macOS

Compliance Spotter

Integration Guide

Agnosys

57 rue Bourguignette

91530 Saint-Maurice-Montcouronne

France

https://www.agnosys.com

Introduction

macOS Compliance Spotter (mCS) is a tool designed for IT teams in both business and educational environments to maintain and monitor ongoing compliance across macOS devices. It performs regular compliance checks using the open-source macOS Security Compliance Project (mSCP), ensuring systems adhere to established security baselines.

In addition to these core checks, mCS includes built-in modules for various compliance areas and offers the flexibility for customers to integrate their own custom modules.

Results are compiled into detailed reports, automatically sent to IT teams via webhooks to Slack and Microsoft Teams. These reports can also be accessed locally through a complete interface or via notifications, offering versatile options for compliance monitoring.

Before reading this documentation, please consult the following pages.

• Introduction

`https://www.agnosys.com/logiciels/mcs-en/`

• Management solutions support

`https://www.agnosys.com/logiciels/mcs-management-solutions-support-en/`

• Capabilities

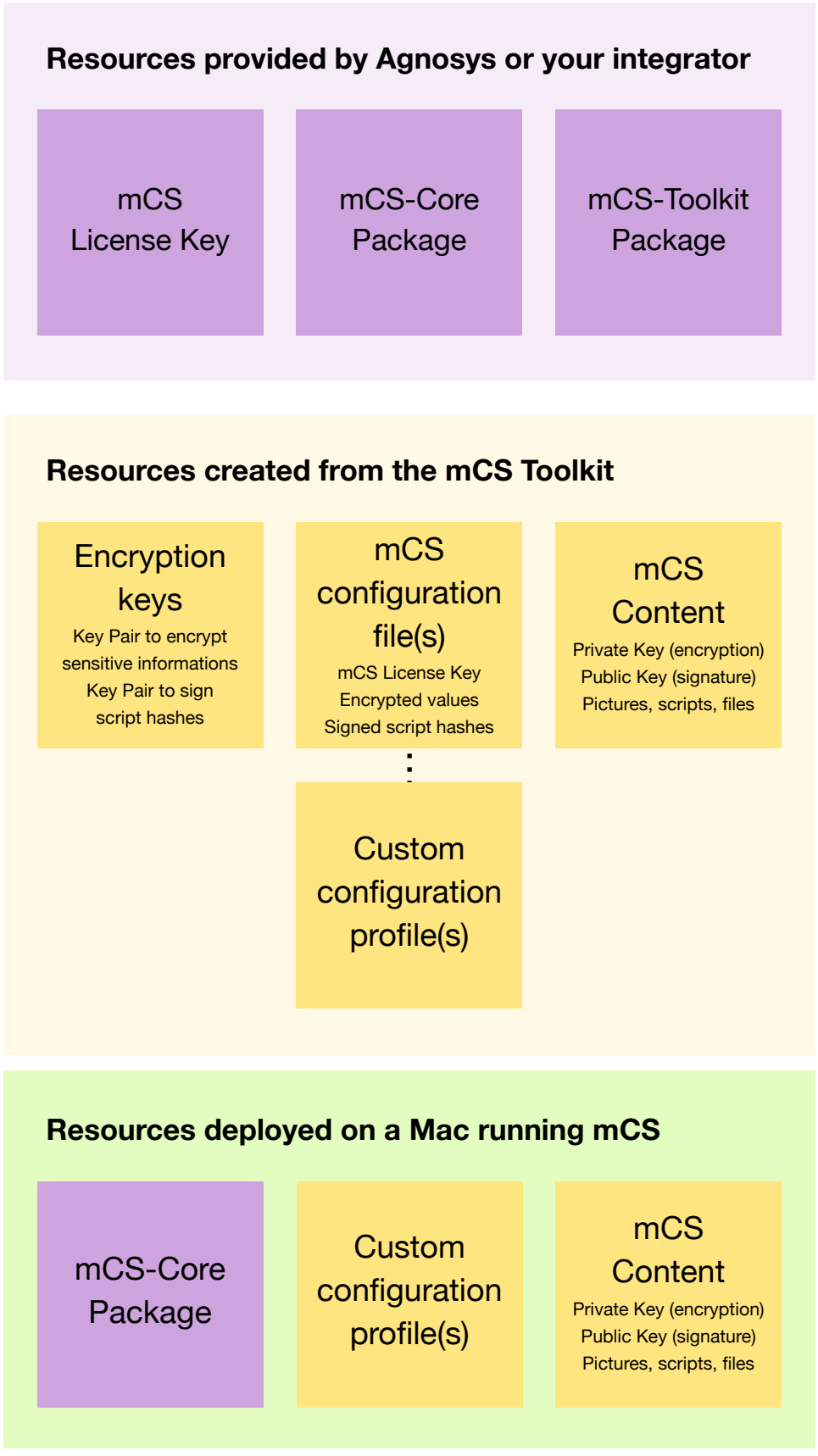
`https://www.agnosys.com/logiciels/mcs-capabilities-en/`

• Offers and pricing

`https://www.agnosys.com/logiciels/mcs-offers-en/`

Synopsis

Resources overview



Implementation workflow

Step	Chapter in this document
Get an mCS License Key	Software requirements
Download the mCS-Core Package	
Download the mCS-Toolkit Package	
Install the Packages app	
Install a Property List editor	
Install the mCS-Toolkit	mCS Toolkit installation
Create the encryption keys	Encryption keys creation
Customize the mCS configuration file	mCS configuration file edition
Create other mCS configuration files if necessary	
Integrate with macOS Security Compliance project	Implementing mSCP compliance
Integrate with Microsoft Teams	Implementing Microsoft Teams integration
Convert configuration files to Custom configuration profiles	mCS configuration files to Custom configuration profiles conversion
Build (and sign) the mCS-Content package	mCS Content building
Provision the MDM for Background Item Management	Configuration profiles requirements
Provision the MDM with mCS components	Provisioning *MDM*
Observe the result of an mCS execution	mCS execution
Renewing the license code (once every year)	Renewing mCS license
Updating mCS to a newer version (year-round)	Updating mCS
Enable logging and display logs	Troubleshooting
Execute mCS manually	

Software requirements

macOS

mCS requires macOS 10.15 and later.

macOS packages

Download (only) the following packages from this URL :

https://www.dropbox.com/scl/fo/8ebmm0mkq41uf1si13z81/AD6DEPKoayPOSC1l9wiEvgM?rlkey=gk3utc0hc9rdooi2yvfhnh2vb&dl=0

- mCS-Core-*version*.pkg

- mCS-Toolkit-*version*.pkg

The installation of mCS-Toolkit is described in the "mCS Toolkit installation" chapter.

mCS requires a license key provided by Agnosys or your integrator.

Packaging editor

Download and install the "Packages" app (free) from this URL :

http://s.sudre.free.fr/Software/Packages/about.html

Property List editor

This documentation refers to the "PLIST Editor" app available on the Mac App Store :

https://apps.apple.com/app/plist-editor/id1157491961

You can use the Property List editor of your choice (e.g. Xcode).

Omnissa Workspace ONE Admin Assistant

If the MDM solution is Omnissa Workspace ONE, download and install Workspace ONE Admin Assistant :

https://customerconnect.omnissa.com/downloads/details?downloadGroup=WSONE-ADMIN-ASSISTANT&productId=1595&rPId=118941

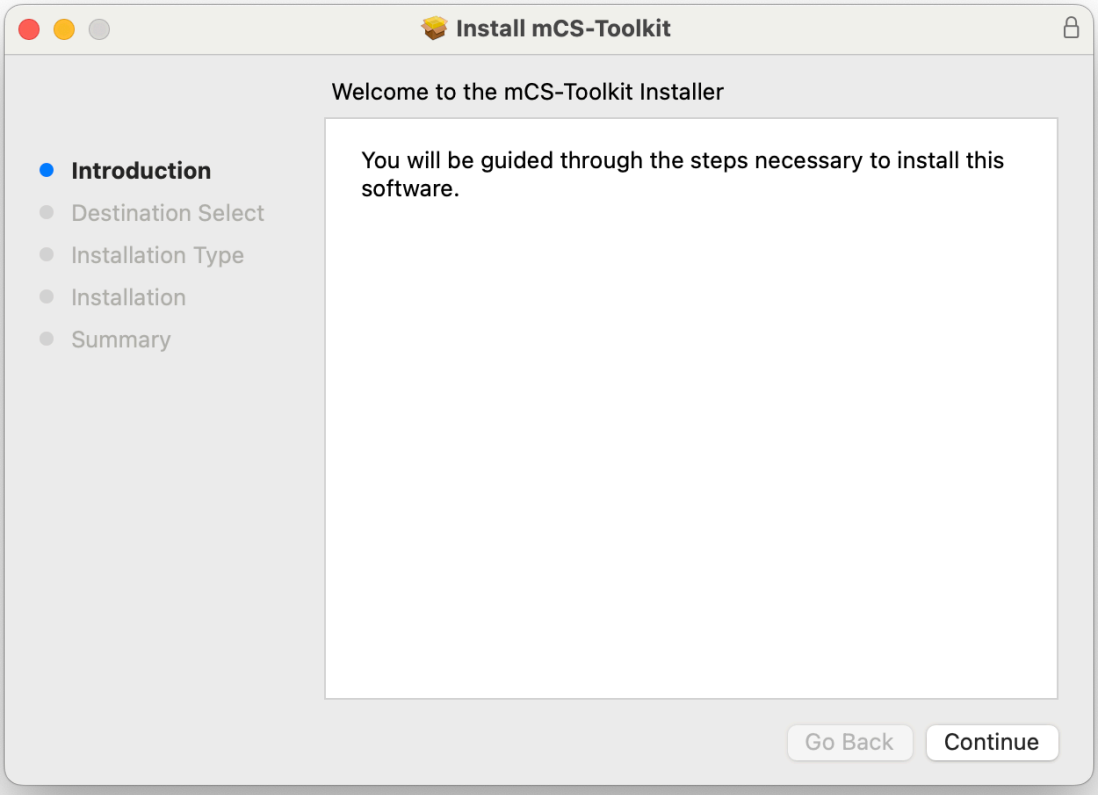
Text Editor

If the MDM solution is Omnissa Workspace ONE, this documentation refers to "Sublime Text" available at this address for the opening of a Plist file :

https://www.sublimetext.com/download_thanks?target=mac

mCS Toolkit installation

Double-click on mCS-Toolkit-*version*.pkg



Enter your administrator password when prompted.

The "mCS-Toolkit" folder is created in /Users/Shared. It contains the following subfolders :
- mcs_configs
- mcs_content
- mcs_library
- mcs_secrets

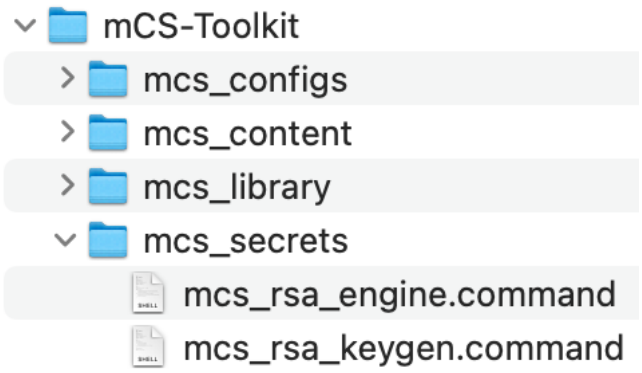
Move the "mCS-Toolkit" folder in a location in your home folder that only you can access.

Do not modify the content of the "mCS-Toolkit" folder unless instructed to do so for specific items.

Encryption keys creation

Sensitive informations in an mCS Property List are protected from direct observation using a RSA encryption method :
- a private / public key pair is created with the "mcs_rsa_keygen" script
- the public key is used when encrypting a value with the "mcs_rsa_engine" script
- the private key is used by mCS to locally decrypt the encrypted values
- the private key is automatically embedded in the mCS-Content package.

Scripts triggered through the External modules are protected from tampering using a RSA signature method :
- a private / public key pair is created with the "mcs_rsa_keygen" script
- the private key is used when signing the hash of a script referenced by an External module
- the public key is used by mCS to locally verify the signatures of the script hashes
- the public key is automatically embedded in the mCS-Content package.

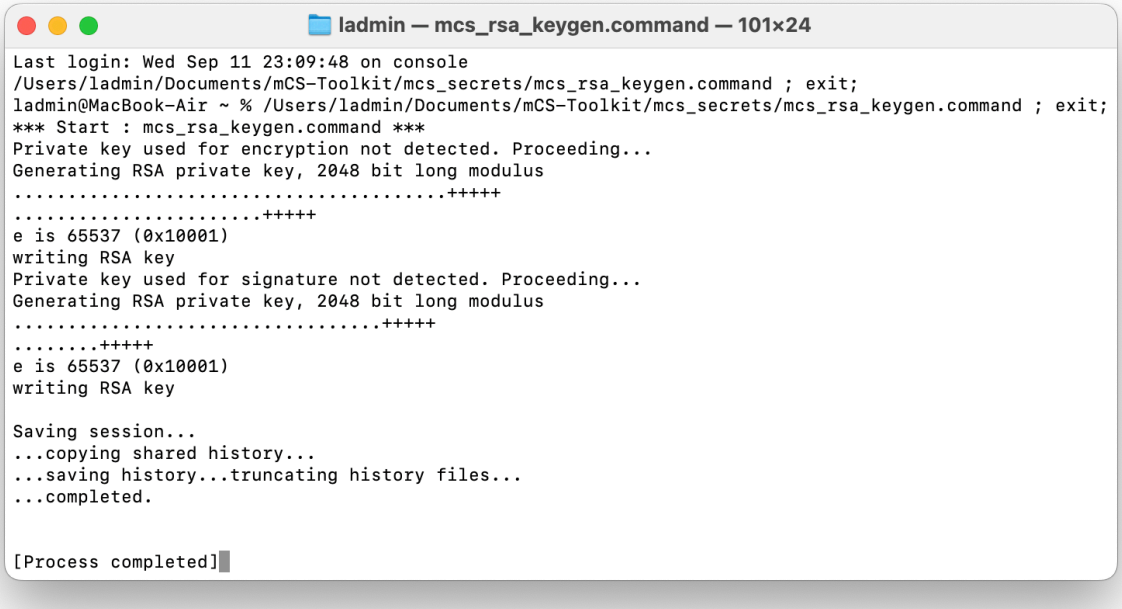


Open the "mCS-Toolkit" folder.

Open the "mcs_secrets" subfolder.

Execute the "**mcs_rsa_keygen**" script (double-click on the .command file).

The script is aimed to be executed only once because the private / public key pairs must be static for the whole mCS integration lifetime.



The private / public key pair used for **encryption** is created at the following path :
mCS-Toolkit > mcs_secrets > mcs_rsa_key.pri and mcs_rsa_key.pub

If you delete the private key at this path, execute the script again. It will generate another private / public key pair with the consequence that you will have to :
- re-encrypt all the sensitive strings
- generate a new mCS-Content package.

The private key is automatically copied at the following path :
mCS-Toolkit > mcs_content > Content > mcs_rsa_key.pri

If you delete the private key at this path, execute the script again. It will copy again the existing private key.

The private / public key pair used for **signature** is created at the following path :
mCS-Toolkit > mcs_secrets > mcs_rsa_key_sign.pri and mcs_rsa_key_sign.pub

If you delete the private key at this path, execute the script again. It will generate another private / public key pair with the consequence that you will have to generate a new mCS-Content package.

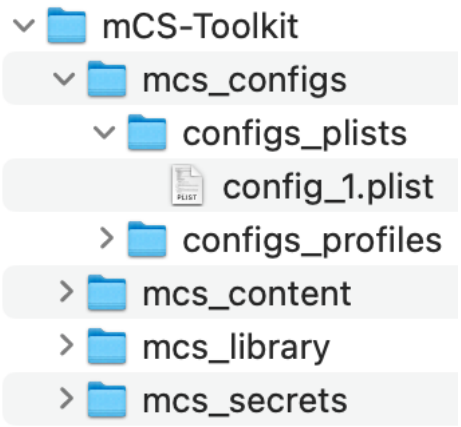
The public key is automatically copied at the following path :
mCS-Toolkit > mcs_content > Content > mcs_rsa_key_sign.pub

If you delete the private key at this path, execute the script again. It will copy again the existing private key.

mCS configuration file edition

The mCS configuration file contains a set of mandatory and optional keys that dictates the functioning of mCS.

Access to the configuration file template



Open the "mCS-Toolkit" folder.

Open the "mcs_configs" subfolder.

Open the "configs_plists" subfolder.

Open the "config_1.plist" property list with you favorite editor.

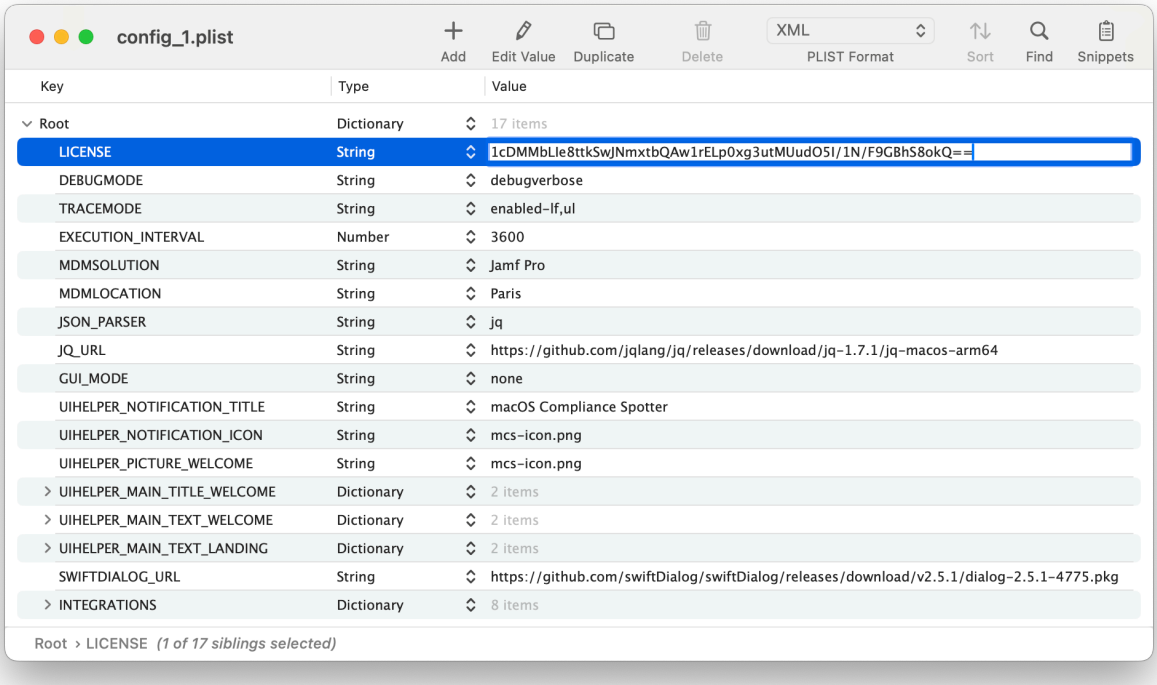
Reference for configuration file keys

Please consult the mCS Dictionary, whose filename is "3. mCS_Dictionary.pdf", to learn how to edit a configuration file.

All keys are important so it is recommended to take the time to read the document completely.

Some keys require extra informations that are detailed in this section.

License key



Paste the mCS license key in the LICENSE key.

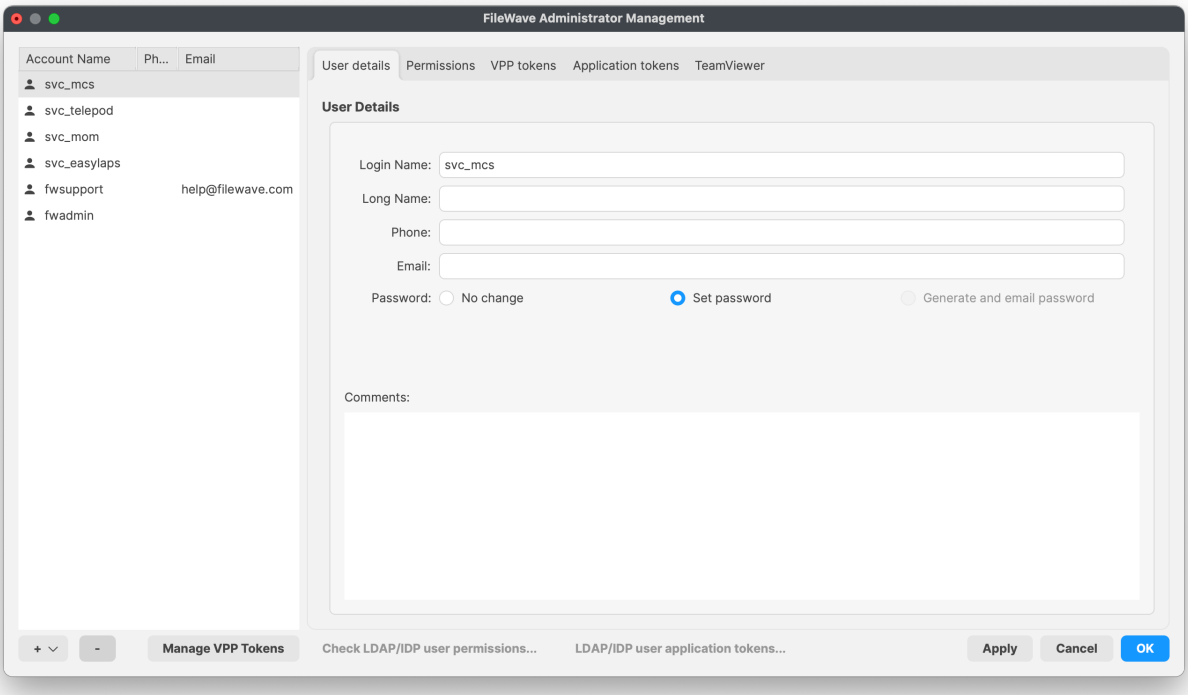
The license key is a one-line string ending exactly with two "=" characters.

FileWave : APIAUTHENTICATIONSTRING key

This section only applies if the management solution is FileWave.

First create a new administrator that will be used by mCS to make API calls.

FileWave Admin > Assistants > Manage Administrators > + Local Account

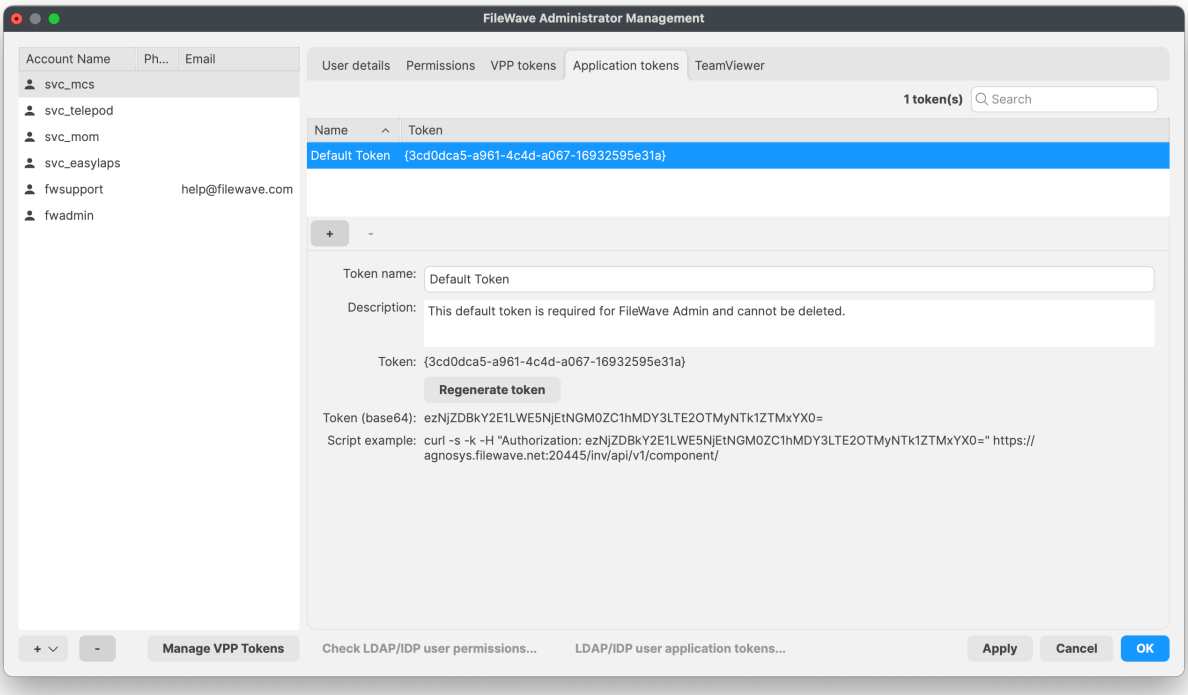


Select "User details" then fill in the "Login Name" field and set a password.

Select "Permissions".

The administrator requires the following permission : "Modify Clients/Groups".

No other permission should be granted to the account.



Select "Application tokens".

Copy the value of "Token (base64)" (exactly) then follow these instructions :
- open the "mCS-Toolkit" folder
- open the "mcs_secrets" subfolder
- execute the "mcs_rsa_engine" script (double-click on the .command file)
- paste the Token
- the Token is encrypted, displayed and then decrypted for sanity check
- copy the encrypted Token (one-line string ending exactly with two "=" characters).

Paste the encrypted value in the APIAUTHENTICATIONSTRING key.

Jamf Pro : APIAUTHENTICATIONSTRING key

This section only applies if the management solution is Jamf Pro.

The authentication mechanism is based on the use of API Roles and Clients available since Jamf Pro 10.49. If the authentication mechanism based on the use of a Jamf Pro User account must be used, please open an mCS support ticket.

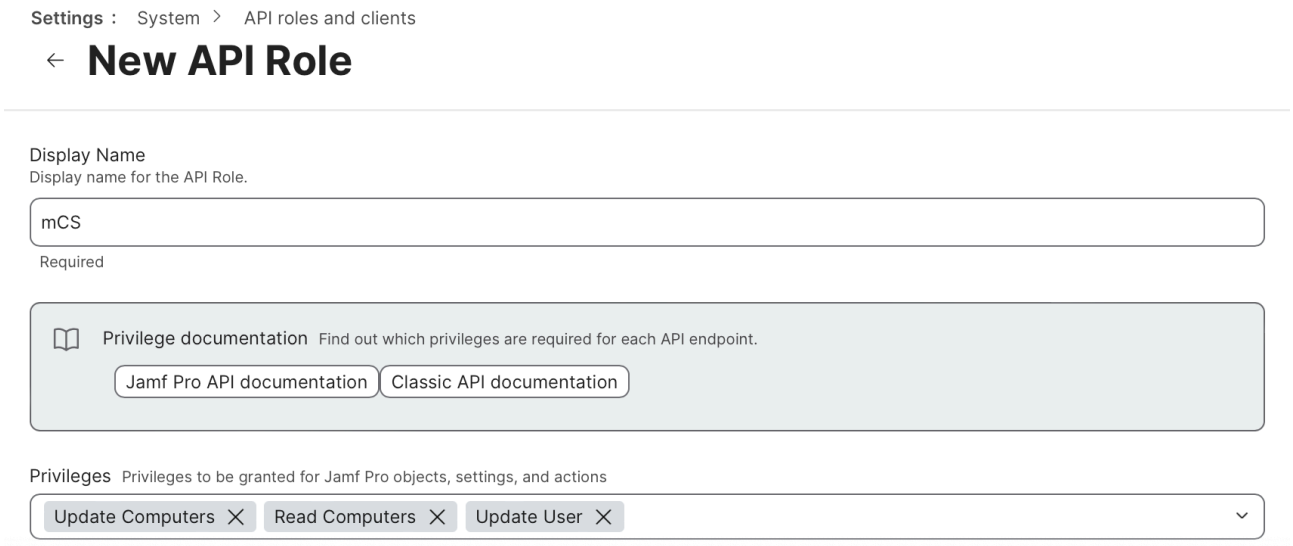
The key will be used by mCS to make API calls.

Create a new text document with 2 lines :
- Client ID :
- Client Secret :

Open Settings then click on "API Roles and Clients".

First create a new role with limited API privileges.

Click on the "API Roles" tab then on the "+ New" button.



Enter a name like "mCS".

