Software Distribution Management

VMware Workspace ONE UEM 2111

vmware®

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

Software Distribution Management for macOS and Win32 Applications

1

Workspace ONE UEM powered by AirWatch offers software distribution that helps you deploy macOS and Win32 applications from the **Resources** section so that you can use the application flow that exists for all the internal applications.

This chapter includes the following topics:

Software Distribution for macOS applications

Workspace ONE UEM offers a flexible deployment through the integration with Munki, a widely renowned open-source tool. All the macOS application file types (`.dmg`, `.pkg`, `.mpkg`) can be managed in the Internal Applications section in the UEM console (**Resources > Apps > Native > Internal**).

The flexible deployment feature resides in the **Assign** sections of the application area and offers advantages to the assigning process:

- Configure deployment assignments.
- Assign multiple deployments simultaneously.
- Order assignments so that critical deployments are not missed due to the limited bandwidth.
- Customize assignments for multiple smart groups.

Software Distribution for macOS application Deployment Requirements

You can deploy macOS applications with the software distribution using the supported file types, platform version, and agents.

Supported Platform Version

macOS 10.10+

Supported File Types

PKG, DMG, MPKG

Supported Agents

- Workspace ONE Intelligent Hub for macOS 3.0
- (Optional) Workspace ONE 1.0 native application

Considerations

- **pkginfo metadata file generation** – You can upload all primary macOS software file types through **Resources > Apps > Internal** application. A PKG file can be a Bootstrap package, or it can be managed through a full lifecycle management. To configure advanced management features for macOS software through the integrated Open-Source Munki library, you must generate a metadata file for the application before uploading the application to the UEM console. You can generate a `pkginfo` metadata file using Generate Metadata Using VMware AirWatch Admin Assistant Tool.
- **Third-Party Integration** – Apart from using the Admin Assistant tool to generate metadata or a pkginfo file, you can also integrate with AutoPkg and AutoPkgr tools that have a ready-made software with configuration features. They perform periodic checks for updates to the third-party software and notify the admins.
- **Migration from Munki setup to Workspace ONE UEM** – You can add the existing application with the direct link of the application on your current Munki Repository server. This method is advantageous, as there is no requirement for an actual upload of the file to Workspace ONE UEM, which uses Workspace ONE UEM File Storage space.
- **CDNs and File Storage Systems** – All deployments use a content delivery network (CDN) to deploy applications. This method has the advantage of sending the content to devices in the network and to remote devices. It also offers an increased download speed and reduces the bandwidth on the Workspace ONE UEM servers.

Configure Workspace ONE UEM for the Software Distribution of macOS applicaton

You can configure Workspace ONE UEM to recognize the deployment of macOS applications through the software distribution method.

Prerequisites

To initiate the software management lifecycle for macOS applications, enable the software management feature (SaaS or on-premises) on the Workspace ONE UEM console.

Procedure

1. Navigate to **Settings > Devices & Users > Apple > Apple macOS > Software Management**.
2. Enable Software Management. At this point, make sure that you verify if the **File Storage** is enabled. If there is no file storage enabled, you are requested to enable it.

3 On-premises environments use a file storage system to store the large macOS applications and also use a CDN to download the applications and to reduce the bandwidth on other servers.

Generate Metadata Using VMware AirWatch Admin Assistant Tool

The VMware Admin Assistant tool uses a Munki command-line utility to give admins an easy way to create the pkginfo metadata files that you must enforce software management.

Prerequisites

Workspace ONE UEM requires pkginfo metadata file with the application file to manage the deployment in the UEM console.

Note The VMware Admin Assistant Tool is available in the UEM console, and at https://my.workspaceone.com/products/Workspace-ONE-Admin-Assistant-Tool/macOS/v2.0.3/awall. The Admin Assistant is also built with an auto-update mechanism, which updates to the latest version based on the AppCast.XML file available at https://getwsone.com/AdminAssistant/VMwareWorkspace ONE UEMAdminAssistant.xml.

Procedure

1 Click open the Admin Assistant tool. The Assistant dialog box asks you to upload the application installer files for the Assistant to parse.

2 Upload an application installer file by dragging and dropping a .pkg, .dmg, .app, or .mpkg file, or browse your local files for an installer file.

   a When you drop or select a file, the tool initiates the process. If needed, you can add more files during this time.

   b If you select an .app file, the tool creates a .dmg containing the file.

   For more information, see Introduction to VMware Workspace ONE Admin Assistant for macOS.

