Mac OS

? macOS Deployment & Management

This book serves as a comprehensive guide for deploying, managing, and troubleshooting macOS applications and devices in enterprise or education environments. Whether you're using MDM solutions like Meraki or performing manual installations, this resource provides step-by-step tutorials, automation scripts, and best practices to streamline your workflows.

Key topics include:

- ? App deployment using shell scripts
- ?? Post-installation checks and logging
- ? Log file analysis and error tracing
- ? Tips for integration with Meraki and other MDM tools

Ideal for IT administrators, support technicians, and anyone managing Apple devices at scale.

App Deployment

App Deployment

? 8x8 Work App Deployment via Script (macOS)

This guide walks you through how to deploy the **8x8 Work** app on macOS using a custom bash script. This is ideal for environments managed via *Meraki* or other MDM tools.

□□ Deployment Script Overview

The following script will:

1. Remove any existing version of `8x8 Work.app`
2. Download the latest installer from 8x8's official server
3. Mount the DMG
4. Copy the app into `/Applications`
5. Unmount the DMG
6. Log each step into `/tmp/8x8_install_status.log`

```
#!/bin/bash

APP_PATH="/Applications/8x8 Work.app"
LOG_FILE="/tmp/8x8_install_status.log"

echo "Starting 8x8 installation status log" > "$LOG_FILE"
date >> "$LOG_FILE"

if [ -d "$APP_PATH" ]; then
  echo "Removing existing 8x8 Work.app" | tee -a "$LOG_FILE"
  rm -rf "$APP_PATH"
fi

if curl -L -o /tmp/8x8Installer.dmg https://work-desktop-assets.8x8.com/prod-
publish/ga/work-dmg-v8.21.3-1.dmg; then
```

```
  echo "Download complete" | tee -a "$LOG_FILE"
else
  echo "Download failed" | tee -a "$LOG_FILE"
  exit 1
fi

if hdiutil attach /tmp/8x8Installer.dmg; then
  echo "Disk image mounted" | tee -a "$LOG_FILE"
else
  echo "Failed to mount disk image" | tee -a "$LOG_FILE"
  exit 1
fi

if ditto "/Volumes/8x8 Work/8x8 Work.app" "$APP_PATH"; then
  echo "Application copied to /Applications" | tee -a "$LOG_FILE"
else
  echo "Failed to copy application" | tee -a "$LOG_FILE"
  exit 1
fi

if hdiutil detach "/Volumes/8x8 Work"; then
  echo "Disk image unmounted" | tee -a "$LOG_FILE"
else
  echo "Failed to unmount disk image" | tee -a "$LOG_FILE"
  exit 1
fi

if [ -d "$APP_PATH" ]; then
  echo "8x8 Work.app installed successfully in /Applications." | tee -a
"$LOG_FILE"
else
  echo "Installation failed" | tee -a "$LOG_FILE"
  exit 1
fi
```

How to Check Installation Logs

The script logs its progress in a plain text file located at `/tmp/8x8_install_status.log`.

View Log Contents

```
cat /tmp/8x8_install_status.log
```

□□ View Log with Paging

```
less /tmp/8x8_install_status.log
```

□□ Follow the Log in Real Time

```
tail -f /tmp/8x8_install_status.log
```

□ Search for Errors

```
grep "Failed" /tmp/8x8_install_status.log
```

? You’re now equipped to deploy and verify the 8x8 Work app installation on macOS systems using a scripted process. Happy deploying!

□□ Line-by-Line Explanation of the Script

```
#!/bin/bash
```

Shebang — Tells the system to run this script using the Bash shell.

```
APP_PATH="/Applications/8x8 Work.app"
LOG_FILE="/tmp/8x8_install_status.log"
```

Variable definitions:

`APP_PATH` stores the expected location of the app.

`LOG_FILE` defines where the script logs its status messages.

```
echo "Starting 8x8 installation status log" > "$LOG_FILE"
date >> "$LOG_FILE"
```

Logging:

Creates a new log file and adds the current date/time.

```
if [ -d "$APP_PATH" ]; then
  echo "Removing existing 8x8 Work.app" | tee -a "$LOG_FILE"
  rm -rf "$APP_PATH"
fi
```

Remove old version:

Deletes the existing app if present, and logs the action.

```
if curl -L -o /tmp/8x8Installer.dmg https://work-desktop-assets.8x8.com/prod-
publish/ga/work-dmg-v8.21.3-1.dmg; then
```

Download the installer:

`curl` fetches the installer and saves it locally.

```
  echo "Download complete" | tee -a "$LOG_FILE"
else
  echo "Download failed" | tee -a "$LOG_FILE"
  exit 1
fi
```