Click in the "Jamf Pro API role privileges" field and select the following privileges :

- Read Computers

- Update Computers

- Update User

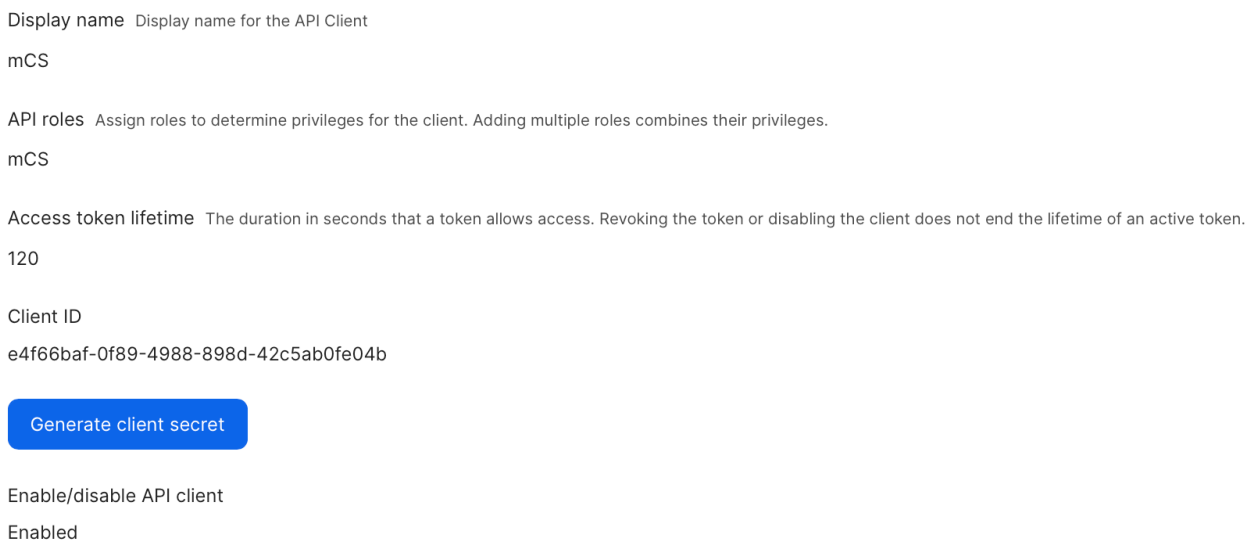
Click on "Save".

Go back to "API Roles and Clients" to create a new API Client associated to the mCS API Role.

Click on the "API Clients" tab then on the "+ New" button.

Enter a name like "mCS", select the mCS API Role and enter "120" (2 minutes) in the "Access Token Lifetime" field.

Click on "Enable API Client" then on "Save".



Click on "Generate Client Secret" then on "Create Secret".

Copy both the Client ID and the Client Secret in the text document then click on "Close".

Concatenate in one string the Client ID and the Client Secret, separated by the character : (colon), then follow these instructions :
- copy the concatenated string (exactly)
- open the "mCS-Toolkit" folder
- open the "mcs_secrets" subfolder
- execute the "mcs_rsa_engine" script (double-click on the .command file)
- paste the concatenated string
- the concatenated string is encrypted, displayed and then decrypted for sanity check
- copy the encrypted concatenated string (one-line string ending exactly with two "=" characters).

Paste the encrypted value in the APIAUTHENTICATIONSTRING key.

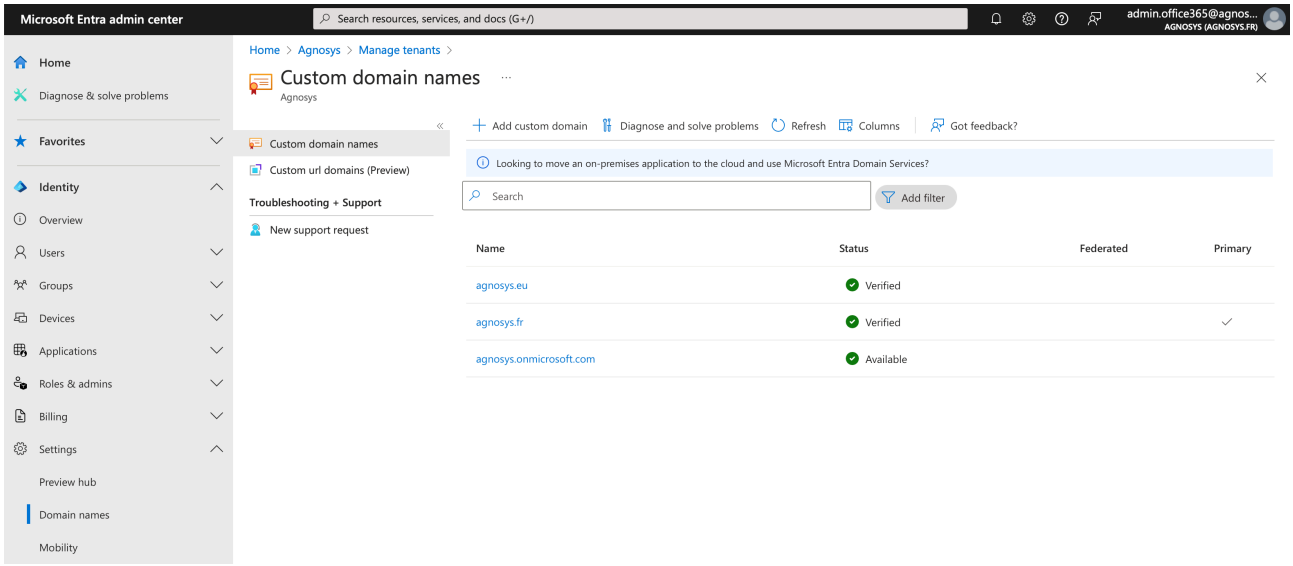
Microsoft Intune : APIAUTHENTICATIONSTRING key

This section only applies if the management solution is Microsoft Intune.

The key will be used by mCS to make API calls.

Create a new text document with 3 lines :
- Tenant domain :
- Application (client) ID :
- Client secret value :

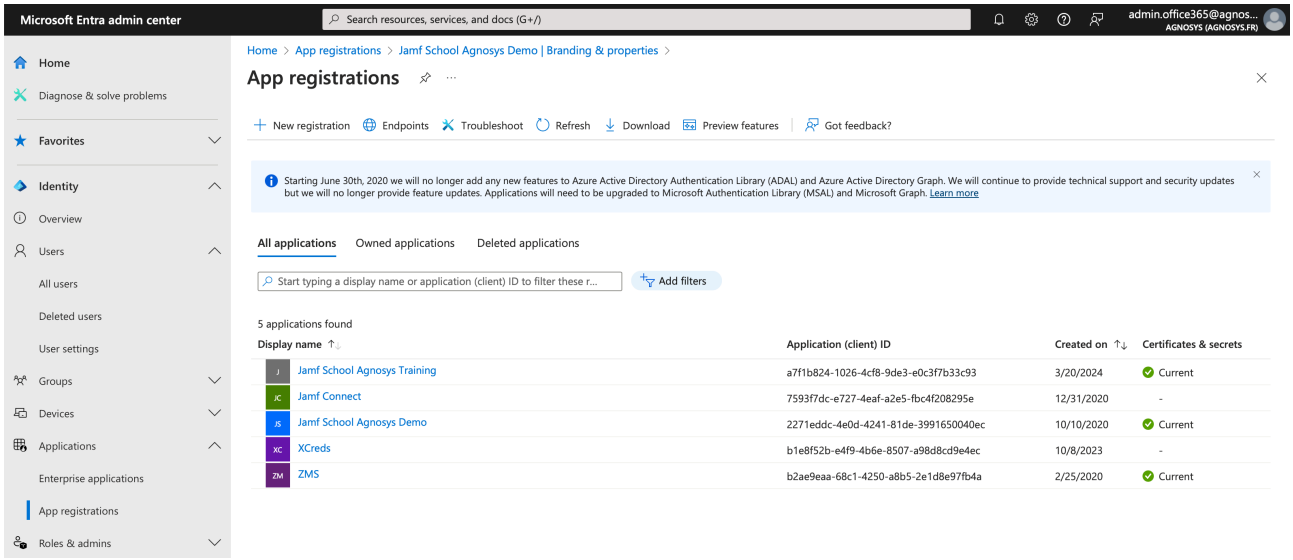
Connect to Microsoft Entra admin center.



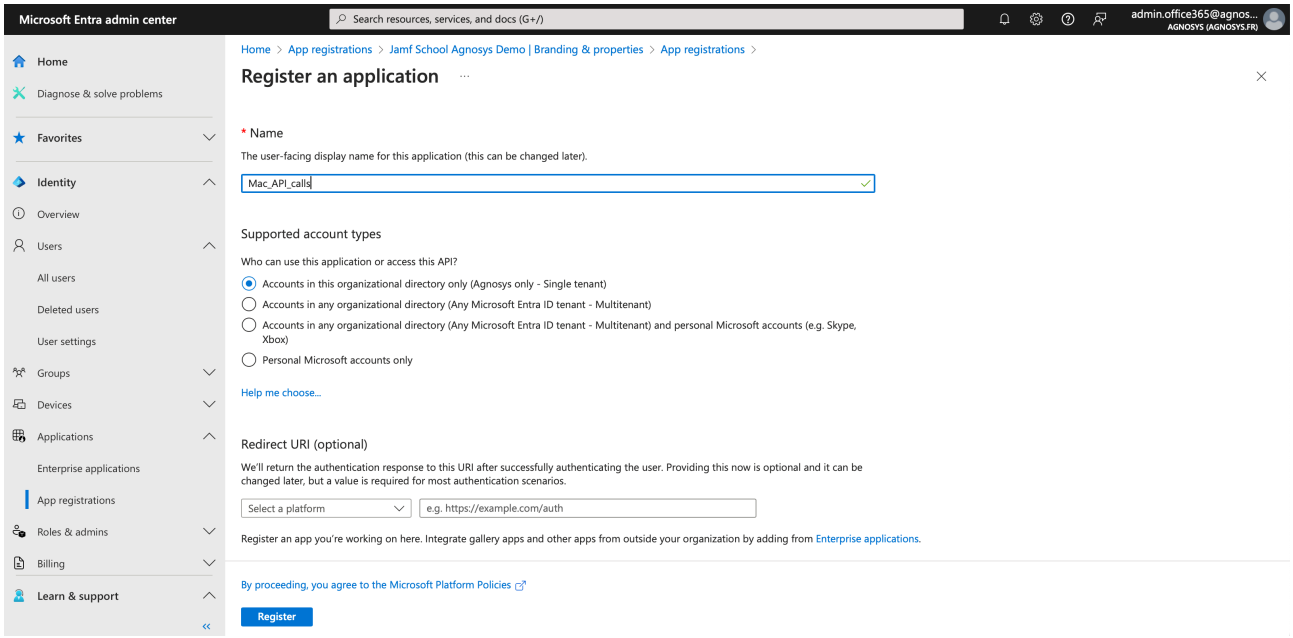
Go to Identity > Settings > Domain names.

Copy / paste the name including the extension ".onmicrosoft.com" in the text document for the value "Tenant domain".

Go to Identity > Applications > App registrations.



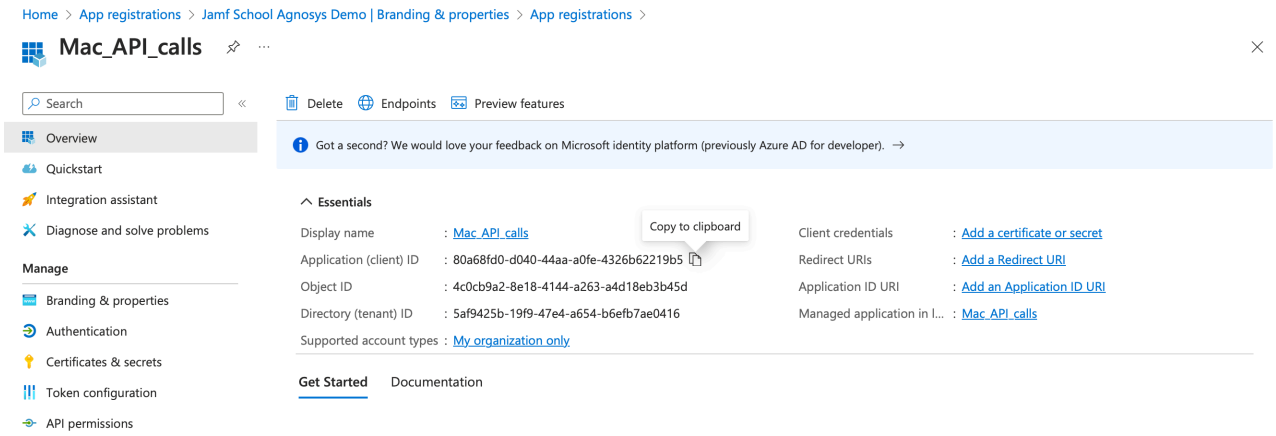
Click on "All applications", then on "New registration".



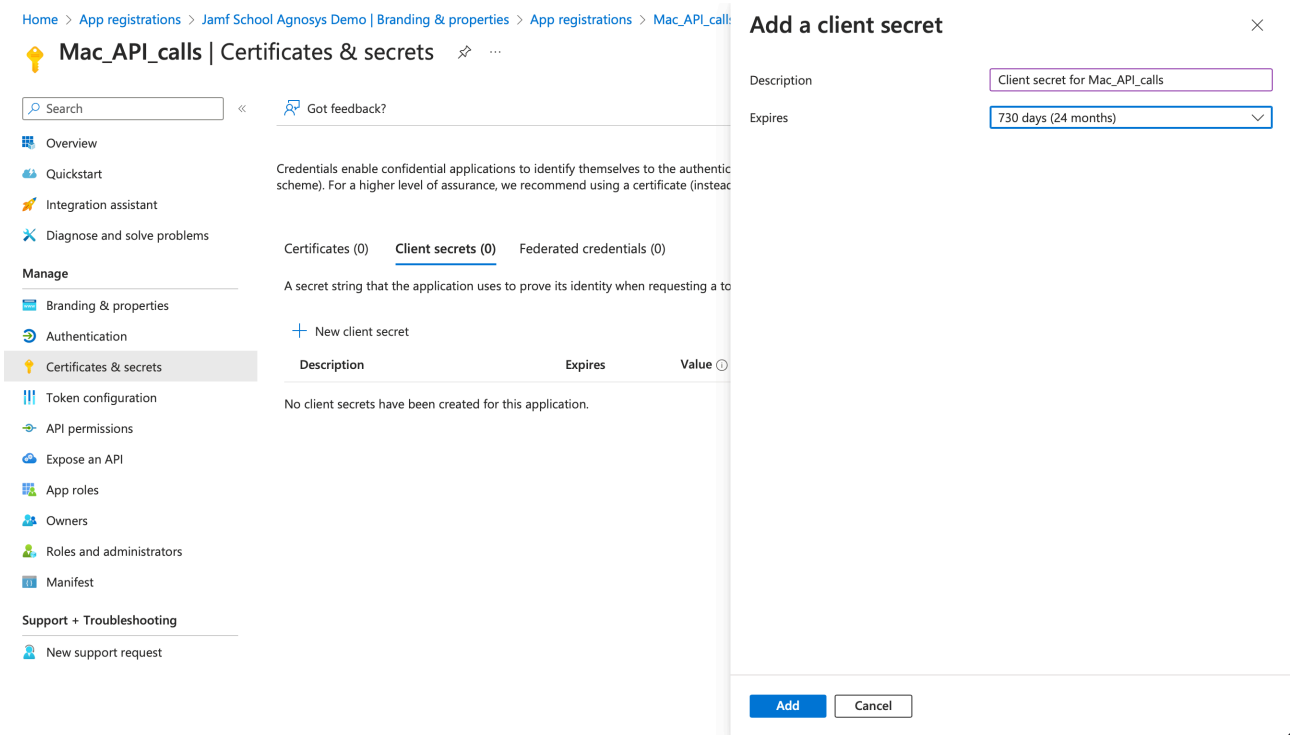
Enter a name for the application.

Select "Accounts in this organizational directory only (*Company* only - Single tenant)".

Click on "Register".



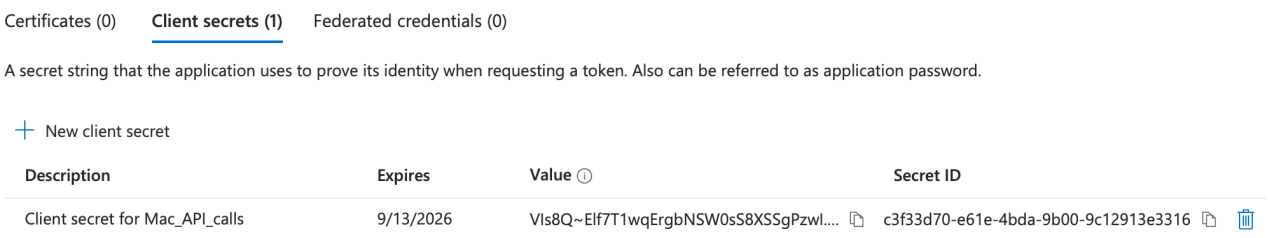
Copy / paste the Application (client) ID in the text document.



Click on "Certificates & secrets" then click on "New client secret".

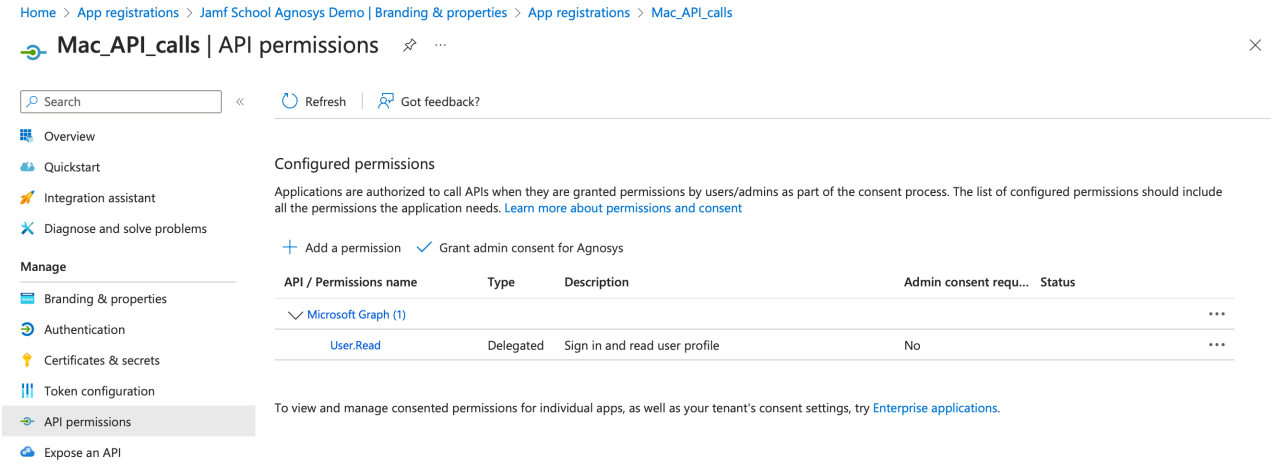
Enter a description and select a life time.

Click on "Add".

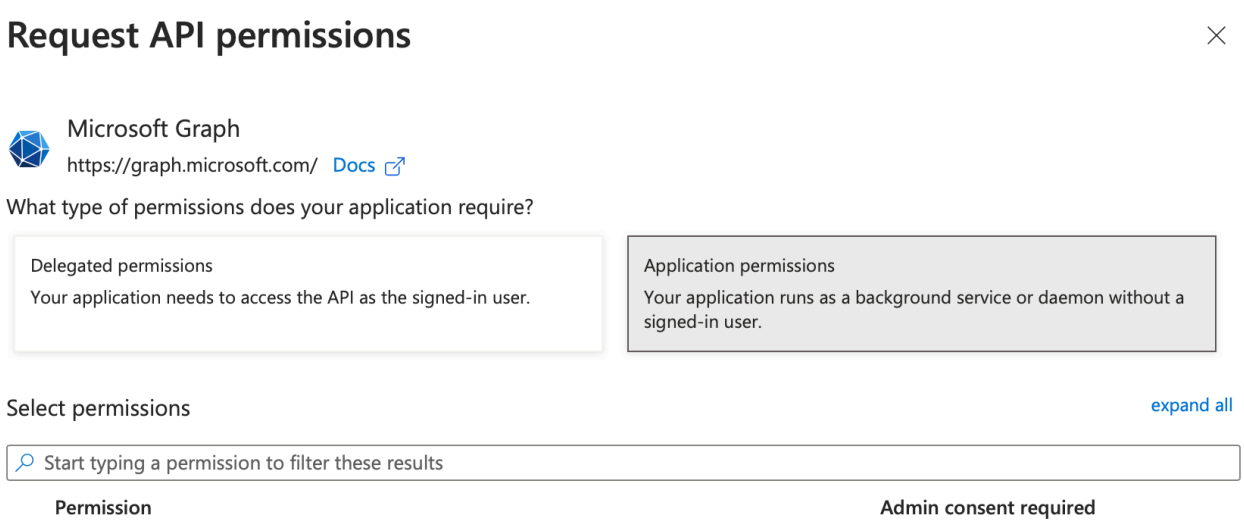


You will see this information **only once**.

Click on the "Copy" button right to the "**Value**" field and paste the value in the text document.



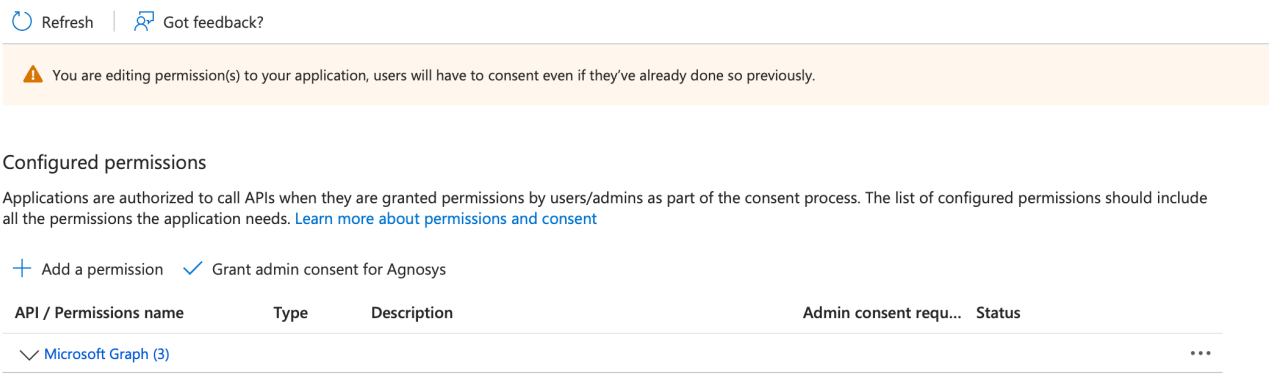
Click on "API permissions", then click on "Microsoft Graph (1)".



Click on "**Application permissions**".

Select the following API permissions :
- "Device.**ReadWrite**.All"
- "DeviceManagementManagedDevices.**Read**.All".

Click on "Update permissions".



Click on "Grant admin consent for *Company*". In the message "Grant admin consent confirmation", click on "Yes".

Check that the "Type" of every permission added is "Application" and that its status is "Granted for *Company*".

```
Tenant domain : agnosys.onmicrosoft.com
Application (client) ID : 66974f69-d2be-4b76-9415-2b8863bc3caa
Client secret value : znGafq6e.V4HT.C34kGN0O9-D~ZV_iyetv

agnosys.onmicrosoft.com,66974f69-d2be-4b76-9415-2b8863bc3caa,znGafq6e.V4HT.C34kGN0O9-D~ZV_iyetv
```

Concatenate the 3 values separated with commas then follow these instructions :
- copy the concatenated string (exactly)
- open the "mCS-Toolkit" folder
- open the "mcs_secrets" subfolder
- execute the "mcs_rsa_engine" script (double-click on the .command file)
- paste the concatenated string
- the concatenated string is encrypted, displayed and then decrypted for sanity check
- copy the encrypted concatenated string (one-line string ending exactly with two "=" characters).

Paste the encrypted value in the APIAUTHENTICATIONSTRING key.

Omnissa Workspace ONE : APIAUTHENTICATIONSTRING key

This section only applies if the management solution is Omnissa Workspace ONE.

The authentication mechanism is OAuth2. If the authentication mechanism is Basic, please open an mCS support ticket.

The key will be used by mCS to make API calls.

Create a new text document with 4 lines :
- Token URL :
- Client ID :
- Client Secret :

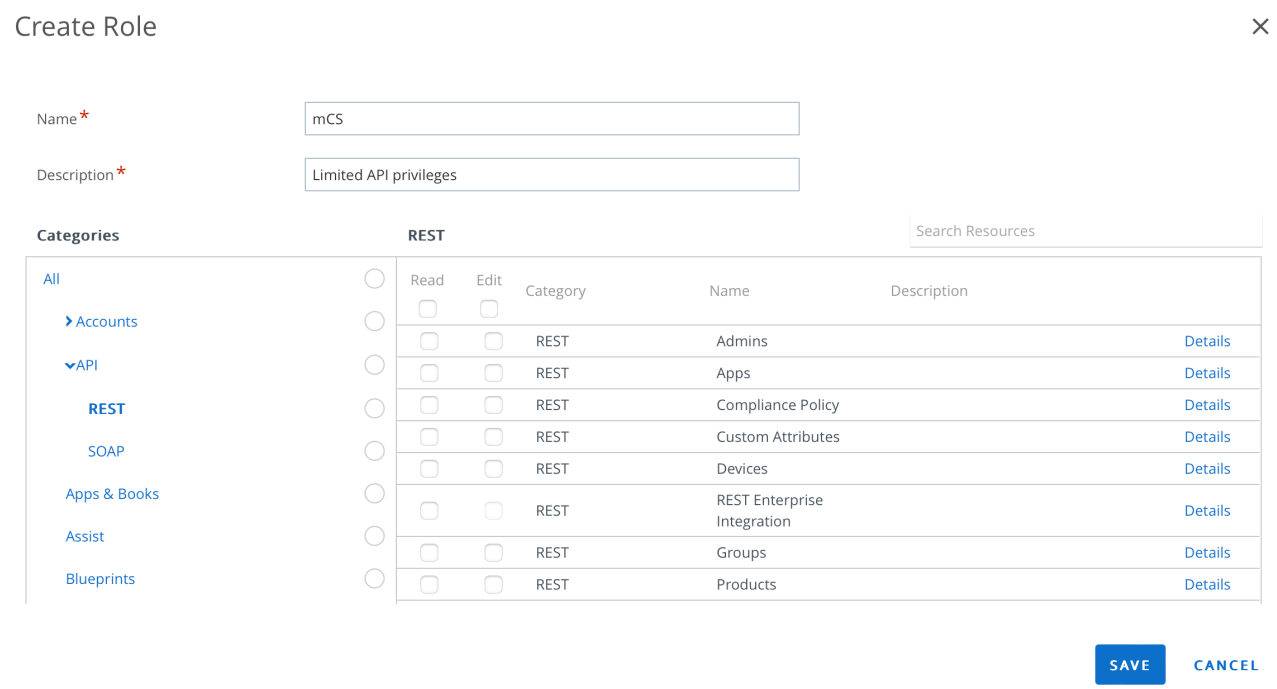
To define the Token URL, please consult this article :

https://docs.omnissa.com/bundle/WorkspaceONE-UEM-Console-BasicsVSaaS/page/UsingUEMFunctionalityWithRESTAPI.html

Create a new role with limited API privileges.

Go to Accounts > Administrators > Roles.

Click on "Add Role".



Enter a specific name like "mCS" and a description, then in the "Categories" sidebar, select All > API > REST.

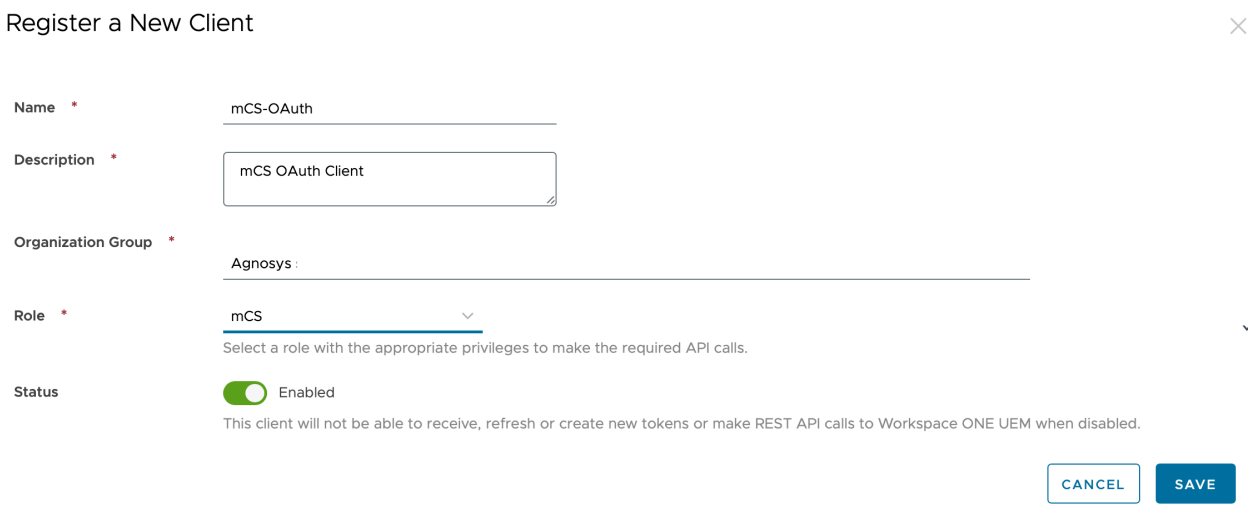
The role requires the following set of privileges :

- Custom Attributes > Details
    - Edit — Rest API Custom Attributes Write
- Devices > Details
    - Read — REST API Devices Read

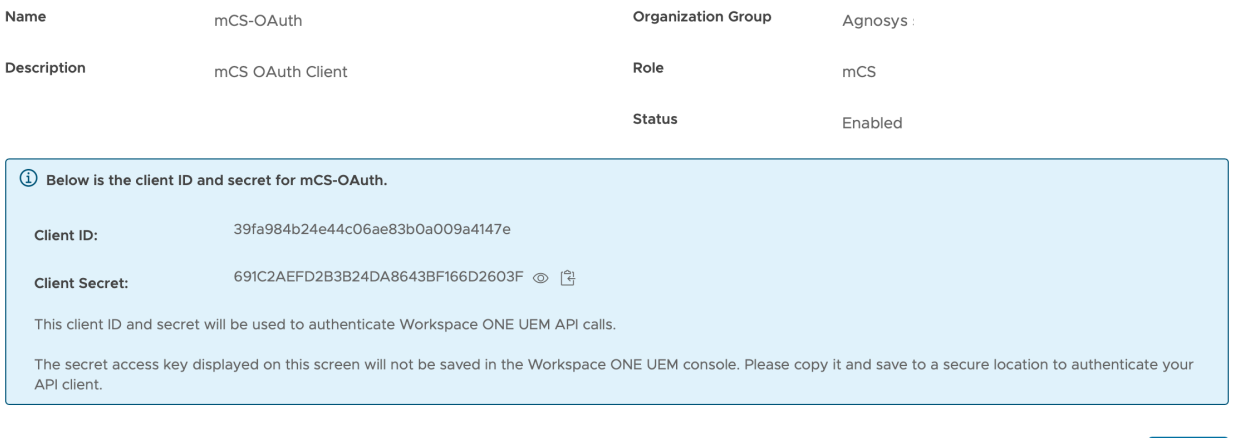
Click on "Save".