What to do next

After the parsing is finished, the tool prompts you to reveal the parsed metadata files in Finder. Store the metadata files in a local folder where you can easily retrieve them during the Software distribution procedure.

Deploy Internal macOS Applications

You can deploy internal applications to your mobile network by upload internal applications with local files in the Workspace ONE UEM console.

1 Navigate to **Resources** > **Apps** > **Native** > **Internal** and select **Add Application**.

2 Select **Upload** > **Local File** and browse for the application file on your system. Select the .dmg, .pkg, or .mpkg file to upload.

3 Upload the required application metadata file (.plist).

To create a metadata file, download and install the VMware Workspace ONE UEM Admin Assistant Tool to your macOS computer.

4 Complete the **Images** tab.

Setting	Description
Mobile Images	Upload or drag application image to display in the AirWatch Catalog for mobile devices.
Tablet Images	Upload or drag the application image to display in the AirWatch Catalog for tablets.
Icon	Upload or drag the application image to display in the AirWatch Catalog as the icon for the application.

5 Configure **Scripts** settings to run the installation, uninstallation, and verification of the application. By providing pre-install scripts and post-install scripts, you can perform additional configuration tasks or install additional items without the need of repacking the applications or software. Simply paste the script and Workspace ONE UEM formats it to be used by Munki.

Setting	Description
Pre-Install Script	Define a pre-install script to run before attempting installation.
Post-Install Script	Define a post-install script to run after a successful installation.
Pre-Uninstall Script	Define a pre-uninstall script to run before an attempted uninstall.
Uninstall Method	Select from the drop-down and customize the behavior of the uninstall methods. The options are: ■ **Remove Packages** ■ **Remove Copied items** ■ **Remove app** ■ **Uninstall script**
Post Uninstall Script	Define a post-uninstall script to run after a successful uninstall. **Note** Failure of the pre-install script cancels the installation attempt and failure of the post-install script logs errors, but the install is considered complete.
	With some software, you have to configure what exactly defines a successful install or uninstall. Munki allows the software configuration through setting an Install or Uninstall Check Script.
Install Check Script	If present, the script runs to determine if the application must be installed. A return code of 0 means install is needed, any other return code causes install to be skipped.
Uninstall Check Script	If present, the script runs to determine if the application must be uninstalled. A return code of 0 means uninstall is needed, any other return code causes uninstall to be skipped.

6 Configure the **Deployment** tab settings.

Setting	Description
Blocking	Activate **Blocking Applications** to define any applications or processes that might block the clean installation of a managed macOS application. Defined applications that must be closed before the installation to prevent those applications from being quit unexpectedly before saving. Additionally, the end users are notified on the device by the Workspace ONE Intelligent Hub to close the defined applications. Deactivate **Blocking Applications** to override any blocking behavior and continue with the installation. If there is an app open which blocks the installation, it will be automatically closed.
	List the blocking applications that must be closed. If the app is in the /Applications/ folder, it can be defined as the app name and the path is automatically discovered. For example, "Firefox" or "Firefox.app". Optionally the full path to the exact process can be used, but is not advised if end users do not can easily quit the app. As a result you must not block any faceless background apps or helper apps.
Restart Action	Select the restart action for the application. The available actions are: ■ **None** ■ **Require Shutdown** ■ **Require Restart** ■ **Recommend Restart** ■ **Require Logout**
Condition	Define the condition for the application to be installed on the device.
Desired State Management	Currently when installing macOS software, administrators can activate or deactivate the Desired State Management settings based on the business needs. Desired State Management is enabled by default to enforce the application management during the macOS software installation. If activated, and if the end user deletes the app, the application is automatically reinstalled on the next Hub sync. If deactivated, and if the end user deletes the app, the application is not automatically reinstalled, unless pushed from the Workspace ONE UEM console or Catalog.

7 Configure the **Terms of Use** tab.

8 Terms of use states specifically how users are expected to use the application. When the application pushes to devices, users view the terms of use that they must accept to use the application. If users do not accept, they cannot access the application.

9 Select **Save & Assign**.

Using macOS Software Distribution Scripts for additional Configuration

Use macOS software distribution scripts to perform additional configurations or validation of tasks in the **Script** section of the **Add or Edit Application** page of the console.

By inserting scripts, you can:

- Avoid repacking installers by using pre-install scripts.
- Avoid post-install user prompts by scripting additional configurations.

- Perform validation.
- Customize the uninstallation.

The following table provides exit code behavior for each script type.

