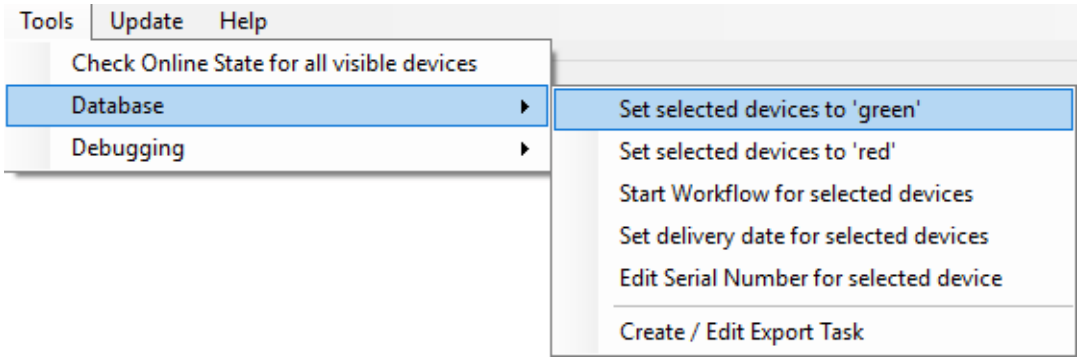
An ICMP packet (ping) will be sent to all devices and if a device is online, an ‘o’ will appear in the Online-Column ‘O’. The online check takes a maximum of 10 seconds for all devices. During the check the text ’Online checks running …’ will be displayed above the device list.

In this sample, the first device is online and the second is not:



Database
In some case it might be useful to write some data directly to the database.

To set a special device status, you can use the functions “Set selected devices to ‘green’” or “Set selected devices to ‘red’” under “Edit Database” in the Tools menu.



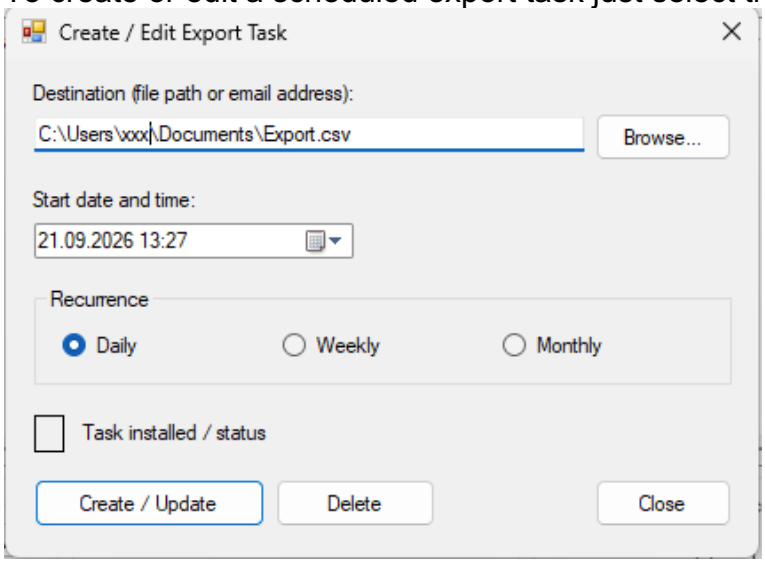
In this sample all selected devices will be set to the state ‘green’.

Start Workflow for all selected devices
To directly invoke an assigned workflow for a device, you can use the third option in this menu called “Start Workflow for all selected devices”.
This command directly sets the device workflow to active state (blue) and sets the first command of the workflow to be the actual command. In addition, the option “Auto” will also be set.

Set delivery date for selected devices
With this function you can set the delivery date for all selected devices.
To get more information about the delivery date function, please refer to “The Device List / Other columns”.

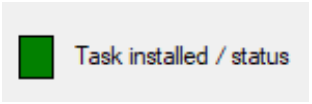
Edit Serial Number for selected device
With this function you can set a device serial number is it could not be retrieved automatically.

Create / Edit Export Task
With this function you can let MOSCA create a scheduled task for automated CSV exports. These exports can be written to disc or sent via email.
(Please refer to the section “Monitoring / Export a CSV-file” to learn more about the export format.)
To create or edit a scheduled export task just select the option from the menu.



Here you can enter or browse to an export file or just enter an email address.
Note: *The Mail Client has to be configured before when using the email option. Please refer to the section “Monitoring / Mail Client” to read more about that.*

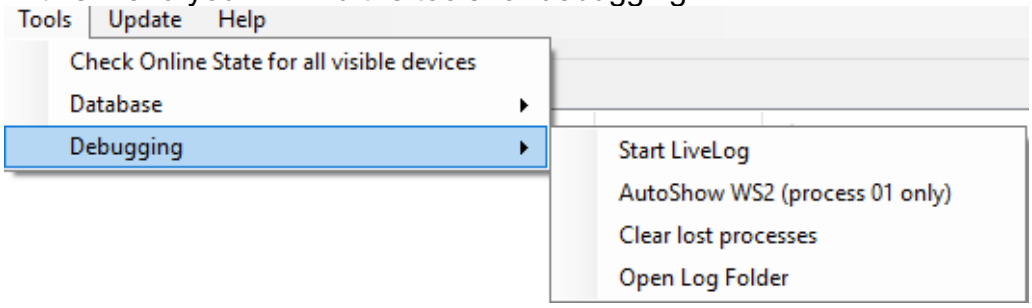
Choose a start date and time and chose Daily, Weekly or Monthly. Then click “Create / Update” to create or update the scheduled task.
After entering the credentials for the scheduled task user, the task will be created.
The “Installed / status” LED will turn green to show that the scheduled task is active.



You can delete the scheduled task by clicking the “Delete” button.

Debugging

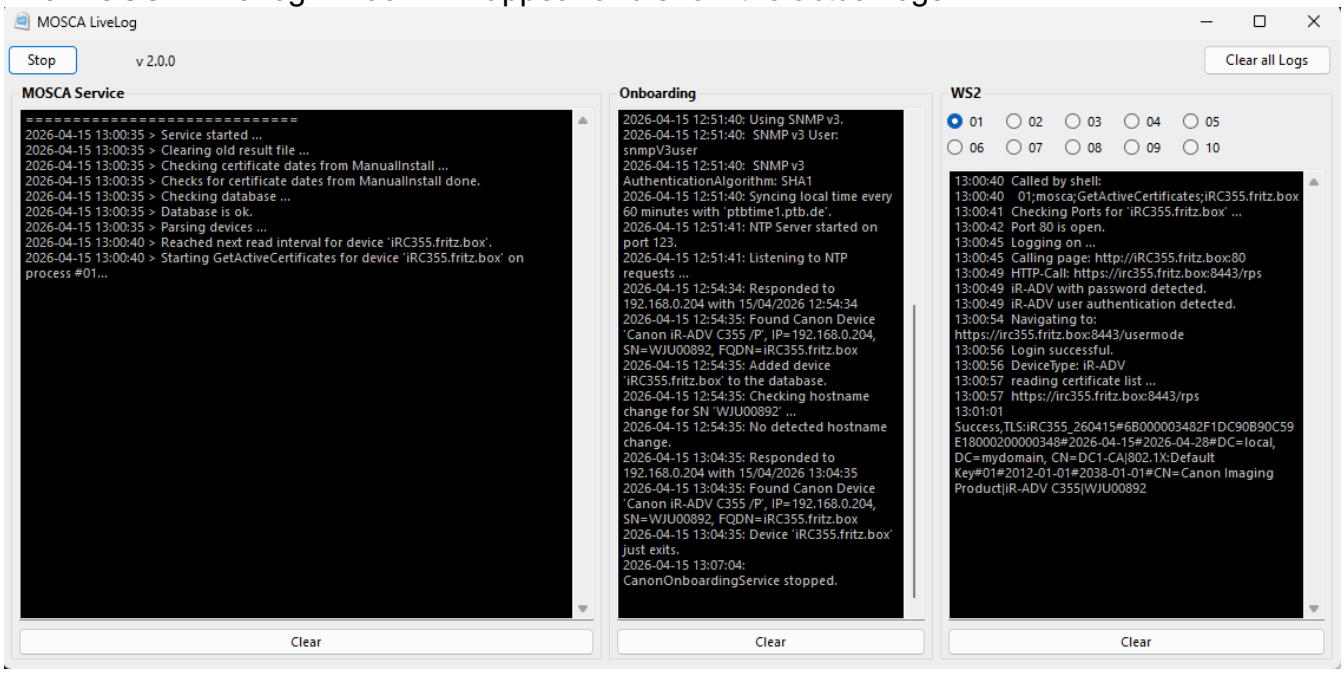
In this menu you will find the tools for debugging.



Start LiveLog

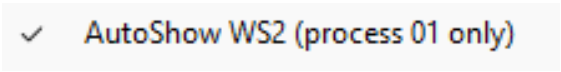
If you run the MOSCA-Service via the scheduled task, the service will not be visible and no log information are shown. This makes it sometimes hard to debug the certificate enrollment process.
Therefore, you can use the Tool “MOSCA LiveLog” which will show various actual log entries. These are the current MOSCA-Service day log, the current Onboarding-Service day log and the selected log of 1 of 10 instances of the WS2.exe.
To start the tool just select it in the Tools menu or start the “MOSCA_LiveLog.exe” in the program directory.

The MOSCA LiveLog window will appear and show the actual logs:



New entries in the log files will immediately be shown in the LiveLog windows. To stop or start the automatic refresh of new log entries press the “Start-Stop” button. To clear the actual logs in the log directory press “Clear” at the bottom of the actual log or “Clear all Logs” to clear all Logs at once. Only the last 1000 lines of each log will be shown.

AutoShow WS2
When the MOSCA-LCM-Service is started in the current user context the first WS2.exe is started in the windows task bar.



With this option tuned on the WS2.exe window will always pop up automatically each time a process is started. Only the 1st process will be shown (“WS2 01”). The option can be disabled by selecting it again or by closing MOSCA-LCM.

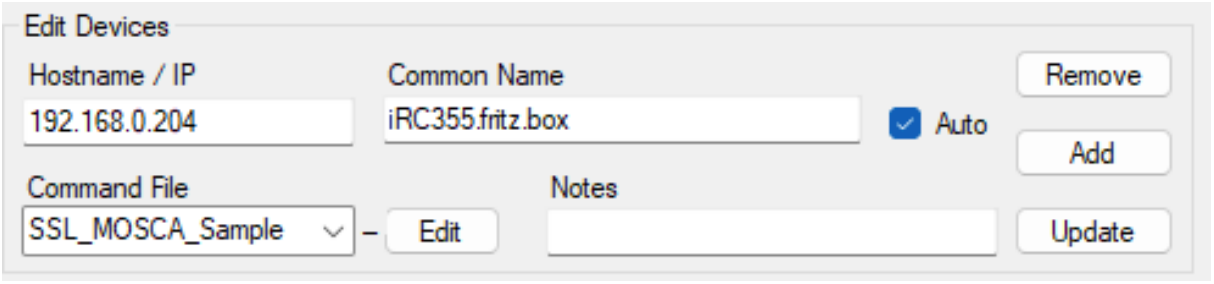
Clear lost processes
If you are in debugging / testing you might exit or kill a process while it is running. Then this process is blocked for some time. To clear all hanging processes please choose “Clear lost processes” before you restart the MOSCA-LCM service again.

Open Log Folder
To open the folder containing all the log files in explorer, just select this option.

Adding/Importing devices

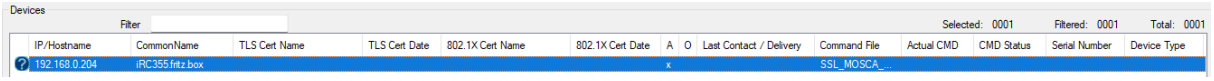
Manually

When you have configured a command file it is time to add devices to MOSCA-LCM. You can type in a “Hostname / IP”, a “Common Name” and optionally a “Note” for your device. Then select a “Command File” from the list and check the “Auto” checkbox.



Then press the “Add” button.

The new device will be added to the device list.



File Import

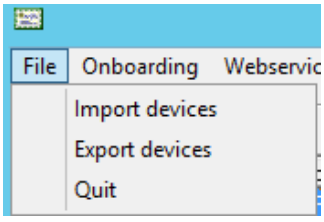
You can also drag and drop a CSV file with one or more devices to the device list to add them. Here is the content of a sample file:

#File for importing devices to MOSCA LCM.
#Add one or more lines in the following format. The first two fields are mandatory.
#IP/Hostname;CommonName;CommandFile;SerialNumber;DeliveryDate;Note
192.168.0.204;iRC355.fritz.box;iR-ADV_SSL_MOSCA

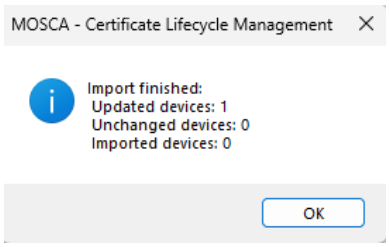
If you provide a command file entry in the import file the “Auto” option will be automatically activated.

If you provide a DeliveryDate entry in the import file, devices will not be processed in MOSCA-LCM before that date. To get more information about the delivery date function, please refer to “The Device List / Other columns”.

The CSV file can also be imported via the program menu:



New devices will be crated, existing devices are updated with new data and same devices will be untouched. The comparison key is always the Common Name.



NTP-Onboarding
If you use the NTP-Onboarding feature, a device will be automatically added to the list as soon as it requests the actual time via the NTPS service included in MOSCA-LCM. Please refer to the section “Base configuration / MOSCA-LCM-Service configuration” and “NTP-Onboarding” to learn more about this feature.

Import from the iW Enterprise Management Console (EMC)
MOSCA-LCM can import devices from the EMC database. Therefore, devices in the EMC must belong to a device group, must have a correct hostname and must be accessible with this hostname (DNS resolution). In addition, you have to define one or more device group names and define a CommandFile name per device group in the MOSCA-LCM-Service configuration.
Please refer to the section “Base configuration / MOSCA-LCM-Service configuration”.

Here is a sample of the section in the MOSCA-LCM-Service configuration for importing all devices from 2 device groups from the EMC:

[AutoEMCImport]

UseAutoEMCImport=1
EMCConnectionString=default

EMCGroup1Name=ADV
EMCGroup1CommandFile=SSL_Ext_Sample

EMCGroup2Name=LBP
EMCGroup2CommandFile=LBP_SSL

The option “UseAutoEMCImport” turns the import function on (1) or off (0).
The option “EMCConnectionString” has to contain the connection string to the EMC database. If you enter the word “default” instead of a connection string, the default data path of the EMC SQLite database is used.

The options “EMCGroup1Name” defines the first group from the EMC to import (“ADV”). All devices in this group will be imported with the command file “SSL_Ext_Sample” from the option “EMCGroup1CommandFile”.

The second group to import is defined with the next 2 lines containing the group name “LBP” and its assigned command file “LBP_SSL”.

You can define as many groups to import as you like by just increasing the number in the group name and the group command file.

The import from the EMC database is done at each service start from the MOSCA-LCM-Service.

Edit devices

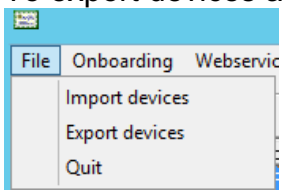
To edit a device entry just select the device in the list. The values will be shown in the “Edit Devices” section. Edit the device values and press the “Update” button.

Remove devices

To remove devices select one or more of them from the device list and hit the “Remove” button.

Export devices

To export devices and device data you can use the program menu:



When exporting devices, a csv file with all device information will be written to disc. Only the actual visible (filtered) devices will be exported. The structure of the file is like the import file structure but with additional information such as the certificate data. When reimporting this export only the base information IP/Hostname, CommonName, CommandFile, SerialNumber, DeliveryDate and Notes will be imported.

Here is a sample export:

IP/Hostname;CommonName;CommandFile;SerialNumber;DeliveryDate;Notes;Status;TLS Cert Name;TLS Cert Date;802.1X Cert Name;802.1X Cert Date;Auto;Online;Device Type
iRC355.fritz.box;iRC355.fritz.box;SSL_Ext_Sample;WJU00892;2026-10-07;;5;Default Key: CN=Canon Imaging Product;2038-01-01;Default Key: CN=Canon Imaging Product;2038-01-01;x;;iR-ADV C355

Here is the same file shown in MS Excel:

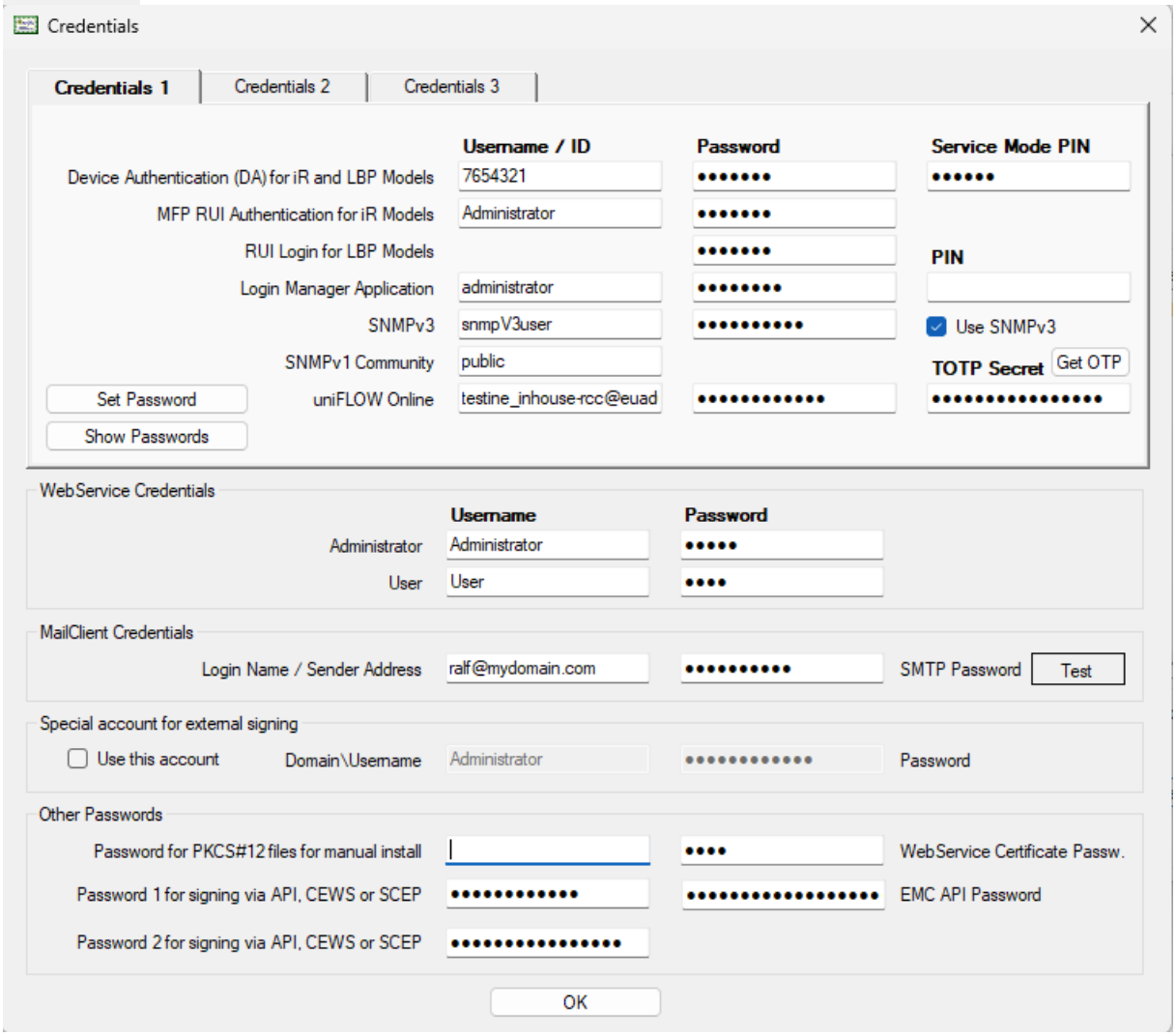
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	IP/Hostname	CommonName	CommandFile	SerialNumber	DeliveryDate	Notes	Status	TLS Cert Name	TLS Cert Date	802.1X Cert Name	802.1X Cert Date	Auto	Online	Device Type
2	iRC355.fritz.box	iRC355.fritz.box	SSL_Ext_Sample	WJU00892	07.10.26		5	Default Key: CN=Canon Im	01.01.38	Default Key: CN=Car	01.01.38	x		iR-ADV C355

Base Configuration

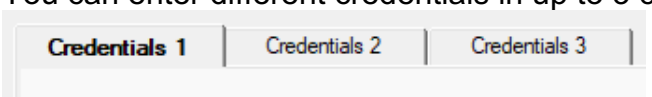
Credentials

To let MOSCA-LCM connect to the Canon devices Remote User Interface (RUI) the software needs to know all credentials to access the RUI of the devices. Press the “Set / Edit” button in the “Credentials” section to open the Credentials configuration.