Go to Groups & Settings > Configurations > OAuth Client Management.

Click on "Add".



Enter a name and a description. Select the Organization Group that encompasses the devices that are to be installed with mCS then select the "mCS" role. Click on "Save".



Copy both the Client ID and the Client Secret in the text document then click on "Close".

```
Token URL     : https://uat.uemauth.vmwservices.com/connect/token
Client ID     : 3c46dbb9377c4491896989ea2fdae1f0
Client Secret : 9A78D983F619CB7873A90908F6AA1409

https://uat.uemauth.vmwservices.com/connect/token,3c46dbb9377c4491896989ea2fdae1f0,9A78D983F619CB7873A90908F6AA1409
```

Concatenate the 3 values separated with commas then follow these instructions :
- copy the concatenated string (exactly)
- open the "mCS-Toolkit" folder
- open the "mcs_secrets" subfolder
- execute the "mcs_rsa_engine" script (double-click on the .command file)
- paste the concatenated string
- the concatenated string is encrypted, displayed and then decrypted for sanity check
- copy the encrypted concatenated string (one-line string ending exactly with two "=" characters).

Paste the encrypted value in the APIAUTHENTICATIONSTRING key.

Implementing mSCP compliance

The macOS Security Compliance Project (mSCP) is a collaborative effort involving the National Institute of Standards and Technology (NIST), the National Aeronautics and Space Administration (NASA), the Defense Information Systems Agency (DISA), and Los Alamos National Lab (LANL). The project aims to develop and maintain security guidance for organizations that must adhere to specific security compliance frameworks and policies.

mCS integrates with mSCP to apply one of the supported security baselines during the workflow. The mSCP compliance report, which includes a compliance score, is displayed in the Landing pane of the complete interface, or in a notification of the informative interface, and can be shared via Slack and Teams webhooks.

References

This chapter outlines the key points for implementing mSCP compliance in mCS.

For more details, please consider the following suggestions :

• Official project documentation
https://github.com/usnistgov/macos_security

• Compliance Made Even Easier | JNUC 2023
https://www.youtube.com/watch?v=Xp7vvhm6fPc

• Implementing mSCP Using Jamf Pro | JNUC 2022
https://www.youtube.com/watch?v=hCq4PbLX0Tc

• Enforcing macOS Security Compliance Project Baselines: Workspace ONE Operational Tutorial
https://techzone.omnissa.com/resource/enforcing-macos-security-compliance-project-baselines-workspace-one-operational-tutorial

• Secure, Contain, Protect... Your Mac: Deploy mSCP with Intune
https://www.intuneirl.com/secure-contain-protect-your-data-deploy-mscp-with-intune/

To get straight to the point with a functional example, follow these instructions to generate a default CIS Benchmark - Level 1 compliance report using Jamf Pro or another MDM.

Step 1 : Generate compliance assets with Jamf Compliance Editor

Jamf Compliance Editor (JCE) is a native macOS app built on mSCP that generates compliance assets needed to assess and enhance the security posture of devices.

Go to https://github.com/Jamf-Concepts/jamf-compliance-editor

Download the latest released package and then install it.

Open the Jamf Compliance Editor located in /Applications.

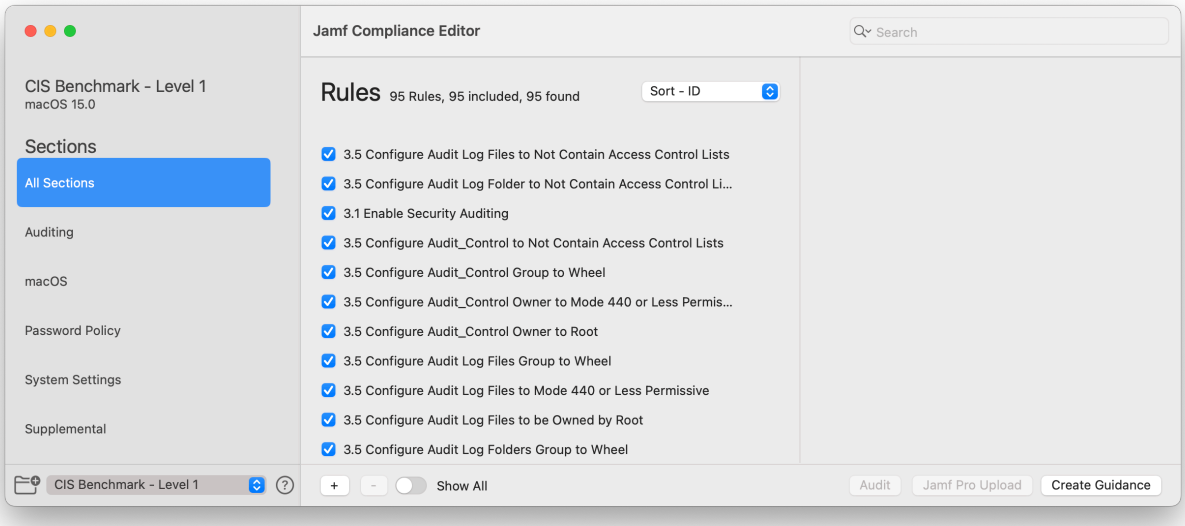
Accept the Terms of Use. On the splash screen, select "macOS" and then click "Create New Project."

When prompted to select a macOS Security Compliance Project branch, choose "sequoia" (or any other version you need) and click "Create".

Note : To configure JCE to show all branches of the mSCP project, run the command `defaults write com.jamf.complianceeditor showAllBranches -bool true` and then re-open JCE ; JCE will then display all branches, including those still under development.

When prompted to select where to save the mSCP directory, choose "Desktop" (or any other folder you prefer) and click "Save".

When prompted to select a Security Benchmark, choose "CIS Benchmark - Level 1" and click "OK".



Click "Create Guidance".

The guidance will be created in Desktop > macOS_security-sequoia/build/cis_lvl1.

Click "View Project" to open this folder in Finder.

Step 2 : Provision Jamf Pro

1/ Identify the compliance assets

The compliance assets to consider in the cis_lvl1 folder are :

• cis_lvl1_compliance.sh : a script to fix and check the security posture

• jamfpro :

    • cis_lvl1.json : a custom schema to configure the baseline

    • compliance-*.xml : Extension attributes (ready to be uploaded)

• mobileconfigs > unsigned > *.mobileconfig : configuration profiles (ready to be uploaded)

2/ Upload the compliance assets

Back in Jamf Compliance Editor, click "Jamf Pro Upload".

Complete the form as follows :
- Display Name : Jamf Pro - Production (a name for the configured server)
- Server URL : https://hostname.jamfcloud.com
- Untick "Use API Role"
- Username : enter the login of an account with administrator privileges
- Password : enter the password for this account
- By default, the script, extension attributes, and configuration profiles are ticked.

Click "Add", and then click "Continue".

You are informed that the json schema must be manually uploaded in Jamf Pro.

Quit Jamf Compliance Editor.

3/ Observe the uploaded compliance assets

Log in Jamf Pro with your administrator account.

Go to Settings > Computer management > Scripts.
Note the script named "Sequoia_cis_lvl1_compliance.sh" associated with the category "Sequoia_cis_lvl1".

Go to Settings > Computer management > Extension attributes.
Note the four Extension Attributes with names starting with "Compliance".
By default, they are displayed in the "Extension Attributes" section of the device records.

Go to Computers > Configuration Profiles.
Note the collection of configuration profiles associated with the category "Sequoia_cis_lvl1".

4/ Make the script available to mCS

Create a policy that mCS will trigger for compliance remediation :

- Options
    - Payload "General"
        - Name : Sequoia_cis_lvl1-fix
        - Category : Sequoia_cis_lvl1
        - Trigger > Custom : Sequoia_cis_lvl1-fix
        - Execution Frequency : Ongoing
    - Payload "Scripts"
        - Sequoia_cis_lvl1_compliance.sh
        - Parameter 4 : `--fix`
- Scope : the devices that will execute mCS.

Create a policy that mCS will trigger for compliance scan :

- Options
    - Payload "General"
        - Name : Sequoia_cis_lvl1-fix
        - Category : Sequoia_cis_lvl1
        - Trigger > Custom : Sequoia_cis_lvl1-fix
        - Execution Frequency : Ongoing
    - Payload "Scripts"
        - Sequoia_cis_lvl1_compliance.sh
        - Parameter 4 : `--check`
- Scope : the devices that will execute mCS.

5/ Distribute the configuration profiles

Scope the configuration profiles to the devices that will execute mCS.

6/ Configure the baseline

This process allows setting exemptions to specific security rules based on your company's unique policies.

Go to Computers > Configuration Profiles.

Click "New".

- Options
    - Payload "General"
        - Name : Sequoia_cis_lvl1-audit
        - Category : Sequoia_cis_lvl1
    - Payload "Application & Custom settings" > External Applications > Add
        - Source : Custom Schema
        - Preference Domain : org.cis_lvl1.audit

Note : To get the domain, open the JSON file and use the value for "Preference Domain"
- Custom Schema > Add schema > Upload > select the JSON file > Save
- Preference Domain Properties
To disable a rule :
- choose "Configured"
- exempt : choose "true"
- exempt reason : enter a reason for disabling the rule (required)

- Scope : the devices that will execute mCS.

Step 3 : Provision another MDM

1/ Identify the compliance assets

The compliance assets to consider in the cis_lvl1 folder are :

• cis_lvl1_compliance.sh : a script to fix and check the security posture

• mobileconfigs :

- preferences > *.plist : plist files used **exclusively with Omnissa Workspace ONE**

- unsigned > *.mobileconfig : configuration profiles used **for all other MDM solutions**

• preferences > org.cis_lvl1.audit.plist : a property list file to configure the baseline

Quit Jamf Compliance Editor.

2/ Embed the compliance script in the mCS-Content

Refer in this documentation to the chapter titled "mCS Content Building" and specifically the section titled "Content gathering" to embed the compliance script in the mCS-Content package.

Multiple scripts for different versions of macOS can be embedded in a single mCS-Content package. mCS will automatically select the correct script based on the configuration of the mSCP integration.

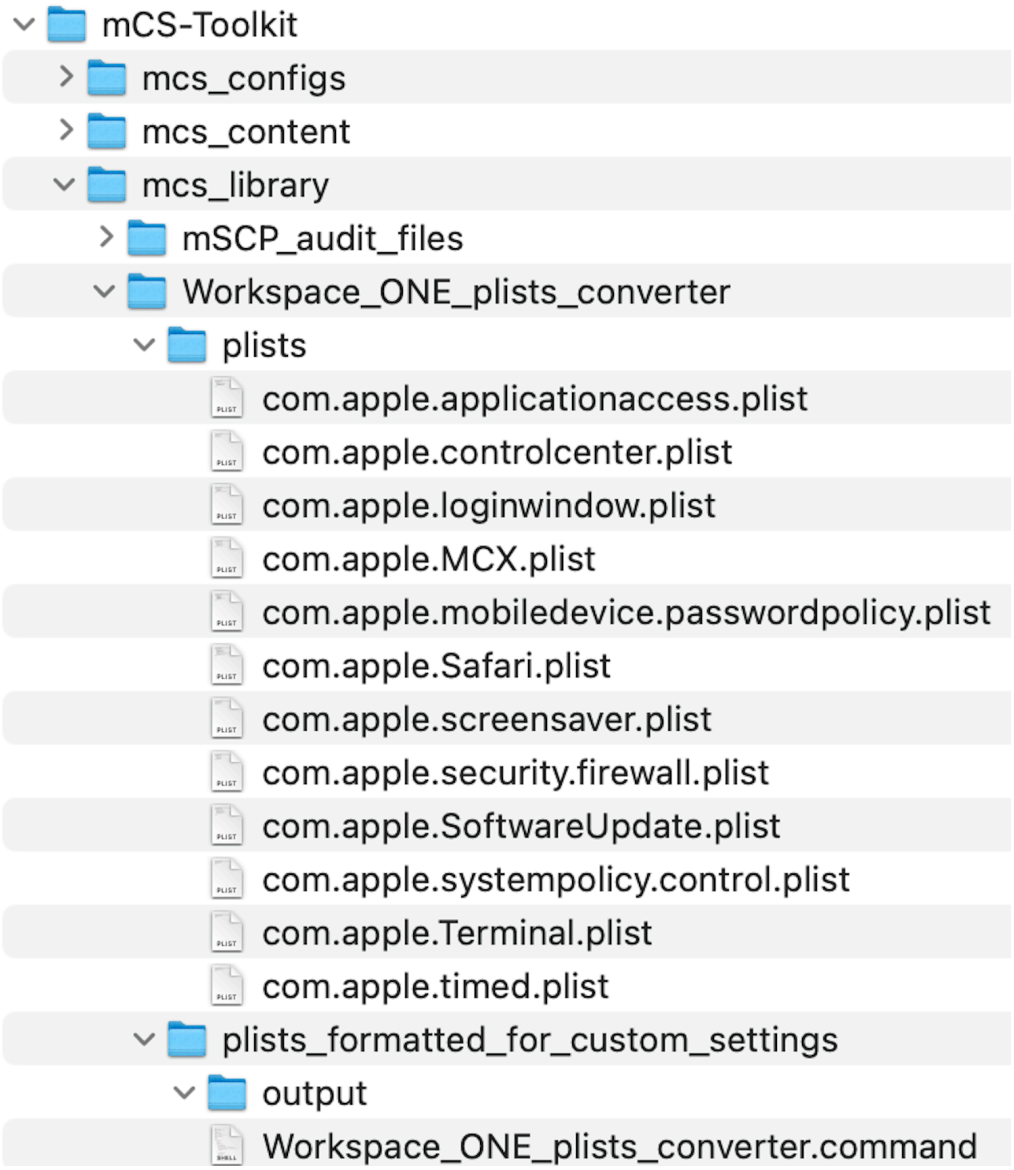
3A/ Omnissa Workspace ONE — Upload and distribute the configuration profiles

Follow these instructions to prepare the plist files for distribution as Custom Settings.

Open the "mCS-Toolkit" folder.

Go to "mcs_library > Workspace_ONE_plists_converter > plists".

Copy the plist files into the "plists" subfolder.

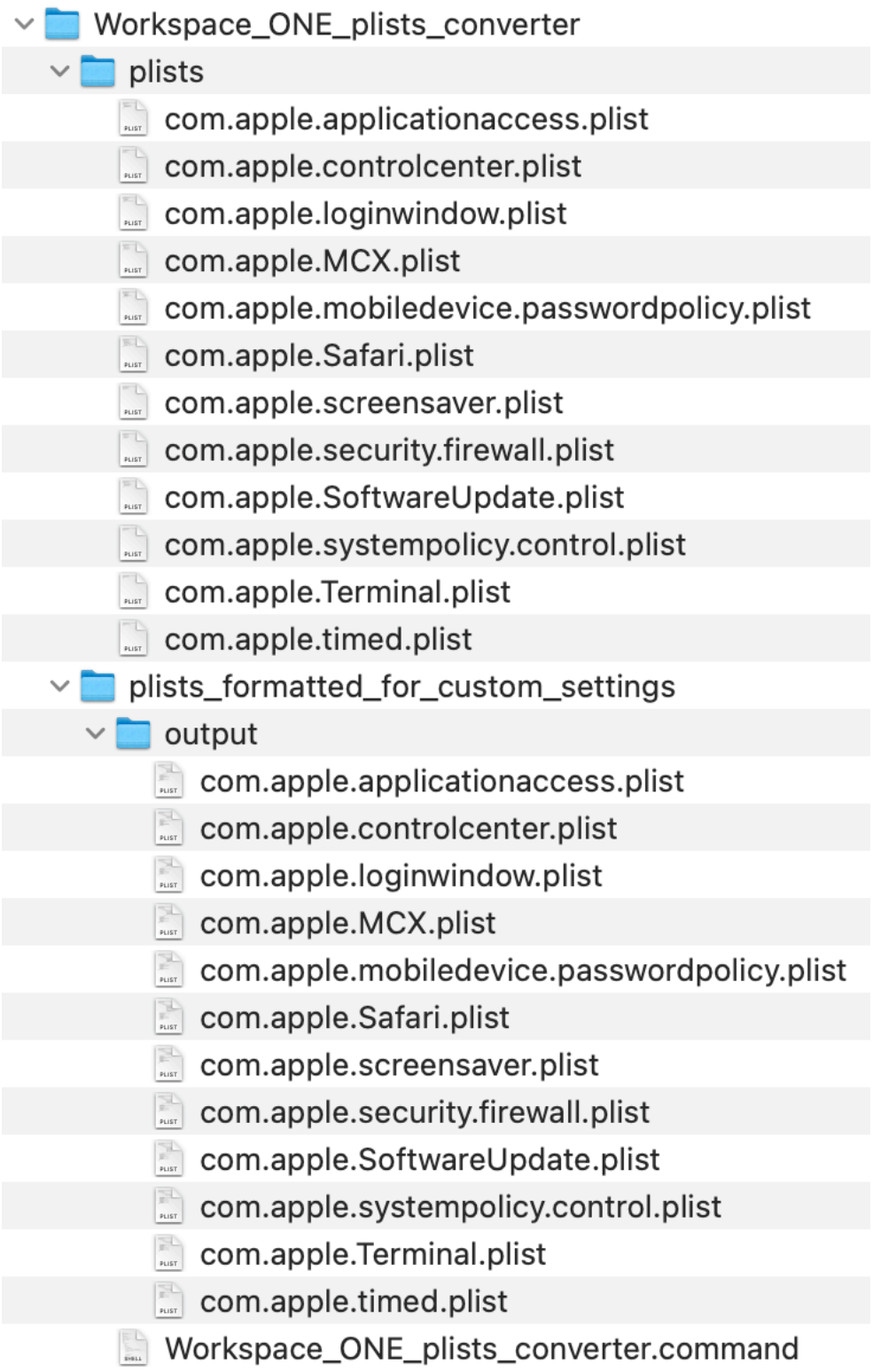


Open the "mCS-Toolkit" folder.

Go to "mcs_library > Workspace_ONE_plists_converter > plists_formatted_for_custom_settings".

Execute the "Workspace_ONE_plists_converter" script (select the script > right-click > Open).

If prompted to authorize the Terminal app to access files in a specific folder like your Desktop folder, click on "OK".



In this example, the script has converted several plist files for distribution as Custom Settings.

Distribute each converted plist file through the Custom Settings payload of a Custom configuration profile to the devices that will execute mCS.

Refer in this documentation to the section titled "Custom configuration profile" included in the chapter titled "Provisioning Omnissa Workspace ONE", and consult the MDM documentation if necessary, to revise the process.

3B/ All other MDM solutions — Upload and distribute the configuration profiles

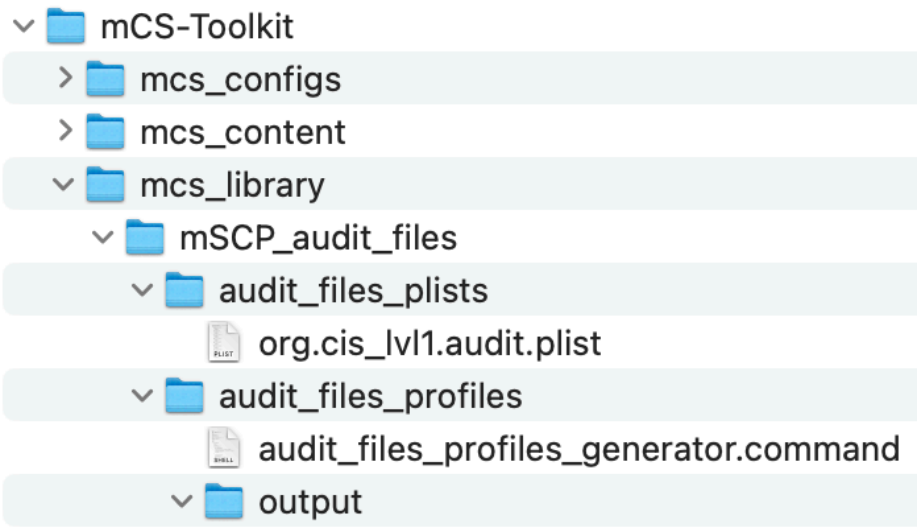
Distribute each configuration profile to the devices that will execute mCS.

Refer in this documentation to the section titled "Custom configuration profile" included in each chapter titled "Provisioning *MDM*", and consult the MDM documentation if necessary, to revise the process.

4/ Copy the audit file in the mCS Library

Open the "mCS-Toolkit" folder.

Go to "mcs_library > mSCP_audit_files > audit_files_plists".

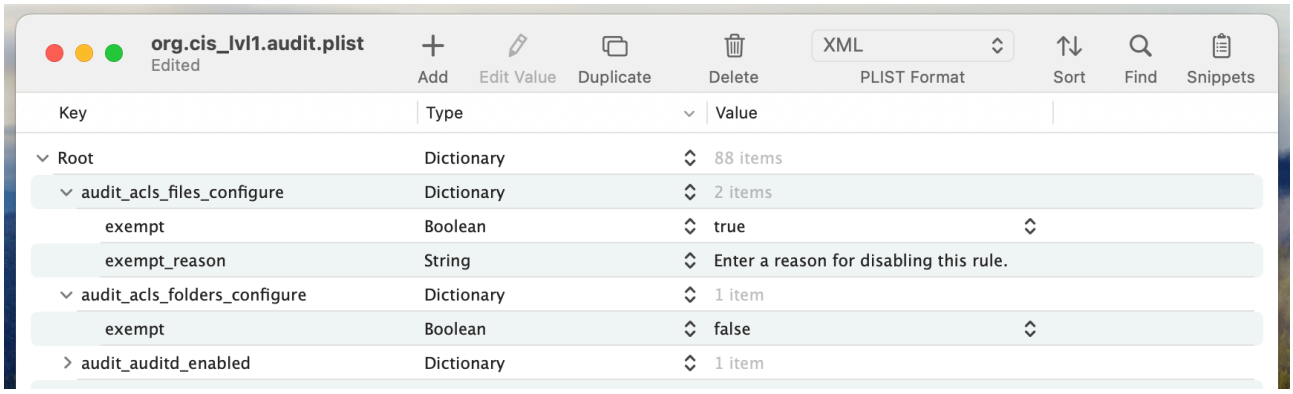


Copy the audit file into the "audit_files_plists" subfolder.

5/ Configure the baseline

This process allows setting exemptions to specific security rules based on your company's unique policies.

Open the audit file located in the "audit_files_plists" subfolder with your preferred Property List editor.



To disable a rule :
- set the "exempt" key to "true"
- add an "exempt_reason" key with the type "String" and enter a reason for disabling the rule (required).

Close the audit file.

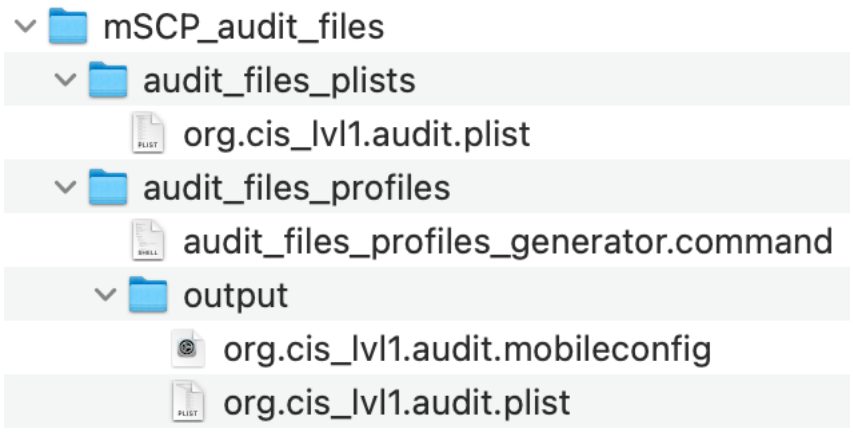
6/ Convert the audit file to a Custom configuration profile

Open the "mCS-Toolkit" folder.

Go to "mcs_library > mSCP_audit_files > audit_files_profiles".

Execute the "audit_files_profiles_generator" script (select the script > right-click > Open).

If prompted to authorize the Terminal app to access files in a specific folder like your Desktop folder, click on "OK".



In this example, the script has converted one audit file into two files :
- one Custom configuration profile with the extension ".mobileconfig"
- one Custom configuration profile with the extension ".plist".

Those two profiles are ready to be deployed by an MDM (only one must be scoped to a specific device) :
- the one with the extension ".plist" is needed if the MDM solution is Omnissa Workspace ONE, and its content is used to populate a Custom Settings payload
- the one with the extension ".mobileconfig" is needed for all other MDM solutions.

7/ Upload and distribute the generated Custom configuration profile

Distribute the generated Custom configuration profile to the devices that will execute mCS.

Refer in this documentation to the section titled "Custom configuration profile" included in each chapter titled "Provisioning *MDM",* and consult the MDM documentation if necessary, to revise the process.

Step 4 : Configure mCS for Jamf Pro or another MDM

Implementing mSCP compliance involves editing keys in the mCS configuration file which are detailed in the Dictionary.

• **Key located inside the INTEGRATIONS Dictionary**

MSCP_INTEGRATION : set to "true"

• **Keys located inside the INTEGRATIONS > MSCP_CONFIGURATION Dictionary**

COMPLIANCE_REMEDIATION : set to "true" to trigger the remediation planned by the compliance script before a scan

COMPLIANCE_REMEDIATION_METHOD
For each macOS version supported in your environment, specify the method to use for executing the script for compliance remediation. Example of a value :
- `script:cis_lvl1_compliance.sh` : name of the script embedded in the mCS-Content
- `event:Sequoia_cis_lvl1-fix` : name of the Jamf Pro policy custom trigger
- `id:100` : Jamf Pro policy ID (alternative to a custom trigger)

Note : Depending on the MDM used, keep only one dictionary named exactly "COMPLIANCE_REMEDIATION_METHOD" from the template.

COMPLIANCE_REMEDIATION_SETTINGS > PREFER_PRIVILEGED_HELPER : set to "true" to delegate the execution of the script to the mCS Privileged Helper

COMPLIANCE_SCAN : set to "true" to trigger a scan

COMPLIANCE_SCAN_METHOD
For each macOS version supported in your environment, specify the method to use for executing the script for compliance scan. Example of a value :
- `script:cis_lvl1_compliance.sh` : name of the script embedded in the mCS-Content
- `event:Sequoia_cis_lvl1-check` : name of the Jamf Pro policy custom trigger
- `id:101` : Jamf Pro policy ID (alternative to a custom trigger)

Note : Depending on the MDM used, keep only one dictionary named exactly "COMPLIANCE_SCAN_METHOD" from the template.

COMPLIANCE_SCAN_SETTINGS > PREFER_PRIVILEGED_HELPER : set to "true" to delegate the execution of the script to the mCS Privileged Helper

COMPLIANCE_SCORE_FAILURE : percentage of compliance score below which a failure message is processed by the Compliance report and webhook alerts

COMPLIANCE_SCORE_WARNING : percentage of compliance score below which a warning message is processed by the Compliance report and webhook alerts

• **Key located inside the INTEGRATIONS > SLACK_CONFIGURATION Dictionary**

MESSAGE_MSCP_COMPLIANCE : message sent for the compliance report

The variable `:mSCPComplianceReport:` is replaced by the report.