Log download result:

Logs success or exits if the download fails.

```
if hdiutil attach /tmp/8x8Installer.dmg; then
```

Mount the DMG: Makes the downloaded disk image accessible.

```
  echo "Disk image mounted" | tee -a "$LOG_FILE"
else
  echo "Failed to mount disk image" | tee -a "$LOG_FILE"
  exit 1
fi
```

Log mount status and handle failure.

```
if ditto "/Volumes/8x8 Work/8x8 Work.app" "$APP_PATH"; then
```

Install the app:

Uses `ditto` to copy the app from the DMG to `/Applications`.

```
  echo "Application copied to /Applications" | tee -a "$LOG_FILE"
else
  echo "Failed to copy application" | tee -a "$LOG_FILE"
  exit 1
fi
```

Log copy status:

Verifies whether the copy worked.

```
if hdiutil detach "/Volumes/8x8 Work"; then
```

Unmount the DMG: Clean-up step after installation.

```
  echo "Disk image unmounted" | tee -a "$LOG_FILE"
else
  echo "Failed to unmount disk image" | tee -a "$LOG_FILE"
  exit 1
fi
```

Log unmount result.

```
if [ -d "$APP_PATH" ]; then
```

Final verification: Confirms the app exists post-install.

```
  echo "8x8 Work.app installed successfully in /Applications." | tee -a
"$LOG_FILE"
else
  echo "Installation failed" | tee -a "$LOG_FILE"
  exit 1
fi
```

Log final result — Either success or failure.

? How to Disable System Integrity Protection (SIP) on macOS

System Integrity Protection (SIP) is a security feature in macOS that helps protect the system by restricting the root user account and limiting access to critical system files. You may need to disable SIP for advanced troubleshooting or when using certain low-level software.

> *“?? **Warning:** Disabling SIP reduces the security of your system. Only disable it if absolutely necessary and re-enable it when done.*

□□ Requirements

- A Mac running macOS
- Administrator access
- A wired or built-in keyboard (wireless keyboards may not work before macOS loads)

□□ Steps to Disable SIP

1. Reboot into Recovery Mode

- Click the Apple ? menu and choose **Restart**.
- Immediately hold down **Command (?) + R** until the Apple logo or a spinning globe appears.
- This boots your Mac into **macOS Recovery Mode**.

2. Open Terminal in Recovery

- In the top menu, click **Utilities > Terminal**.

3. Run the SIP Disable Command

In the Terminal window, type the following command:

```
csrutil disable
```

Press **Enter**. You should see a message confirming that SIP has been disabled.

4. Reboot Normally

Type:

```
reboot
```

Or select **Apple menu > Restart** from the top-left corner.

Confirming SIP is Disabled

After rebooting, open **Terminal** and run:

```
csrutil status
```

You should see:

```
System Integrity Protection status: disabled.
```

To Re-enable SIP

Repeat the steps above, but in step 3, run:

```
csrutil enable
```

Notes

- SIP does **not** affect user-level apps, only critical system paths like `/System`, `/bin`, `/sbin`, and some root-level processes.
- Disabling SIP may be required for third-party kexts, system utilities, or recovery tasks.

Google LDAP Deployment directions Instructions

google-ads-sizes-update-2021.jpg

These instructions follow a Series of Guides by Google on LDAP. The article demonstrates many types of Systems Including PaperCut-MF . The instructions Below are part of the macOS Deployment article Deployment phase which comes after the Preparation Phase which was completed on a CCA Device beforehand.

This Deployment phase instructs as follows:

System requirements

- The macOS must be Catalina Version 10.15.4 or later.
- A Google super admin user ID is required to complete step 1 in the preparation phase. (already completed)
- You need local admin permissions to perform this configuration.

1. Copy Files

Copy the Mac Profile file `GOOGLE_LDAP_PROFILE2 (1).mobileconfig`, and the XML config *ldap.google.com.plist* file generated, and the python script *ldaps_macos_script.py* to the `/tmp/` directory on the macOS device.