Enter all known credentials for the different authentication methods of the Canon devices.
You can enter different credentials in up to 3 credential slots.



MOSCA will try them one after the other to gain device access. An empty entry will not be used. MOSCA will remember the successful login parameters and preferably use these for the upcoming logins.

The SNMP section is needed only for the Auto-Onboarding features described later.
The “WebService Credentials” section is for the use of the included webservice described later.

The “Special account for external signing” is only needed, when you do not want to use the (service-) account of the schedule task for the MOSCA service to authenticate against your CA.
Sample: You want to sign certificates against an MS-CA with certreq commands. You want to run the scheduled task of MOSCA-LCM in the local system account, who has no access to the certificate template of the CA. Then you have to enable “Use this account” and type in the credentials of an account with the needed enrol rights.
Note: The password of this account will never be shown as readable text in MOSCA-LCM, even when using “Show Passwords” (see below).

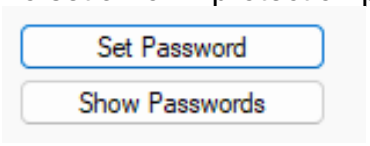
The “MailClient Credentials” section is for the authentication of the MOSCA-LCM Mail Client described later.

The “Other Passwords” section is used for special workflows described in the “Command Files” section and for the WebService Certificate, used by the included webservice described later.

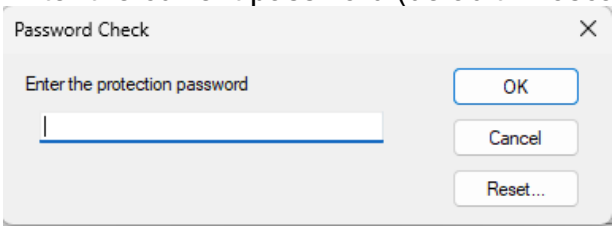
Login Manager Note1: If you want to use the Universal Login Manager with a PIN instead of username and password, you have to leave the username field empty.

Passwords are normally hidden. To show them, press the “Show Passwords” button and enter “mosca” as default password.

To set an own protection password just click “Set Password”.



Enter the current password (default “mosca”)

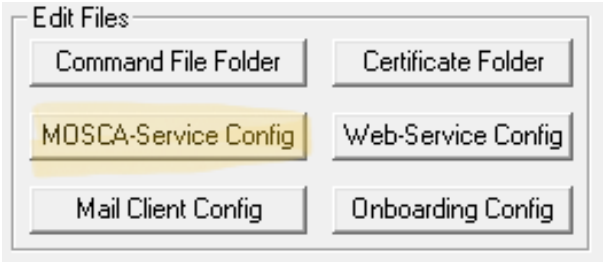


and enter a new protection password.

If you forgot the protection password you can click “Reset …” so set the default password again, but then all current credentials in all 3 slots will be erased.

MOSCA_LCM_Service settings

The MOSCA-LCM-Service configuration is found in the “MOSCA_LCM_Service.ini” file inside the program directory. To open the file just click on the “MOSCA-Service Config” button in the “Edit Files” section. Here you can set some base parameters for the MOSCA-LCM-Service.



[Settings]
MaxDaysSinceLastContact=5
DeleteAfterInactiveDays=0
ReadIntervalDays=1
MaxProcesses=10
ProcessTimeoutMinutes=4
MaxErrors=3
MaxOffineErrors=2
MaxLoginManagerErrors=5
ErrorRetryMinutes=1
RetryDaysBeforeFinalError=2
Logging=1
EnableDayLog=1
ExternalSigningTimeout=10
ExternalCommandTimeout=10
CloseAfterInactiveMinutes=10
AutoEscapeDoubleQuotes=0
CIVTimeout=5
GenerateCSRWithSingleKeys=0

ManualInstallCertificateValidityMonths=24
ManualInstallCheckCertificateDates=0

;Minimized,Normal,Hidden
OpenServiceWindow=Normal
CallWriteSettings=Minimized

[AutoOnboarding]
UseAutoOnboarding=1
OnboardingCommandFile=iR-ADV_SSL_MOSCA
LiveOnboarding=0
DeleteAfterHostnameChange=0
UseIPForHostname=0

[AutoEMCImport]
UseAutoEMCImport=1
EMCConnectionString=default
EMCGroup1Name=Canon
EMCGroup1CommandFile=SSL_Ext_Sample

[EMCOnboarding]
Servername=
APIUser=
APIPASS=
IgnoreErrors=0
DiscoveryTimeout=30

[SANs]
;AppendSANs 1=during signing, 2=resign CSR using EA certificate, 3=only in MOSCA generated CSR
AppendSANs=0
AutoGenerateIPAddress=0
AppendIPAddressFromHostNameField=0
AutoGenerateShortName=1
AutoGenerateFQDN=0
FQDNDomain=mydomain.local
AppendUserPrincipalName=0
UserPrincipalNameSuffix=@mydomain.local
SuppressDeviceSANEntry=0

General settings

MaxDaysSinceLastContact
Sets the timeout in days which a device can be offline until it will be marked yellow-red in the device list.

DeleteAfterInactiveDays
With this parameter you can let MOSCA automatically delete devices that could not be accessed for given period of time. By default, this parameter is set off (0). If you set it to 180 for example, MOSCA will delete devices that could not be contacted for more than 180 days.

ReadIntervalDays
Sets the interval in days after which the devices certificates will be checked again.

MaxProcesses
Sets the number of processes that can be started by MOSCA-LCM-Service in parallel to query devices.

ProcessTimeoutMinutes
Sets the timeout in minutes after a single process has to be finished before it will be forced to close.

MaxErrors
Sets the maximum retries if an error occurs during certificate enrollment process. If the maximum is reached, the device will be marked ‘red’ and no longer processed.

MaxOfflineErrors
Sets the maximum retries if a device is offline. If the maximum is reached the device will be queued for the next run time.

MaxLoginManagerErrors
If a device is running the uniFLOW Login Manager application and the Login Manager is temporary disconnected from uniFLOW, MOSCA cannot login to the

device. If this case MOSCA will directly queue the device for next runtime. The parameter “MaxLoginManagerErrors” sets the maximum of retries for Login Manager login failures. If the maximum is reached, the device will be marked ‘red’ and no longer processed.

ErrorRetryMinutes
Sets the retry time in minutes after the enrollment process got an error.

RetryDaysBeforeFinalError
Sets the amount of retries in days before a repeated error gets a final error. When the internal FinalError counter has not exceeded the RetryDaysBeforeFinalError setting, the device will be marked orange and forced to be retried the next day. If the error still persists and the internal FinalError counter gets equal to the RetryDaysBeforeFinalError setting, the device will finally be marked red and excluded from the further processing.

Logging
Enables the debug log of the service. Each action will be logged into the file “MOSCA_LCM_Service_DebugLog.txt” in the program directory.

EnableDayLog
When logging and daylog are active, a log file with the day name in it will be created for each day logging occurs. After one week the logs will be overwritten.

ExternalSigningTimeout
Timeout in seconds when waiting for the answer of an external signing command (RPC).

ExternalCommandTimeout
Timeout in seconds when waiting for the answer of an external command.

CloseAfterInactiveMinutes
Defines the inactive time of the MOSCA-LCM-Service before the process will end itself in case of a program failure.

AutoEscapeDoubleQuotes
PowerShell commands to sign certificates on an external CA API, might need escaped double quotation marks (“) in the PowerShell command. A “\” character is used for that. This can be done automatically by MOSCA by setting this parameter to “1”.

CIVTimeout
This parameter sets the timeout in seconds (default=5) for the code integrity verification (CIV). On some servers it might be helpful to increase the value if you get CIV errors without no reason.

GenerateCSRWithSingleKeys
When using GenerateCSR to let MOSCA generate the CSR for a certificate in OpenSSL, normally a given private key is used for all requests. If you need device individual private keys for the CSRs, set this parameter to “1”. MOSCA will then generate a device individual private key for the request and store it password

protected as a *.ppk-file (protected private key) with the CSR in the certificate folder. After the conversion to PKCS#12 the signed certificate and the ppk-file will be deleted automatically.

ManualInstallCertificateValidityMonths
Defines the certificate end date to be set in months from now, when running a workflow with the command ManualInstall. Please refer to the sample command file “SSL_Ext_CertOnly_Sample.mlc” in the command file section and the section “How to use MOSCA-LCM with non-supported devices” in the appendix.

ManualInstallCheckCertificateDates
When set to “1” MOSCALCM will check the entry of manually installed certificates against their validity end date and mark devices yellow if they run out of time. Please refer to the sample command file “SSL_Ext_CertOnly_Sample.mlc” in the command file section and the section “How to use MOSCA-LCM with non-supported devices” in the appendix.

OpenServiceWindow
When starting the MOSCA-LCM-Service manually this option defines whether the program windows is visible or not. You can set Minimized, Normal or Hidden.

CallWriteSettings
When starting the MOSCA-LCM-Service manually this option defines whether the single process (WS2.exe) run in debug mode and are visible or not. You can set Minimized, Normal or Hidden.

Onboarding Settings

UseAutoOnboarding
Defines if auto onboarding is used if the Canon Onboarding Service is running. Please refer to the sections “Adding/Importing devices / NTP Onboarding” and “NTP Onboarding”.

OnboardingCommandFile
Defines the default command file to be used when devices are auto onboarded.

LiveOnboarding
When this option is set on (1) the onboarding service will invoke the MOSCA_LCM_Service via the windows scheduled task immediately after a new device is added to the database and check the device certificates. If this option is tuned off (0) the certificate check on the new device will be done at the next normal runtime of the scheduled task.

DeleteAfterHostNameChange
When this onboarding option is on (1) the onboarding service will delete a device from the device list when a new device with the same serial number but a different common name (CN) has been just added. This can be useful if devices are renamed to a different CN due to an internal move to another department. The new device will be added and the old device will be removed.

UseIPForHostname
Normally MOSCA uses the Common Name as hostname. When setting UseIPForHostname on (1) then MOSCA-LCM uses the IP-Address instead of the Common Name when Auto-Onboarding.

EMC Import settings

UseAutoEMCImport
The option “UseAutoEMCImport” turns the import function from the iWEMC database on (1) or off (0). Please refer to the section “Adding/Importing devices / Import from the iW Enterprise Management Console (EMC)”.

EMCConnectionString
If you wat to use the AutoEMCImport function the option “EMCConnectionString” has to contain the connection string to the EMC database. If you enter the word “default” instead of a connection string, the default data path of the EMC SQLite database is used.

EMCGroup1Name and EMCGroup1CommandFile
The options “EMCGroup1Name” defines the first group from the EMC to import (“Canon”). All devices in this group will be imported with the command file “SSL_Ext_Sample” from the option “EMCGroup1CommandFile”.

EMCOnboarding settings

Please refer to the EMCOnboarding section.

SAN settings

AppendSANs
Defines whether and with which method MOSCA-LCM should try to append Subject Alternative Names (SANs) to the Certificate Signing Request (CSR). A value of “1” means “during signing”, a value of “2” means “resign CSR using EA certificate”, a value of “3” means “only in MOSCA generated CSR” and a value of “0” switches the AppendSANs function off. Please refer to the “Requirements / External Signing” section. The value “1” is available for the signing process against a Microsoft Certificate Authority and when signing in MOSCA (self-signed with OpenSSL). The value “2” is only available for the signing process against a Microsoft Certificate Authority. The third value “3” enables the addition of SANs to MOSCA generated CSR’s, e.g. when signing against a none MS CA using an API.
When appending SANs, MOSCA-LCM will always append the CN of the device as DNS Name entry by default.
Note: When generating a CSR and signing this CSR in MOSCA (self-signed with OpenSSL, e.g. for tests) and want to add SANs, please choose value “1”.

AutoGenerateIPAddress
If AppendSANs is used, this parameter defines whether MOSCA is doing a reverse lookup for the Common Name (CN) to get the IP Address to append it as SAN.

AppendIPAddressFromHostNameField
If you are using fixed IP addresses and no DNS-Server, you can append the IP address to the SANs by settings this option to 1. The IP address from the “Hostname / IP” field from the device list will be appended as SAN. Note: The option “AutoGenerateIPAddress” has to be set to 0 to activate this function.

AutoGenerateShortName
If AppendSANs is used, this parameter defines whether MOSCA is taking the first part of the Common Name (if an FQDN) to append it as SAN.

AutoGenerateFQDN
If AppendSANs is used and you are using a CN (not FQDN) as the host name in MOSCA, only the CN will be added to the SANs. If you want to have the FQDN as an additional SAN entry, you can use this function. It generates an FQDN entry from the CN and a fixed domain part (see FQDNDomain).
When using this function, the function AutoGenerateShortName as to be set to “0”.

FQDNDomain
FQDNDomain defines the fixed domain part, that will be added to the CN when using the function AutoGenerateFQDN. The domain has to be entered without a leading dot, e.g. “mydomain.local”.

AppendUserPrincipalName
Depending on your login method for 802.1X you need to append the DNS Name (when the device is logging in with a computer account) or the User Principal Name (when the device is logging in with a user account). If AppendUserProcipalName is set to 1, MOSCA will create a SAN entry for the User Principal Name (UPN) with the Common Name (CN).

Note: The UPN will only be appended as the CN (short name), not as the FQDN. To add the domain to a UPN entry please see the “UserPrincipalNameSuffix” option. This option needs the “AutoGenerateShortName” option to be set to “1” as well.

UserPrincipalNameSuffix
When appending a UPN to the SAN, only the CN (short name) will be used. To add the domain to the UPN you can use this configuration entry and add any text, for example “@mydomain.local”.

SuppressDeviceSANEntry
Some Canon Devices support adding a SAN when generating a signing request (CSR). By default, MOSCA is adding the CN as SAN on these devices. If you have a mixed fleet of devices that support adding a SAN on the device and devices who don’t support adding a SAN on the device, it might be us useful to change the default behaviour. When setting the “SuppressDeviceSANEntry” to “1”, MOSCA will not add a SAN on the device CSR at all, so that the other methods for appending a SAN can be used for all devices in the same way.

WS2 settings

WS2 is doing the communication to the Canon devices. This task is started up to 10 times in parallel and automates MS Edge or Chromium headless browsers for the device communication.
The configuration for WS2 is done in the WS2.ini file inside of the program directory.

It might be necessary to do changes in the [Settings]-section of this file to fit your environment and needs. The shown values here are the default values.

[Settings]
CertificateBasePath=
Logging=1
KeepLogs=0
UseOnlyCanonPorts=0
FirstPageTimeout=15
RebootAfterActivation=1
UseChromiumHeadless=1

CertificateBasePath
Is set automatically by MOSCA-LCM.

Logging
A value of 0 switches the writing of WS2 logs off.

KeepLogs
Normally the logs of the WS2 tasks are overwritten at each run. Setting this value to 1 will keep the logs and append new entries. Keep in mind, that you hard disc might run full with this option.

UseOnlyCanonPorts
Canon devices use port 8000 and 8443 for browser communication. Normal browser calls (80 and 443) are redirected to these ports. If port 80 or 443 are blocked in your network, you can tell MOSCA to use port 8000 and 8443 directly by setting this value to 1.

FirstPageTimeout
This entry defines the timeout in seconds for the first page of the device to be accessed. If you have slow devices or networks you might need to increase this value.

RebootAfterActivation
When a certificate is enrolled the device is rebooted my MOSCA-LCM. If you don’t want to have the devices rebooted after enrol, you can set this value to 0.
Keep in mind, that for 802.1x a reboot is mandatory. So when using 802.1x you need to reboot the devices manually or e.g. with an iWEMC task.

UseCromiumHeadless
For background tasks the WS2 is using Chromium Headless Shell instead of MS Edge to save system sources. If you do not want to use Chromium Headless Shell and have maximum MS Edge compatibility you can set this value to 0.

Command Files

MOSCA LCM Commands (.mlc)*

What are the requirements of a certificate to check? What is the process for a new certificate enrollment? The answers for these questions are all inside the command files. These files have to be customized to fit to your environment and finally each device in the list has to be assigned to a command file for an auto enrollment.

There are **10** sample command files in the command file folder. You can edit a command file by selecting it in the “Command File” drop down control and hitting the “Edit Command File” button. Alternatively, you can press the “Open Command File Folder” button and open the wanted file in the folder with a text editor.

Here is the content of the 1st sample file “SSL_Ext_Sample.mlc”:

```
[SSLCertificateRequirements]
;certificate will only be checked if IssuedBy has a value
IssuedBy=DC1-CA
MaxDaysUntilExpiry=10
CheckCertificateName=1

[SSLCertificateRequest]
SigAlg=SHA256
KeyAlg=RSA2048
O=Canon
OU=SBG
L=Gehrden
S=Nds
C=DE

[SSLWorkflow]
Elements=8
1=RequestCSR
2=SignExternal:SignDC1
3=InstallCertificateFromCSR
4=InstallCACertificate:ca-root.cer
5=ActivateSSLCertificate
6=Wait 3m
7=RemoveUnusedCertificates
8=GetActiveCertificates
```

The first section “SSLCertificateRequirements” defines the certificate requirements of the active SSL device certificate. You can define where the certificate has to be issued from with the entry “IssuedBy” and what are the maximum days until expiry for a certificate with the entry “MaxDaysUntilExpiry”. With the entry “CheckCertName” you can define, whether to check the certificate name against the common name in the certificate (set to “1”), or not to check the certificate name (set to “0”).