• **Key located inside the INTEGRATIONS > TEAMS_CONFIGURATION Dictionary**

MESSAGE_MSCP_COMPLIANCE : message sent for the compliance report

The variable `:mSCPComplianceReport:` is replaced by the report.

Implementing external modules

mCS supports executing external modules in the form of shell scripts.

This capability requires additional configuration steps, which are outlined in this chapter.

Step 1 : Prepare the scripts

A template named "placeholder.sh" is available in mCS-Toolkit > mcs_library.

The script is executed by the shell interpreter specified in the shebang (e.g. /bin/sh).

The entire parameter string sent to the script is stored in a variable named "PARAMETERS".

The result of the processing is sent back to mCS using an exit code and a line formatted like this :

```
echo "<result>Script executed with parameters $PARAMETERS</result>"
```

The name between the <…> and </…> (with a forward slash) is not important as long as it remains consistent before and after the enclosed string.

Disclaimer : The customer acknowledges and agrees that they are solely responsible for the actions and outcomes of the scripts, especially when executed through the Privileged Helper. It is the customer's responsibility to ensure compliance with all applicable laws and regulations.

Step 2 : Embed the scripts into the mCS Content

Refer to the chapter titled "mCS Content building" to learn how to embed the scripts into the mCS Content distributed to devices installed with mCS. Note that the scripts must have distinct names, must avoid special characters, and must be stored at the root of the "content" folder.

Step 3 : Declare the scripts in the mCS configuration file(s)

As detailed in the mCS Dictionary, each script is configured as an item in the EXTERNAL_MODULES > LIST array, using a dictionary that contains several keys :
- TYPE : set to "script"
- DISPLAYNAME : the name displayed for the script in the stages list
- ICON : the icon embedded in the mCS-Content which is displayed for the script in the stages list
- FILENAME : the name of the script embedded in the mCS-Content which is executed
- PARAMETERS : the parameters (options and arguments) passed to the script upon execution
- PREFER_PRIVILEGED_HELPER : set to "true" to delegate the execution of the script to the mCS Privileged Helper
- TIMEOUT : the maximum wait time in seconds before waiting is interrupted.

Bear in mind that mCS will interrupt any script that is still running after the defined timeout, so set the timeout value accordingly.

Step 4 : Signature of the embedded scripts

To prevent the execution of tampered scripts, mCS integrates a mechanism that verifies the script's signature before execution.

First, a signed hash of each script referenced by the mCS configuration file and present in the "content" folder is calculated and automatically stored in an additional SIGNATURE key within the script dictionary.

mCS-Core and mCS-Content are deployed alongside the mCS configuration file converted to a Custom configuration profile.

Before a deployed script is executed by a workflow, its hash is calculated and compared to the signed hash stored in the Custom configuration profile. If the hashes match, the script is executed.

When the script terminates with a zero exit code, the Flight Recorder reports the display name and the string that the script sends to standard output, which is enclosed within tags of any name, following the structure <tag>string</tag> (e.g. <result>Printer management privileges granted</result>).

When the script terminates with a non-zero exit code, the Flight Recorder reports the display name and the exit code.

When the script terminates due to tampering detection, the Flight Recorder reports the display name along with the message "Unverified signature".

When the script terminates due to a timeout, the Flight Recorder reports the display name along with the message "Timeout exceeded".

When script execution is delegated to the Privileged Helper, its signature is verified before invocation. If the script is not executed due to tampering detection, the Flight Recorder reports the script display name along with the message "Unverified Privileged Helper". The Privileged Helper will also abort script execution if it cannot verify the signature of the mCS core script.

The signature of the embedded scripts is processed during the conversion described in the chapter titled "mCS configuration files to Custom configuration profiles conversion".

Please note that **if the MDM solution is Jamf Pro**, following the instructions in this chapter is mandatory. However, once the conversion is complete, you can choose to ignore the output because the original .plist file now contains the required script signatures.

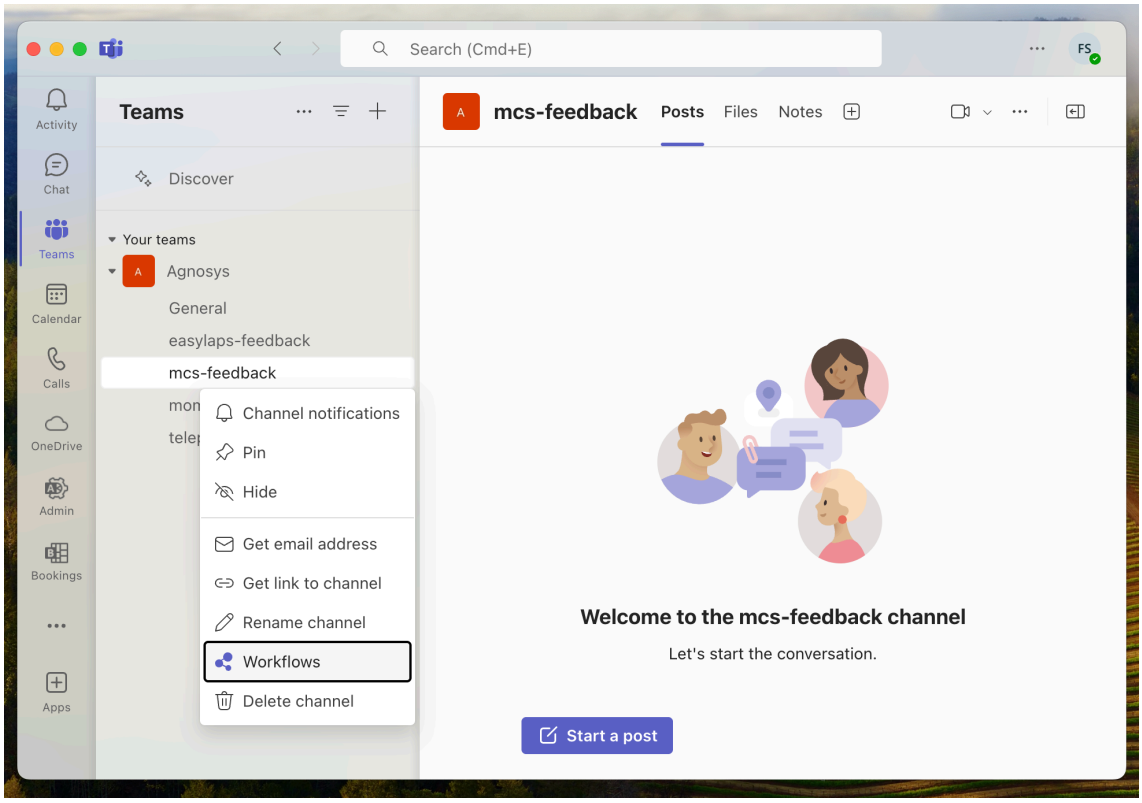
Step 5 : Maintenance of the embedded scripts

If you need to distribute updated versions of the embedded scripts, please keep in mind that you need to :
- build the mCS-Content with the updated scripts, then distribute it
- execute a new conversion to distribute an updated Custom configuration profile that includes the signatures of the updated scripts.

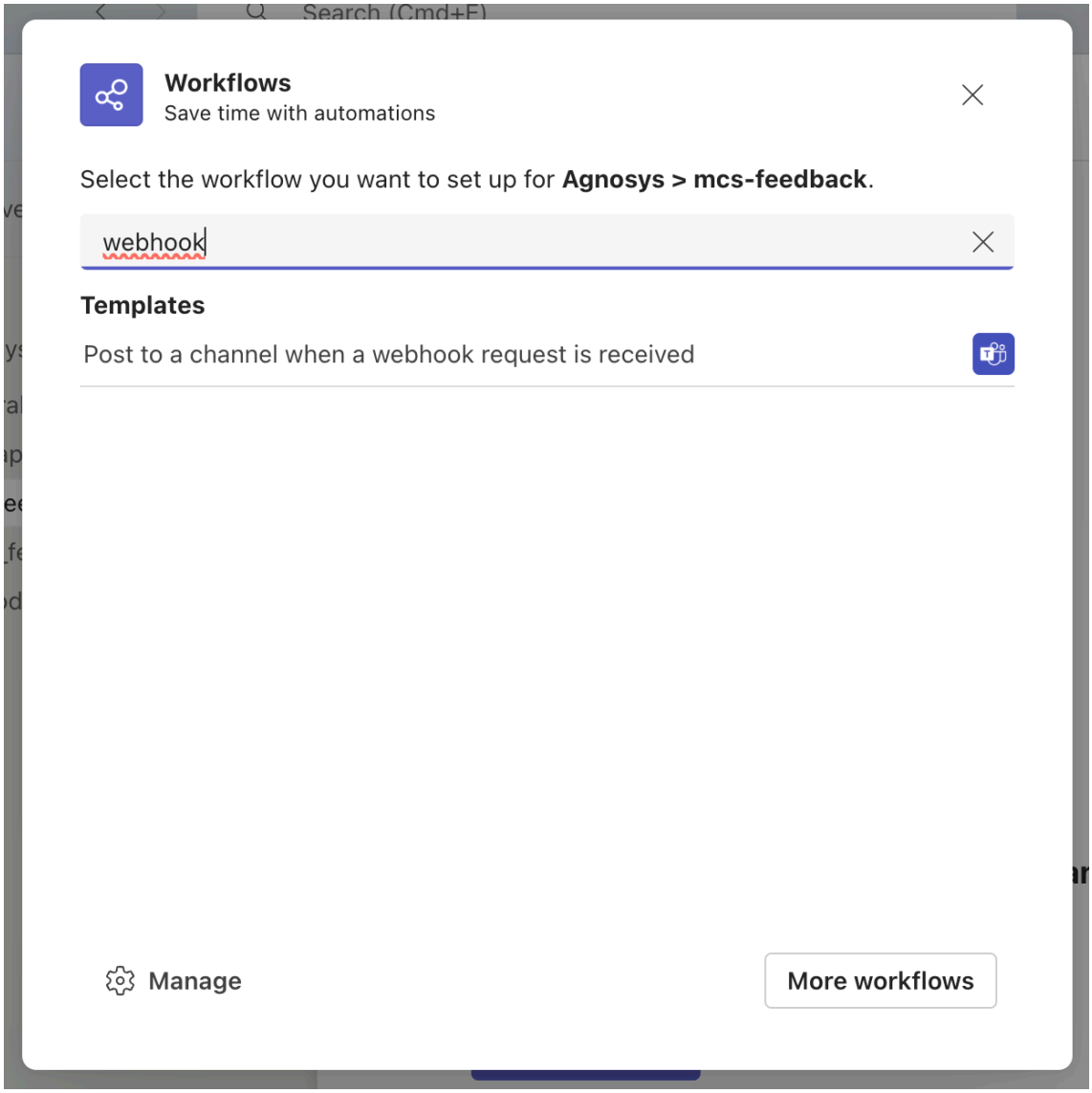
Implementing Microsoft Teams integration

mCS can report to a dedicated Microsoft Teams channel the successive status of a running workflow.

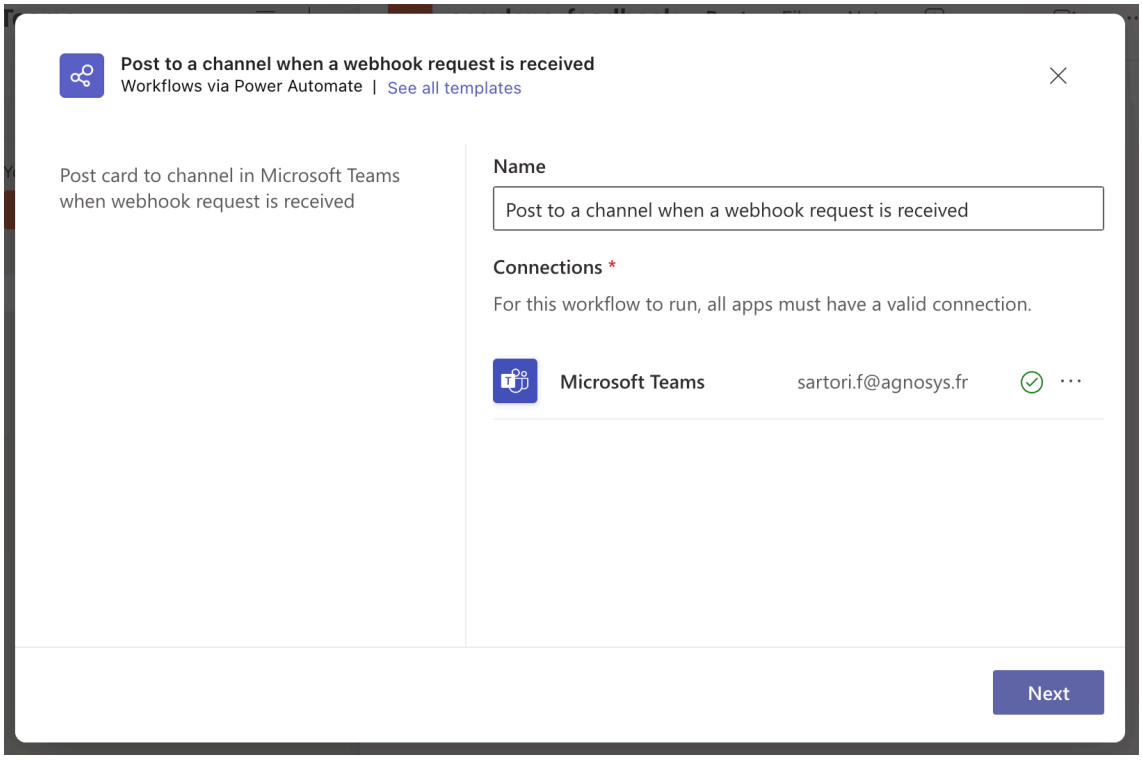
First create a dedicated Microsoft Teams channel of type "Standard" (everyone on the team has access).



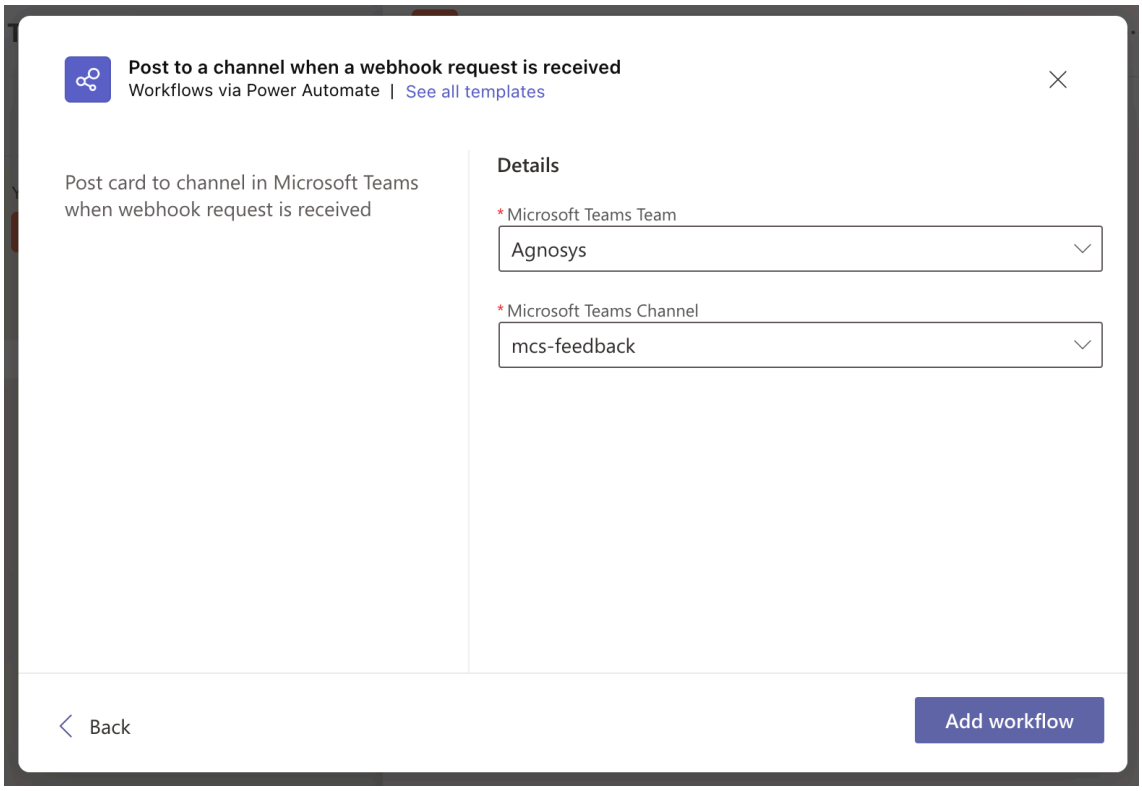
Click on the "..." button to the right of the channel name, then select "Workflows".



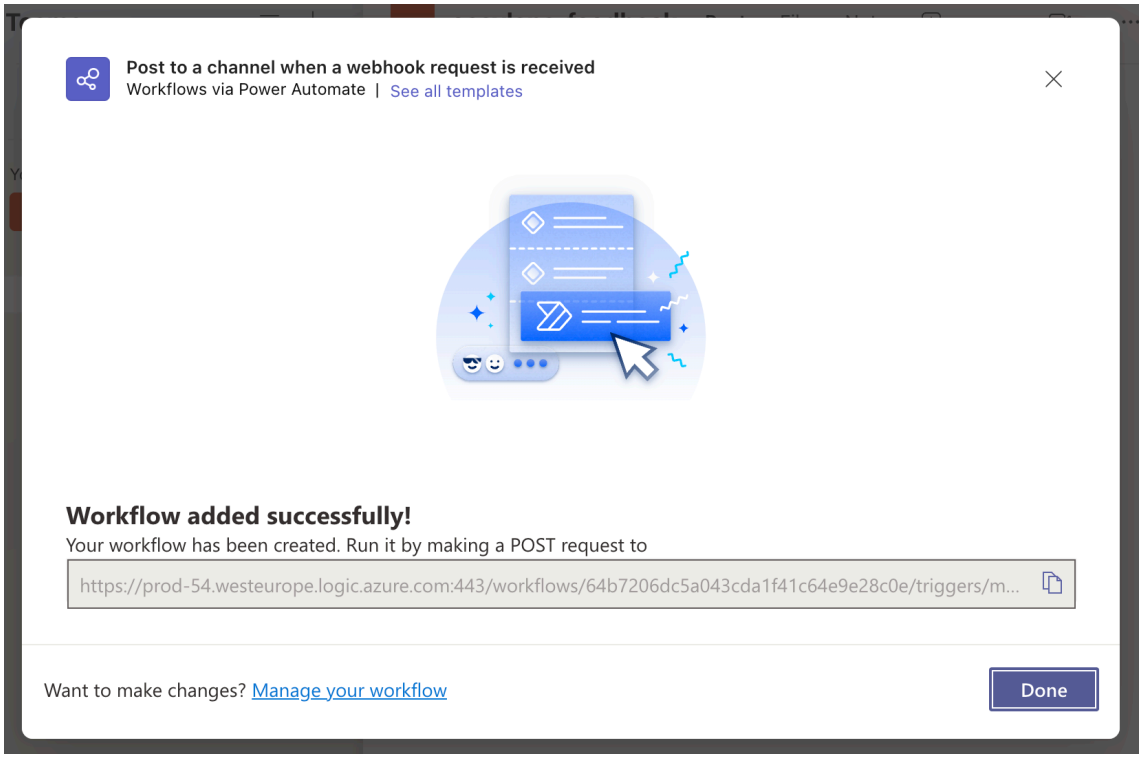
Type "webhook" in the search field, then click on "Post to a channel when a webhook request is received".



Once the connection is indicated as valid with a green tick, click on "Next".



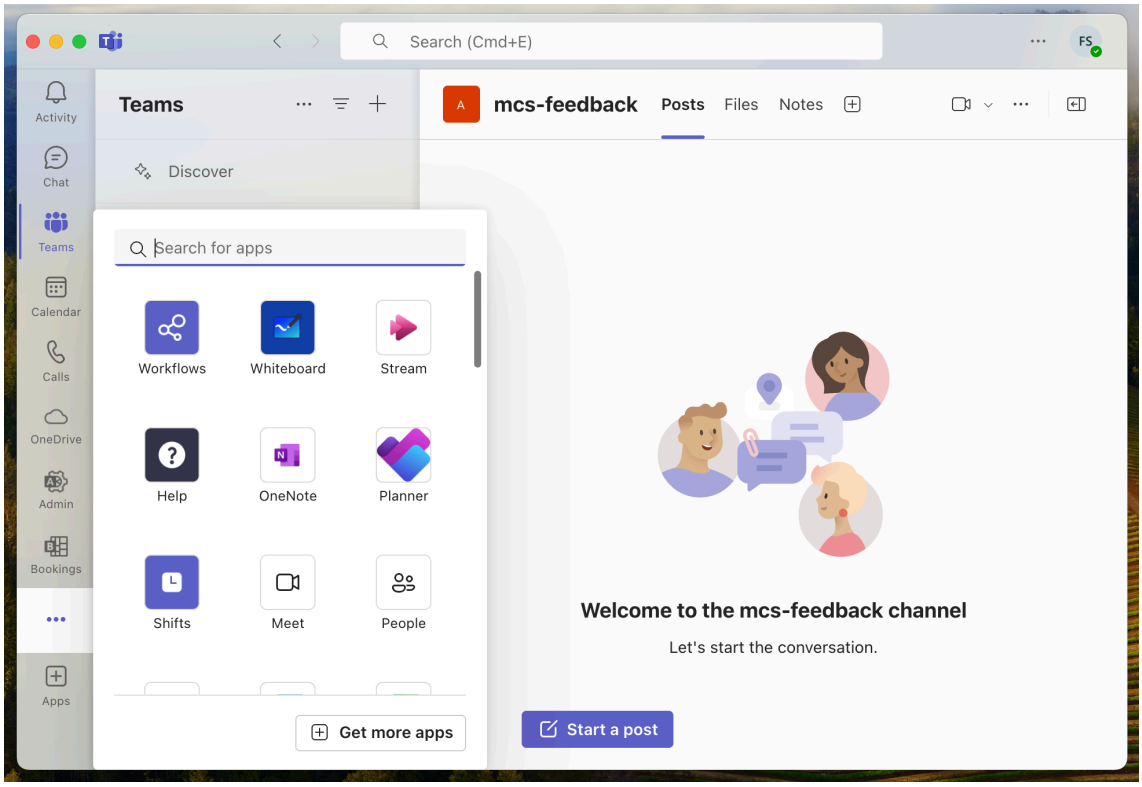
Check the Microsoft Teams team and the Microsoft Teams channel, then click on "Add workflow".



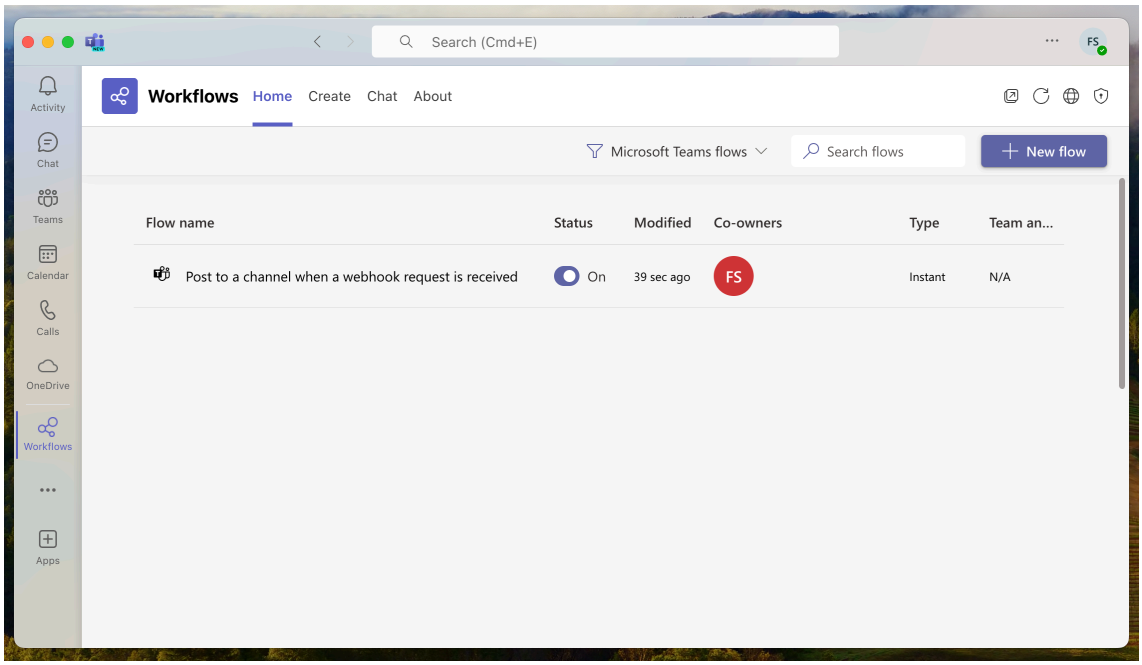
Click on the button to the right of the URL displayed to copy it, then follow these instructions :
- open the "mCS-Toolkit" folder
- open the "mcs_secrets" subfolder
- execute the "mcs_rsa_engine" script (double-click on the .command file)
- paste the copied URL
- the URL is encrypted, displayed and then decrypted for sanity check
- copy the encrypted URL (one-line string ending exactly with two "=" characters)
- paste the encrypted URL in the INTEGRATIONS > TEAMS_CONFIGURATION > INCOMING_WEBHOOK_URL key.

Make sure that the INTEGRATIONS > TEAMS_INTEGRATION key is set to "true".

Back in the pane, click on "Done".



Click on the "..." Button in the sidebar, then click on "Workflows" to display this app.



Click on the created workflow to display its details if you want to.

mCS configuration files to Custom configuration profiles conversion

If the MDM solution is Jamf Pro — This step is **optional** because this MDM offers to upload an mCS configuration file directly into a Configuration profile that includes an "Application & Custom Settings" payload. However, if you prefer to upload in Jamf Pro a pre-built Custom configuration profile, follow these instructions.

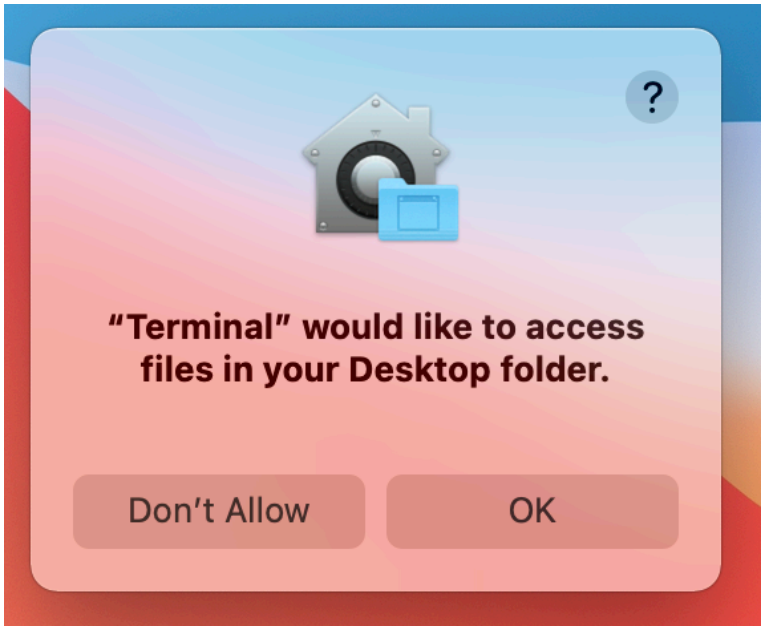
If the MDM solution is not Jamf Pro — This step is **required**. Follow these instructions.

Open the "mCS-Toolkit" folder.

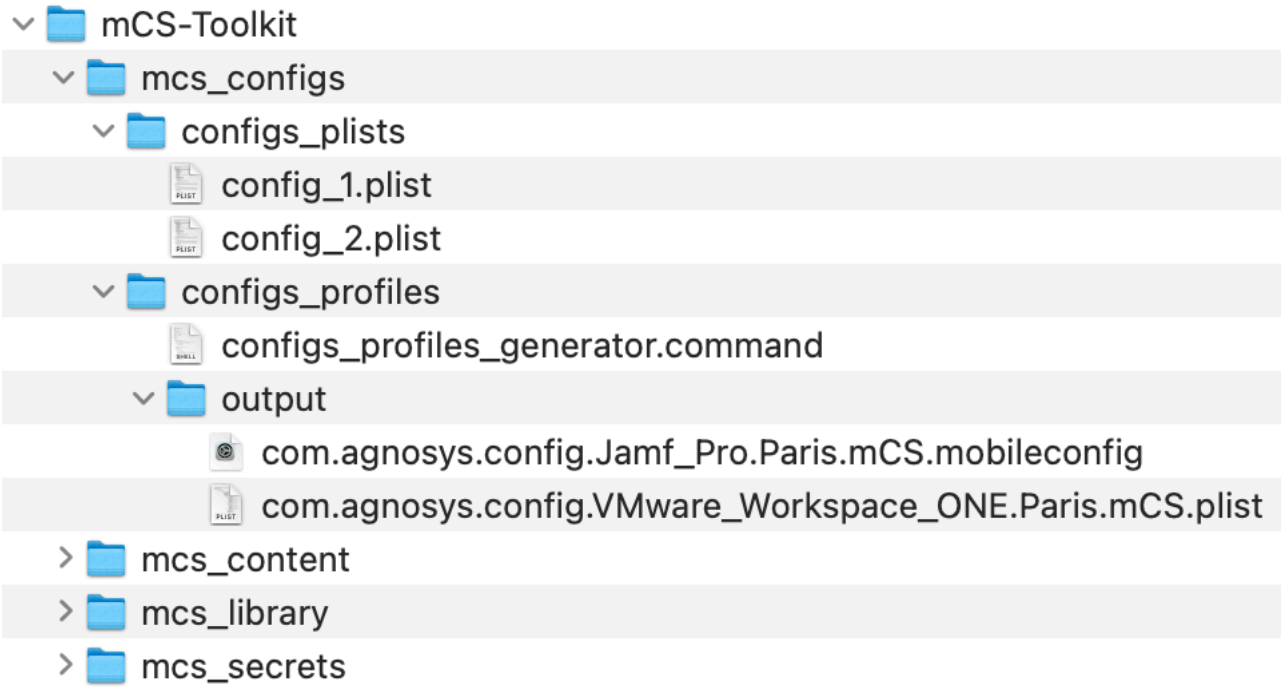
Open the "mcs_configs" subfolder.