Script Type	Exit Code 0 Behavior	Other exit Code Behavior
Pre-Install	Continue Install	Skip Install
Post-Install	Successfully Installed	Installed Successfully with Warnings
Pre-Uninstall	Continue Uninstall	Skip Uninstall
Post-Uninstall	Uninstall Successfully	Uninstall successfully with Warnings
Install Check Script	Install is Needed	Skip Install
Uninstall Check Script	Uninstall is Needed	Skip Uninstall

macOS Software Distribution Conditions

macOS software distribution conditions are a set of attributes provided by the integrated open-source Munki library for determining the install applicability. Conditions are defined at a per-application level and are evaluated before download and install of the software.

There are some built-in conditions supported by Munki.

Conditions are written in the format:

```
machine_type == "laptop" AND os_vers BEGINSWITH "10.7"
```

Conditional Comparison Attributes

Attribute	Type	Description	Example Comparison
hostname	String	Hostname	hostname=="Lobby imac"
arch	String	Processor architecture. For example: 'powerpc', 'i386', 'x86_64'	arch=="x86_64"
os_vers	String	Full OS Version. For example: "10.7.2"	os_vers BEGINSWITH "10.7"
os_vers_major	Integer	Major OS Version. For example: '10'	os_vers_major == 10
os_vers_minor	Integer	Minor OS Version. For example:'7'	os_vers_minor == 7
os_vers_patch	Integer	Point release version. For example: '2'	os_vers_patch >=2
machine_model	String	'MacMini1,1', 'iMac4,1', 'MacBookPro8,2'	machine_model == "iMac4,1"
machine_type	String	'laptop' or 'desktop'	machine_type == "laptop"

ipv4_address	Arrays of string	Contains current IPv4 addresses for all interfaces.	ANY ipv4_address CONTAINS '192.168.161.'
munki_version	String	Full version of the installed munkitools	munki_version LIKE '*0.8.3*'
serial_number	String	Machine serial number	serial_number =="W9999999U2P"
date	UTC date string	Date and time. Note the special syntax required to cast a string into an NSDate object.	date>CAST("2013-01-02T00:00:00Z", "NSDATE")

Assign Applications to your macOS Devices

Once you configure an application, add a single assignment or multiple assignments. If you add multiple assignments, prioritize the importance of the assignment by moving its place in the list up for most important or down for least important.

1 Navigate to **Resources** > **Apps** > **Native** > **Internal** or **Public**.

2 Upload an application and select **Save & Assign** or select the application and choose **Assign** from the actions menu.

3 Select **Add Assignment** and complete the following options.

Setting	Description
Select Assignment Groups	Type a smart group name to select the groups of devices to receive the assignment.
App Delivery Method	■ **On Demand** – Deploys content to a catalog or other deployment Hub and lets the device user decide if and when to install the content. This option is the best choice for content that is not critical to the organization. Allowing users to download the content when they want helps conserve bandwidth and limits unnecessary traffic. ■ **Automatic** – Deploys content to a catalog or other deployment Hub on a device upon enrollment. After the device enrolls, the Workspace ONE Intelligent Hubautomatically installs the app without needing user interaction. This option is the best choice when it is critical to your organization and its mobile users.
Deployment Begins On Internal Applications	Set a day of the month and a time of day for the deployment to start. The **Priority** setting governs which deployments push first. Workspace ONE UEM then pushes deployments according to the **Effective** configuration. To set a beginning date with enough bandwidth for successful deployment, consider the traffic patterns of your network .

4 Select **Add**.

5 Use the **Move Up** and **Move Down** options to order assignments if you have more than one. Place critical assignments at the top of the list. This configuration displays as the **Priority**.The **Priority** setting takes precedence when there are conflicting deployments assigned to a single device.

6 Select **Save & Publish**.

Manage your macOS Software Distribution Updates

Once the macOS application or software is deployed, the deployed application or software can be managed from the Workspace ONE UEM console. You can manage updates by uploading a new version of the file onto the UEM console.

1 Navigate to **Resources** > **Apps** > **Native.**

2 Select the application that you want to update.

3 On the top right of the **Details** page, select **Add Version**.

4 Upload the installer and the .pkginfo file of the new version.

5 If necessary, perform additional changes and then **Save**.

6 Select **Save & Assign**.

Install macOS Applications using `Munki`

`Munki` uses information from the `pkginfo` file and looks for the software items to decide whether or not a given item must be installed. To create a functional `pkginfo` item, understand the methods used by `Munki` to check the list of software items.

Important Most of the content under this section are from the `Munki` website.