In the section “SSLCertificateRequest” you can define your settings for a certificate request on the device. Please refer to the device RUI or the device manual to see which settings the device offers for the signature and key algorithms (SigAlg and KeyAlg).
The correct notation for the signature algorithm is “SHAx” where the x has to be replaced by your algorithm length (e.g. “SHA256”).

The correct notation for the key algorithm is “RSAx” or “ECDSAPx”, depending on whether you want to use RSA or ECDSA (ElipticCurveDigitalSigningAltorithm). The x has to be replaced by the algorithm length (e.g. “RSA2048” or “ECDSAP256”.
Note: Keep in mind, that not all Canon devices support ECDSA.

Please also set Organisation (O), Organisational Unit (OU), Location (L), State (S) and Country (C) as you need it.

The last section defines the process for a new certificate enrollment. You have to provide a list of commands with an incrementing number before each command. Finally, an entry “Elements” with the total number of commands has to be created.

Possible commands are:
RequestCSR[-XXX], SignMOSCA[-XXX], SignExternal[:{SignCommand}][-XXX], InstallCertificateFromCSR[-XXX], ActivateSSLCertificate[-XXX], Activate8021XCertificate[-XXX], ActivateSSLAnd8021XCertificate, ActivateExisting8021XCertificate, Wait xm, RemoveUnusedCertificates, GetActiveCertificates, InstallCACertificate:{CertificateName}, GenerateCSR[-XXX], ConvertToPKCS12[-XXX], ConvertToPKCS12MI[-XXX], ConvertToDER[-XXX], ConvertCSRToDER[-XXX], ManualInstall, InstallPKCS12Certificate[-XXX], RunExternalCommand:{CommandFile}, EMCOnboarding, End

"-xxx" is the optional certificate suffix if SSL and 802.1X certificates should have different names on the same device. "End" is only necessary if GetActiveCertificates or ManualInstall are not the last commands.

All possible commands should be covered by the following samples.

In this first sample file the process is as follows:

1=RequestCSR
A certificate request will be created on the device and transferred to MOSCA-LCM.

2=SignExternal:SignDC1
The request will be signed externally with a shell command against the external CA. The shell command to be called by MOSCA is in a text file called “SignDC1.txt” which is also as a sample file inside command folder. The shell commands will be described later in this chapter under “External Signing Commands”.

3=InstallCertificateFromCSR
This command will install the signed certificate on the device but not activate it.

4=InstallCACertificate:ca-root.cer
This command will install the certificate with the file name “ca-root.cer” of a certificate authority into the trusted root certificate store of the device. The certificate has to be available in the LCMCertificates folder under the MOSCA-LCM program folder.

5=ActivateSSLCertificate
This command will activate the certificate for SSL and activate SSL on the device. Furthermore a reboot of the device will be triggered.

6=Wait 3m
This command instructs MOSCA-LCM to wait for 3 minutes until the process continues. This command is necessary because of the reboot in step 5.

7=RemoveUnusedCertificates
Unused certificates on the device will be removed before new ones are installed. Removed certificates are all certificates which are currently not in use for SSL or 802.1X usage. Also all not signed certificate requests will be removed.

8=GetActiveCertificates
This command lets MOSCA-LCM read all active certificates on the device and check them against the SSLCertifacteRequirements as described before.

There are 8 more sample files provided:

“SSL_MOSCA_Sample.mlc”
This sample is pretty much the same as the first sample file. But the signing will be done with OpenSSL inside of MOSCA. The corresponding command is “SignMOSCA”. In addition, the workflow does not install a CA certificate.

“SSL+8021X_SingleUse_Sample.mlc”
This sample also covers the deployment of an SSL certificate but additional it configures the deployment of an 802.1X certificate. Most sections are equivalent as for the SSL deployment. Only a new section “8021XSettings” covers the configuration of the radius server name and the login name for the 802.1X authentication.
In this command file two different certificates for SSL and 802.1X will be requested and deployed. Each one of them for one single use case. One for SSL and one for 802.1X. With this sample, signing will take place in MOSCA itself.
If the login name is not fix you can use the placeholder {CN} and {FQDN} in the configuration file. MOSCA will then take the data from each device in the device list and use the Common Name (CN) or Full Qualified Domain Name (FQDN) as login name.
[8021XSettings]
RadiusServer=Radius
Login={CN}

“SSL+8021X_Ext_DualUse_Sample.mlc”
This sample is like the last one but only one certificate for SSL and 802.1X will be requested and deployed (dual use). Therefor the command “ActivateSSLAnd8021XCertificate“ is used in the SSLWorkflow to activate the signed certificate for SSL and 802.1X in one stroke. The “8021XWorkflow” section is only needed, if an SSL certificate is just present but not activated for 802.1X yet. The command “ActivateExisting8021XCertificate” is then activating this SSL certificate for 802.1X usage. The signing of the certificate is made against an external CA.

[SSLWorkflow]
Elements=7

1=RequestCSR
2=SignExternal:SignDC1
3=InstallCertificateFromCSR
4=ActivateSSLAnd8021XCertificate
5=Wait 3m
6=RemoveUnusedCertificates
7=GetActiveCertificates

[8021XWorkflow]
Elements=3
1=ActivateExisting8021XCertificate
2=Wait 2m
3=GetActiveCertificates

“WatchOnly_Sample.mlc”
This sample shows a configuration to let MOSCA-LCM just watch your device certificates to get an overview of your fleet. No workflow for the certificate enrollment is defined in the file and therefore MOSCA-LCM will only mark wrong or expired certificates:
[SSLCertificateRequirements]
;certificate will only be checked if IssuedBy has a value
IssuedBy=DC1-CA
MaxDaysUntilExpiry=10

[8021XCertificateRequirements]
IssuedBy=DC1-CA
MaxDaysUntilExpiry=10

“8021X_Ext_Sample.mlc”
This sample is like the first one but only for 802.1X certificate deployment instead of an SSL certificate deployment. In addition this workflow does not deploy a CA certificate.

“SSL_Ext_PKCS12_Sample.mlc”
Some canon devices have problems importing certificates signed from a device certificate request. In these cases, it might be a work around to import the certificates in a PKCS#12 format as a password protected file, containing the private key of the certificate request.
This sample file shows the special workflow for these cases, where the certificate requests are not generated on the device but in MOSCA with OpenSSL. After signing the certificates are converted to PKCS#12 format and then imported to the device with a secret password.

In this sample file the process is as follows:

1=GenerateCSR
A certificate request will be created with OpenSSL inside of MOSCA.

2=SignExternal:SignDC1Base64
The request will be signed externally with a shell command against the external CA. The shell command to be called by MOSCA is in a text file called “SignDC1Base64.txt” which is also as a sample file inside command folder.

The difference of this file to the file “SignDC1.txt” is that the “-binary”-option is removed. This is necessary because MOSCA can only convert the signed certificate to a PKCS#12 format when the signed certificate in in Base64 format.

3=ConvertToPKCS12
This command takes the signed certificate in Base64 format and the private key of the certificate request from OpenSSL and converts them into a password protected certificate in PKCS#12 format. The used password is only known by MOSCA.

4=InstallPKCS12Certificate
This command will install a certificate in PKCS#12 format on the device. The certificate will but not activate it yet. The used password is only known by MOSCA.

5=ActivateSSLCertificate
This command will activate the certificate for SSL and activate SSL on the device. Furthermore a reboot of the device will be triggered.

6=Wait 2m
This command instructs MOSCA-LCM to wait for 2 minutes until the process continues. This command is necessary because of the reboot in step 5.

7=RemoveUnusedCertificates
Unused certificates on the device will be removed before new ones are installed. Removed certificates are all certificates which are currently not in use for SSL or 802.1X usage. Also, all not signed certificate requests will be removed.

8=GetActiveCertificates
This command lets MOSCA-LCM read all active certificates on the device and check them against the SSLCertifacteRequirements as described before.

“SSL_Ext_CertOnly_Sample.mlc”
This special workflow starts like the last one and is also finally generating a PKCS#12 certificate. But after the certificate creation, the process stops and prevents the device from being automatically processed again by disabling “Auto” in the device list. You can use this workflow to let MOSCA-LCM request and sign a certificate against an external CA and save it in PKCS#12 format in the LCMCertifcates folder. From there the certificates could be installed manually on the devices. This process might also work for non-Canon devices.
At the end of the workflow the certificate end date will be written to the database calculated from the MOSCA_LCM_Service.ini entry called “ManualInstallCertificateValidityMonths”. If another entry, called “ManualInstallCheckCertificateDates”, is set to “1” and the name of the workflow contains the phrase “CertOnly”, then MOSCA will check the certificate end date at each service start and mark a device yellow, if its end date reaches the MaxDaysUntilExpiry entry in the workflow config file.

For this workflow the option “Auto” must not be set. The workflow can only be invoked with the function “Start workflow for all selected devices” in the menu “Tools / Edit Database”. Please refer to the section “Tools” and the section “How to use MOSCA-LCM with non-supported devices” in the appendix.

In this sample file the process from element 3 on is as follows:

3=ConvertToPKCS12MI
This command takes the signed certificate in Base64 format and the private key of the certificate request from OpenSSL and converts them into a password protected certificate in PKCS#12 format for manual install (MI). The used password for the manual install has to be set in the Credentials section in MOSCA-LCM under "Password for PKCS#12 files for manual install".

4=Manual Install
This command stops the workflow and disables "Auto" option for the device. The device state will be "7" displayed as purple, which signals a forced workflow end.

"SSL_Ext_API_Sample.mlc"
This sample is like the first one but instead of calling the Shell commands inside of SignDC1.txt file to sign via RPC against a MS certificate authority (CA), the SignExternal command refers to the SignAPI.txt file. This SignApi.txt file contains the shell curl commands to sign the CSR on an external CA via an API. The API is answering with a JSON file normally containing a Base64 encoded certificate (PEM format).
The API in this sample requires the CSR in DER format (binary) but the Canon devices deliver the CSR only in PEM format (Base64). Therefore, the command "ConvertCSRToDER" is used before signing.
To keep the most compatibility to all Canon devices the final certificate format should be DER (binary). To convert the certificate from the API answer in PEM format (Base64) to DER format the command "ConvertToDER" is used in this sample file.

“SSL_Ext_Sample_Embedded.mlc”
Since version 4.3.0 of MOSCA you can set the signing commands directly in the command file instead of calling the external *.txt command file.
This sample show the syntax:

[SSLWorkflow]
Elements=7
1=RequestCSR
2=SignExternal
3=InstallCertificateFromCSR
4=ActivateSSLCertificate
5=Wait 3m
6=RemoveUnusedCertificates
7=GetActiveCertificates

[SSLSignExternal]
SignCommand=certreq -submit -binary -rpc -q -f -config "DC1\DC1-CA" -attrib "CertificateTemplate:Computer802" "{FILENAMECSR}" "{FILENAMECERT}"
PolicyCommand=certreq -policy -rpc -q -f -config "DC1\DC1-CA" "{FILENAMECSR}" "{POLICY.INF}" "{TEMP.CSR}"
RetrieveCommand=certreq -retrieve -binary -rpc -q -f -config "DC1\DC1-CA" "{ID}" "{FILENAMECERT}"

The workflow is the same as we know, except element 2. Here we have no reference to the second external *.txt file “SignDC1.txt” from our first sample. If this file name is missing, MOSCA will search for the signing commands in the command file itself.
So instead of the file reference, we have a section [SSLSignExternal] holding all three parameters from the “SignDC1.txt” file:
SignCommand = line 1 of the *.txt file
PolicyCommand = line 2 of the *.txt file (only for AppendSANs=2)
RetrieveCommand = line 3 of the *.txt file (for approval workflows)

For API signing use the sections [SSLAPICall_1..3] / [SSLAPIAuthenticationCall] (contents as in the SignAPI*.txt files).
For SCEP the section [SSLSCEP] (contents as in the SignDC1_SCEP.txt file).

In 802.1X workflows the prefix is 8021X instead of SSL, e.g. [8021XSignExternal], [8021XAPICall_1], [8021XSCEP].

External Signing Commands using RPC

As described before MOSCA-LCM is using shell commands to sign, resign or retrieve certificates from the CA. Here is a description of the content of the sample file “SignDC1.txt” using a Remote Procedure Call (RPC) connection to the CA.
The first line in this file:

certreq -submit -binary -rpc -q -f -config "DC1\DC1-CA" -f -attrib "CertificateTemplate:Computer802" "{FILENAMECSR}" "{FILENAMECERT}"

- “DC1\DC1-CA” represents the domain name of the server with the CA on it (before the backslash) and the name of the CA (after the backslash).
- “-binary” will enforce a binary certificate file which is most compatible to all Canon devices.
- “-rpc” will enforce the use of the “Remote Procedure Call” interface.
- “-q” runs the command silent without message boxes.
- “-f” forces to overwrite the certificate file.
- The certificate template to be used while signing is named after “CertificateTemplate:”. “{FILENAMECSR}” and „{FILENAMECERT}” are placeholder and must not be changed.

The sample file could also contain a second line for resigning a CSR using an enrollment agent certificate to add subject alternative names (SANs):
certreq -policy -rpc -q -config "DC1\DC1-CA" "{FILENAMECSR}" "{POLICY.INF}" "{TEMP.CSR}"

- “DC1\DC1-CA” represents the domain name of the server with the CA on it (before the backslash) and the name of the CA (after the backslash).
- “{FILENAMECSR}”, „{POLICY.INF}” and „{TEMP.CSR}” are placeholder and must not be changed.

A possible 3rd line in the sample file is for an approval workflow. In an approval workflow the certificate will not be signed directly, but the signing will be requested to an CA operator. If MOSCA is getting the answer form the CA that a certificate needs to be approved before issuing, it will halt the process for the current device and try to retrieve the certificate at the next run time:
certreq -retrieve -binary -rpc -q -f -config "DC1\DC1-CA" "{ID}" "{FILENAMECERT}"

- “DC1\DC1-CA” represents the domain name of the server with the CA on it (before the backslash) and the name of the CA (after the backslash).
- “{ID}” and „{FILENAMECERT}” are placeholder and must not be changed.

External Signing Commands using CEWS

The external signing commands using the Certificate Enrollment WebServices (CEWS) are pretty much the same as the signing commands using RPC. But instead of using the server name and instance name of the CA, a URL for the https call to the CA is used.
Here is a sample command for the first line in the command file according to the first line of the RPC command file:

certreq -submit -binary -q -f -config "https://dc2-ca/DC2-CA_CES_Kerberos/service.svc/CES" -attrib "CertificateTemplate:Computer802" "{FILENAMECSR}" "{FILENAMECERT}"

- “https://dc2-ca/DC2-CA_CES_Kerberos/service.svc/CES” represents the URL to call the CEWS of the CA. Here a Kerberos authentication is used.
- “-binary” will enforce a binary certificate file which is most compatible to all Canon devices.
- “-q” runs the command silent without message boxes.
- “-f” forces to overwrite the certificate file.
- The certificate template to be used while signing is named after “CertificateTemplate:”. “{FILENAMECSR}” and „{FILENAMECERT}” are placeholder and must not be changed.

External Signing Commands using an API

If you have a Certificate Authority (CA) offering an API for signing certificate requests, you can also connect MOSCA to it. The external signing commands using API are a full flexible way for connecting MOSCA to a none Microsoft CA.
To use this functionality the API has to be compatible to “CURL”- or “Invoke-Webrequest”-commands. Up to three of these commands plus an authentication command can be configured to support APIs that need more than one call for signing a CSR, e.g. when submission and retrieval of the certificate need multiple calls.
Note: The name of a command file including the API signing commands needs to include the letters “API” to be processed correctly. Sample: “SignDC1_API.txt”

Sample 1 – SignAPI.txt

Here is a sample command file for a two-step certificate signing against an API using CURL:

[APICall_1]
Command=curl 'https://cert-manager.com/api/ssl/v1/enroll' -i -X POST -H "Content-Type: application/json;charset=utf-8" -H "login: MyAccount" -H "password: {APIPASSWORD}" -H "customerUri: TEST" -d "{\"orgId\":\"9999\",\"subjAltNames\":\"{SANS}\","\certType\":18,\"numberServers\":0,"\serverType\":-1,"\term\":365,"\comments\":\"\","\externalRequester\":\"\",\"customFields\": [],\"csr\":\"{CSR}\"}

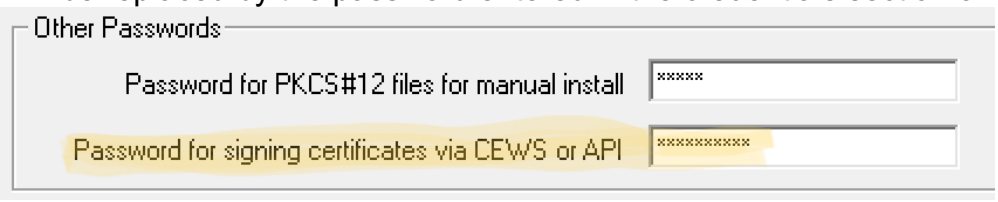
SuccessContains=200 OK
CallAnswer1=RequestID
WaitAfter=2

[APICall_2]
curl 'https://cert-manager.com/api/ssl/v1/collect/{CALLANSWER1}/binary' -i -X GET -H "Content-Type: application/json;charset=utf-8" -H "login: MyAccount" -H "password: {APIPASSWORD}" -H "customerUri: TEST" -o {CERTFILENAME}

SuccessContains=200 OK
WaitAfter=0

In this sample the curl commands will be called in the API call order. So “APICall_1” will be served first, firing the shown curl command. In the command there are 2 placeholder that MOSCA replaces before firing the command:

- {APIPASSWORD}
  Will be replaced by the password entered in the credentials section of MOSCA



- {CSR}
  Will be replaced by the certificate request from the device.
- {SANS}
  Will be replaced by the SANs configured in the MOSCA_LCM_Service.ini.

In the “APICall_1” there are 2 more configured parameters that will be used after the curl answer is received:

- Success contains=200 OK
  Defines a part of the received answer that needs to be included to declare the curl command as successful. Here “200 OK” as the http-answer for a successful call.
- CallAnswer1=RequestID
  Defines a JSON key contained in the curl answer, that will be stored in a variable “CallAnswer1” and can be used in the next API call. In this sample the curl answer contains the RequestID referring the signing request and is needed in the second API call, to retrieve the requested certificate.
- WaitAfter=2
  Defines the number of seconds (here 2) to wait after the curl answer is received before processing the “APICall_2”.

After the 2 seconds pause, the “APICall_2” section will be processed. The second curl command will be fired containing the RequestID from the first call, replaced with the {CALLANSWER1}. In addition, the {APIPASSWORD} and the {CERTFILENAME} will be replaced. The curl answer of the second API call also has to contain the “200 OK” as positive http answer and the received certificate will be stored in MOSCAs certificate folder for further processing. In this sample the curl answer is directly containing the certificate as attached file.