Open the "configs_profiles" subfolder.

Execute the "configs_profiles_generator" script (select the script > right-click > Open).



If prompted to authorize the Terminal app to access files in a specific folder like your Desktop folder, click on "OK".



In this example, the script has converted two mCS configuration files into two Custom configuration profiles ready to be deployed by the MDM (only one must be scoped to a specific device).

If the MDM solution is Omnissa Workspace ONE, please note that the file extension is ".plist" instead of ".mobileconfig". The content of the file is used to populate a Custom Settings payload.

mCS Content building

All pictures (.png), scripts (.sh) and files referenced in the mCS configuration file must be embedded in the mCS-Content package.

The mCS-Content package is deployed alongside the Custom configuration profile derived from the mCS configuration file, via the MDM.

Depending of the MDM used and the distribution method implemented, the signature of the package, even always recommended, may become a requirement. However, the notarization is never required.

Package signature requirement

• **FileWave**

No signature required.

• **Hexnode UEM**

Signature required.

• **Jamf Now**

Signature required.

• **Jamf Pro**

This documentation plans that the mCS-Content package is deployed via the Packages payload of a policy which does not require that the package is signed.

• **Jamf School**

No signature required.

• **JumpCloud**

No signature required.

• **Kandji**

No signature required.

• **Meraki Systems Manager**

No signature required.

• **Microsoft Intune**

No signature required when the package is provisioned as a macOS app.

• **Miradore**

Signature required.

• **Mosyle Business**

No signature required unless the option "Install with Apple Protocol" is enabled in the deployment configuration.

• **Mosyle Manager**

No signature required unless the option "Install with Apple Protocol" is enabled in the deployment configuration.

• **Omnissa Workspace ONE**

This documentation plans that the mCS-Content package is deployed as a regular package with the "Full Software Management" deployment type which does not require that the package is signed.

• **SimpleMDM**

Signature required.

Package signature options

These are some options to sign the mCS-Content package.

• Subscribe to an Apple Developer program, create a "Developer ID Installer" certificate and use it to sign the package.

• With Jamf Pro : create a certificate with the Jamf Pro's Built-in CA and use it to sign the package with these informations in mind :

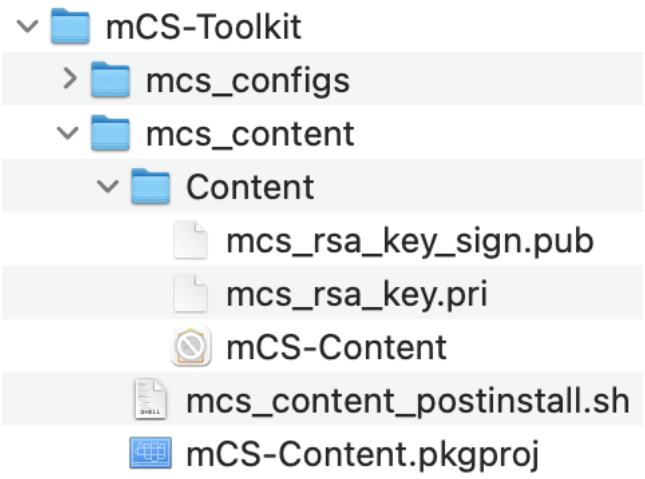
- the certificate forged with the Jamf Pro's Built-in CA can be validated by a device only if it is already enrolled in Jamf Pro
- for more information, please consult this article : https://docs.jamf.com/technical-articles/Creating_a_Signing_Certificate_Using_Jamf_Pros_Built-in_CA_to_Use_for_Signing_Configuration_Profiles_and_Packages.html
- once the signing identity is available in the "login" keychain, click on "Certificates" to check the certificate associated with the private key, then sign the unsigned package produced by the Packages app with the following command :

```
productsign --sign "name_of_certificate" mCS-Content.pkg mCS-Content_signed.pkg
```

- ignore the section below titled "Signing configuration" as the package is now signed.

• Open an mCS support ticket to get the package signed by Agnosys or your integrator.

Content gathering



Open the "mCS-Toolkit" folder.

Open the "mCS_content" subfolder.

Open the "Content" folder.

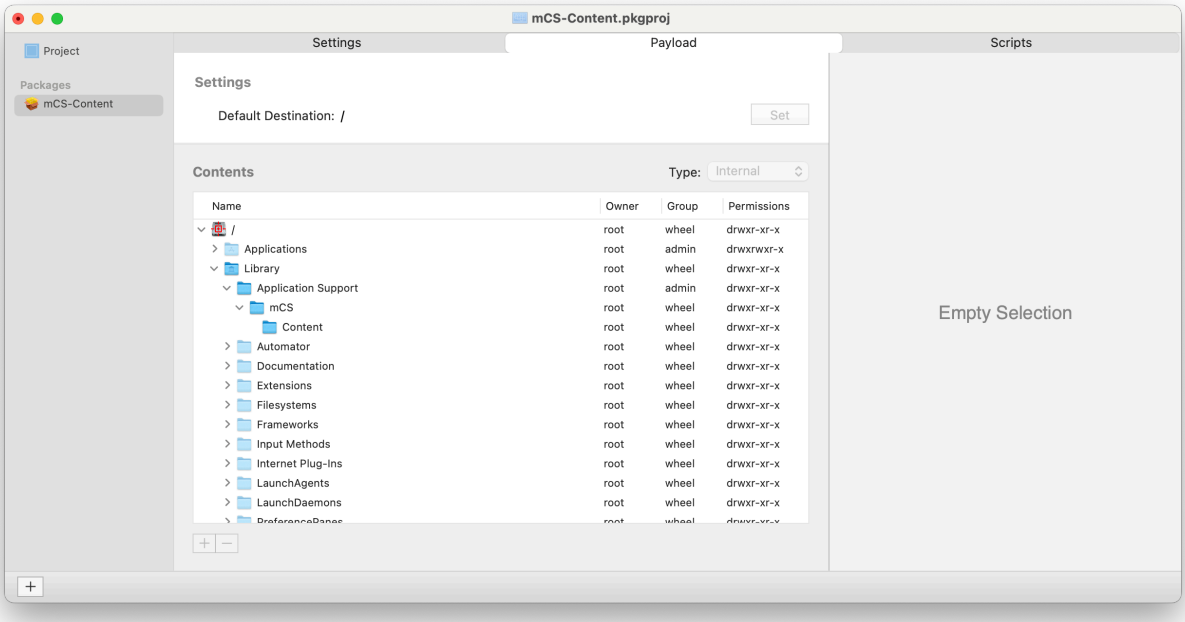
Copy your content in the "Content" folder, alongside the "mcs_rsa_key_sign.pub" file, the "mcs_rsa_key.pri" file, and the detection app named "mCS-Content.app".

Project opening

Open the "mCS-Toolkit" folder.

Open the "mcs_content" subfolder.

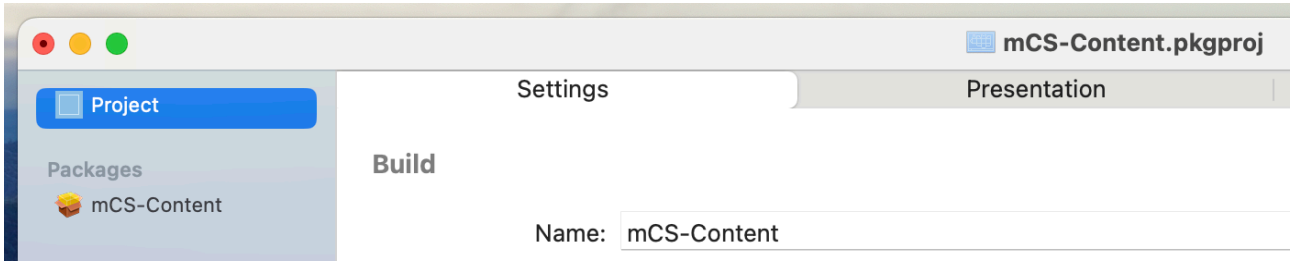
Open the "mCS-Content.pkgproj" file with the Packages app.



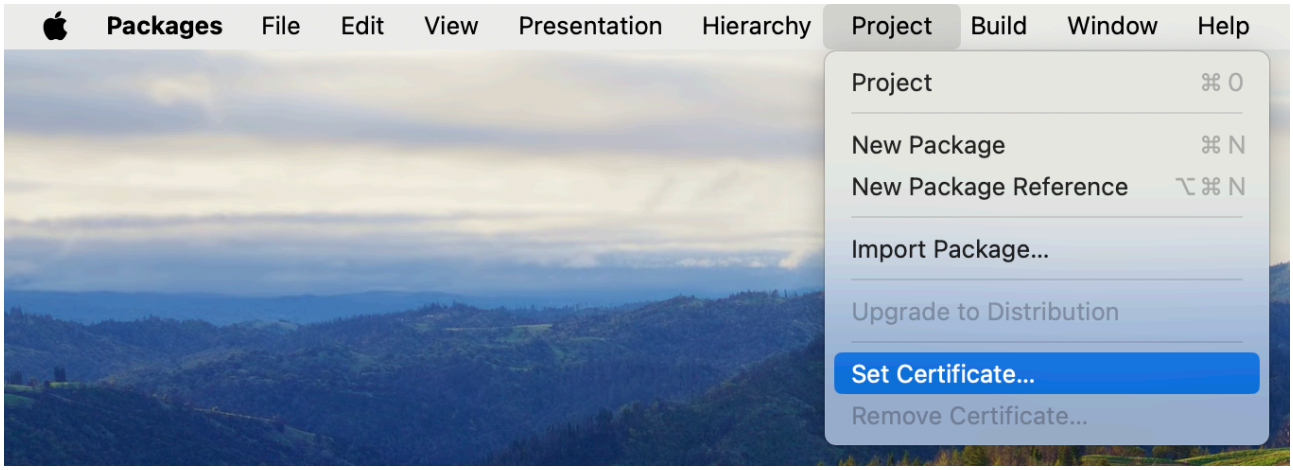
The generated package will embed the "Content" folder for an installation in /Library/Application Support/mCS/

Signing configuration

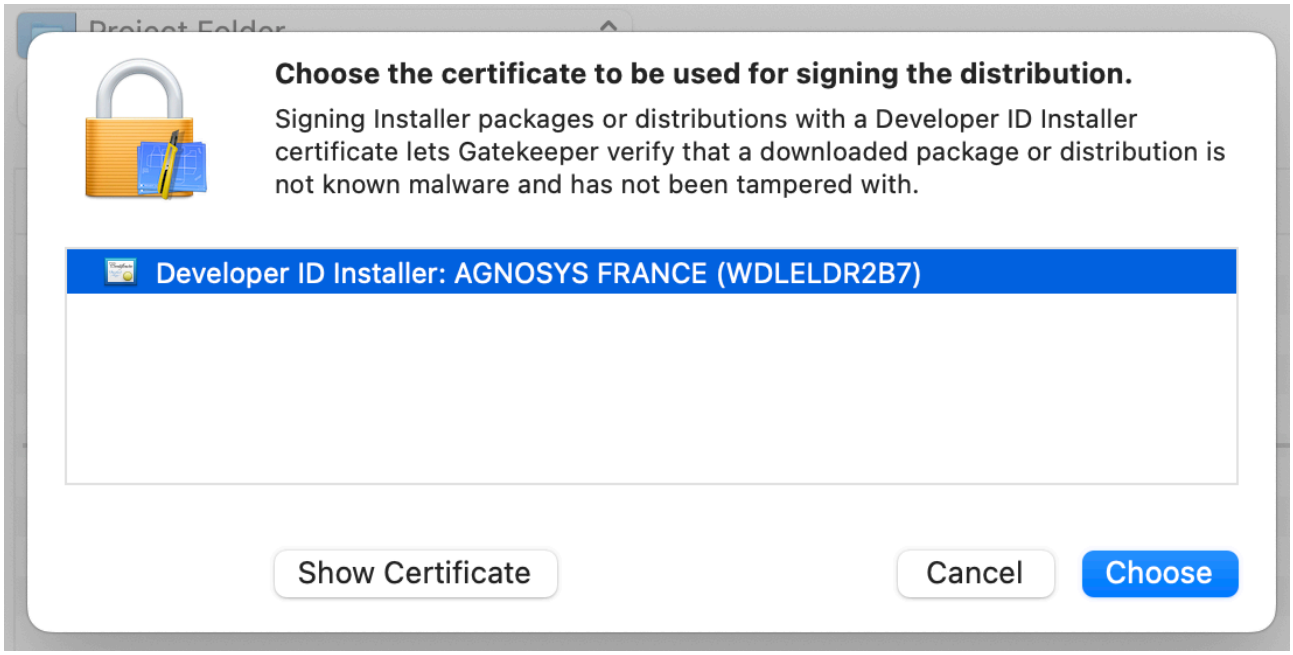
Open the Keychain Access app and check that your "Developer ID Installer" certificate is installed in the "login" keychain.



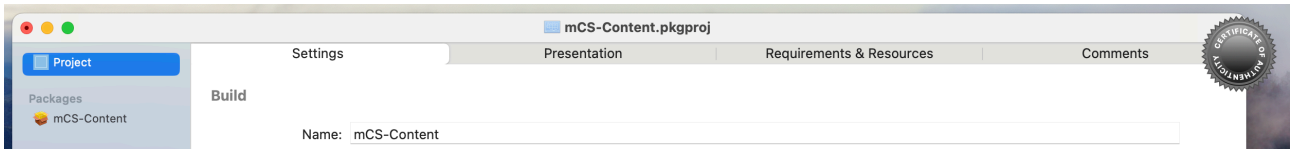
Click on "Project" then select the "Settings" tab.



Select Project > Set Certificate.

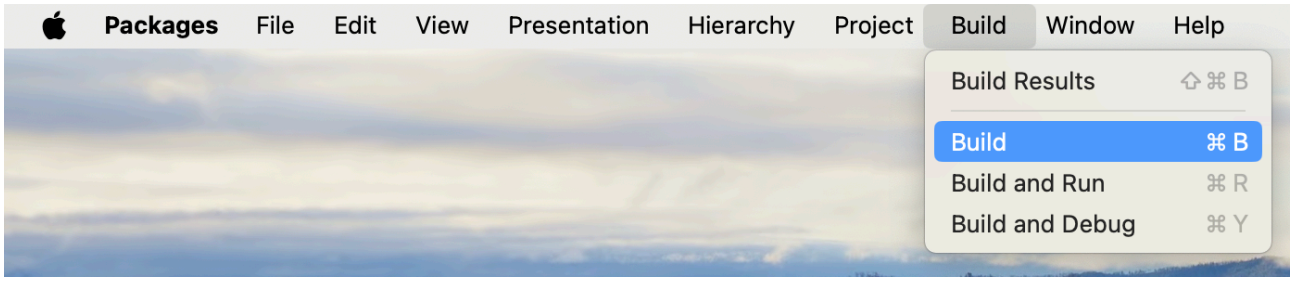


Select your "Developer ID Installer" certificate and click on "Choose".

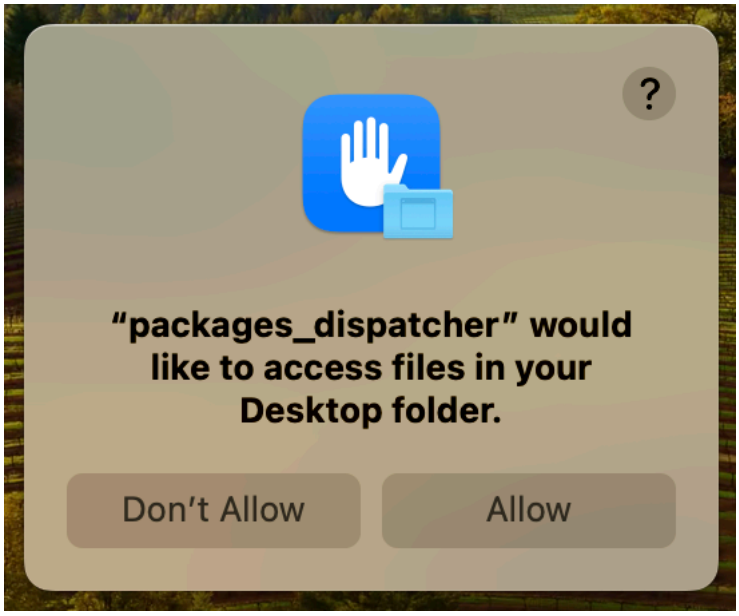


A "Certificate of authenticity" badge is now visible in the upper right corner of the project window.

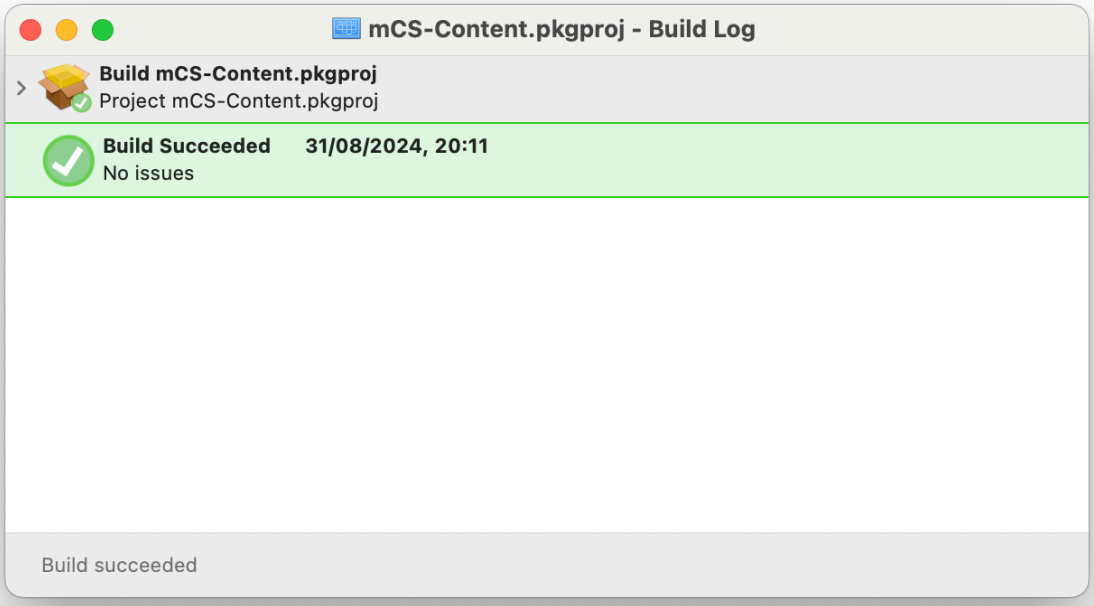
Project building



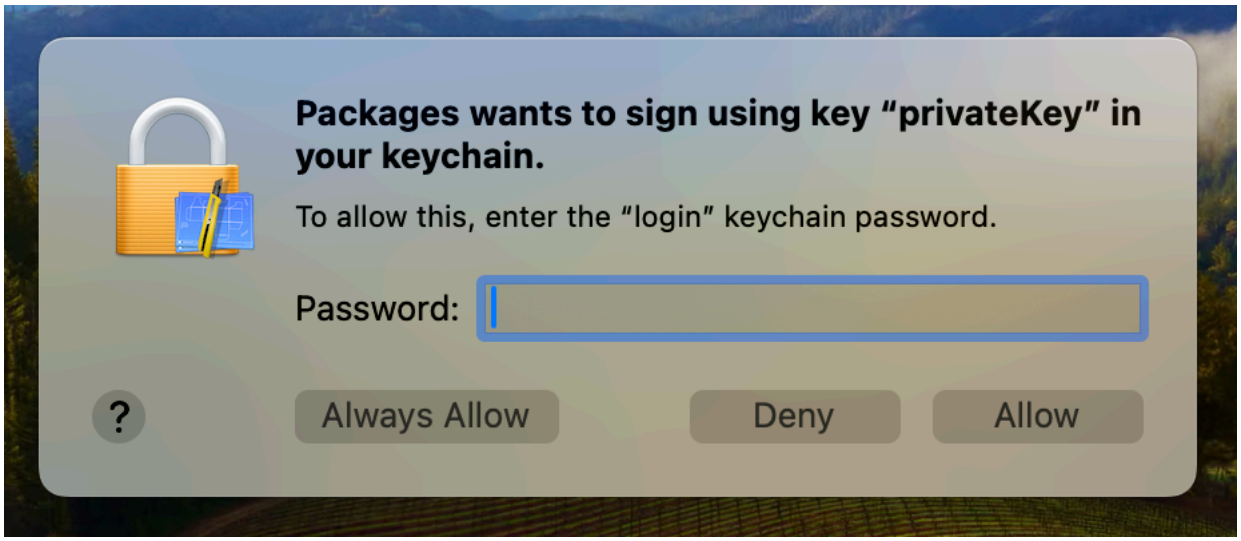
Select Build > Build.



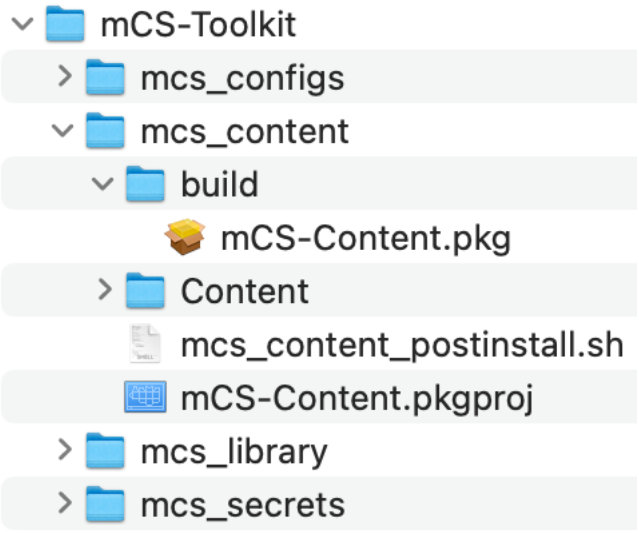
If prompted to authorize the Packages app to access files in a specific folder like your Desktop folder, click on "OK".



The Build Log must display "Build Succeeded — No issues".



During the building, if you previously set a signing certificate, you may be prompted to authorize Packages to access the private key of your "Developer ID Installer" certificate. Enter your account's password and click on "Always Allow".



The package is built at the following path :
mCS-Toolkit > mcs_content > build

You can now quit the Packages app. Choose to save the changes made to the project if you are offered to do so.

Configuration profiles requirements

This section details the configuration profiles required by mCS in addition to the mCS Custom configuration profile.

Privacy Preferences Policy Control

mCS includes a Privileged Helper to which it can delegate sensitive operations requiring extended privacy settings.

Scripts defined by the mSCP integration or declared in external modules can benefit from this Privileged Helper on demand.

Payload required : Privacy Preferences Policy Control

- Identifier Type : Bundle ID

    - Identifier : com.agnosys.mcs_privileged_helper

    - Code Requirement : `identifier "com.agnosys.mcs_privileged_helper" and anchor apple generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = WDLELDR2B7`

        Note : The string is a single-line string when pasted into the Code Requirement field, with no line breaks.

    - Validate the Static Code Requirement : No

    - App or Service :

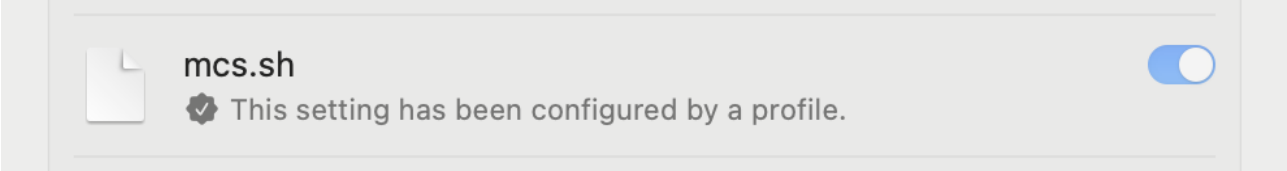
        - System Policy All Files : Allow

Background Item Management

With macOS 13 and later, a Background Item Management configuration profile must be deployed so that the system does not display a notification that mCS has installed a Login item that can run in the background and that can be managed in System Settings.

Payload required : Background Item Management

Rule Type : Label
Rule Value : com.agnosys.mcs



Once the configuration profile is deployed on a device running mCS, open System Settings > Login Items and check that the Login items "mcs.sh" is enabled and cannot be disabled.

Notifications

A Notifications configuration profile must be deployed to allow the informative interface to display notifications through swiftDialog instead of AppleScript.

Payload required : Notifications

Bundle Identifier : au.csiro.dialog
Settings :
- Notifications : Enable
- Style : Banner
- Show in Notification Center : Enable

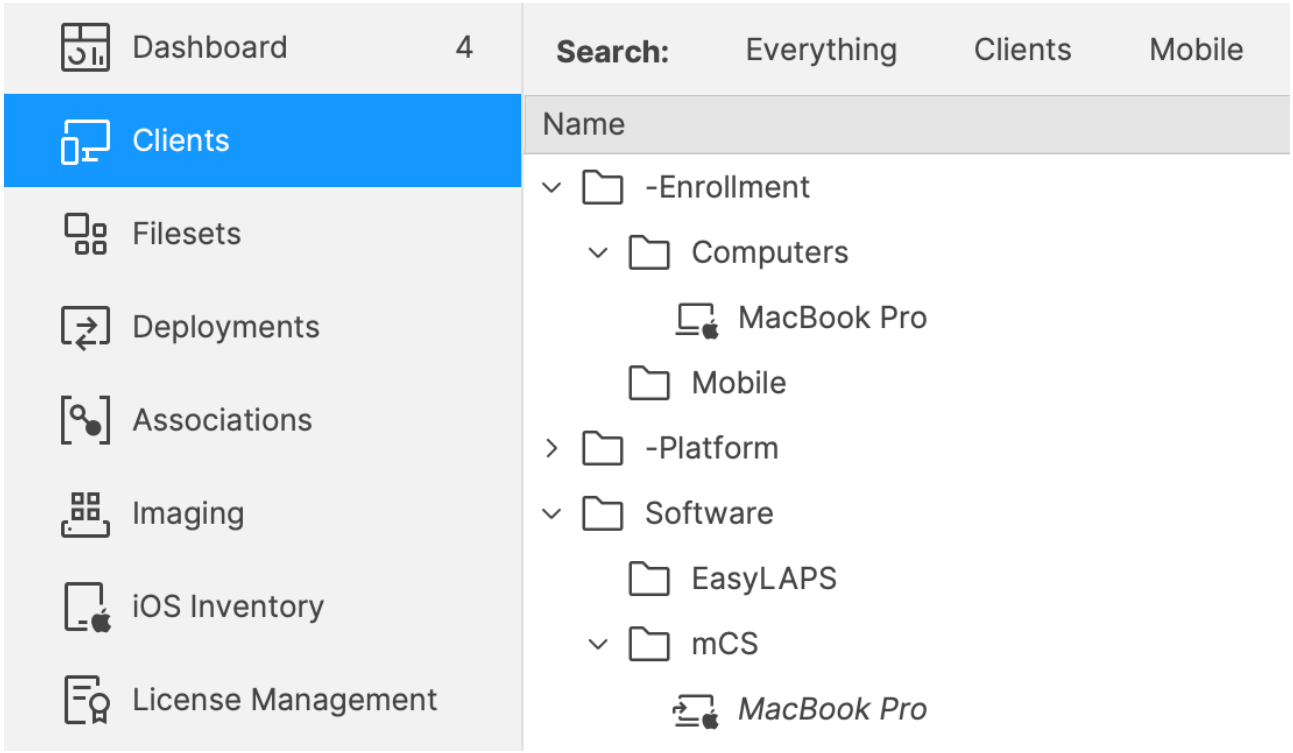
Provisioning FileWave

Three components must be automatically deployed to the devices :
- a Custom configuration profile
- an mCS-Content package
- the mCS-Core package.