> Files attached in this Document or this Link

2. Install Mobile Profile

This step involves installing the mobile profile, which is crucial for integrating with the Secure LDAP server.

```
GOOGLE_LDAP_PROFILE2 (1).mobileconfig
```

3. Install Python 3

Download and install Python 3 from the official Python website.

```
https://www.python.org/ftp/python/3.13.5/python-3.13.5-macos11.pkg
```

4. Install Dependencies

Once Python 3 is installed, open a terminal and run the following command to install the required `pyobjc-framework-opendirectory` dependency:

```
python3 -m pip install pyobjc-framework-opendirectory
```

5. Execute Python Script

Run the Python script to configure the Secure LDAP settings:

```
sudo python3 /tmp/ldaps_macos_script.py /tmp/ldap.google.com.plist
```

6. Restart your Machine

Restart the macOS machine

7. Connect to Secure LDAP and Create Mobile Account

After the script executes, run the following command to connect to the Secure LDAP server and set up a home path and mobile account(s):

```
sudo
/System/Library/CoreServices/ManagedClient.app/Contents/Resources/createmobil
eaccount -n $uid -v
```

> **Tip:** Replace *$uid* with the username part of the email address associated with the user's Google account. For example, jsmith is the username part for jsmith@solarmora.com.

When prompted for the *SecureToken admin user name*, enter your admin username, and enter your password in the next prompt. This will add **$uid** into the FileVault. This is needed if the macOS disk is encrypted.

(Optional) Set the login screen preference

1. Go to **System preferences > Users & Groups > Login Options** at the bottom left.
2. Unlock the lock by providing admin credentials.
3. Change the *Display login window as* to **Name and password.**

8. Limitations and guidelines

- For users signing in to macOS using their Google credentials, their Workspace account username must be different from their macOS user profile user ID, or sign-in is blocked.
- Once a user starts signing in to macOS using Google credentials, user password management (reset or recovery) must happen on the Google website (for example, at *myaccount.google.com* or in the Google Admin console). If you choose to do password management using a third-party solution, then make sure the latest password is synchronized with Google.
- If the admin creates a new user or resets an existing user's password with the *Ask for a password change at the next sign-in* setting turned on, the user cannot sign in to Mac using the temporary password set by the admin.

Workaround: The user needs to sign in to Google using another device (for example, their mobile device or other desktop device), set a permanent password, and then sign in to macOS using the new password.

- The Mac must be connected to a working internet connection so that *ldap.google.com* is reachable during the first sign-in after the above configuration. Any subsequent sign-ins won't need Internet access as long as you opted to set up a mobile account.

- Google Secure LDAP integration with macOS is tested on macOS Catalina, Big Sur, and Monterey.

Apple Push Notification

Apple Push Notification

Migrating an APNS certificate from one Apple ID to another Apple ID

As part of a recent change, I needed to migrate an APNS certificate from being associated with one Apple ID to now being associated with another Apple ID. Apple has a KBase article available which provides contact information for this, which is available via the link below:

https://support.apple.com/HT208643

For those folks with AppleCare support plans, you can also submit a ticket to AppleCare. That's the route I took. Regardless of which support avenue you pursue, Apple will request the following information from you.

- APNS Certificate Subject DN
- APNS Certificate CN
- APNS Certificate Serial Number
- APNS Certificate Expiration Date
- The Apple ID you want to migrate from
- The Apple ID you want to migrate to

For more information, please see below the jump:

You can obtain the following information from the Apple Push Certificates Portal:

- APNS Certificate Subject DN
- APNS Certificate CN
- APNS Certificate Serial Number
- APNS Certificate Expiration Date

To see how to do this, please use the following procedure:

1. Log into the Apple Push Certificates Portal using the Apple ID you want to migrate from.

Screenshot 2023 04 11 at 3 48 59 PM

2. Make a note of the current certificate’s expiration date.

Screenshot 2023 04 11 at 3 49 59 PM

3. Click the **(i)** button to display the certificate information.

Screenshot 2023 04 11 at 3 50 22 PM

4. Make a note of the APNS certificate’s serial number.

Screenshot 2023 04 11 at 3 50 23 PM

5. Make a note of the APNS certificate’s Certificate Subject DN.

Note: Even though it may be displayed in the Portal site as being multiple lines, the Certificate Subject DN should be a one-line entry when you send it to Apple.

Screenshot 2023 04 11 at 3 50 24 PM

6. Make a note of the APNS certificate’s CN.

Note: The CN is included as part of the Certificate Subject DN information. It will be a string with information similar to this:

```
CN=APSP:0e77f39b-e9c8-42f9-8e8b-b5508c4abe95
```

view raw

gistfile1.txt

hosted with by GitHub

Screenshot 2023 04 11 at 3 50 25 PM

For example, if you have an APNS certificate with the following information:

APNS Certificate Subject DN: C=US, CN=APSP:dc1a3263-443c-4779-a3c3-18c95dd11264, UID=com.apple.mgmt.External.dc1a3263-443c-4779-a3c3-18c95dd11264
APNS Certificate Serial Number: 3bb763753df5d8dd
APNS Certificate Expiration Date: January 4, 2024

You would convert that to the following information for Apple:

Serial Number: 3bb763753df5d8dd
Subject CN: CN=APSP:dc1a3263-443c-4779-a3c3-18c95dd11264
Subject DN: C=US, CN=APSP:dc1a3263-443c-4779-a3c3-18c95dd11264, UID=com.apple.mgmt.External.dc1a3263-443c-4779-a3c3-18c95dd11264
Expiration Date: January 4, 2024

view raw

gistfile1.txt

hosted with by GitHub

The last part is identifying the Apple ID you want to migrate from, and the Apple ID you want to migrate to. For example, if you want to migrate an APNS certificate with the information listed above from an Apple ID of **oldappleid@company.com** to an Apple ID of **newappleid@company.com**, you could send in the following request via email:

Email subject: [Apple Push Notification Service] Transferring APNS certificate with serial number 3bb763753df5d8dd from one Apple ID to another Apple ID
Email body:
I need to transfer the following APNS certificate from one Apple ID to another Apple ID:
Serial Number: 3bb763753df5d8dd
Subject CN: CN=APSP:dc1a3263-443c-4779-a3c3-18c95dd11264
Subject DN: C=US, CN=APSP:dc1a3263-443c-4779-a3c3-18c95dd11264, UID=com.apple.mgmt.External.dc1a3263-443c-4779-a3c3-18c95dd11264
Expiration Date: January 4, 2024
Current Apple ID: oldappleid@company.com
New Apple ID: newappleid@company.com
Please let me know if you need any additional information.
Thanks,
Your Name Goes Here

view raw

gistfile1.txt

hosted with by GitHub

That should provide all the information Apple should need for a successful migration of an APNS certificate.

ScaleFusion

ScaleFusion

Scalefusion macOS PPPC Configuration (Privacy Permissions)

This guide explains how to configure **Privacy Preferences Policy Control (PPPC)** in Scalefusion for macOS devices. This allows permissions like Accessibility, Full Disk Access, and Screen Recording to be granted silently via MDM.

Overview

PPPC policies control macOS privacy permissions for applications such as:

- MDM agents
- Remote support tools
- Monitoring and security applications

Common applications:

- **Scalefusion MDM Client**
- **Remote Support**

> **Important:** Devices must be **Supervised** and enrolled via **Apple Business Manager / Apple School Manager (ADE)**. Otherwise, macOS will ignore or partially apply these settings.

Prerequisites

- Device enrolled in Scalefusion MDM
- Device is supervised
- Access to Terminal on a Mac
- Application installed or available on the system

Step 1 – Get Bundle ID

Scalefusion MDM Client:

```
osascript -e 'id of app "Scalefusion-MDM Client"'
```

Remote Support:

```
osascript -e 'id of app "Remote Support"'
```

Example output:

```
com.promobitech.scalefusion.mac
```

Step 2 – Get Code Requirement

Scalefusion MDM Client:

```
codesign -dr - "/Applications/Scalefusion-MDM Client.app" 2>&1
```

Remote Support:

```
codesign -dr - "/Applications/Remote Support.app" 2>&1
```

Example output:

```
designated => identifier "com.promobitech.scalefusion.mac" and anchor apple gener
```

Important: Copy everything after `designated =>` as a single line.

Step 3 – Configure in Scalefusion

Navigate to:

```
Device Profiles & Policies → Apple Configurations → Privacy Preferences (PPPC)
```

Step 4 – Create App Permission Entry

Field	Value
Identifier Type	Bundle ID
Bundle ID	From Step 1
Code Requirement	From Step 2
Static Code	Unchecked
State	Grant

Step 5 – Required Permissions

Scalefusion MDM Client:

- Accessibility ? Grant
- Full Disk Access ? Grant
- Reminders ? Grant

Remote Support:

- Accessibility ? Grant
- Screen Recording ? Grant
- Full Disk Access ? Grant (optional)

Common Mistakes

- **Using "/" in Code Requirement** ? Incorrect
- **Wrong app name/path** ? Must match exactly
- **Multi-line Code Requirement** ? Must be single line
- **Wrong Bundle ID** ? Each app is different

Verification

1. Open System Settings
2. Go to Privacy & Security
3. Check:
    1. Accessibility
    2. Full Disk Access
    3. Screen Recording
4. Confirm the app is already enabled

Troubleshooting

- Ensure device is supervised
- Ensure ADE enrollment
- Verify Bundle ID and Code Requirement
- Ensure app is installed via MDM

Summary

To successfully configure PPPC in Scalefusion:

- Use correct Bundle ID
- Use exact Code Requirement
- Ensure proper MDM supervision

When done correctly, permissions are granted automatically with no user interaction required.