Please note, that in windows curl commands, all double quotation marks that are inside outer double quotation marks, have to be escaped with a “\” character.
-d "{\"orgId\":\"9999\",\"subjAltNames\":\"{SANS}\"}”

Sample 2 – SignAPI_PS.txt

Here is a second sample for a one step certificate signing using the PowerShell command “Invoke-WebRequest”:

[APICall_1]
PSCommand=(Invoke-WebRequest -CertificateThumbprint A7B5C9489FC85586957A2BD5F5AE978D2AE86CEA -Method Post -UseBasicParsing -Uri 'https://api.test.pki.com/certificates/internal/tls/p10/issue' -Headers @{'Accept' = 'application/json'; 'Content-Type' = 'application/json'} -Body '{"structKeyId": "1234-5678", "email": "test@pki.com ", "csr": "{BASE64ENCODE(CSR)}", "validity": {"months": 6}}').RawContent

SuccessContains=200 OK
CertificateTag=Base64Decode(certificate)
CertificateLineFeed=CRLF

The call is a “PSCommand” instead of the “Command” in the first sample. It is telling MOSCA to use Window PowerShell to run the command. The authentication to the API endpoint is done here with a certificate installed on the machine MOSCA is running on. The “CertificateThumbprint” option is referring to that certificate in the local certificate store.
In the command the placeholder “{BASE64ENCODE(CSR)}” is used telling MOSCA to Base64 encode the CSR before replacing it in the command.

Finally, the parameter “CertificateLineFeed” tells MOSCA which characters are used for a line feed in the CSR PEM format. “CRLF”, “CR”, “LF” or any other custom characters like “\n” might be used.

In this sample the API is answering with a JOSN file containing the certificate. To extract the certificate from the API answer, the parameter “CertificateTag” is used. It tells MOSCA to search for a tag (key) in the JSON called “certificate” and extracts the value (data) behind it. Because in this sample the certificate from the API is Base64 encoded the additional parameter “Base64Decode” followed by the JSON key in brackets “(certificate)” is used.

Sample 3 – SignAPI_Venafi_PS.txt

This sample shows the configuration for communicating to a Venafi CA API.

```
[APIAuthenticationCall]
PSCommand=Invoke-WebRequest -UseBasicParsing -Method Post -Uri 'https://cert-
management.system-a.local/vedauth/authorize/oauth' -ContentType 'application/json' -Body
'{"client_id":"CanonMOSCA","username":"Test","password":"{APIPASSWORD}","scope":"certificate:m
anage"}'
SuccessContains=200 OK
APIToken=access_token

[APICall_1]
PSCommand=(Invoke-WebRequest -UseBasicParsing -Method Post -Uri 'https://cert-
management.system-a.local/vedsdk/certificates/request' -ContentType 'application/json' -Headers
@{Authorization = 'Bearer {APITOKEN}'} -Body '{"PolicyDN":"\\VED\\Policy\\Certificates\\Locked
Groups\\CanonMOSCA","PKCS10":"{CSR}","ObjectName":"{FQDN}"}').RawContent
SuccessContains=200 OK
AuthErrorContains=401
CallAnswer1=CertificateDN
WaitAfter=10
CertificateLineFeed=\n

[APICall_2]
PSCommand=(Invoke-WebRequest -UseBasicParsing -Method Post -Uri 'https://cert-
management.system-a.local/vedsdk/certificates/retrieve' -ContentType 'application/json' -Headers
@{Authorization = 'Bearer {APITOKEN}'} -Body
'{"CertificateDN":"{CALLANSWER1}","Format":"Base64","IncludeChain":false,"RootFirstOrder":false}').
RawContent
SuccessContains=200 OK
CertificateTag=Base64Decode(CertificateData)
```

The Venafi API uses OAuth, which means, that a token is needed for signing certificates. Here is the workflow in MOSCA:

- MOSCA is trying to request the signing of the CSR with [APICall_1]. MOSCA has no token yet and will get an answer containing the message “401 unauthorized”.
- Since the parameter “AuthErrorContains” is “401”, MOSCA detects the error as authentication error and will invoke the [APIAuthenticationCall] with the given API password from the credentials section.
- With the parameter “APIToken=access_token” MOSCA extracts the API token from the API answer and stores is for further use.
- Now MOSCA is running the [APICall_1] again with the stored API token and extracts the CertificteDN from the API JSON answer to the CallAnswer1

variable with the parameter “CallAnswer1=CertificateDN”. With the parameter “WaitAfter=10” MOSCA waits 10 seconds after the successful API answer.

- Now the [APICall_2] is processed for retrieving the certificate from the API using the stored CertificateDN as reference. If the request is successful, the certificate will be extracted and base 64 decoded from the JSON answer with the parameter “CertificateTag=Base64Decode(CertificateData)” and finally stored to disc.

List of API parameters

Here is a full list of all parameters when using an API for signing:

- Command — Invoke Shell command.
- PSCommand — Invoke PowerShell command.
- SuccessCointains — To define a successful answer.
- AuthErrorContains — To define an authentication error answer.
- CertificateTag — To extract the certificate from an API JSON answer.
- CertificateLineFeed — Defines which characters to use for a line feed in a pem formatted CSR.
- WaitAfter — Defines the number of seconds to wait after the call.
- CallAnswer1-3 — Defines the key in the JSON answer to extract the value and store it in a variable “CallAnswer1-3“. The variable can be used in the next API call.
- APIToken — To extract the API token from an API JSON answer.

Here is a full list of all functions when using an API for signing:

- Base64Decode() — Used with the parameter “CertificateTag” to Base64 decode the certificate after extraction from the JSON answer.
- WellFormat() — Used with the parameter “CertificateTag” to well format the certificate after extraction from the JSON answer. Line feeds and the “begin” and “end” tags will be added if missing.

Here is a full list of all paceholder when using an API for signing:

- {APIPASSWORD} — Replaced with the API password for signing certificates.
- {APITOKEN} — Replaced with the token from an OAuth authentication which initially used the API password.
- {CSR} — Replaced with the CSR from the device.
- {BASE64ENCODE(CSR)} — Like {CSR}, but the CSR will be Base64 encoded.
- {STRIP(CSR)} — Like {CSR}, but the “BEGIN” and “END” tags will be removed from the CSR.
- {CSRFILENAME} — Replaced with the path & filename of the CSR.
- {CERTFILENAME} — Replaced with the path & filename of the certificate.
- {SANS} — Replaced with the SANs configured in MOSCA.
- {CN} — Replaced with the common name (short name).
- {FQDN} — Replaced with the full qualified domain name or the common name if no FQDN is given.

- {IP} Replaced by the devices IP address from a name resolution against the DNS server.
- {CALLANSWER1} Replaced with the content of the variable “CallAnswer1” of the previous API call.
- {CALLANSWER2} Like before but with “CallAnswer2”.
- {CALLANSWER3} Like before but with “CallAnswer3”.

External Signing using NDES SCEP

When using Microsoft NDES SCEP for signing certificates you have to provide a special command file holding the links and configuration.
Here is a description of the content of the sample file “SignDC1_SCEP.txt”:

[SCEP]
AdminURL=http://dc1/certsrv/mscep_admin/
OperationURL=http://dc1/certsrv/mscep/mscep.dll
DigestAlgorithm=SHA256
ContentEncryptionAlgorithm=3DES
FixedChallenge=
UseWindowsAuthForAdminPage=1
UseWindowsAuthForOperation=1

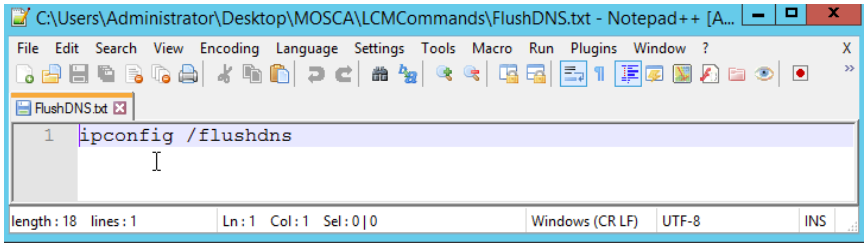
- AdminURL = URL of the NDES admin page that hands out the one-time SCEP challenge password (fetched with an HTTP GET for every signing request).
- OperationURL = URL of the NDES SCEP operational endpoint (mscep.dll) that receives the actual PKIOperation enrollment request.
- DigestAlgorithm = Digest algorithm used to sign the outer SCEP message: SHA256 (default) or SHA1.
- ContentEncryptionAlgorithm = Symmetric algorithm used to encrypt the CSR for the RA: 3DES (default) or AES256CBC.
- FixedChallenge = Optional fixed challenge value instead of reading it from AdminURL. Supports placeholders {APIPASSWORD} and {APIPASSWORD2} (values from [OtherPasswords] APIPass/APIPass2). If set, AdminURL is ignored.
- UseWindowsAuthForAdminPage = 1 sends the service's own Windows credentials when retrieving the challenge password (needed only if that page is configured for Windows Authentication instead of anonymous access). Default 0.
- UseWindowsAuthForOperation = 1 sends the service's own Windows credentials for the mscep.dll calls (GetCACert/PKIOperation) too. Default 0.

Note: The name of a command file including the SCEP signing links needs to include the letters “SCEP” to be processed correctly. Sample: “SignDC1_SCEP.txt”

You will find the detailed requirements of using SCEP with MOSCA in the section “Appendix / SCEP Requirements”.

External Commands

MOSCA-LCM can run external commands during the auto enrollment process. An example for an external command is to flush the DNS cache after a device got a certificate for 802.1X authorization and therefore enters a new network segment and got a new IP address after reboot.
An external command has to be put in a text file which is stored in the command file folder of MOSCA-LCM. In our sample we put the shell command “ipconfig /flushdns” in a file called “FlushDNS.txt”:



In the command file of the certificate enrollment process, we now put the call “RunExternalCommand:” with the name of the command text “FlushDNS” behind the command “ActivateSSLAnd8021XCertificate”:

```
Elements=8
1=RequestCSR
2=SignMOSCA
3=InstallCertificateFromCSR
4=ActivateSSLAnd8021XCertificate
5=RunExternalCommand:FlushDNS
6=Wait 2m
7=RemoveUnusedCertificates
8=GetActiveCertificates
```

Now the DNS cache will be cleared every time a device is restarted after a certificate has been installed.

In another sample, a device web call was needed before a new certificate could be enrolled. The content of the external command was a curl command:
curl.exe --insecure --no-keepalive --user admin:admin --url https://{CN}:8443/mymfp/updatesettings

In this sample a placeholder tag “{CN}” is used. The placeholder is replaced by MOSCA-LCM with the common name of the actual device before the calling the external command.

Possible tags:
{DeviceID} {Status} {Hostname} {CommonName} {Notes} {CertNameTLS} {CertDateTLS}
{CertName802} {CertDate802} {CommandFile} {SerialNumber} {DeviceType} {AppPath}
{CN} is supported as alias for {CommonName}.
Put tags in double quotes, values may contain spaces. Double quotes inside values are removed.
With these tags you may for example invoke a command like this:
powershell.exe -NoProfile -ExecutionPolicy Bypass -File "{AppPath}\LCMCommands\TestScript.ps1" "{CommonName}" "{DeviceID}"
In this call a powershell script is executed with the given parameter CommonName and DeviceID (the ID of the device in the MOSCA database).

EMCOnboarding

MOSCA-LCM can onboard new devices to the Canon iW Enterprise Management Console software (iWEMC). A possible scenario for this function might be this:

- The device onboards itself in MOSCA with the AutoOnboarding feature
- MOSCA enrolls a certificate for 802.1x and reboots the device.
- The device authenticates to the network with 802.1x and need to be added to the iWEMC to get its final configuration

The workflow in the CommandFile could look like this:

```
[8021XWorkflow]
Elements=8
1=RequestCSR
2=SignExternal:SignDC1
3=InstallCertificateFromCSR
4=Activate8021XCertificate
5=Wait 3m
6=RemoveUnusedCertificates
7=EMCOnboarding
8=GetActiveCertificates
```

To use EMCOnboarding, you need to configure the section [EMCOnboarding] in the MOSCA_LCM_Service.ini file.

[EMCOnboarding]
Servername=
APIUser=
APIPASS=
IgnoreErrors=0
DiscoveryTimeout=30

Servername
The servername or IP of the iWEMC server

APIUser
The name of the API user that need to be created in the iWEMC to allow API access.

APIPass
The password for the API user.

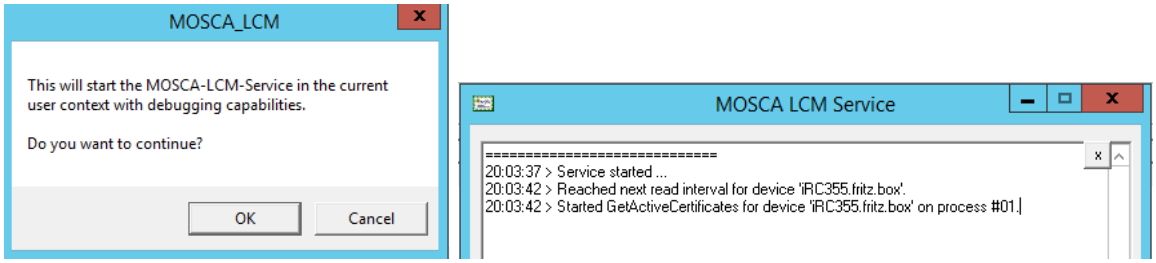
IgnoreErrors
1=Log EMC errors and continue the workflow, 0=treat as workflow error (retry).

DiscoveryTimeout
Sets the seconds to wait until a discovered device appears in the iWEMC.

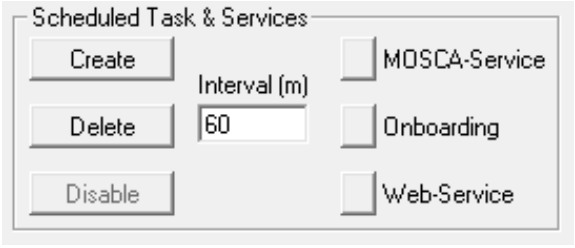
Scheduled Task & Service

In this section you can create and control a scheduled task for the MOSCA-LCM-Service and monitor the task activity. You can start the MOSCA-LCM-Service manually.
You can also monitor the activity of the Canon Onboarding Service (COS) and the MOSCA-LCM Web-Service.

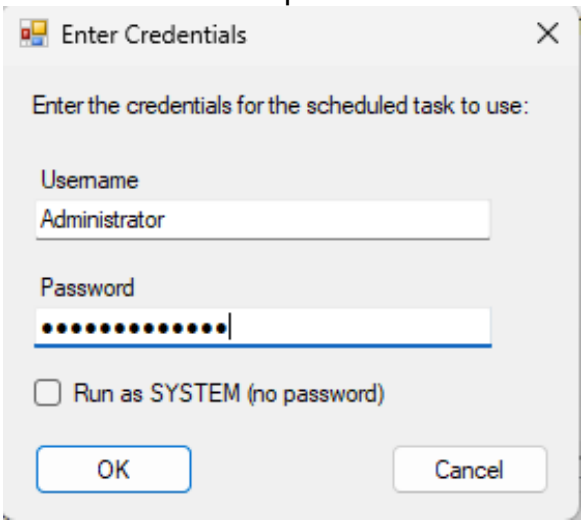
To start the MOSCA-LCM-Service manually just click on the “MOSCA-Service” button.
The MOSCA-LCM-Service will immediately start and run all open tasks to do. After that the service will automatically end. If you have set the MOSCA-LCM-Service window to be shown in manual mode in the “MOSCA-LCM-Service.ini” (refer to the configuration section) then the MOSCA-LCM-Service window will open up and log all activities on screen.



You can also create a scheduled task to run the MOSCA-LCM-Service periodically.



By pressing the “Create” button a scheduled task for the MOSCA-LCM-Service will be created with the entered repeat interval in minutes. You will be prompted for a user account and a password.



The scheduled task is executed under this account. The account needs to have local access rights on the MOSCA-LCM folders and also needs to have access to the CA

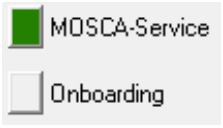
interface. Alternatively, you can use configure the “Special account for external signing”. Please refer to the section “Base configuration / Credentials”.

The used scheduled task account also needs the local policy right “Run as a batch job” (see Requirements).

If you want to use a globally managed service account (gMSA) just enter the Username with the “$” sign at the end and no password will be needed to be entered.

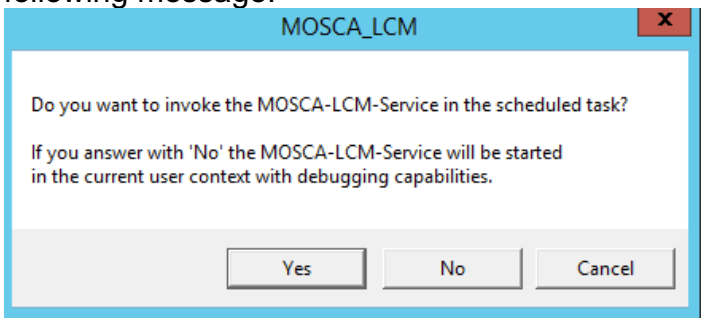
If you want to run the scheduled task under local system account just select “Run as System”. Keep in mind, that “system” has no access to certificate templates or might have no access to CA webpages for API, CEWS or SCEP. Here you must configure the “Special account for external signing”. Please refer to the section “Base configuration / Credentials”.

After the scheduled task is created, the MOSCA-Service button will turn dark green.



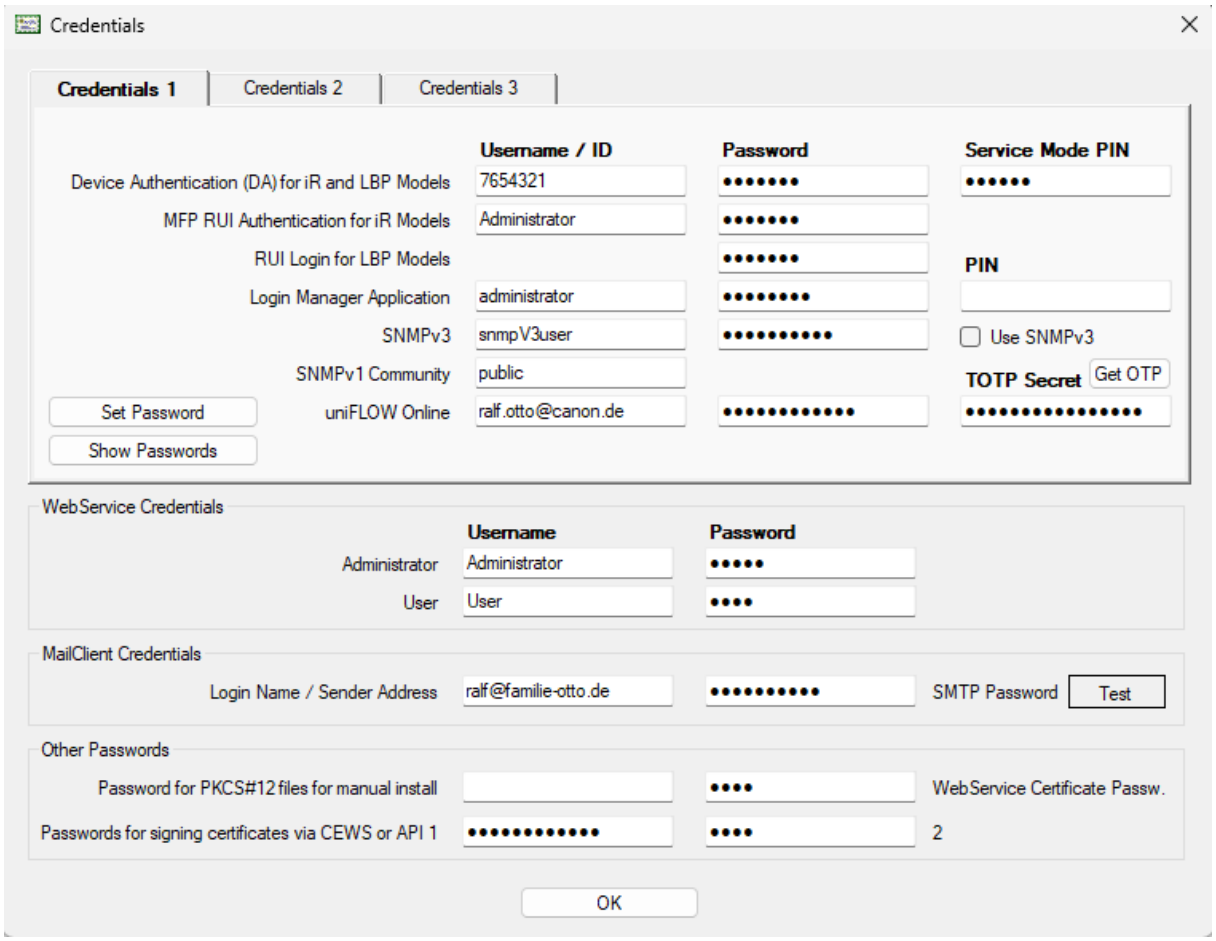
A light green button shows that the MOSCA-LCM-Service is just running in the scheduled task. A red button shown that the scheduled task has been disabled.