This section outlines the key points for the provisioning of these three components in FileWave. Please refer to FileWave documentation for details not specific to mCS.

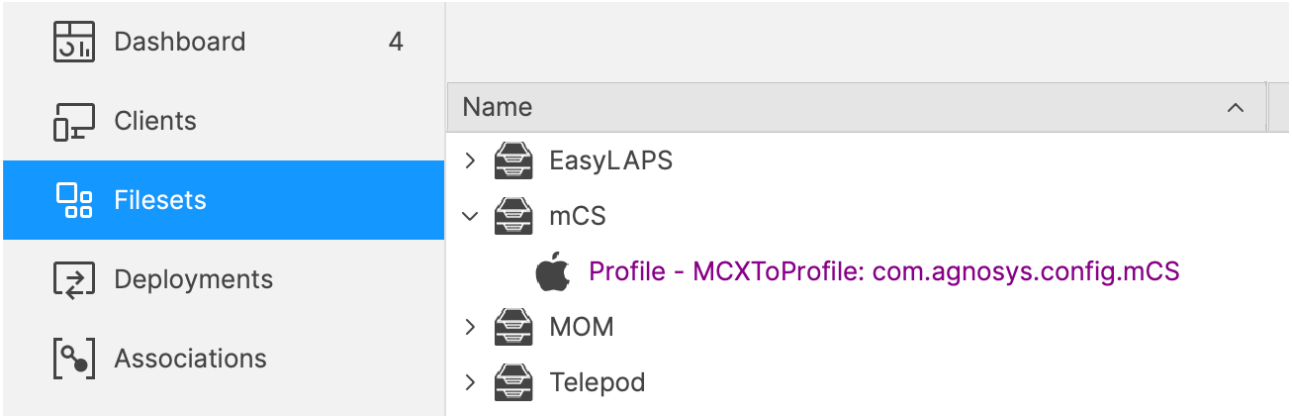
General configuration

In this example, the devices are part of group named "mCS" following this structure.



Custom configuration profile

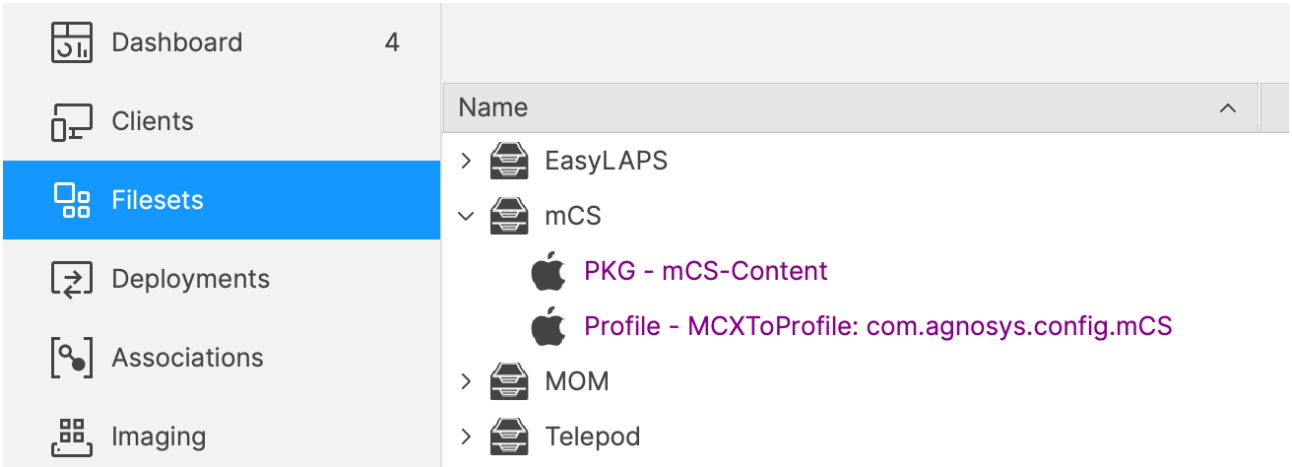
The Custom configuration profile is provisioned with the following steps :
- Filesets > New Fileset Group > Name : mCS
- Select the Fileset Group "mCS"
- Click on "New Desktop Fileset" then click on "Profile"
- In the Profile Editor, click on "Load Profile"
- Select the file : com.agnosys.config.FileWave.Paris.mCS.mobileconfig > Open
- Back to the Profile Editor, click on "Save".



The Custom configuration profile is linked to the "mCS" Fileset Group.

mCS-Content package

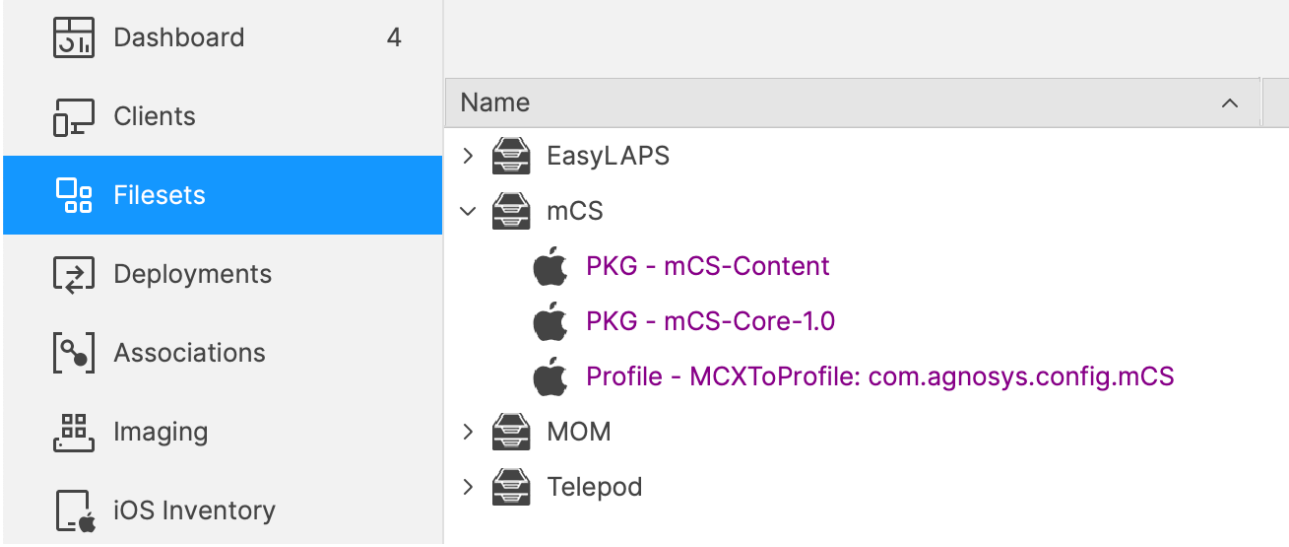
The mCS-Content package is defined with the following steps :
- Filesets > mCS
- Click on "New Desktop Fileset" then click on "MSI / PKG"
- Select the file : mCS-Content.pkg > Open.



The mCS-Content package is linked to the "mCS" Fileset Group.

mCS-Core package

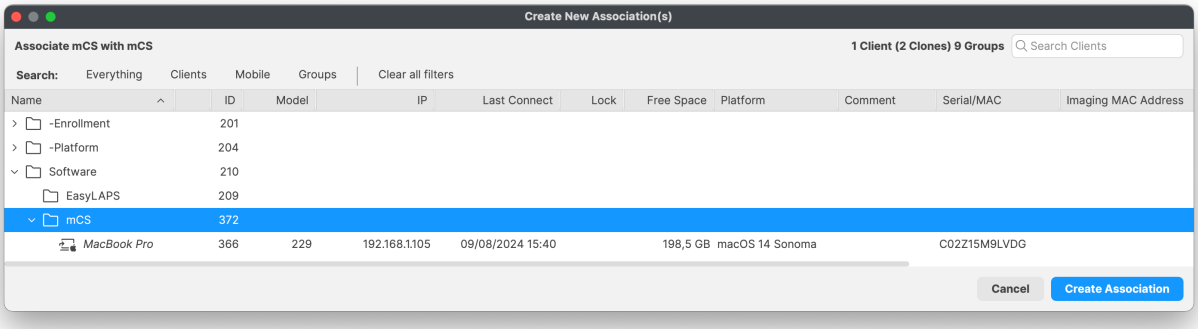
The mCS-Core package is defined with the following steps :
- Filesets > mCS
- Click on "New Desktop Fileset" then click on "MSI / PKG"
- Select the file : mCS-Core.pkg > Open.



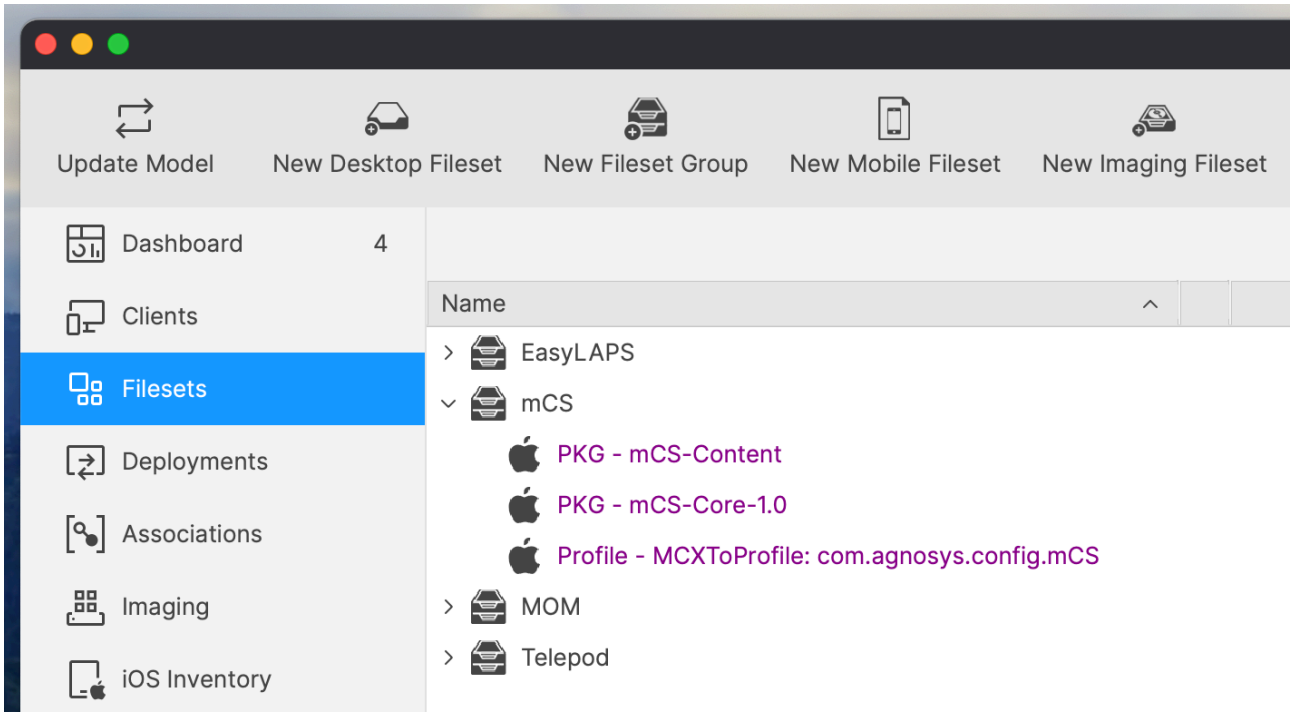
The mCS-Core package is linked to the "mCS" Fileset Group.

Deployment on the mCS group

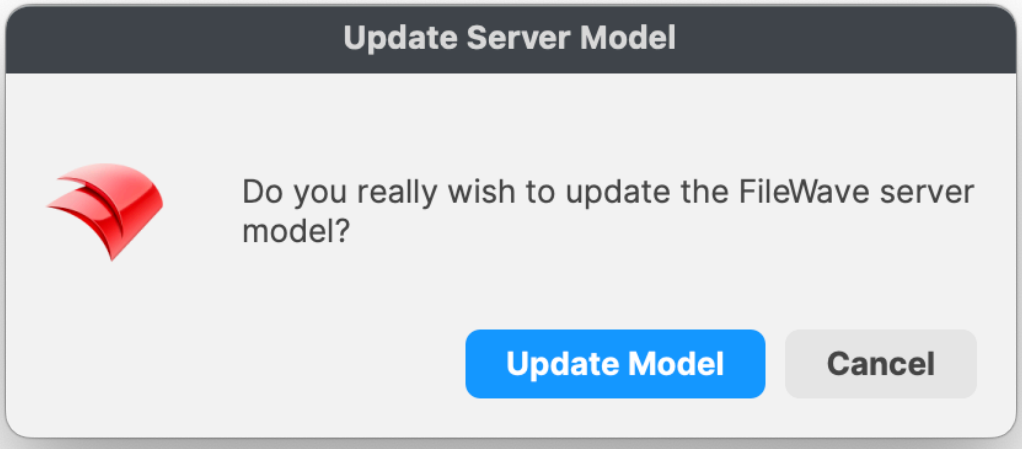
The mCS Fileset is associated to the mCS group with the following steps :
- Filesets > mCS
- In the toolbar, click on "New Association".



Select the mCS group and click on "Create Association".



In the toolbar, click on "Update Model".



Click on "Update Model".

Provisioning Jamf Pro

Three components must be automatically deployed to the devices :
- a Custom configuration profile
- an mCS-Content package
- the mCS-Core package.

This section outlines the key points for the provisioning of these three components in Jamf Pro. Please refer to Jamf Pro documentation for details not specific to mCS.

General configuration

In this example, the devices are part of a computer group named "mCS".

Custom configuration profile : importing a .plist file

Follow these instructions if you want to upload in Jamf Pro an mCS configuration file (.plist file) via a Configuration profile that includes an "Application & Custom Settings" payload.

The Custom configuration profile is provisioned with the following steps :
- Computers > Content Management > Configuration Profiles > New
- General
    - Name : a name of your choice (e.g. mCS-Custom configuration profile)
    - Level : Computer Level
    - Distribution Method : Install Automatically
- Application & Custom Settings > Upload > Add
    - Preference Domain : com.agnosys.config.mCS
    - Upload > config_1.plist
- Scope
    - Targets > Specific Computers
    - Add > Computer Groups > mCS > Add
- Save

Custom configuration profile : importing a .mobileconfig file

Follow these instructions if you want to upload in Jamf Pro a pre-built Custom configuration profile (.mobileconfig file) generated by an mCS configuration file to Custom configuration profile conversion.

The Custom configuration profile is provisioned with the following steps :
- Computers > Content Management > Configuration Profiles > Upload
- Choose File : com.agnosys.config.Jamf_Pro.Paris.mCS.mobileconfig
- General
    - Name : a name of your choice (e.g. mCS-Custom configuration profile)
    - Level : Computer Level
    - Distribution Method : Install Automatically
- Scope
    - Targets > Specific Computers
    - Add > Computer Groups > mCS > Add
- Save

Please note that the "Upload" button to use is the one positioned to the right of the "New" button in the upper right corner of the Configuration Profiles window and not the "Upload" button available inside an "Application & Custom Settings" payload.

mCS-Content package

The mCS-Content package is defined with the following steps :
- Settings > Computer Management > Packages > New
- General
    - Display Name : mCS-Content
    - Filename > browse for a file : mCS-Content.pkg
- Save

The mCS-Content package is provisioned with the following steps :
- Computers > Policies > New
- Options
    - General
        - Display Name : mCS-Content Install
        - Trigger : Recurring Check-in — Execution Frequency : Once per computer
    - Packages
        - Configure > mCS-Content.pkg > Add
- Scope
    - Targets > Specific Computers
    - Add > Computer Groups > mCS > Add
- Save

mCS-Core package

The mCS-Core package is defined with the following steps :
- Settings > Computer Management > Packages > New
- General
    - Display Name : mCS-Core
    - Filename > browse for a file : mCS-Core.pkg
- Save

The mCS-Core package is provisioned with the following steps :

- Computers > Policies > New
- Options
    - General
        - Display Name : mCS-Core Install
        - Trigger : Recurring Check-in — Execution Frequency : Once per computer
    - Packages
        - Configure > mCS-Core.pkg > Add
- Scope
    - Targets > Specific Computers
    - Add > Computer Groups > mCS > Add
- Save

Provisioning Microsoft Intune

Three components must be automatically deployed to the devices :
- a Custom configuration profile
- an mCS-Content package
- the mCS-Core package.

This section outlines the key points for the provisioning of these three components in Microsoft Intune. Please refer to Microsoft Intune documentation for details not specific to mCS.

The packages must be provisioned as macOS apps. More informations about this new type of provisioning are available at https://learn.microsoft.com/en-us/mem/intune/apps/macos-unmanaged-pkg

General configuration

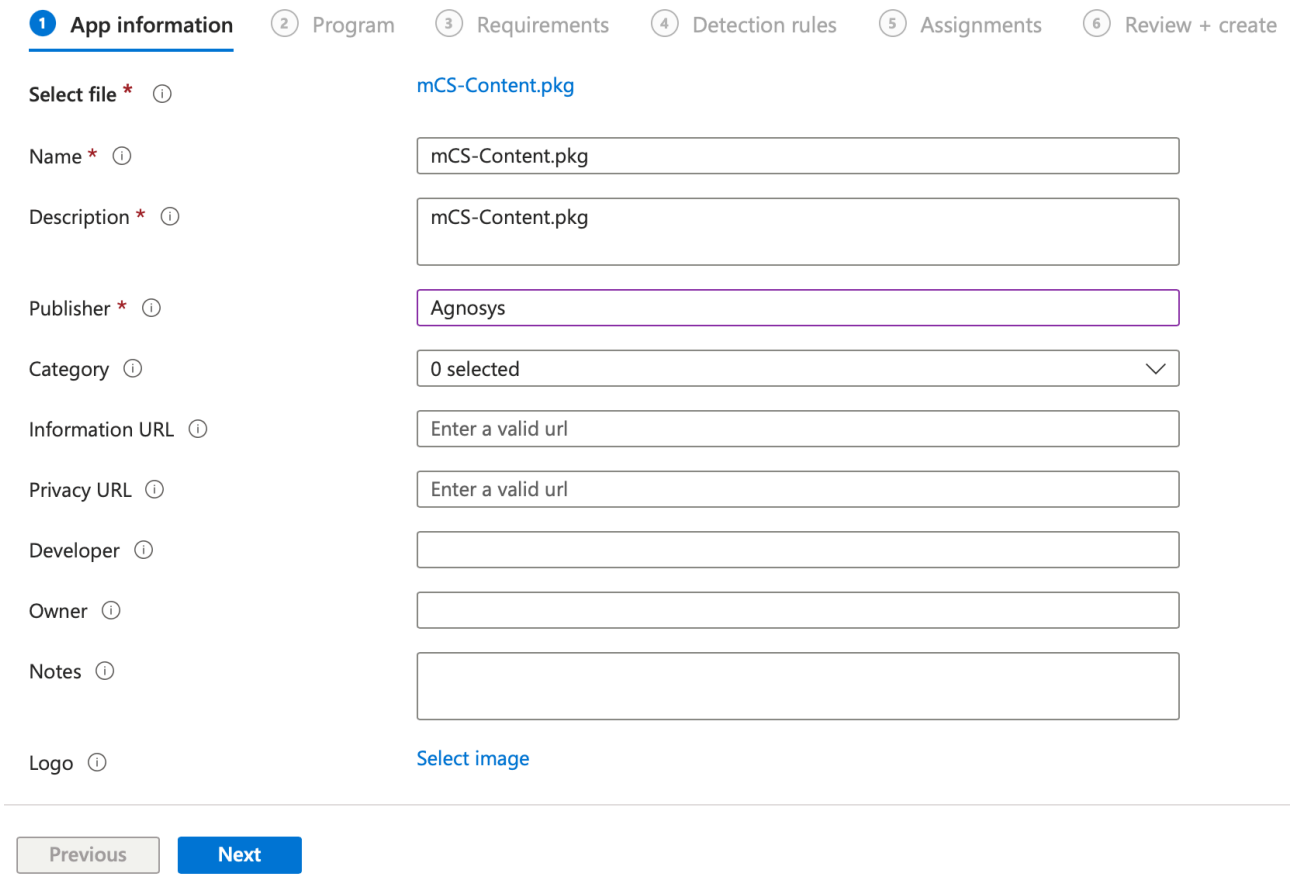
The scope of the components installation is here all devices.

Custom configuration profile

The Custom configuration profile is provisioned with the following steps :
- Devices > macOS > Configuration > Create > New Policy
- Profile type : Templates
- Select "Custom" > Create
- Basics
    - Name : mCS-Custom configuration profile
- Configuration settings
    - Custom configuration profile name : mCS-Custom configuration profile
    - Deployment channel : Device channel
    - Select a configuration profile file : com.agnosys.config.Microsoft_Intune.Paris.mCS.mobileconfig
- Assignments
    - Included groups : Add all devices
- Review + create
    - Create

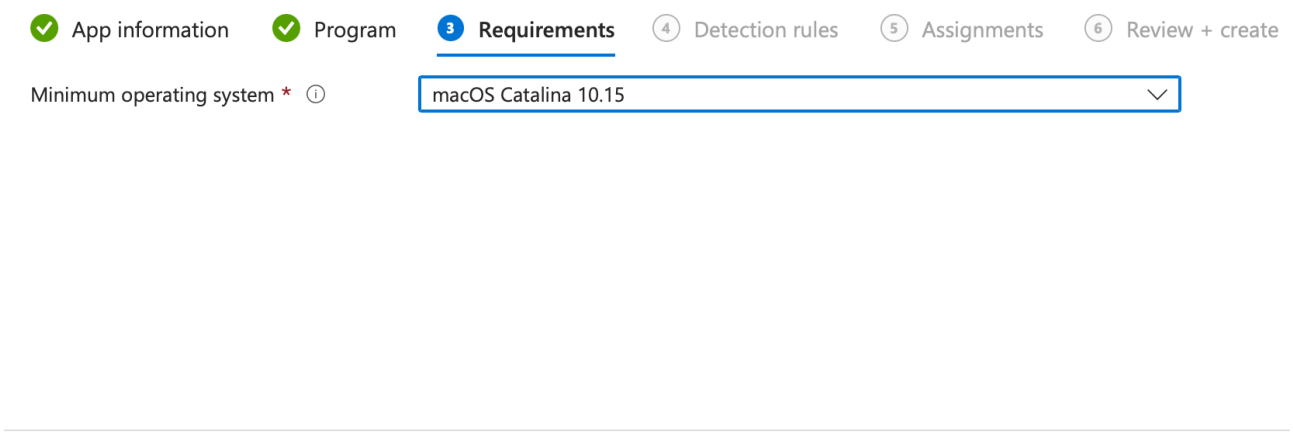
mCS-Content package

The mCS-Content package is provisioned with the following steps :
- Apps > macOS > Add
- App type : macOS app (PKG) > Select

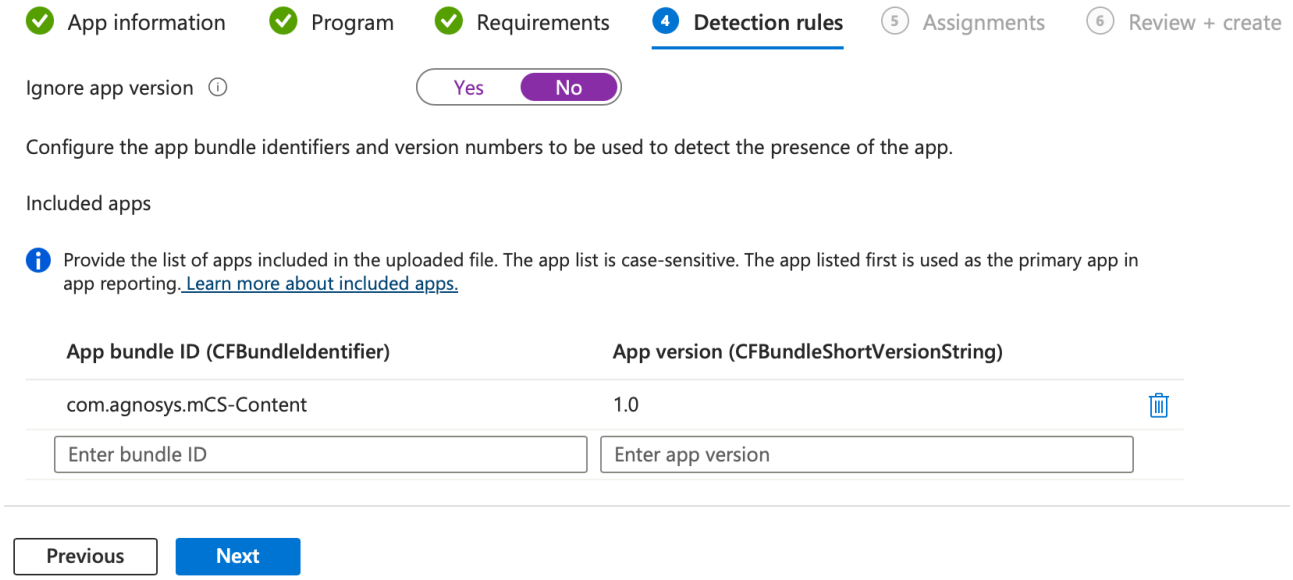


- App information
    - Select file > Select app package file
    - App package file > Select a file > mCS-Content.pkg > OK
    - Publisher : Agnosys

- Program : no scripts need to be configured



- Requirements
    - Minimum operating system : macOS Catalina 10.15



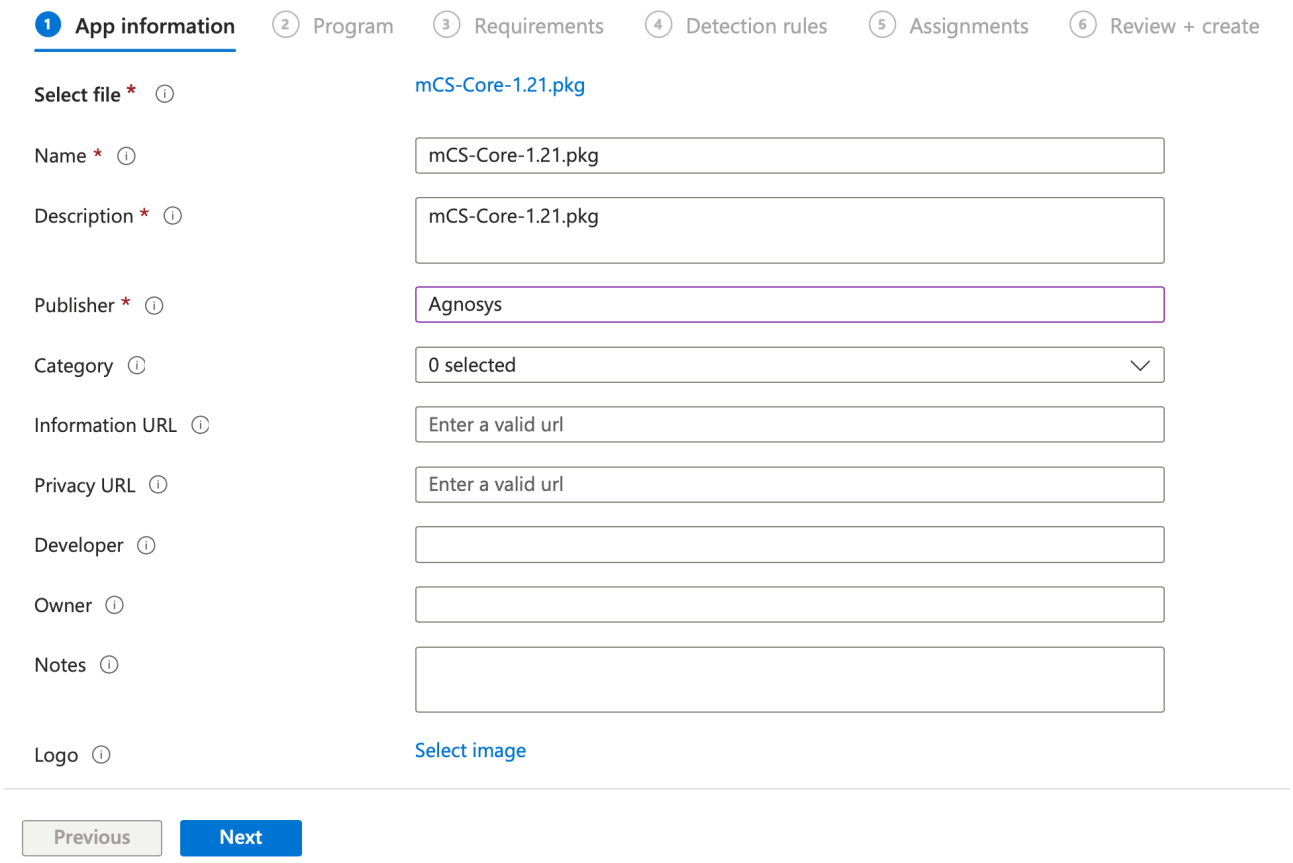
- Detection rules
    - Ignore app version : **No**
    - Detection method table :
        - App bundle ID : **com.agnosys.mCS-Content**
        - App version : keep current value (e.g. 1.0)