If you press the “MOSCA-Service” button and the MOSCA-LCM-Service is installed as a scheduled task and the “MOSCA-Service” button is dark green, you will get the following message:



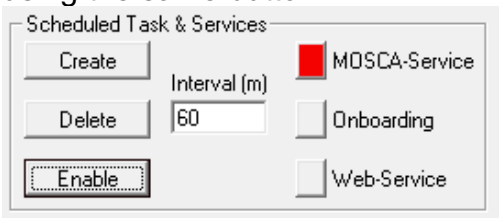
You can now decide, whether to invoke the MOSCA-LCM-Service inside the scheduled task (by pressing “Yes”) or to start the MOSCA-LCM-Service in the current user context (by pressing “No”).

When the MOSCA-LCM-Service is running in a scheduled task and the “MOSCA-Service” button is light green you can press the “MOSCA-Service” button to stop the actual running scheduled task:

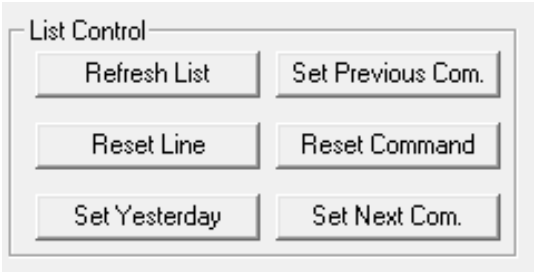


To delete the scheduled task, press the “Delete” button. The scheduled task will be deleted and the “LCM-Service” button will turn grey again.

To disable the scheduled task, press the “Disable” button. The scheduled task will be disabled and the “MOSCA-Service” button will turn red. The text on the button will turn to “Enable” to give you the possibility to reenable the scheduled task again by using the same button.



List Control



By pressing the “Refresh” button MOSCA-LCM will reload all devices and states from the database. Furthermore the “Command File” drop down will be reloading all files in the command file folder.

When in testing or debugging it might be useful to reset the actual workflow that is running on a device in the list. To do so just select the device and press the “Reset Line” button. The workflow will be reset and the device state will turn to unknown.

Normally MOSCA-LCM just checks a device once a day. If you want to force a device check, select a device from the list and press the “Set Yesterday” button. This sets the day of the last check to yesterday so that MOSCA-LCM has to check the device today again.

When in testing or debugging it might be useful to reset the error counter of the failed command of a complete workflow. To do so just select the device and press the “Reset Command” button. The actual command will be reset and the device state will turn to yellow. The command will be repeated the next time the MOSCA-LCM service is running.

The buttons “Set Previous Com.” and “Set Next Com.” are doing the same as “Reset Command” but they will reactivate the previous or the next command in the command list of the workflow.

NTP Onboarding

Description

With this feature new devices can be added automatically to the MOSCA-LCM database. Therefor MOSCA-LCM uses the Canon Onboarding Service (COS) which is in easy words a time server (NTP server). The process works as follows:

- MOSCA-LCM installs the Canon Onboarding Service (COS).
- Every Canon device has to have the Hostname or IP address of the server where MOSCA-LCM is running on, in its time server entry (NTPS) in the network configuration.
- When a device boots up or after a configurable period of time, it requests the actual time from the COS.
- COS answers with the correct time and will also check, whether the device is just known or not. If it is not known, COS requests the hostname, the domain,

the serial number and the model name from the device via SNMP (v1 or v3). If the device is a canon device it will be added as new device to MOSCA-LCM.

- The MOSCA-LCM-Service starts at the next run time with the defined process for new devices and missing certificates will be automatically requested and deployed.
  As an option the MOSCA-LCM-Service can also be invoked immediately after a new device has been added to the database (Refer to LiveOnboarding).

When using the NTP-Onboarding feature, incoming UDP port 123 has to be opened in the firewall on the server MOSCA-LCM is running on. Other NTP services on the same server, like Windows Server's "Windows Time" service have to be stopped and set to disabled state. It is recommended, to unregister the "Windows Time" service with the command "w32tm /unregister". Furthermore, SNMP v1 or v3 have to be configured on the Canon devices.
The time server entry on the Canon devices may also be configured and deployed via DHCP.
When the "Windows Time" service is disabled, the server would not get the actual time itself anymore. Therefor COS can also synchronize the local time of the server it is running on. The time server address (TimeServerAddress) and the interval (SyncTimeMinutes) have to be configured if you want to use this feature of COS. See "Configure Canon Onboarding Service" to get a full description of these two options.

Configure Canon Onboarding Service

Before using the Canon Onboarding Service, the file “CanonOnboardingService.ini” in the directory “COS” has to be edited. This can be done by simply pressing the button “Onboarding Config” in then “Edit Files” section.



[CanonOnboardingService Config File]
Version=1.3

[Settings]
DebugLog=1
EnableDayLog=1
SNMPType=v1
SNMPCommunity=public
SNMPv1Fallback=0
SNMPv3AuthenticationAlgorithm=SHA1
SNMPRetrySeconds=10
SNMPv3User=
SNMPv3Pass=
TimeServerAddress=ptbtime1.ptb.de
SyncTimeMinutes=60

With DebugLog set to 1 (default) you can enable the debug logging of COS. A file “DebugLog.txt” will be created in the Logs folder when COS is starting.
If EnableDayLog is set to 1 (default) COS will create a log file for each day of the week. Then the files will be overwritten after one week.

Select the SNMPType you want to use. Options are “v1” or “v3”.
If you want to use “v1”, the setting “SNMPCommunity” has to be set also.
If you want to use “v3”, the settings “SNMPv3AuthenticationAlgorithm” has to be set to fit your on the device configured authentication algorithm. “SHA1”, “SHA256” or SHA512” can be choosen.
In addition, “SNMPv3User” and “SNMPv3Pass” have to be set, but this must not be done in the “CanonOnboardingService.ini”. These settings have to be set with the “Set / Edit” button in the “Credentials” section of the GUI to encrypt the password.





If the Canon Onboading Service is not able to query the device information after a device reboot, the SNMP interface of the device might not be ready fast enough. To increase the wait time, you may change the parameter “SNMPRetrySeconds”.

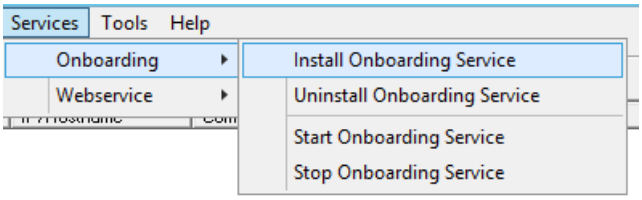
Setting the parameter “SNMPv1Fallback” to 1 activates the fallback to SNMPv1 if the use of SNMPv3 is configured and the SNMPv3 access fails.

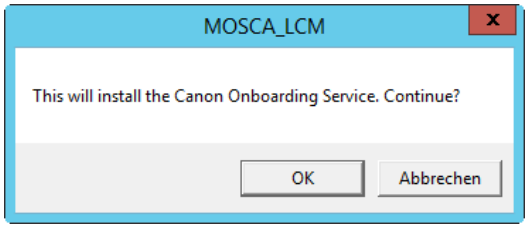
If you want COS to synchronize the local time with a time server you have to set the parameter “TimeServerAddress” with the address of your local or external time server.

The parameter “SyncTimeMinutes” sets the interval in minutes in which the time is requested from the time server.

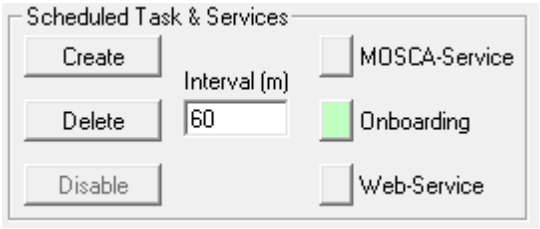
Install Canon Onboarding Service

To install COS, MOSCA-LCM needs to be started with the option “Run as administrator”. Then just select “Install Onboarding Service” from the program menu.





The COS will be installed and started immedetely. The running service will be displayed with the green “Onboarding” button in the section “Scheduled Task & Service”.



The service is also visible in the windows service manager. It will run in the “local system” context and its start type is set to “Automatic”.

You can control (start, stop) the COS from the service manager or also from the Onboarding menu in MOSCA-LCM. A stopped COS will be displayed with a red “Onboarding” button in the section “Scheduled Task & Services”.

Uninstall Canon Onboarding Service

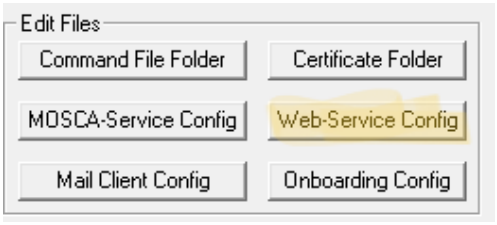
To uninstall the COS, just select the option “Unistall Onboarding service” from the MOSCA-LCM Onboarding menu.
The “Onboarding” button in the section “Scheduled Task & Service” will turn to grey.

Webservice

MOSCA-LCM comes with a web service component to view and control the status of the device certificates with a browser. MOSCA-LCM includes its own web server based on .net Core's Kestrel and needs no MS Internet Information Server.

Configuration of the Webservice

To configure the Webservice press the "Web-Service Config" button in the "Edit Files" section.



In the opening "appsettings.json" file you have to set the wanted ports for http and https connection. Please ensure, that the ports you enter are not used by any other application or service (e.g. the MS Internet Information Server).

```
"Ports": {
  "HTTPPort": 80,
  "HTTPSPort": 443
}
```

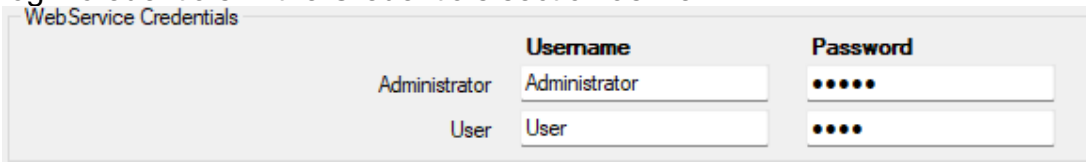
For an SSL connection to the webservice a self-signed certificate is included in MOSCA-LCM. If you want to use your own certificate, you have to provide an SSL server certificate in PKCS#12 (*.pfx) format in the webservice directory of MOSCA-LCM. You then have to configure the name of the certificate file in the "appsettings.json" file as well. If you want to force the use of SSL connections to the webservice you can set the 'FORCEHTTPS' option to '1'

```
"Security": {
  "ForceHTTPS": 0,
  "CertificateName": "MOSCA-LCM.pfx"
}
```

The password for the certificate can be set in the Credentials section of MOSCA-LCM in the field "WebService Certificate Passw.". Please refer to "Base Configuration / Credentials".



If you want to secure the webpage from anonymous access you can provide two login credentials in the Credentials section as well:

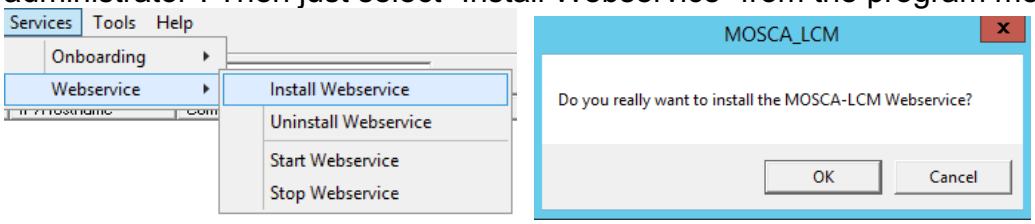


With the role “User” you can access the webpage with no possibility of adding or editing data. With role “Administrator” you have full access to the webpage. You will be able to add or edit data on the webpage.

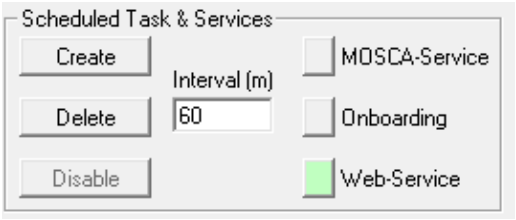
Each time you access the MOSCA-LCM webpage the password will be requested. The session will kept open until you close the browser or reload the whole page.

Install Webservice

To install the Webservice, MOSCA-LCM needs to be started with the option “Run as administrator”. Then just select “Install Webservice” from the program menu.



The Webservice will be installed and started immedetely. The running service will be displayed with the green “Web-Service” button in the section “Scheduled Task & Services”.

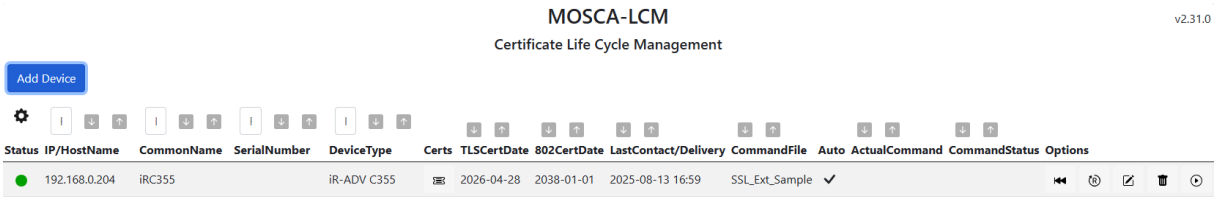


Uninstall Webservice

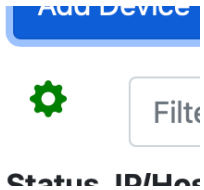
To uninstall the Webservice, just select the option “Unistall Webservice” from the MOSCA-LCM Webservice menu.
The “Webservice” button in the section “Scheduled Task & Services” will turn to grey.

Use the Webservice

To browse to the Webservice page just open a browser and type in the IP or hostname of the server MOSCA-LCM is running on. If you are not using the standard ports (80 and 443) you have to add them to your address entry.
Example when using port 81: https://192.168.0.125:81
On the server itself you may just enter ‘localhost’.



In the top left of the device list, you can see a service gear in one of 4 different colors showing the state of the MOSCA-LCM-Service on the MOSCA-LCM host.



The 4 different colors present the following states:

- dark green: The service is installed and ready.
- light green: The service is just running.
- red: The service is installed but disabled.
- black: The service is not installed, or the state is unknown.

These states are the same as the states of the MOSCA-Service “LED” shown in the MOSCA-LCM GUI.

With the Filter boxes and the sort buttons you can filter and sort the device list.



When you are logged in as administrator, you can add and edit devices and start the LCM-Sercvice for a single device:

Add new devices to the list by pressing the “Add Device” button.



The 5 buttons in each device line have the following functions:



‘Rewind’ sets the last readout time to yesterday. This is the same function as with the button “Set Yesterday” in the MOSCA-LCM program GUI.

‘Reset’ resets the error counter. This is the same function as with the button “Reset Line” in the MOSCA-LCM program GUI.

‘Edit’ lets you edit a device line entry.

‘Delete’ deletes the entry from the device list.

‘Start’ invokes the scheduled task for MOSCA-LCM-Service on the MOSCA-LCM host just for the selected device. This is the same function as the one from the device context menu in the MOSCA-LCM GUI (“Run MOSCA-Service for this device via Scheduled Task”).

The status of a devices certificate is displayed in the status column according to the status column in the MOSCA-LCM program GUI.



Monitoring

To monitor the certificate states and the results of the certificate enrollments, MOSCA-LCM offers four different methods. They are using a SysLog server, sending status e-mails, using the Web-API or exporting to a CSV-file.
If you look for debugging capabilities, please refer to the section “LiveLog”.

SysLog

As the first monitoring method MOSCA-LCM can log events to a SysLog service. The configuration for the SysLog setup is also found in the “MOSCA_LCM_Service.ini” in the section “[SysLog]”.

```
[SysLog]
SysLogServer=192.168.0.125
SysLogPort=514

;Facility Numbers, possible numbes are:
;0 - Kernel messages, 1 - User-level messages, 2 - Mail System, 3 - System daemons,
;4 - Security/authorization messages, 5 - Messages generated internally by syslogd,
;6 - Line printer subsystem, 7 - Network news subsystem, 8 - UUCP subsystem, 9 - Clock daemon,
;10 - Security/authorization messages, 11 - FTP daemon, 12 - NTP subsystem, 13 - Log audit,
;14 - Log alert, 15 - Clock daemon

FacilityNo=7

;Warning Level:
;Enter a Warning level for each message. Leave empty for no message.
;1 - Alert: action must be taken immediately"
;2 - Critical: critical conditions"
;3 - Error: error conditions"
;4 - Warning: warning conditions"
;5 - Notice: normal but significant condition"
;6 - Informational: informational messages"

StartStopService=6
AddedDevice=6
StartedEnrollment=6
SuccessfulEnrollment=6
EnrollmentError=3
ExceededMaxDaysSinceLastContact=4
ServiceError=2
```

SysLogServer
Enter the Name or the IP-Address of the server running the SysLog service.

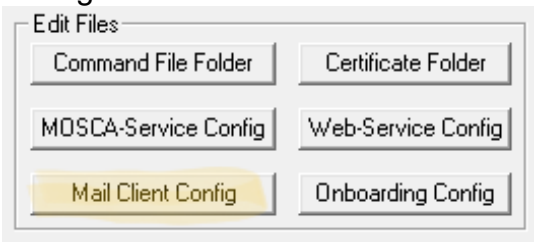
SysLogPort
Enter the Port number for the outgoing messages via UDP (default 514).

FacilityNo
Enter the facility number under which MOSCA-LCM is sending log data to the SysLog service.

StartStopService to ServiceError
Enter a SysLog warning level between 1 and 6 for each event, that MOSCA-LCM should send to the SysLog service. An empty event will not be sent to the SysLog service.

Mail Client

The second method for monitoring events in MOSCA-LCM is using a Mail Client for sending status e-mails. The Mail Client is configured in the “MailClient.ini” file in the program directory. The file can be accessed by pressing the button “Mail Client Config” in the “Edit Files” section.



[MOSCA-LCM MailClient.ini]
Version=1.0.0

[SMTPServer]
Address=smtp.strato.de
Port=587
UseSSL=1

[Mail]
EnableErrorMails=1
EnableWarningMails=1
Recipients=ralf.otto@canon.de
SenderMail=mosca@canon.de

;possible variables are:
;WarningLevel,ErrorMessage,HostName,CommonName,SerialNumber

Subject={WarningLevel} on device {CommonName}
Body={WarningLevel} occured in MOSCA-LCM for the following device:/nCN: {CommonName}/nSN: {SerialNumber}/n/nError: {ErrorMessage}

[Credentials]
UserName=mosca@canon.de
Password=49877CD9A957A4D98F937E281AB7DA901A8431F378

The section “SMTPServer” defines the hostname or IP address of the SMTP server, the port to be used and the use of SSL for encrypting the mail server connection.

The section “Mail” defines if mails for a special warning level should be send or not. If “EnableErrorMails” is set to 1, an e-mail will be sent, every time an error during the certificate check or certificate enrollment occurs. Error e-mails will only be sent, if a device got a final error and is turned to the state “red”.

If “EnableWarningMails” is set to 1, a warning e-mail will be sent if a device exceeds the maximum number of days without contact to MOSCA-LCM. The device will then be marked grey, and the warning e-mail will be sent.

The entry “Recipients” defines the recipients for the e-mails. Multiple recipients can be entered comma separated.

If you need a special sender address you can set in in the “SenderMail” entry. If you leave this field empty, the entry from the “Login Name / Sender Address” in the credentials section will be used (Please see below).

The entry “Subject” defines the subject of the e-mail. You have several variables that can be combined with own text to create a subject line.

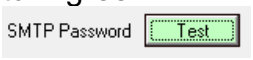
The entry “Body” defines the body of the e-mail. You have several variables that can be combined with own text to create an e-mail body. If you want wo insert a line feed in your body text, please use the characters “/n” in the body configuration entry.

Finally, the “Login Name / Sender Address” and “SMTP Password” (if needed) to authenticate against the SMTP-Server have to be set in the “Credentials” section. This must not be done in the “MailClient.ini” but with the “Set / Edit” button in the “Credentials” section of the GUI.

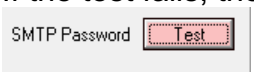




Here you will also find a “Test” button to test the Mail Client configuration and SMTP authentication. If the test is positive, a test mail will be sent. The “Test” button will turn green.

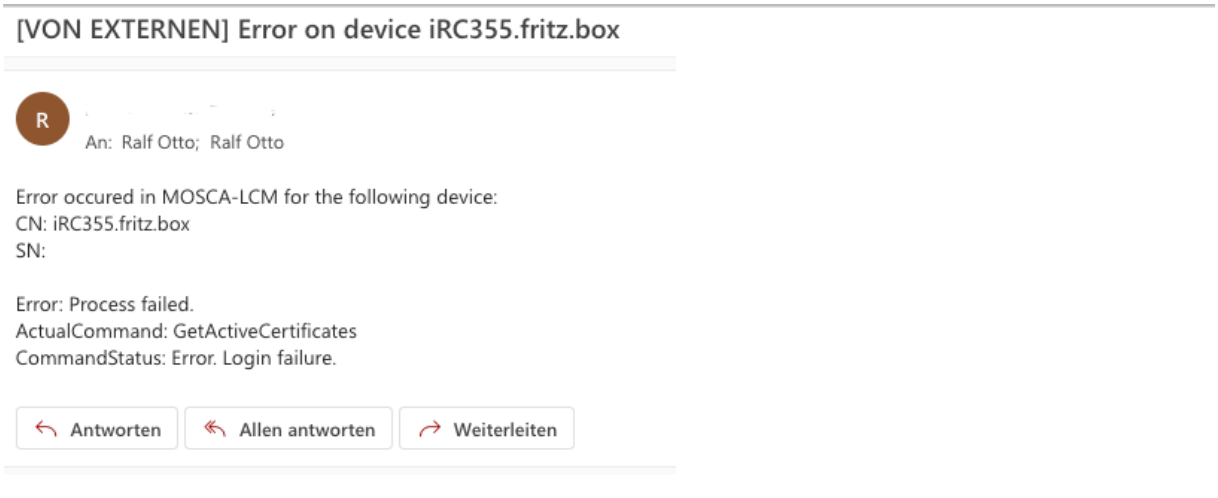


If the test fails, the button will turn red.

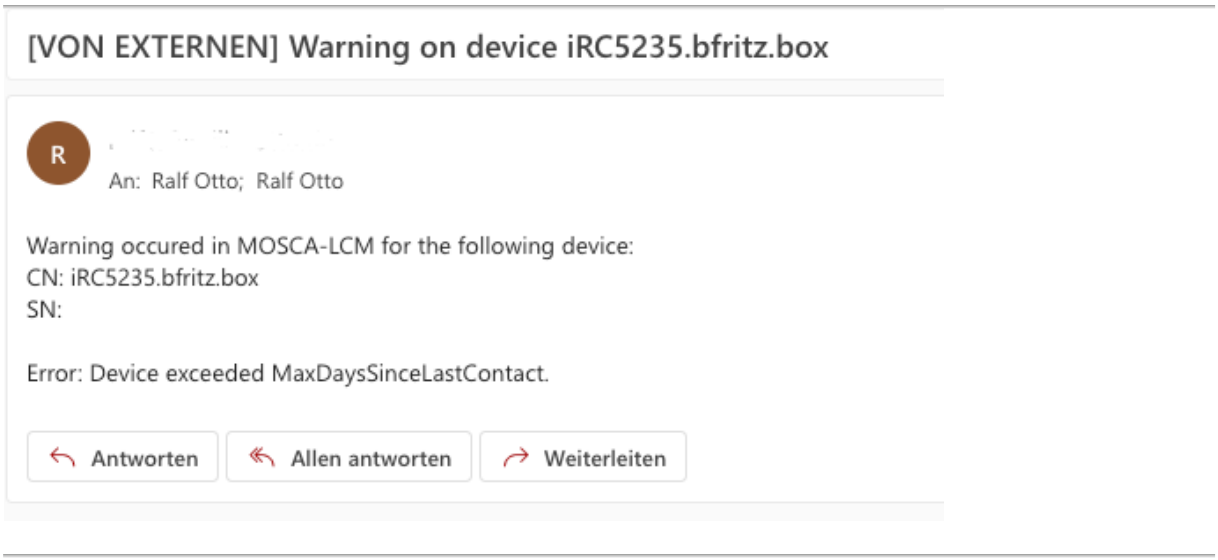


Here are two sample e-mails from MOSCA-LCM configured like above.

Error e-mail after a login failure at the device RUI due to a wrong password:



Warning e-mail because the device was offline for too many days:

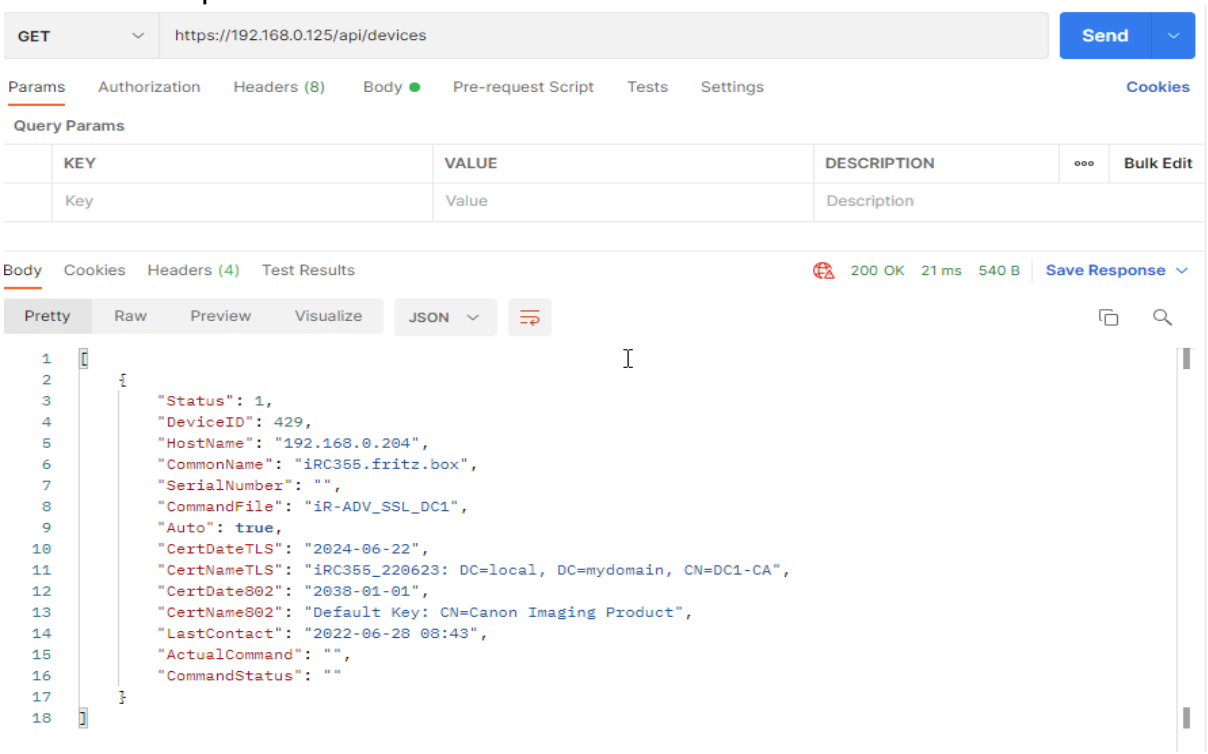


Web-API

The third method to monitor all devices in MOSCA-LCM, is to periodically query the Web-API from MOSCA-LCM and analyze the results.
Querying the device list is not password protected. Therefore, a simple API call via HTTP or HTTPS is possible, depending on the configuration in MOSCA-LCM (refer to section Web-Service).

API-Call

You can call the API with the HTTP(S)-Get call: https://hostname/api/devices
“hostname” is the placeholder for the IP address, or the hostname of the server MOSCA-LCM is running on.
Here is a sample answer from a Postman test:



The result will be a device list in JSON format, containing all known device information. In this sample only one device is available in MOSCA-LCM.

JSON Answer

Status Column

In the first field the actual device state will be displayed. This field gives you information about the certificate state and the enrollment process state. There are 8 different states:

- State 0 (or null)
  The device is new and has not been read by MOSCA-LCM.
- State 1
  The device has been checked. The found active certificates are all valid and the dates of the certificate expiry are out of the range of the

MaxDaysUntilExpiry entry in the command file. In the web interface this state is shown dark green.

-  State 2
  The device has been checked. One found active certificates is valid and the other one is not. In the web interface this state is shown light green.
  If “Auto” is checked and a certificate enrollment workflow for the failed certificate type is present, the certificate enrollment process will be started immediately.

-  State 3
  All certificates to check are invalid. The date of the certificate expiry is in the range of the MaxDaysUntilExpiry entry in the command file or the certificate issuer is wrong. If “Auto” is checked and a certificate enrollment workflow for the failed certificate types is present, the certificate enrollment process will be started immediately.

-  State 4
  The device last contact exceeded the “MaxDaysSinceLastContact” setting in the “MOSCA_LCM_Service.ini”.

-  State 5
  MOSCA found invalid certificates and started the workflow for deploying a new certificate. This workflow is now running.

-  State 6
  An active certificate is invalid (see state 3, yellow dot) and the enrollment process had a final error or “Auto” is not checked.

-  State 7
  The workflow was forced to end by the workflow commands “End” or “ManualInstall”.

-  State 8
  An active certificate is invalid (see state 3, yellow dot) and the enrollment process had an error. The enrollment process will be retried the next days as many times as configured with the “RetryDaysBeforeFinalError” setting in the “MOSCA_LCM_Service.ini”. Please refer to “Base Configuration / MOSCA_LCM_Service / RetryDaysBeforeFinalError” for further information.

Other columns

The field “DeviceID” contains the MOSCA-LCM internal ID of the device.
The fields “HostName”, “CommonName” and “Serial Number” displays the devices host name, common name and serial number, if entered.
The field “CommandFile” shows the name of the assigned command file for the auto enrollment process.
The field “Auto” is true if the certificate auto enrollment is active for this device.

The field “TLS Cert Name” displays the certificate details of the active TLS/SSL certificate. The field “TLS Cert Date” displays the TLS/SSL certificate expiry date.
The field “802.1X Cert Name” displays the certificate details of the active 802.1X certificate. The field “802.1X Cert Date” displays the 802.1X certificate expiry date.

The field “Last Contact” shows the last contact to the device.
The field “Actual CMD” shows which command of the enrollment process is active by now and the field “CMD Status” shows the status of the active command.

Export a CSV-file

The fourth method to monitor all devices in MOSCA-LCM, is to periodically export all device lines to a CSV-file and analyze the results. The exported file can also be sent via email.
The export can be done with a shell command on MOSCA_LCM.exe itself. So, it is possible to run the export periodically with a scheduled task, for example.

In the Tools / Database menu there is an option to auto create such a scheduled task. Please refer to the section “The Device List / The Tools Menu”.

Shell call
You can run the export by calling
MOSCA_LCM.exe -export

With this call MOSCA-LCM will create an export file with the name “Export.csv” in the MOSCA-LCM program directory.

If you want a different path and file name you can add your destination to the shell command:
MOSCA_LCM.exe -export “c:\Administrator\Desktop\MyName.csv”

If the Mail Client is configured in MOSCA you can also export and send the file automated via email. Just add an email address instead of a file name:
MOSCA_LCM.exe -export “test@mydomain.local”

Here is a sample export file containing 2 devices:

Status;DeviceID;IP/Hostname;CommonName;CommandFile;SerialNumber;Status;TLS Cert Status;DeviceID;IP/Hostname;CommonName;CommandFile;SerialNumber;TLS Cert Name;TLS Cert Date;802.1X Cert Name;802.1X Cert Date;Auto;Last Contact;Actual Command;CMD Status;Device Type;Notes

5;896;192.168.0.204;iRC355.fritz.box;SSL_Ext_Sample;;Default Key: CN=Canon Imaging Product;2038-01-01;Default Key: CN=Canon Imaging Product;2038-01-01;x;2023-12-12 16:13:00;RequestCSR;in progress;

5;897;192.168.0.205;iRC5235.fritz.box;SSL_Ext_Sample;;;;;;x;;GetActiveCertificates;in progress;

Status Column

In the first field the actual device state will be displayed. This field gives you information about the certificate state and the enrollment process state. There are 8 different states:

-  State 0 (or null)
  The device is new and has not been read by MOSCA-LCM.
-  State 1
  The device has been checked. The found active certificates are all valid and the dates of the certificate expiry are out of the range of the MaxDaysUntilExpiry entry in the command file. In the web interface this state is shown dark green.
-  State 2
  The device has been checked. One found active certificates is valid and the other one is not. In the web interface this state is shown light green.
  If “Auto” is checked and a certificate enrollment workflow for the failed certificate type is present, the certificate enrollment process will be started immediately.
-  State 3
  All certificates to check are invalid. The date of the certificate expiry is in the range of the MaxDaysUntilExpiry entry in the command file or the certificate issuer is wrong. If “Auto” is checked and a certificate enrollment workflow for the failed certificate types is present, the certificate enrollment process will be started immediately.
-  State 4
  The device last contact exceeded the “MaxDaysSinceLastContact” setting in the “MOSCA_LCM_Service.ini”.
-  State 5
  MOSCA found invalid certificates and started the workflow for deploying a new certificate. This workflow is now running.
-  State 6
  An active certificate is invalid (see yellow dot) and the enrollment process had a final error or “Auto” is not checked.
-  State 7
  The workflow was forced to end by the workflow commands “End” or “ManualInstall”.
-  State 8
  An active certificate is invalid (see state 3, yellow dot) and the enrollment process had an error. The enrollment process will be retried the next days as many times as configured with the “RetryDaysBeforeFinalError” setting in the “MOSCA_LCM_Service.ini”. Please refer to “Base Configuration / MOSCA_LCM_Service / RetryDaysBeforeFinalError” for further information.

Other columns

The field “DeviceID” contains the MOSCA-LCM internal ID of the device.
The fields “HostName”, “CommonName”, “Serial Number”, “Device Type” and “Notes” displays the devices host name, common name, serial number, device type and notes if entered.

The field “CommandFile” shows the name of the assigned command file for the auto enrollment process.
The field “Auto” is true if the certificate auto enrollment is active for this device.

The field “TLS Cert Name” displays the certificate details of the active TLS/SSL certificate. The field “TLS Cert Date” displays the TLS/SSL certificate expiry date.
The field “802.1X Cert Name” displays the certificate details of the active 802.1X certificate. The field “802.1X Cert Date” displays the 802.1X certificate expiry date.

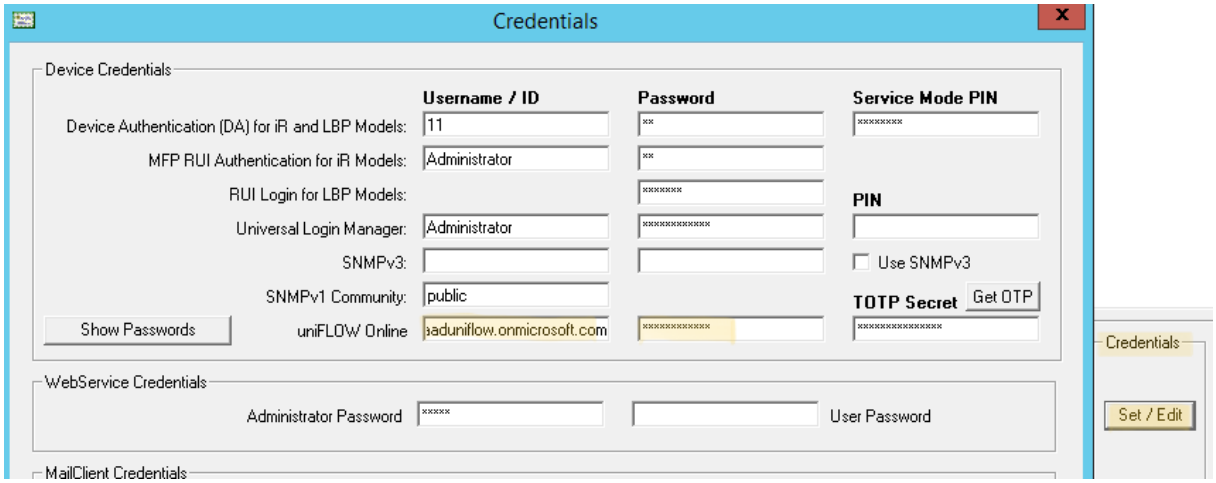
The field “Last Contact” shows the last contact to the device.
The field “Actual CMD” shows which command of the enrollment process is active by now and the field “CMD Status” shows the status of the active command.