- Assignments
    - Required : Add all devices

- Review + create
    - Create

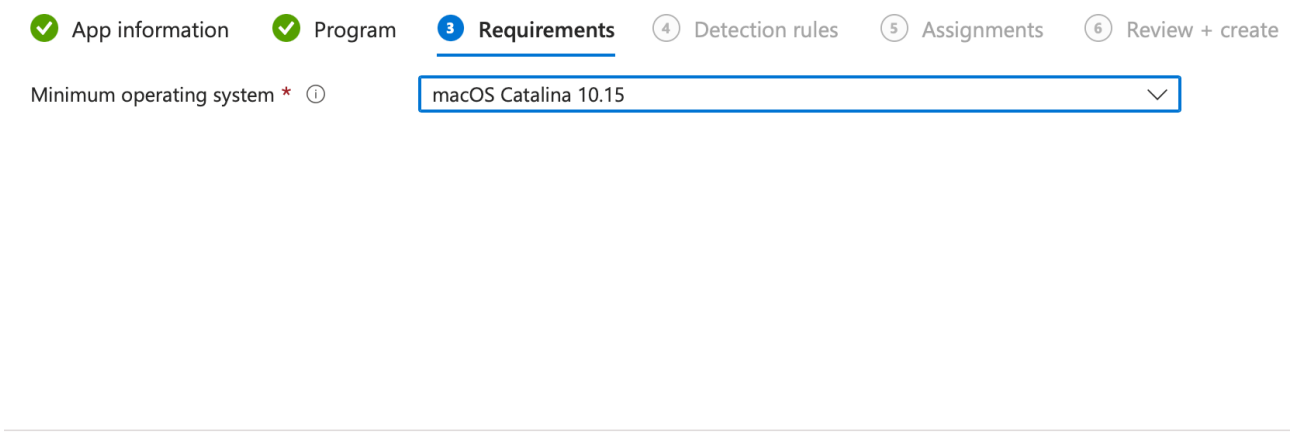
mCS-Core package

The mCS-Core package is provisioned with the following steps :
- Apps > macOS > Add
- App type : macOS app (PKG) > Select

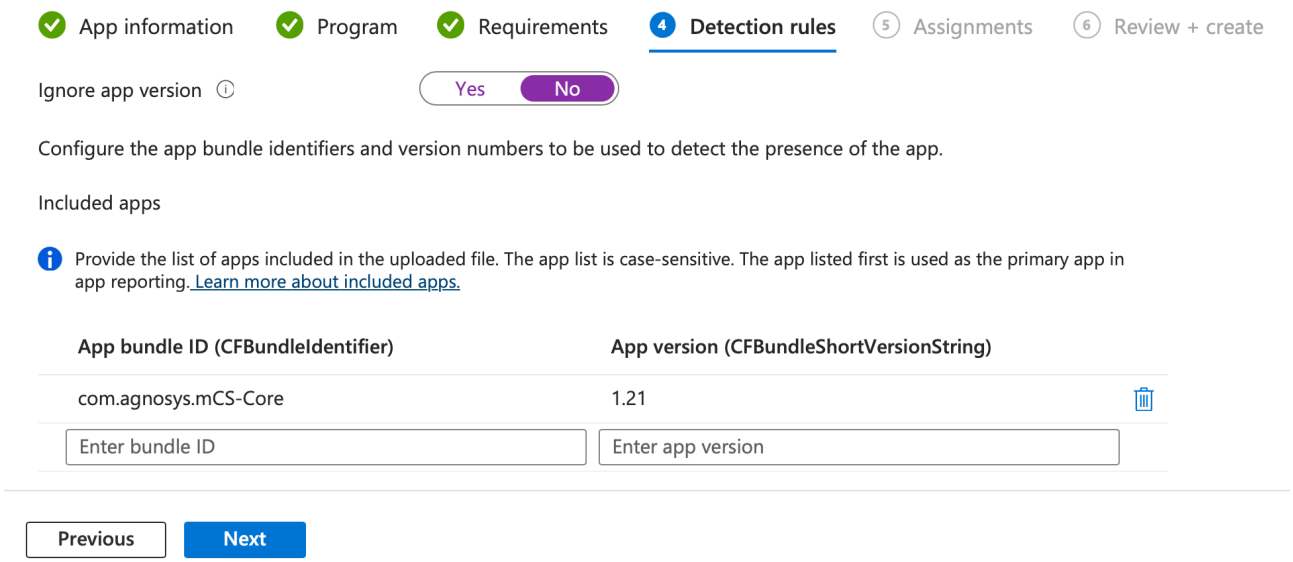


- App information
    - Select file > Select app package file
    - App package file > Select a file > mCS-Core.pkg > OK
    - Publisher : Agnosys

- Program : no scripts need to be configured



- Requirements
    - Minimum operating system : macOS Catalina 10.15



- Detection rules
    - Ignore app version : **No**
    - Detection method table :
        - App bundle ID : **com.agnosys.mCS-Core**
        - App version : keep current value (e.g. 1.21)

- Assignments
    - Required : Add all devices

- Review + create
    - Create

Provisioning Omnissa Workspace ONE

Three components must be automatically deployed to the devices :
- a Custom configuration profile
- an mCS-Content package
- the mCS-Core package.

This section outlines the key points for the provisioning of these three components in Omnissa Workspace ONE. Please refer to Omnissa Workspace ONE documentation for details not specific to mCS.

General configuration

The scope of the components installation is here all devices.

Custom configuration profile

Open the Omnissa Workspace ONE console.

Go to Resources > Profiles & Baselines > Profiles.

Click on "Add" > "Add Profile".

Select the platform "macOS" then click on "Device Profile".

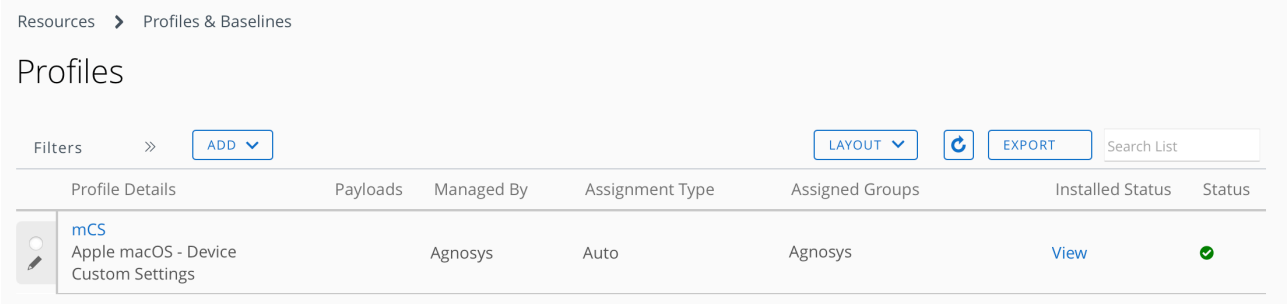
Name the configuration profile (e.g. "mCS") and optionally add a description (e.g. "mCS configuration").

Inside the "Custom Settings" payload, click on "Add" to reveal the XML field.

Open the Custom configuration profile (extension ".plist") with a Text Editor like Sublime Text, then copy and paste the whole content in the XML field.

Click on "Next".

In the "Assignment" section, click in the "Smart Group" field to add the Organization Group that encompasses the devices that are to be installed with mCS. Click on "Save and Publish".

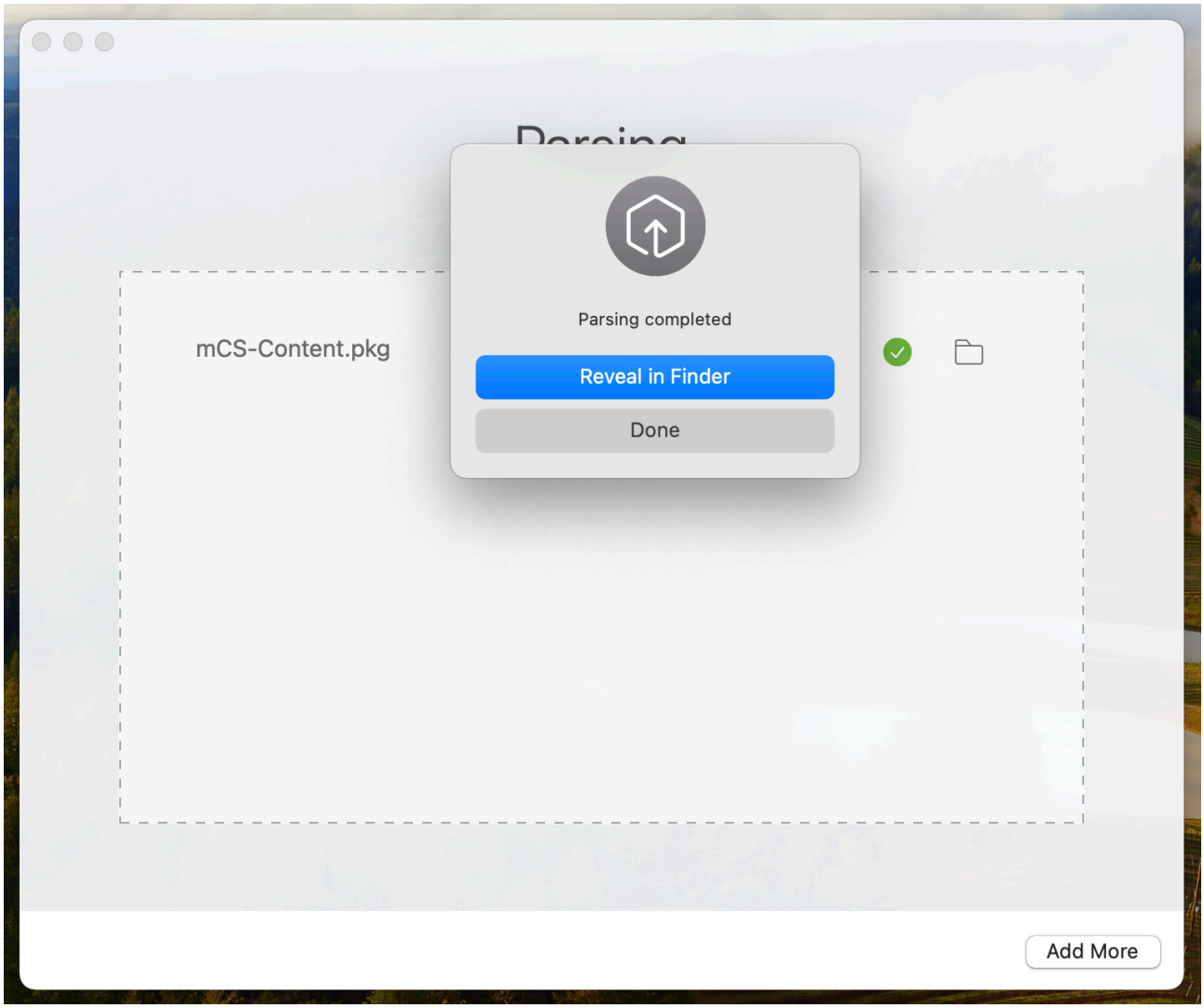


Check that the Custom configuration profile is published and assigned.

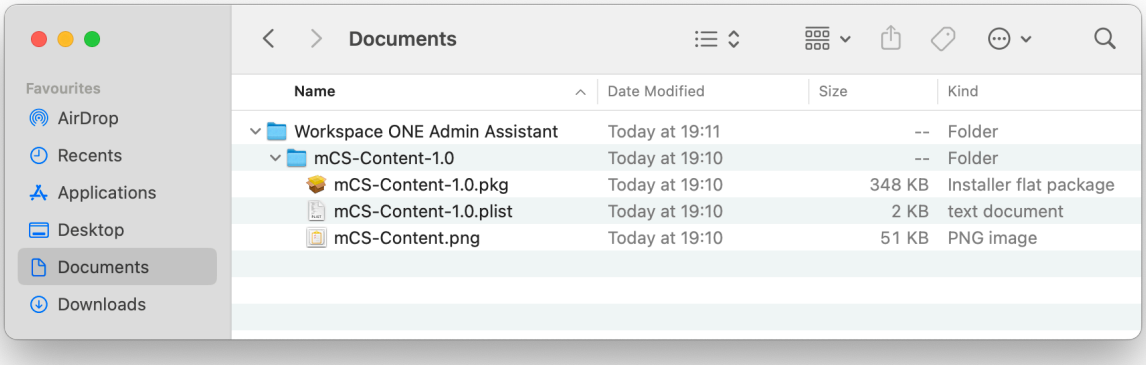
mCS-Content package

Open /Applications/Workspace ONE Admin Assistant.

Drag and drop mCS-Content.pkg in the main window.



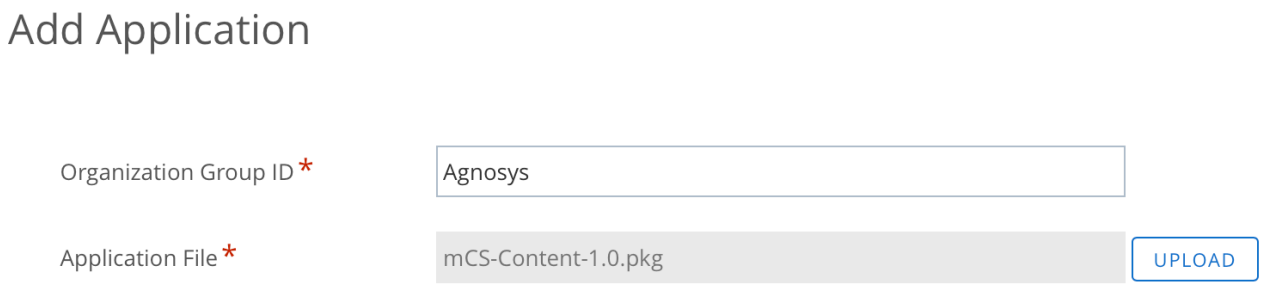
Once the parsing is completed, click on "Reveal in Finder".



Identify the package and its associated property list file that are both going to be uploaded.

Open the Omnissa Workspace ONE console.

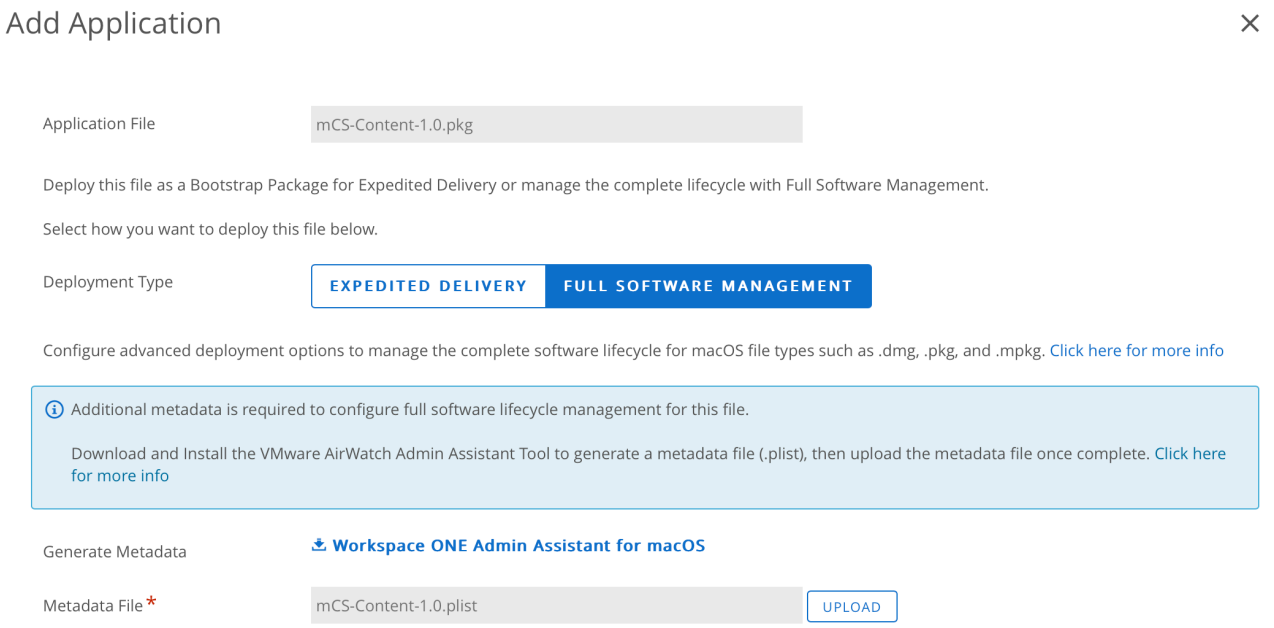
Go to Resources > Apps > Native.
Click on "Add" > "Application File".



Select the Organization Group that encompasses the devices that are to be installed with mCS.

Click on "Upload" and upload mCS-Content.pkg (Type : Local File).

Click on "Continue".

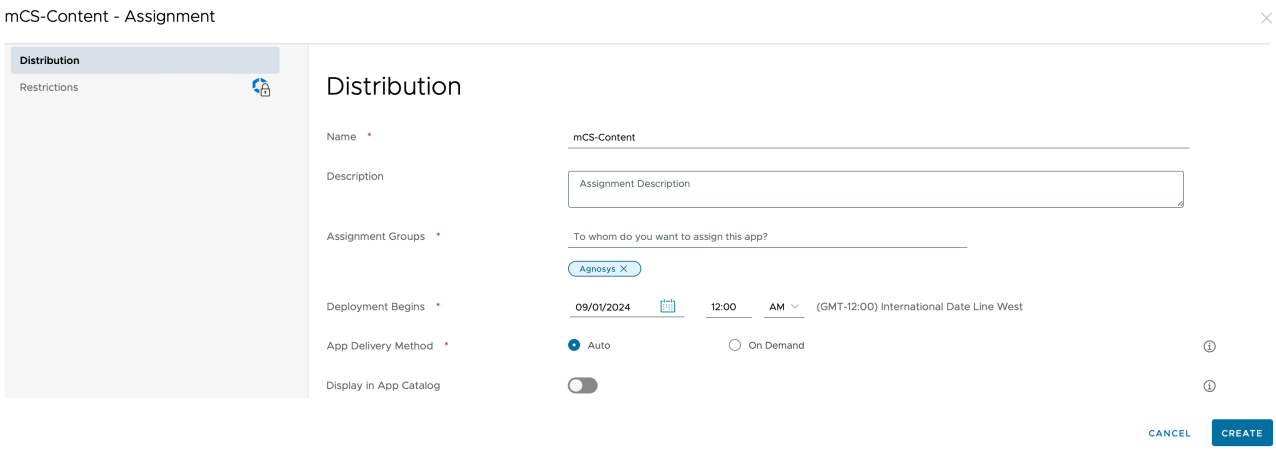


Select "Full Software Management".

Click on "Upload" and upload the associated property list file.

Click on "Continue".

In the Settings pane, click on "Save & Assign".

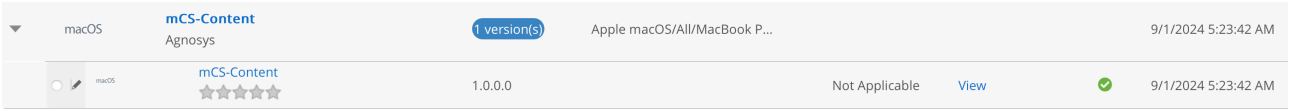


Complete the assignment form :
- Name : mCS-Content
- Assignment Groups : select the Organization Group that encompasses the devices that are to be installed with mCS
- App Delivery Method : Auto
- Display in App Catalog : disabled.

Click on "Create".

In the Assignment pane, click on "Save".

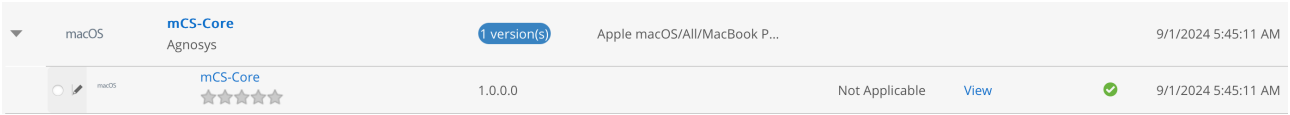
In the Preview Assigned Devices pane, click on "Publish".



Go to Resources > Apps > Native and check that the application is published and assigned.

mCS-Core package

Reproduce the same steps as for the mCS-Content package :
- use Workspace ONE Admin Assistant to parse the mCS-Core package
- add the mCS-Core package as a native application
- deploy the application to the same devices using "Full Software Management".



Go to Resources > Apps > Native and check that the application is published and assigned.

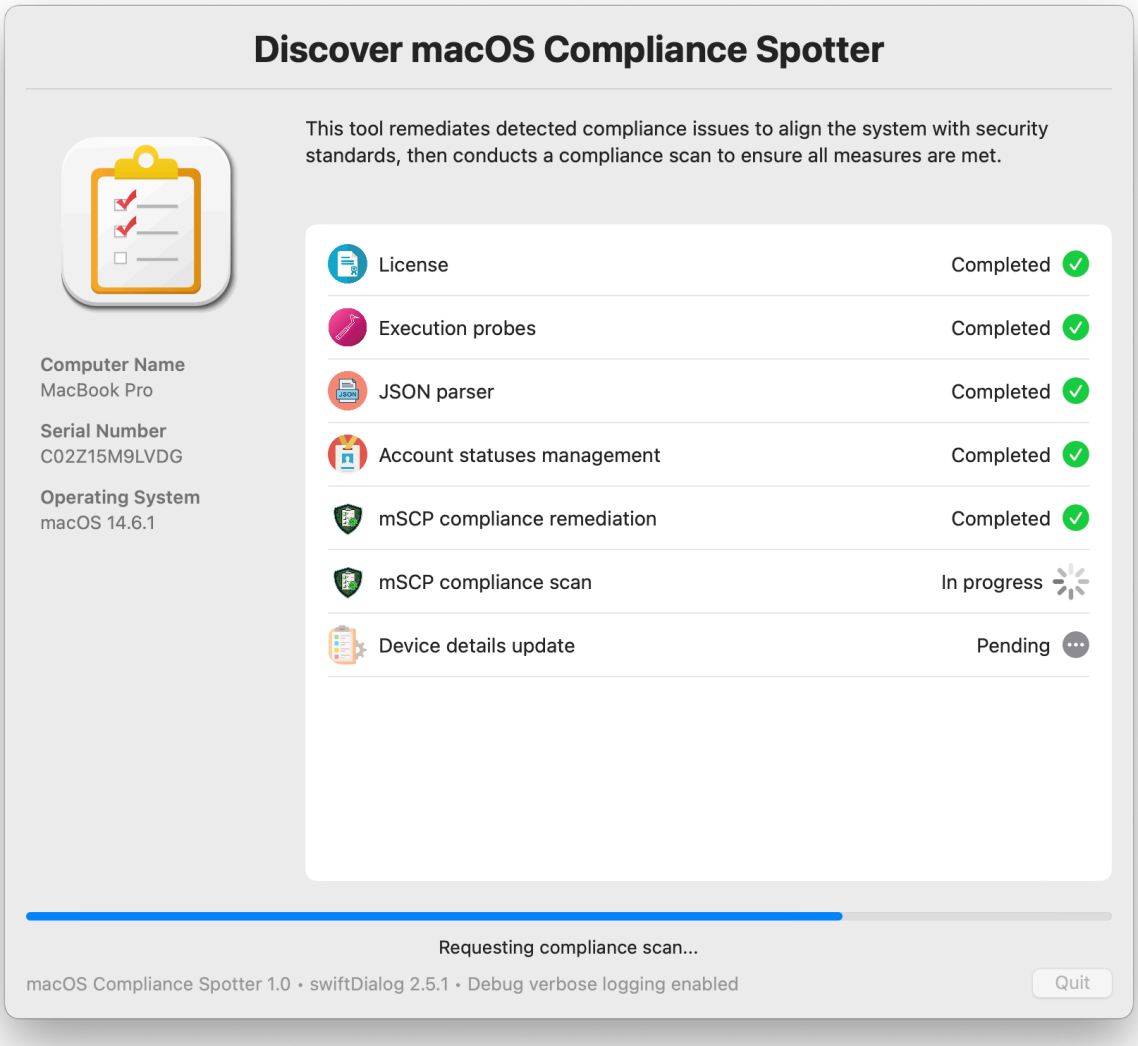
mCS execution

mCS is executed during installation and then, by default, automatically every day when the computer is awake. The maximum interval that can be set between two executions is 30 days.

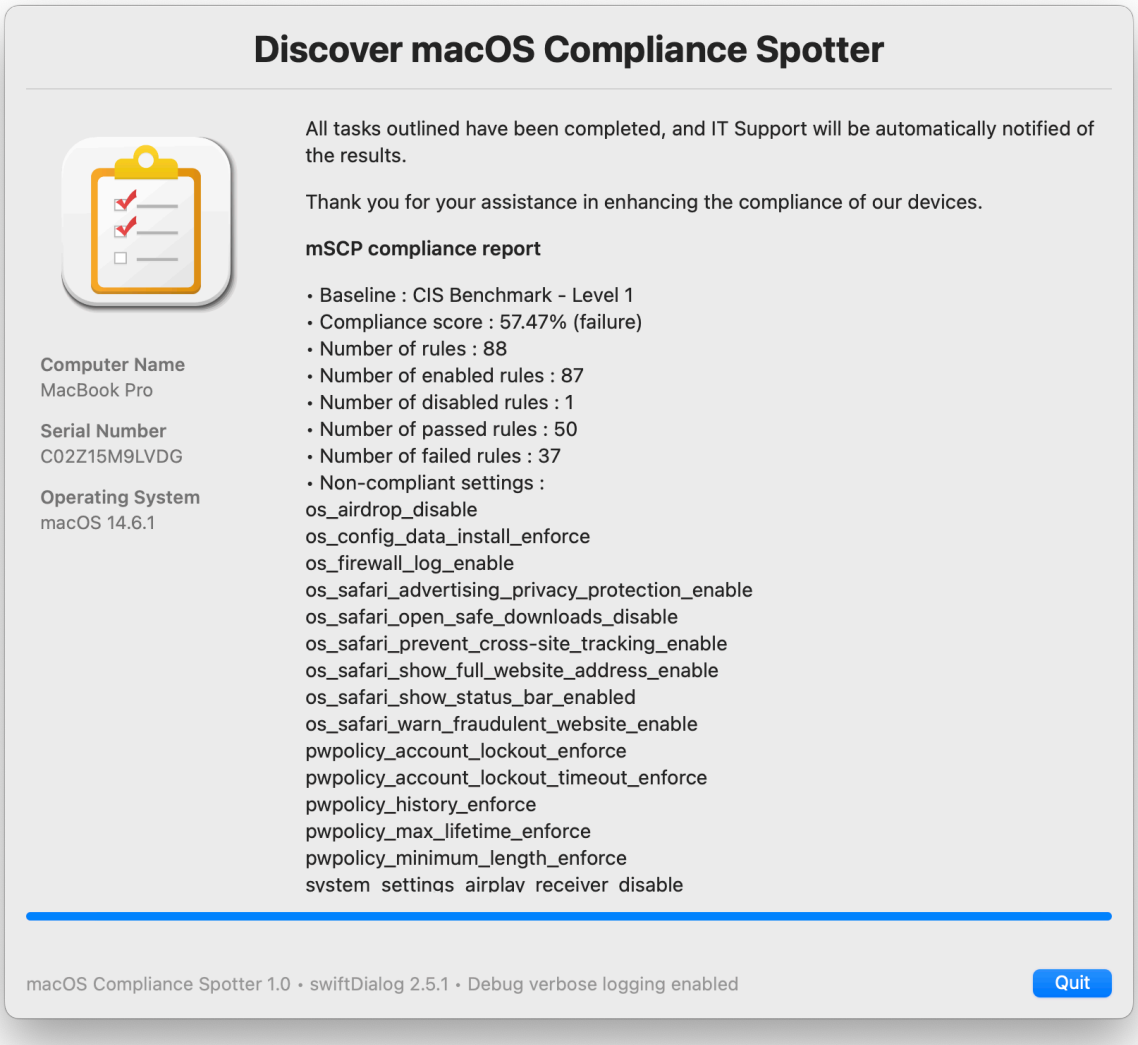
Depending on the visibility level chosen for the graphical user interface, when mCS is executed, no interface is displayed, only notifications are displayed, or the complete interface is displayed.

As soon as Slack or Teams integrations are configured, separate webhooks that allow sharing the Flight recorder report and the mSCP compliance report are sent to the dedicated support channels, regardless of the graphical user interface setting.

Execution with the complete interface display



The stages of the planned workflow are listed, with progress indicated by a status icon to the right of each item and a progress bar at the bottom.

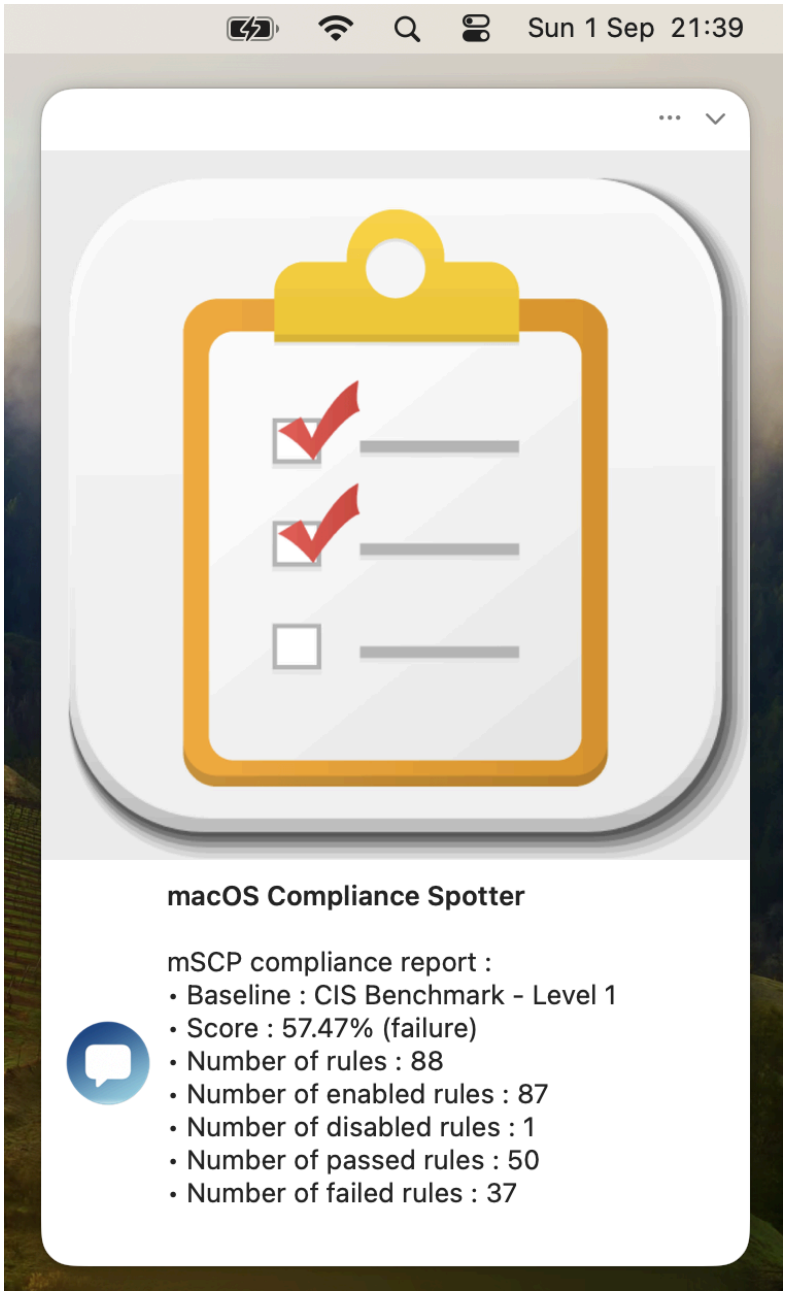


The Landing pane displays the requested mSCP compliance report, which includes the name of the applied baseline, the compliance score followed by a warning or failure status based on the chosen values, statistics about the rules, and a list of non-compliant settings.

Execution with only notifications displayed

The workflow is executed silently, and then a notification is displayed. This notification includes the name of the applied baseline, and the compliance score followed by a warning or failure status based on the chosen values.

Move the pointer to the upper right corner of the notification to reveal the disclosure triangle that allows displaying more details.



The details include the name of the applied baseline, the compliance score followed by a warning or failure status based on the chosen values, and statistics about the rules.

Localizing mCS

mCS offers two methods to localize the strings displayed during a workflow, a basic one to quickly localize a couple of key strings in the configuration files and an advanced one to fully localize all strings.

Localization of the configuration files for one language

Once the mCS configuration file is built, localize the strings in the UIHELPER and INTEGRATIONS sections.

As indicated in the mCS Dictionary, use \r to create a line break and \r\r to create a blank line, except for the UIHELPER_MAIN_TEXT_HELP help where you use exactly two spaces followed by \n for a line break and \n\n for a line break followed by an empty line.

As indicated in the mCS Dictionary, use \r to create a line break and \r\r to create a line break followed by an empty line, except in the UIHELPER_MAIN_TEXT_HELP key, where exactly two spaces followed by \n create a line break, and \n\n creates a line break followed by an empty line.

Take care of the variables used. Variables must be written exactly as indicated in the placeholders, keeping the starting and leading columns (:) otherwise their substitutions by expected values will fail.

Localization of the configuration files for multiple languages

The following keys can be localized for multiple languages :
- UIHELPER_MAIN_TITLE_WELCOME
- UIHELPER_MAIN_TEXT_WELCOME
- UIHELPER_MAIN_TEXT_HELP
- UIHELPER_BUTTON_LABEL_HELP
- UIHELPER_MAIN_TEXT_LANDING

1. Identify the language codes to use in the configuration file(s) :

    - in the Language & Region System Setting (Preference), set the Preferred languages

    - open a Terminal Window

    - type the following command :

```
defaults read .GlobalPreferences.plist AppleLanguages
```

- read the output for a user which preferred languages are French then English :

```
(
        "fr-FR",
        "en-FR"
)
```

2. Convert the Strings to Dictionaries and add one entry for each language supported.

For each entry of the Dictionary, the key name is one of the codes identified at step 1, the key type is a string and the key value is the text to display.

When mCS is executed, the previous command is used to define the preferred languages, read from top to bottom. Each time a translation is supported, a localizable string is searched according to the order of the preferred languages. If no string is available in the preferred languages, the fallback is firstly the string in English (en), secondly the first string found in the Dictionary, and thirdly the built-in string in English.

Advanced localization

Once familiar with the basic localization, you can go with the advanced localization. This localization is based on building a custom PO file from a template POT file.

The localization is aimed firstly to translate the built-in strings of mCS in English to another language, but can also be diverted to just customize those strings in English.

The POT file is provided in the mcs_library subfolder of the mCS Toolkit folder. An example of a PO file for French language is available at the same place.

To create a new PO file for your language with the POedit application, please follow these steps.

1. Download Poedit : https://poedit.net/download

2. Open Poedit

3. File > New From POT/PO File...

4. Select mCS Toolkit > mcs_library > mcs.pot > Open

5. Language of the translation > select the targeted language (e.g. "French")

6. Translate offered strings

In the translations, use \n to create a line break, and \n\n to create a line break followed by an empty line. Poedit automatically manages the \n when inserting a carriage return.

Take care of the variables used. Variables must be written exactly as indicated in the placeholders, keeping the starting and leading columns (:) otherwise their substitutions by expected values will fail.

If a translation is blank in the PO file, the fallback is the built-in string in English.

7. Identify the language code to use in the PO filename :

- in the Language & Region System Setting (Preference), set the Preferred languages

- open a Terminal Window

- type the following command :

```
defaults read .GlobalPreferences.plist AppleLanguages
```

- read the output for a user which preferred languages are French then English :

```
(
        "fr-FR",
        "en-FR"
)
```

8. File > Save > Save As : name the file "mcs_**languagecode**.po" (e.g. "mcs_**fr**.po")

9. Add the PO file to the mCS Content (MO files are not supported, see POedit Preferences to stop their compilation)

10. To update an existing PO file with the strings of an updated mcs.pot file : Translation > Update from POT File

When mCS is executed, the previous command is used to define which PO file must be invoked. The languages are read from top to bottom. As a PO file matches the language read, it is cached for the length of the workflow and the evaluation stops. If the language read is English and no PO file is available for English, the built-in strings in English are displayed.

Renewing mCS license

An mCS license is valid for one year. An annual license can be purchased at any time before the current license expires. The message sent on completion of the order contains both the new license code and the new expiration date. The new license code can be used as soon as it is supplied. To apply the new license code, please follow these steps.

Editing the LICENSE key of the deployed Custom configuration profile

If your MDM solution offers an interface for directly modifying, and not just reading, the keys of the deployed Custom configuration profile, you can choose to carefully replace the current license code with the new license code in the LICENSE key.

Warning : Bear in mind that the mCS configuration file stored in the mCS Toolkit will still contain the current and probably soon-to-expire license code, unless you also update this file as well.

• **Jamf Pro**

The Custom configuration profile can be edited with the following steps :
- Computers > Content Management > Configuration Profiles
- Click on the name of the profile (e.g. mCS-Custom configuration profile)
- Application & Custom Settings > Upload
- Edit
- Carefully replace the current license code with the new license code in the LICENSE key.
- Save.

• **Omnissa Workspace ONE**

Open the Omnissa Workspace ONE console.

Go to Resources > Profiles & Baselines > Profiles.

Click on the mCS profile to display its details then click on "Add version".

Click inside the "Custom Settings" payload then carefully replace the current license code with the new license code in the LICENSE key.

Click on "Next".

Click on "Save and Publish".

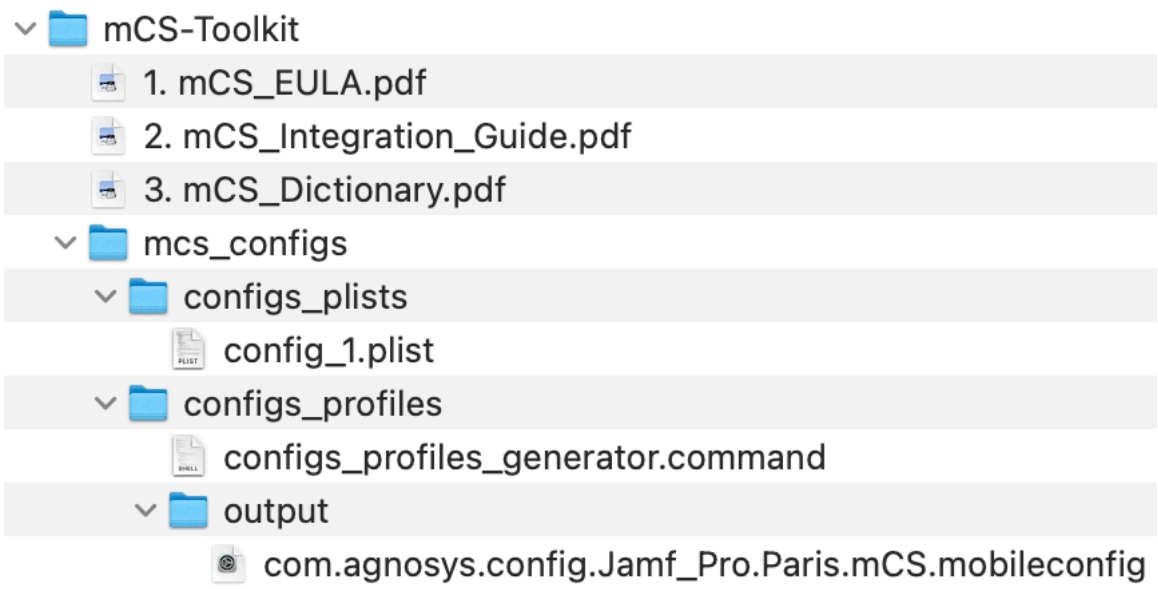
• **SimpleMDM**

The Custom configuration profile can be edited with the following steps :
- Configs > Profiles > mCS-Custom configuration profile
- Carefully replace the current license code with the new license code in the LICENSE key.
- Save.

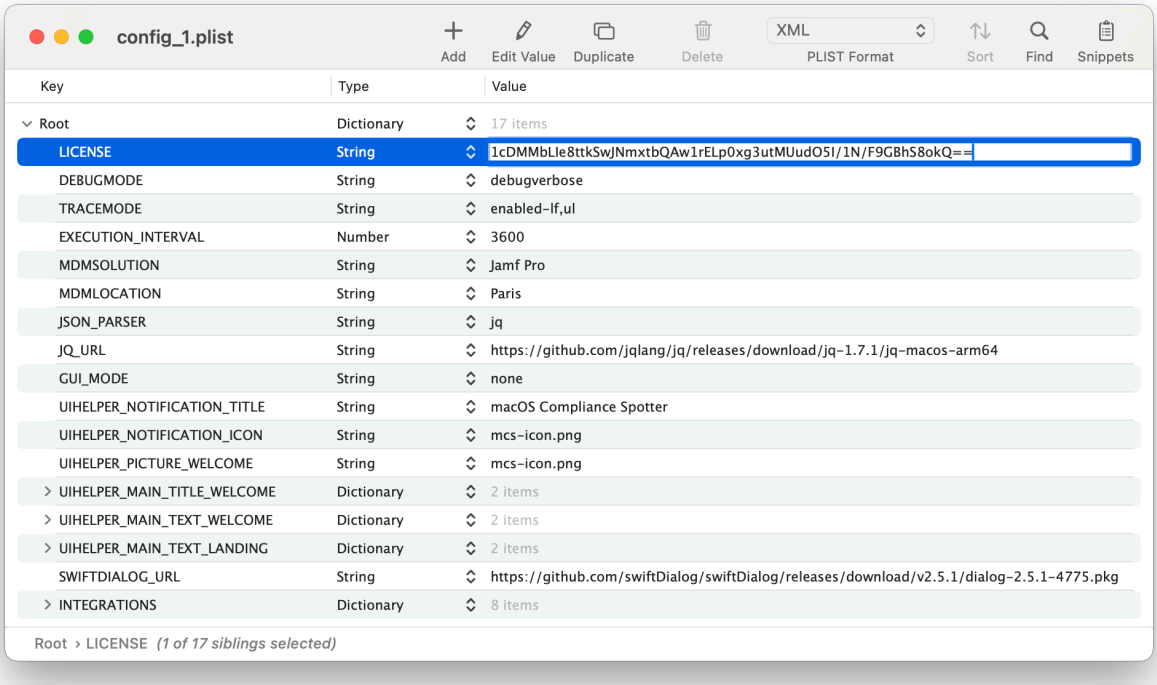
Editing the mCS configuration file(s)

If your MDM solution does not offer an interface for directly modifying, and not just reading, the keys of the deployed Custom configuration profile, follow these instructions.

First of all, backup your current mCS-Toolkit folder.



It contains particularly the mCS configuration file named by default "config_1.plist" and the current Custom configuration profile that was generated from the mCS configuration file using the script named "configs_profiles_generator.command".



Open the mCS configuration file and replace the current license code with the new license code in the LICENSE key. The license key is a one-line string ending exactly with two "=" characters.

Save the mCS configuration file, then follow these instructions :

1. Refer in this documentation to the chapter titled "mCS configuration files to Custom configuration profiles conversion" to convert your updated mCS configuration file to a Custom configuration profile, if applicable.
2. Refer in this documentation to the section titled "Custom configuration profile" included in each chapter titled "Provisioning *MDM",* and consult the MDM documentation if necessary, to distribute the updated Custom configuration profile, replacing the previous one.

Updating mCS

To safely update your mCS implementation with the latest version of the product, please follow these instructions carefully.

The components to be updated are respectively :
- mCS Toolkit
- mCS configuration file(s)
- Custom configuration profile(s)
- mCS Content package
- mCS Core package.

However, there is a shortcut. If you want to distribute a newer version of mCS without modifying the configuration, go directly to the "mCS Core package" section below.

mCS Toolkit

First of all, backup your current mCS-Toolkit folder. It contains ressources that must be preserved during the update.

1. Rename your current mCS-Toolkit folder, adding a suffix like "_previous"

2. Download and install the updated version of mCS Toolkit

3. Place the updated mCS-Toolkit folder next to the previous mCS-Toolkit folder

4. Copy from the previous folder the **.plist files** stored in mcs_configs > configs_plists to the updated folder in mcs_configs > configs_plists. Be sure to keep the updated template named "config_1.plist".

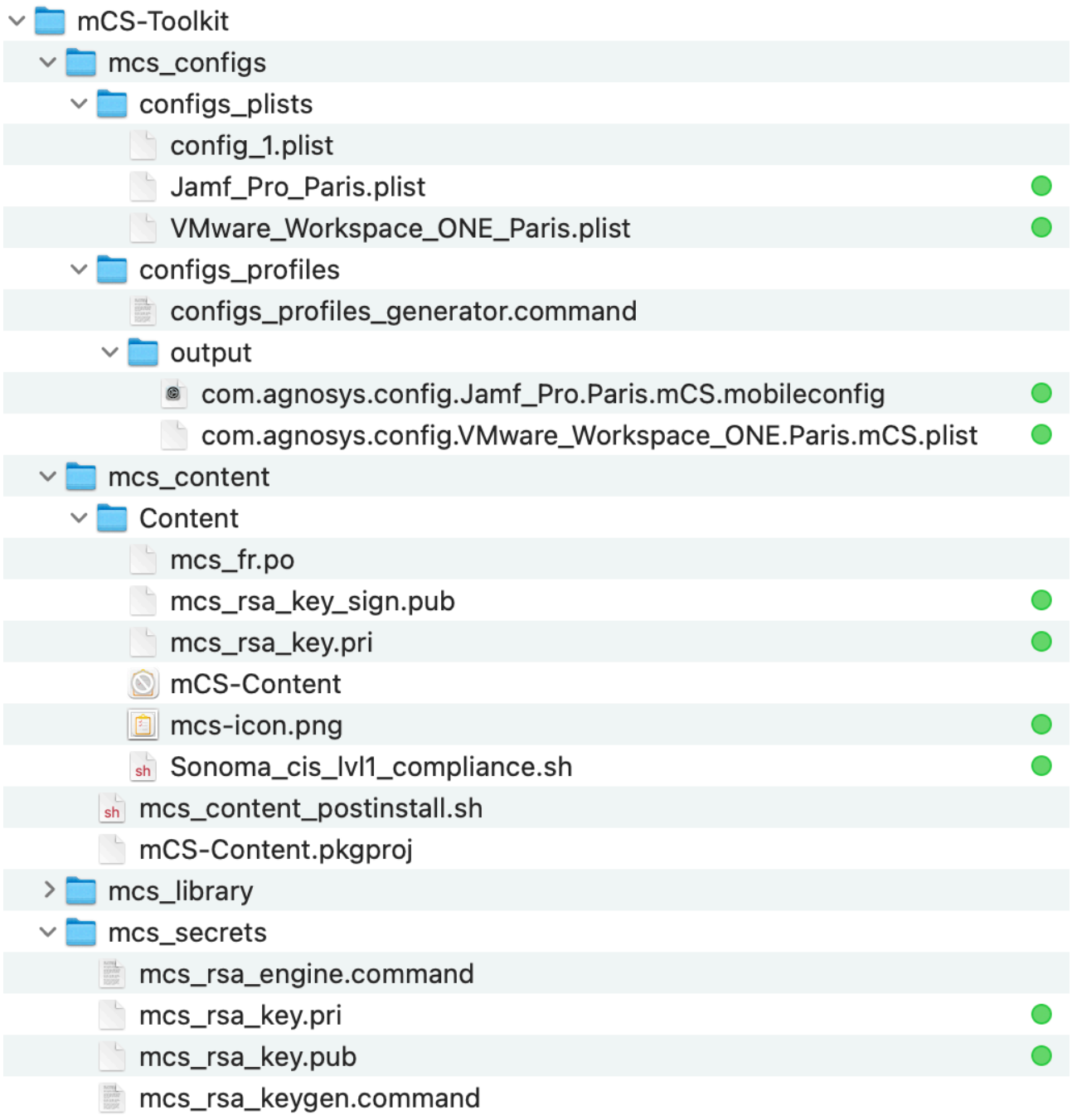
5. Copy from the previous folder the **.mobileconfig and .plist files** stored in mcs_configs > configs_profiles > output to the updated folder in mcs_configs > configs_profiles > output

6. Copy from the previous folder **the content of the folder** mcs_content > Content except **mCS-Content.app and original .po files** to the updated folder in mcs_content > Content

7. Copy from the previous folder **the following 4 files** stored in mcs_secrets to the updated folder in mcs_secrets :

    - mcs_rsa_key_sign.pri
    - mcs_rsa_key_sign.pub
    - mcs_rsa_key.pri
    - mcs_rsa_key.pub

To summarize, the figure below shows the copied resources with green dots.


mCS-Toolkit
mcs_configs
configs_plists
config_1.plist
Jamf_Pro_Paris.plist
VMware_Workspace_ONE_Paris.plist
configs_profiles
configs_profiles_generator.command
output
com.agnosys.config.Jamf_Pro.Paris.mCS.mobileconfig
com.agnosys.config.VMware_Workspace_ONE.Paris.mCS.plist
mcs_content
Content
mcs_fr.po
mcs_rsa_key_sign.pub
mcs_rsa_key.pri
mCS-Content
mcs-icon.png
Sonoma_cis_lvl1_compliance.sh
mcs_content_postinstall.sh
mCS-Content.pkgproj
mcs_library
mcs_secrets
mcs_rsa_engine.command
mcs_rsa_key.pri
mcs_rsa_key.pub
mcs_rsa_keygen.command

mCS configuration file(s)

Complete your current .plist file(s) to implement as desired new capabilities of mCS, helping you with the updated mCS Dictionary and the updated config_1.plist file. Do not hesitate to contact mCS support if you need any clarifications.

Warning : Ensure that your mCS configuration file(s) contain(s) your current mCS license, as it may have been updated directly in the MDM solution, but not in the mCS configuration file(s).

Even if your MDM solution offers an interface for directly modifying, and not just reading, the keys of the deployed Custom configuration profile(s), it is recommended to update the mCS configuration file(s) using your preferred Property List editor to ensure no structural errors are introduced accidentally. The only key supported for direct modification is the license code.

Custom configuration profile(s)

1. Refer in this documentation to the chapter titled "mCS configuration files to Custom configuration profiles conversion" to convert your updated mCS configuration file(s) to Custom configuration profile(s), if applicable. This one / these ones will reuse the same identifier(s) as the previous Custom configuration profile(s), thanks to the content of the output folder copied at the expected location.

2. Refer in this documentation to the section titled "Custom configuration profile" included in each chapter titled "Provisioning *MDM",* and consult the MDM documentation if necessary, to distribute the updated Custom configuration profile(s), replacing the previous one(s).

mCS Content package

As the private key contained in the file "mcs_rsa_key.pri" is static and if you didn't update the management account picture, no action is necessary.

If you implemented an advanced localization, you may have to revise your PO files after importing the updated POT file provided. Otherwise, built-in strings in English may be unexpectedly displayed.

If you updated the other resources :
- refer to this documentation to build an updated mCS-Content package
- refer to this documentation and the MDM documentation to deploy the updated package.

mCS Core package

1. Download the updated version of mCS Core.

2. Refer to this documentation and the MDM documentation to deploy the updated package.

Troubleshooting

When using the commands described below, pay attention to use "**straight**" double quotes and not "curly" double quotes often generated automatically by word processing applications.

Display the follow-up file

Open the Terminal utility located in /Applications/Utilities.

Type the following command :

```
sudo defaults read "/Library/Application Support/mCS/mcs.plist"
```

The informations displayed are :

- LASTEXECUTIONDATE : last mCS execution date in the format "YYYY-MM-DD HH:MM:SS"

- LASTEXECUTIONDATEEPOCHTIME : last mCS execution date in the format "epoch time"

- MSCPCOMPLIANCEREMEDIATIONDATE : compliance remediation date in the format "YYYY-MM-DD HH:MM:SS" (mSCP integration)

- MSCPCOMPLIANCEREMEDIATIONDATEEPOCHTIME : compliance remediation date in the format "epoch time" (mSCP integration)

- MSCPCOMPLIANCESCANDATE : compliance scan date in the format "YYYY-MM-DD HH:MM:SS" (mSCP integration)

- MSCPCOMPLIANCESCANDATEEPOCHTIME : compliance scan date in the format "epoch time" (mSCP integration)

- MSCPCOMPLIANCESCORE : compliance score floored to an integer (mSCP integration)

- STATUSMESSAGE : exit message of the latest mCS execution (raw message for a success, information or error event)

Enable the debug logging manually

Open the Terminal utility located in /Applications/Utilities.

Two level of debug logging can be enabled, depending of the debug flag created.

To enable a standard debug logging, type :
```
sudo touch "/Library/Application Support/mCS/debug"
```

To enable a verbose debug logging, type :
```
sudo touch "/Library/Application Support/mCS/debugverbose"
```

Enter your password.

The debug logs are written in /private/var/log.

The files titled mCS-mcs_*date*.log and mCS-mcs_starter_*date*.log contain sensitive informations for troubleshooting purpose only, must not be left unattended and can be read by admin users only.

Warning : Do not forget to delete the mCS debug logs and the debug flag created once the log analysis is completed.

Enable the debug logging with Custom configuration profile

Two levels of debug logging can be enabled, depending of the value of the DEBUGMODE key.

To enable a standard debug logging, set the DEBUGMODE key to "debug".

To enable a verbose debug logging, set the DEBUGMODE key to "debugverbose".

The debug logs are written in /private/var/log.

The files titled mCS-mcs_*date*.log and mCS-mcs_starter_*date*.log contain sensitive informations for troubleshooting purpose only, must not be left unattended and can be read by admin users only.

The corresponding debug flag is automatically created in /Library/Application Support/mCS.

Warning : Do not forget to disable the debug logging then delete the mCS debug logs and the debug flag created once the log analysis is completed.

Note that this setting is ignored if the debug logging has been already enabled by the manual creation of a debug flag.

Display the debug logs from the Console utility

Open the Console utility located in /Applications/Utilities.

Select Reports > Log Reports > a file named mCS-mcs_*date*.log

Display the debug logs from the Terminal utility

Select Finder > Go > Go to Folder > /private/var/log

Open the Terminal utility located in /Applications/Utilities.

Type : `sudo cat`

Type a space then drag and drop a file named mCS-mcs_*date*.log

Enter your password.

In the context of a request for support, please attach a **verbose** debug log to your message.

Execute mCS manually

Open the Terminal utility located in /Applications/Utilities.

Type one of the following command and enter your password when prompted.

To get mCS options :

```
sudo sh "/Library/Application Support/mCS/Core/mcs.sh" --help
```

To execute mCS normally :

```
sudo sh "/Library/Application Support/mCS/Core/mcs.sh"
```

To execute mCS ignoring the execution intervals :

```
sudo sh "/Library/Application Support/mCS/Core/mcs.sh" --ignoreintervals
```

To bypass the GUI mode defined in the Custom configuration profile, a GUI option must be passed with the command.

```
sudo sh "/Library/Application Support/mCS/Core/mcs.sh" --ignoreintervals
--gui none|informative|interactive
```

When mCS is executed manually from the Terminal, the Terminal utility must be granted Full Disk Access to allow the Privileged Helper to perform sensitive operations that require such access. To do this, go to System Settings > Privacy & Security > Full Disk Access, add the Terminal utility if it is not already listed, and ensure that the toggle is enabled.

Enable the trace log with Custom configuration profile

To enable the trace log for informative purposes, set the TRACEMODE key with one of these values :
- "enabled-lf" or "enabled" (exactly) to log in the log file /private/var/log/mCS-mcs_trace.log
- "enabled-ul" to log in the Unified Logging under the process "logger"
- "enabled-lf,ul" to log to both destinations (the two options can be placed in any order but must be separated by a comma).

These logs do not contain sensitive informations, can be used to keep track of execution history and can be read by any standard or admin user.

Display the trace log of type "log file" from the Console utility

Open the Console utility located in /Applications/Utilities.

Select Reports > Log Reports > mCS-mcs_trace.log

Display the trace log of type "Unified Logging" from the Terminal utility

Open the Terminal utility located in /Applications/Utilities.

Type the following command to get the log.

```
log show --predicate 'process="logger" and eventMessage contains "mCS"'
```

To get the log for the last day only, add the option "--last 1d".

To get the log for the last 60 minutes only, add the option "--last 60m".

Solve a non-reinstallation issue

This section only applies if the management solution is Microsoft Intune.

To help the MDM determine that the mCS-Core package and/or the mCS-Content package have been successfully installed so these last are not reinstalled in loop at each sync, mCS includes two detection apps :
- /Library/Application Support/mCS/Core/mCS-Core.app
- /Library/Application Support/mCS/Content/mCS-Content.app.

The side effect of their presence is that they may prevent a reinstallation of these packages.

To solve this issue, open the Terminal utility located in /Applications/Utilities, type one of the following command depending of the package to be reinstalled and enter your password when prompted.

```
sudo rm -Rf "/Library/Application Support/mCS/Core/mCS-Core.app"
```

```
sudo rm -Rf "/Library/Application Support/mCS/Content/mCS-Content.app"
```

Then trigger an MDM sync to reinstall the targeted package(s).

Support

Paid support included in mCS offers

Send your support request to mcs.support@agnosys.fr

Support is delivered by email in English and French.

Support is opened Monday to Friday 10:00-17:00 Time Zone Europe/Paris.

The first callback is targeted to be done within 4 hours after the reception of the support request.

Free community support

Join our Slack channel at https://macadmins.slack.com/archives/C07K4PNHKQF

The free support is offered as time permits for basic cases, bug report studies and feature request discussions.

The community is encouraged to help the other adopters and share its findings.

mCS announcements and public release notes, which are a summary of the release notes, are published in the Slack channel.

Release notes

The release notes are available in the Dropbox folder where the software is available for download. They contain a detailed log of the changes introduced with the different released versions and the one in development.