Devices in uniFLOW Online

If you have devices, that are attached to a uniFLOW Online tenant, you have to gain access for MOSCA to uniFLOW Online to log on to the devices.

Account without MFA

If you have an account **without** multi factor authentication (MFA) you can just enter the username and password for a device manager account in the “Credentials” section by pressing “Set / Edit”.

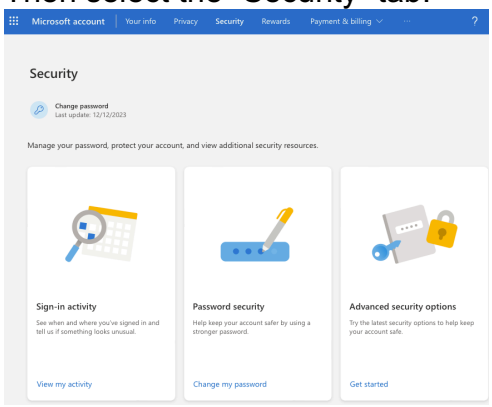


Existing Account with MFA

If you want to use an account **with** MFA, you also have to enter username and password, but in this case MOSCA additionally needs the TOTP (Time based One Time Password) Secret. MOSCA can then generate a code, if a multi factor authentication is requested by the device login.

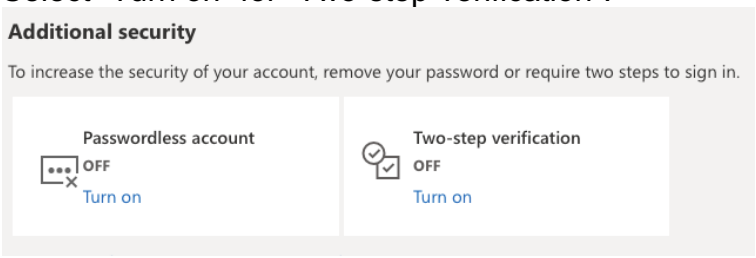
The TOTP secret can be accessed when creating or editing the multi factor authentication for an account. Here is a sample for activating MFA for an MS online account.

Login to your MS online account and choose “My Microsoft account”.
Then select the “Security” tab.

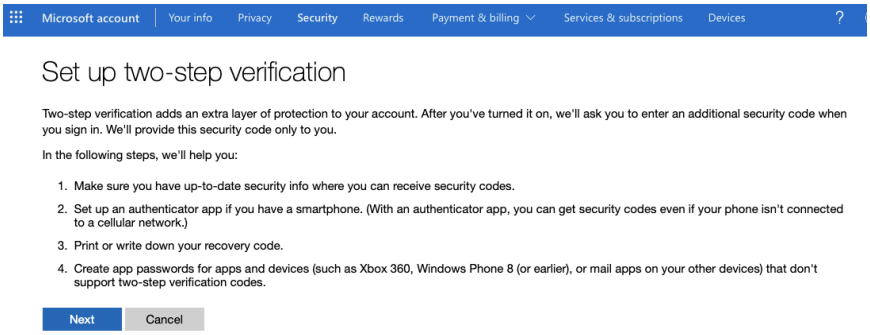


Choose “Advances security options”.

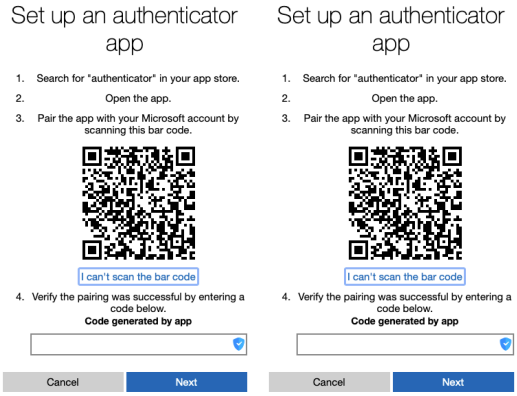
Select “Turn on” for “Two-step-verification”:



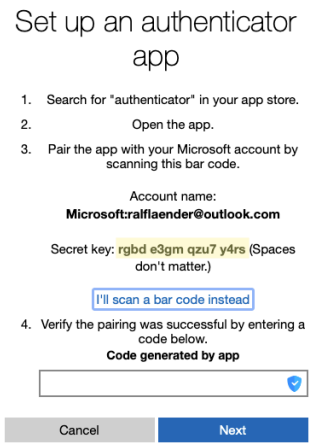
Click “Next”



For the second factor choose “An app” and press on “set up a different Authenticator app”.



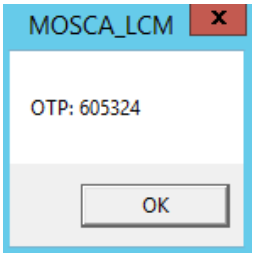
When the barcode is shown, you can use it with your common authenticator app, but you also need to choose “I can’t scan the bar code” to show the TOPT secret key behind the bar code.



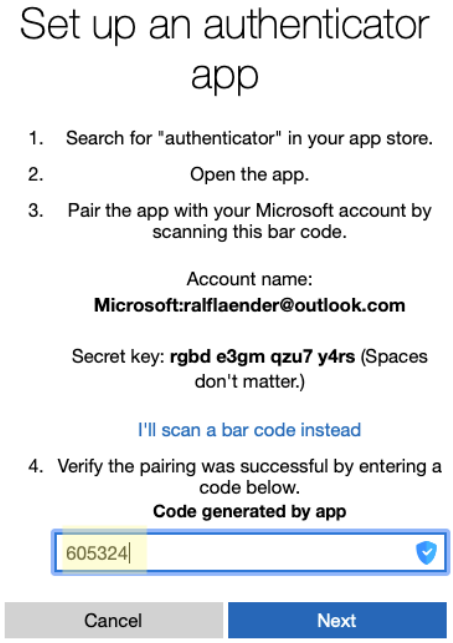
Copy the “Secret key” and enter it without the spaces in the “TOTP Secret” field in the credentials form in MOSCA-LCM.



You can then click the “Get OTP” button to get an OTP generated by MOSCA.



Enter this code to finish the MFA set-up.



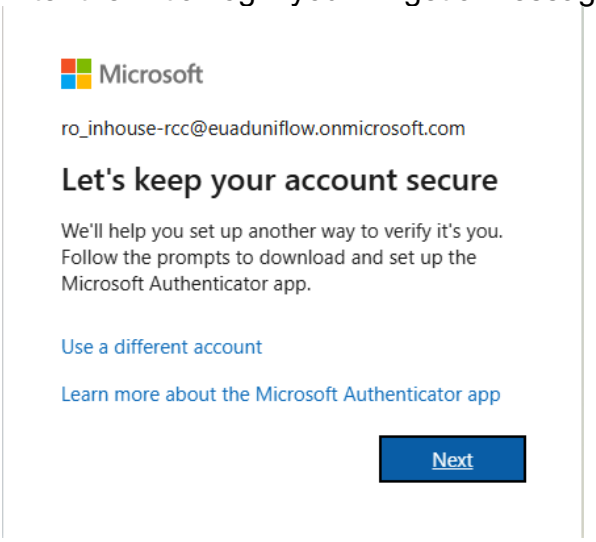
The account used, has to be set as device manager in uniFLOW Online.

MOSCA-LCM will then be able to generate one time passwords (OTPs) and login to any devices connected to this uniFLOW Online tenant.

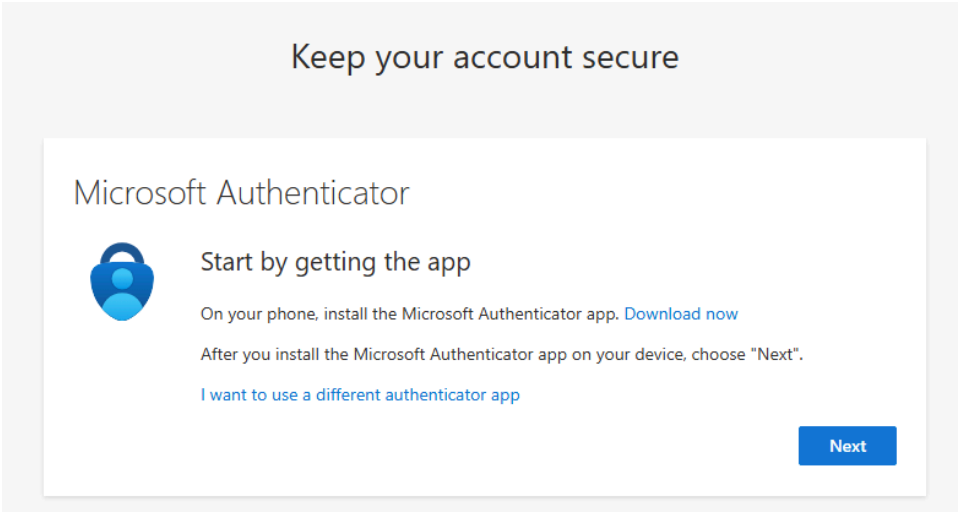
First login to a new account enabling MFA

Here is a sample of login on to a new account and enabling MFA.

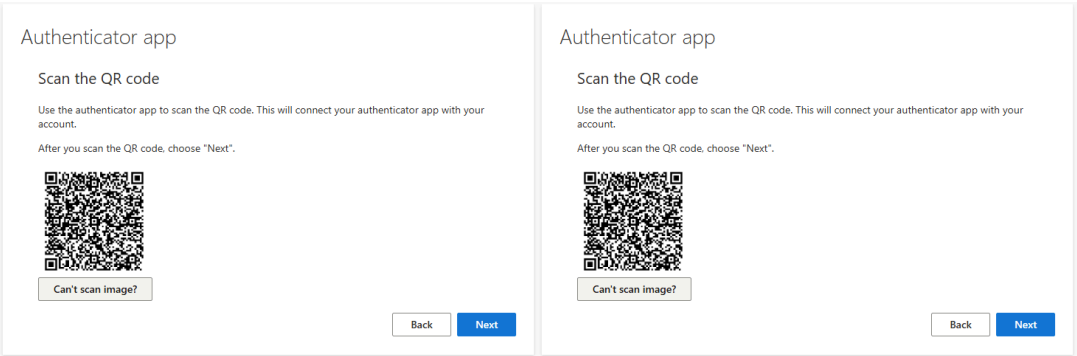
After the initial login you will get a message like this:



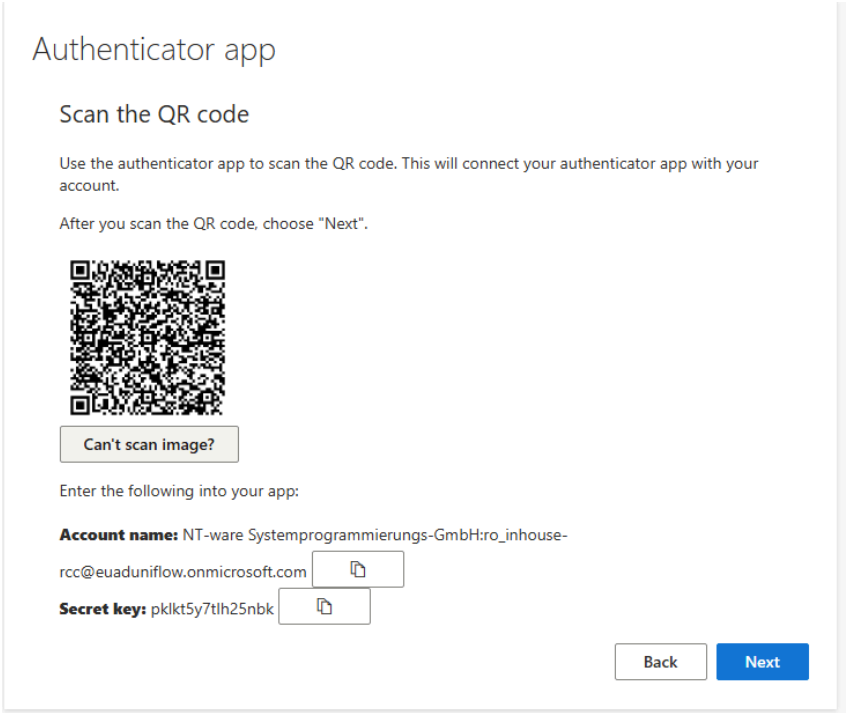
Click “Next”.



Click “I want to use a different authenticator app”.



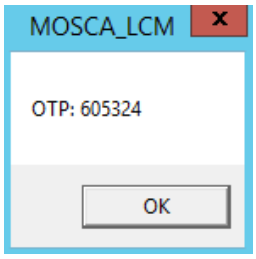
Click “Next” and the click “Can’t scan image?”



Copy the “Secret key” and enter it without the spaces in the “TOTP Secret” field in the credentials form in MOSCA-LCM.



You can then click the “Get OTP” button to get an OTP generated by MOSCA.



Enter this code to finish the MFA set-up.

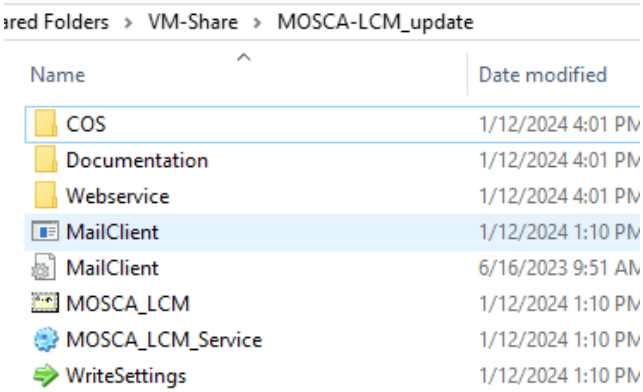
Update MOSCA-LCM

If you want to update the software, you need an update package called “vX_X_X_MOSCA-LCM_update.zip”.

Manual Update

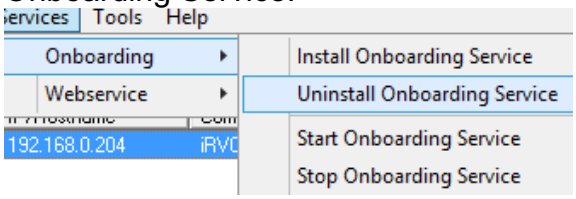
This update is recommended if you are updating a version prior to 2.21.0 or if you want to update single components of MOSCA-LCM. Otherwise, you should use the “In place Update” (see next chapter).

Unzip the package to a single folder:

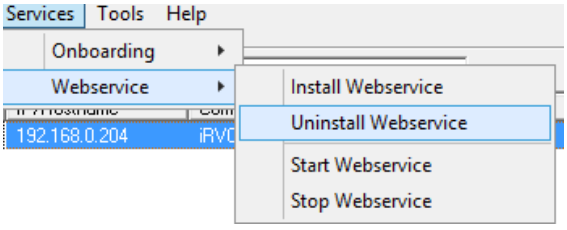


Then uninstall any of the services from the service menu in MOSCA-LCM.

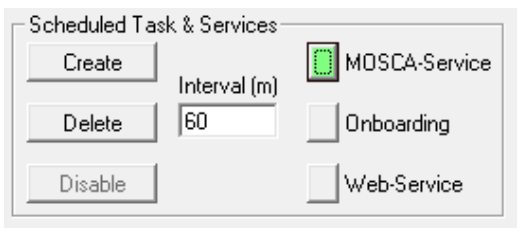
Onboarding Service.

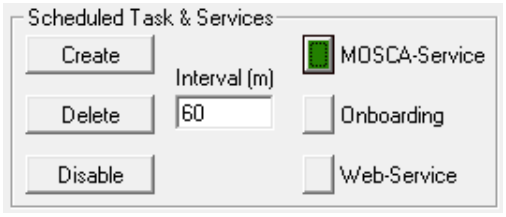


Webservice

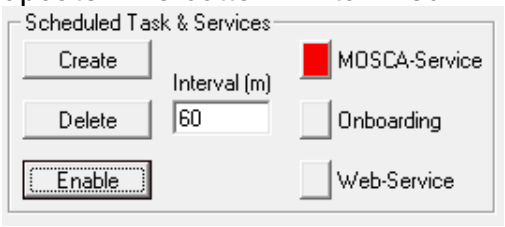


If the MOSCA-LCM-Service is just running (light green MOSCA-Service) click on it to stop the MOSCA-LCM-Service.

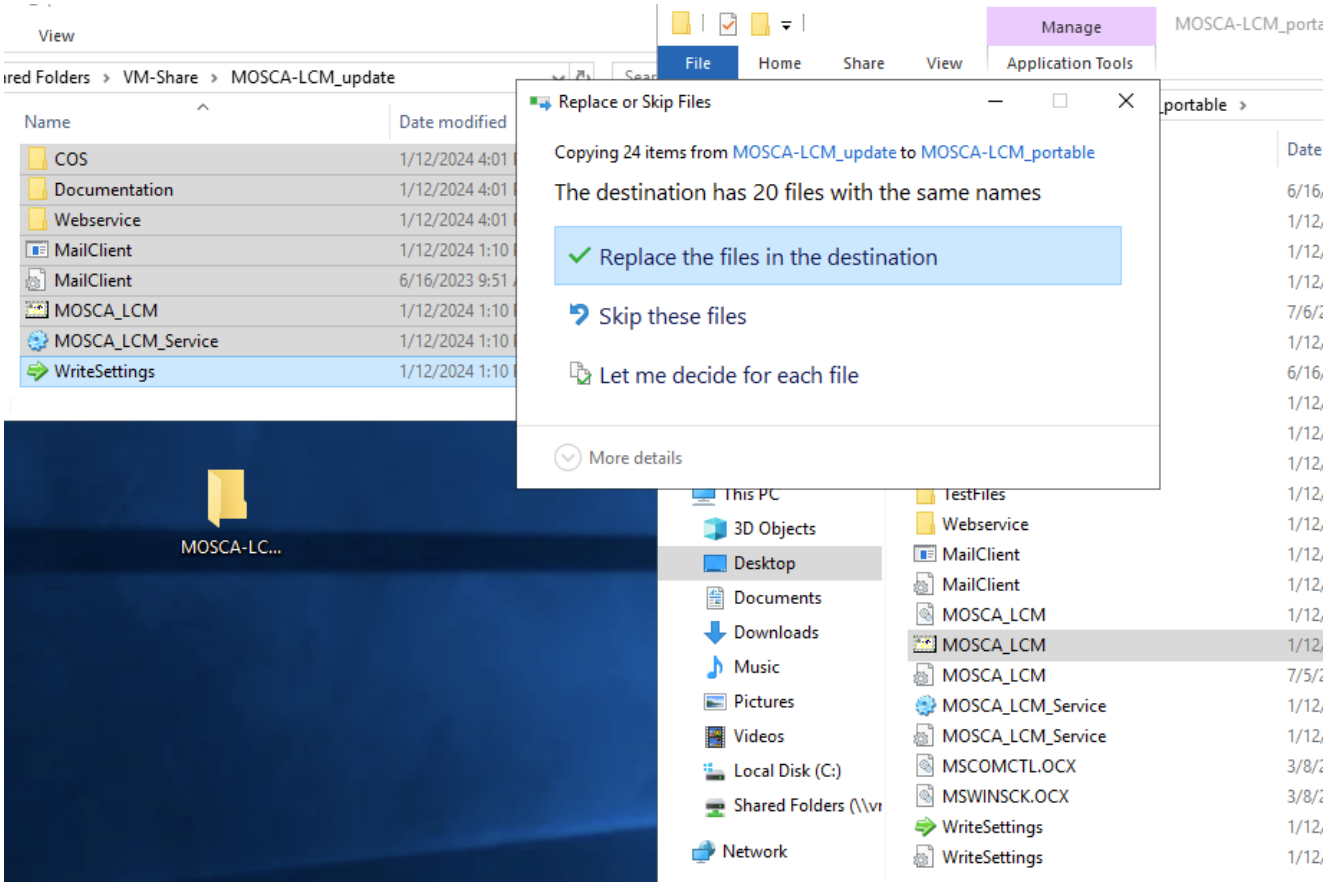




Then click “Disable” to prevent the MOSCA-LCM-Service to be started during the update. The button will turn red.

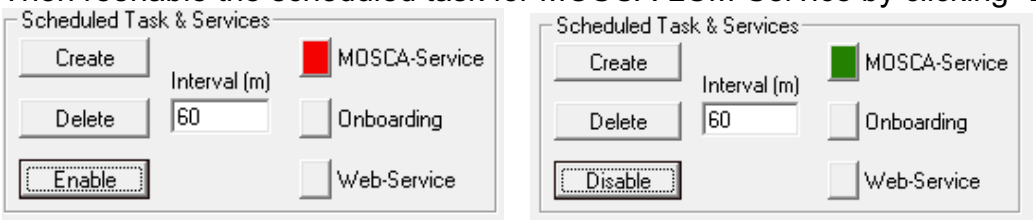


Mark all files and folders you want to update in the update folder and move/copy them to the MOSCA-LCM folder you want to update. Click “Replace all files in the destination”.



Your update is finished. Start MOSCA-LCM and check if all configurations and credentials are still right. It might happen, that after an update some of the settings or passwords have changed and need to be corrected.

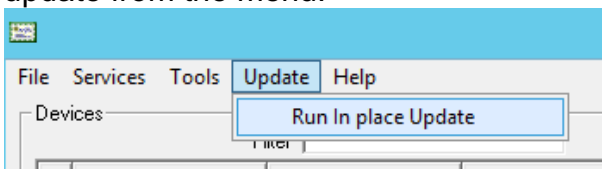
Then reenable the scheduled task for MOSCA-LCM-Service by clicking “Enable”.



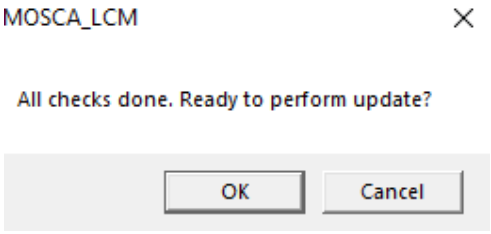
Finally reinstall your needed services from the service menu.

In place Update

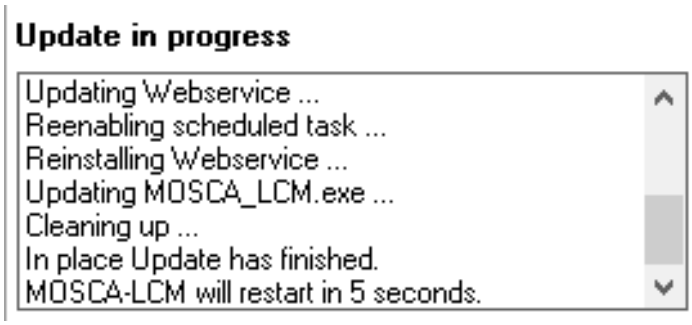
This is a fully automated update using the update package called “vX_X_X_MOSCA-LCM_update.zip”. Just start MOSCA-LCM with administrative rights and start the update from the menu.



Then select the update zip file “MOSCA-LCM_update.zip”. Please do **not** uncompress the file before.
MOSCA-LCM is doing some preflight checks and if everything is fine, you are finally asked to start the update.



Press OK and the update process starts.
MOSCA-LCM will stop and uninstall all running services. The scheduled task will be disabled.



After the update has finished, all tasks and services will be restored to their prior state.

Finally, MOSCA-LCM will restart in the updated version.

If update messages appear after the update, please read them carefully and follow instructions if shown.

Appendix

List of supported devices

Here you will find of a list of all by MOSCA-LCM supported and tested devices. Other untested devices may also work with MOSCA-LCM but with no warranty.

Device Name	Canon Platform
LBP1127C	NCA3.3 SFP
LBP1238	NCA3.3 SFP
LBP1440	NCA4.0 SFP
LBP1861/1871	XPT2-Lite
LBP21x	NCA3.2 SFP
LBP226 / 227 / 228	NCA3.3 SFP
LBP25x	NCA3.0 SFP
LBP311/312	XPT1 SFP
LBP351	XPT1 SFP
LBP6670/6680/6780	XPT1 SFP
LBP635C	NCA3.1 SFP
LBP647C	NCA4.0
LBP674C	NCA4.0
LBP71x	XPT1 SFP
iR C1325	NCA2.0
iR1435	NCA2.0
iR1643	NCA3.3
iR1643 II	NCA3.4
iR2224	NCA4.0
iR2725/2730/2745	XPT2-Lite
iR2925/2930/2945	XPT2-Lite
iRC3326	XPT2-Lite
MF1127C	NCA3.3
MF635C/735C	NCA3.1
MF72xC	NCA2.0
MF1238	NCA3.3
MF1333C	NCA4.0
MF1440	NCA4.0
MF41x	NCA3.0
MF42x	NCA3.2
MF44x	NCA3.3
MF510	NCA3.0
MF5900/MF6100	NCA1.0
MF645C	NCA3.3
MF742C/745C/746C	NCA3.3
i-SENSYS X C1533P	XPT2-Lite
i-SENSYS X C1538P	XPT2-Lite
iR-ADV xxxx	iR-ADV
iR-ADV Cxxxx	iR-ADV
iR-ADV-DX xxxx	iR-ADV (Gen3)
iR-ADV-DX Cxxxx	iR-ADV (Gen3)

imageForce xxxx	imageFORCE
imageForce Cxxxx	imageFORCE
imagePRESS Vxxx	iR-ADV
imagePRESS Cxxx	iR-ADV
Lexmark MX622/MX632	Lexmark
Lexmark CX635/CX735	Lexmark
Lexmark CX922/CX963	Lexmark
Lexmark CS963	Lexmark

Used Ports and Protocols

Port	Protocol	Service	Source	Destination	Mandatory	Intended Use
123	UDP	NTP	Printer	MOSCA	only when using Onboading	Auto onboarding via NTPS request
161	UDP	SNMP	MOSCA	Printer	only when using Onboading	Get device Information for onboarding
80	TCP	HTTP	MOSCA	Printer	optional	- Get active certificates - if certificate is invalid, request CSR on device and download CSR to MOSCA - Deploy and activate signed certificate
8000	TCP	HTTP	MOSCA	Printer	only for devices with TLS off	- Get active certificates - if certificate is invalid, request CSR on device and download CSR to MOSCA - Deploy and activate signed certificate
443	TCP	HTTPS	MOSCA	Printer	optional	- Get active certificates (encrypted communication) - if certificate is invalid, request CSR on device and download CSR to MOSCA (Encrypted communication) - Deploy and activate signed certificate (Encrypted communication)
8443	TCP	HTTPS	MOSCA	Printer	yes	- Get active certificates (encrypted communication) - if certificate is invalid, request CSR on device and download CSR to MOSCA (encrypted communication) - Deploy and activate signed certificate (encrypted communication)
135	TCP	RPC	MOSCA	Certificate Authority	only when signing via RPC	Sign CSR against CA via Microsoft Remote Procedure Call (RPC) and by using an assigned certificate template
49152-65535	TCP	RPC	MOSCA	Certificate Authority	only when signing via RPC	Dynamic high ports for RPC. Can be set to a single port on the CA.
443	TCP	HTTPS	MOSCA	Certificate Authority	only when signing via API, CEWS or SECP	Sign CSR against CA
443	TCP	HTTPS	Client computer	MOSCA	only when using webservice	Accessing MOSCA-LCM via a web browser (encrypted communication). Port can be configured.
80	TCP	HTTP	Client computer	MOSCA	optional	Accessing MOSCA-LCM via a web browser. Port can be configured.
25	TCP	SMTP	MOSCA	e-mail Server	only if Mail Client is used	When Mail Client is used for monitoring, for communication to the SMTP Server. Port can be configured.
587	TCP	SMTP	MOSCA	e-mail Server	only if Mail Client is used	When Mail Client is used for monitoring, for TLS communication to the SMTP Server. Port can be configured.
514	UDP		MOSCA	SysLog Server	only if SysLog is used	When using SysLog for monitoring, for communication to the SysLog Server.
-	ICMP	PING	MOSCA	Printer	optional	Online check of devices.

How to use MOSCA-LCM with non-supported devices

If you have non-supported or non-Canon devices you can even use MOSCA-LCM to support you to get the needed certificates on these devices. MOSCA-LCM can assist you to get the certificates from your CA. Finally, MOSCA-LCM can monitor the certificate validity end date to inform you if a certificate is about to expire.

The non-supported devices need to be able to get a PKCS#12 file installed.

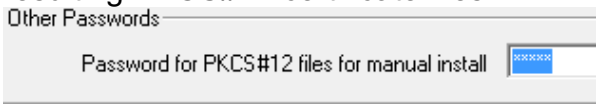
The process will be as follows:

- You create an entry in MOSCA-LCM for each non supported device.
- MOSCA-LCM will generate a CSR.
- MOSCA-LCM will sign the CSR against your CA.
- MOSCA-LCM will convert the signed certificate in PKCS#12 (*.pfx) format.
- You have to install the certificate manually.
- You tell MOSCA-LCM that the certificate is installed.
- MOSCA-LCM will monitor the certificate validity end date.

Pre-Configuration

The following settings need to be configured before running the process:

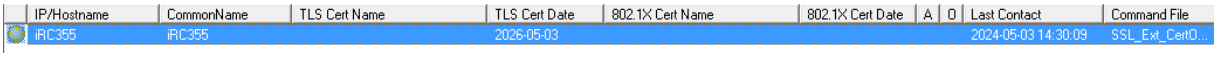
- Enter a password in the credentials section that will be used to encrypt the resulting PKCS#12 certificate files.



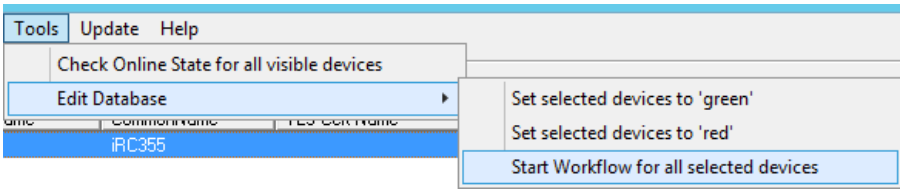
- Configure the entry “ManualInstallCertificateValidityMonths” in the “MOSCA_LCM_Service.ini” to let MOSCA-LCM automatically create a certificate end date in the device list. Please refer to the section “MOSCA_LCM_Service” to learn more about the configuration options.
- Configure the entry “ManualInstallCheckCertificateDates” to “1” in the “MOSCA_LCM_Service.ini” to let MOSCA-LCM automatically monitor the certificate end date at each service start. Please refer to the section “MOSCA_LCM_Service” to learn more about the configuration options.
- Create the device entry in MOSCA and assign a special workflow for letting MOSCA-LCM just putting the certificates for manual installation in the certificate folder. There is a sample command file called “SSL_Ext_CertOnly_Sample.mlc” which should be used. You need to configure this workflow to fit your environment. Please refer to the section “Command files” to learn how to do that.
- Do not set the device in the device list to “Auto” as this workflow will be started by a tool command.

To enrol a certificate, you need to do the following for each device in the list you want to get a certificate for:

- Select one or more devices.



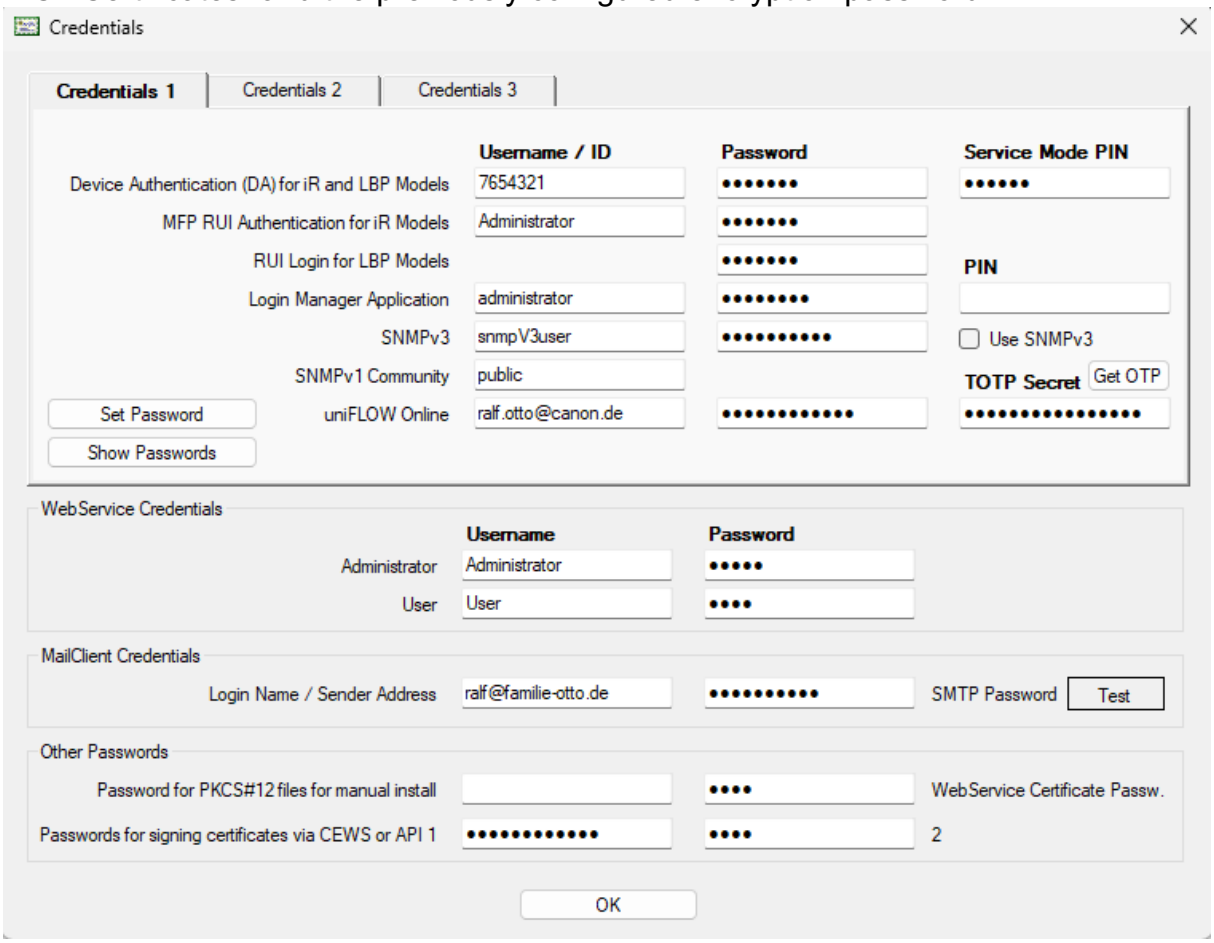
- Go to menu “Tools / Edit Database” and select “Start Workflow for all selected devices.



- Start the MOSCA-Service or wait for the scheduled service start.
- Wait for the process to finish. You will get a purple state for each device a certificate has been requested for. In addition, these devices got an “Actual CMD” entry called “ManualInstall”, telling you that the certificate is ready for manual install.



- Install the certificate manually on the device using the *.pfx file in the folder “LCMCertificates” and the previously configured encryption password.



- After manual installation, tell MOSCA-LCM that the certificate is installed by using the menu “Tools / Edit Database” and select “Set selected devices to ‘green’”.
- MOSCA-LCM will then show device state green and check all devices with the phrase “CertOnly” in the command file name at each service start.
- If a device reaches the in the command file configured “MaxDaysUntilExpiry” entry before the certificate validity end date, the device state will be turned to yellow. Then you have to start the whole process again.

SCEP Requirements

To let MOSCA sign device certificates via SCEP, the customer's environment must provide the following:

1. NDES role installed and reachable
   - Network Device Enrollment Service (NDES) role installed on a server (typically alongside or registered with an Enterprise CA).
   - Two URLs must be reachable by the MOSCA_LCM_Service host:
     Admin/challenge-password page: http://<server>/certsrv/mscep_admin/
     SCEP operational endpoint: http://<server>/certsrv/mscep/mscep.dll

2. Certificate template configured
   - NDES uses one fixed template for all SCEP-issued certificates, configured via registry (GeneralPurposeTemplate, or separate EncryptionTemplate / SignatureTemplate), not selectable per request.
   - The template must be supported for CSR-based (non-AD-tied) enrollment and contain the required EKUs (e.g. Server Authentication, and/or Client Authentication) and key usages needed by the devices.
   - The template must allow the key size/algorithm MOSCA generates (RSA 2048/4096).

3. Challenge password mechanism available or fixed challenge is used
   - The admin page must be reachable and return a one-time challenge password (16-char hex string) on each GET request.
   - Access to the admin page must be permitted for whatever identity MOSCA_LCM_Service uses (anonymous, or Windows-authenticated — see below).
   - If a fixed challenge password is set in the configuration, the admin page is not used for the retrieval of the challenge password.

4. Authentication mode decided and configured consistently
   - If the admin page and/or mscep.dll require Windows Authentication (common default), the MOSCA_LCM_Service process must run under an account with the necessary access rights, and UseWindowsAuthForAdminPage / UseWindowsAuthForOperation must be set to 1 in the SCEP command file.
   - If anonymous access is used instead, these must be 0, and the customer must explicitly allow anonymous access to those two IIS paths.
   - Alternatively, you can use configure the “Special account for external signing”. Please refer to the section “Base configuration / Credentials”.

5. GetCACert must return usable RA/CA certificates
   - mscep.dll?operation=GetCACert must return the CA chain including the RA encryption and RA signing certificates (or a single RA cert used for both roles).
   - NDES must have valid, non-expired RA certificates (Registration Authority (RA) templates auto-enrolled to the NDES service account).

6. Workflow-side constraint (customer awareness, not a CA setting)
   - SCEP-based signing only works for workflows using MOSCA's GenerateCSR (key pair generated and held locally by MOSCA), because the SCEP challenge password must be embedded and signed inside the CSR.

Workflows using RequestCSR (device generates its own key pair) cannot use SCEP, since the private key never leaves the device.

7. Network/firewall access
   - Outbound HTTP(S) access from the MOSCA_LCM_Service host to the NDES server's admin and mscep.dll URLs must be permitted (no proxy interference, correct DNS resolution).

Optional/recommended:
   - Hardening the admin page (Windows auth, IP restriction, HTTPS, dedicated service account) since anyone who can reach it can obtain a valid enrollment password.