Methods

In the order of precedence, `Munki` uses the following methods to determine if the given item needs to be installed (or removed):

1 Install macOS Applications using the Check Script

2 Install macOS Applications using the Install Items

3 Install macOS Applications using the Receipts

When combining these methods, only the highest priority method is used. For example, if a given `pkginfo` item has both an "installs" list and a "receipts" list, the receipts is ignored for purposes of determining the installation status. Even in this case, though, receipts may be used when removing an item, as they help `Munki` determine exactly which files were installed.

Install macOS Applications using the Check Script

A pkginfo item may optionally contain an **installcheck_script**. Install ckeck script provides a method for determining if an software item needs to be installed, where providing **installs/ receipts** is inadequate or impractical.

Command-line tools typically installed through port (macports) or Python modules installed using easy_install or pip are prime examples as they provide no easy method for determining their installed version.

An install check_script should be written such that an exit code of 0 indicates that the item is currently not installed and should therefore be installed. All non-zero exit codes indicate that the item is installed.

An example of installcheck_script illustrating a check to determine if the current version of the argparse Python module is installed.

```
#!/bin/sh# Grab current version of installed python moduleversion="$(python -c 'import
argparse;print argparse.__version__' 2>/dev/null)"# Compare with the version we want to
installif [ ${version:-0} < 1.2.1 ]; thenexit 0elseexit 1fi
```

Optionally, an explicit **uninstallcheck_script** can be provided to determine whether or not an software item should be removed. In this case, the script with an exit code of 0 indicate that the item is currently installed and that removal should occur. All non-zero exit codes indicate that the item is not installed.

Install macOS Applications using the Install Items

The install items list is generated by the VMware AirWatch Admin Assistant for some types of installation items(.dmg), but not for Apple packages (.pkg or .mpkg). You can generate (or modify) this list and is the most flexible mechanism for determining the installation status.

The **installs** list can contain any number of items such as applications, preference panes, frameworks, or other bundle-style items, info.plists, simple directories, or files. You can use any combination of items to help `Munki` determine if an item is installed or not.

An example of an auto-generated "installs" list for Firefox 6.0

```
<key>installs</key><array><dict><key>CFBundleIdentifier</key><string>org.mozilla.firefox</
string><key>CFBundleName</key><string>Firefox</string><key>CFBundleShortVersionString</key><string>6.0</
string><key>minosversion</key><string>10.5</string><key>path</key><string>Applications/Firefox.app</
string><key>type</key><string>application</string></dict></array>
```

To determine if Firefox 6 is installed or not, `Munki` checks for an application with a CFBundleIdentifier of org.mozilla.firefox and if found, verifies that its version (CFBundleShortVersionString) is at least 6.0. If `Munki` cannot find the application or its version is lower than 6.0, it considers Firefox-6.0 as not installed. Installs lists can contain multiple items. If any item is missing or has an older version, the item is considered not installed. You can manually generate items to add to an **installs** list using the following `pkginfo`:

```
/Library/Application\ Support/AirWatch/Data/Munki/bin/makepkginfo -f /Library/Interne
t\ Plug-Ins/Flash\ Player.plugin
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Prope
rtyList-1.0.dtd">
<plist version="1.0">
<dict>
```

```
 <key>installs</key>
 <array>

  <dict>

   <key>CFBundleShortVersionString</key>
   <string>10.3.183.5</string>
   <key>path</key>
   <string>/Library/Internet Plug-Ins/Flash Player.plugin</string>
   <key>type</key>
   <string>bundle</string>

  </dict>

 </array>

</dict>
</plist>
```

Copy and paste the entire **installs** key and value, or copy the `dict` value and add it to an existing installs list inside your `pkginfo` file. `Munki` checks for the existence of /Library/Internet Plug-Ins/Flash Player.plugin and if found, check its version. If the version is lower than 10.3.183.5, the item is be considered not installed. You can generate installs items for any filesystem item, but `Munki` only knows how to determine the versions for bundle-style items that contain an Info.plist or version.plist with the version information.

For other filesystem items, `Munki` can only determine the existence of a non-bundle directory, or can calculate a checksum (for files). For files with checksums, the test fails (and therefore the item is considered not installed) if the checksum for the file on disk does not match the checksum in the `pkginfo`.

```
<key>installs</key>
<array>

 <dict>

  <key>md5checksum</key>
  <string>087fe4805b63412ec3ed559b0cd9be71</string>
  <key>path</key>
  <string>/private/var/db/dslocal/nodes/MCX/computergroups/loginwindow.plist</s
  tring>
  <key>type</key>
  <string>file</string>

 </dict>

</array>
```

If you want Munki to only check for the existence of a file and do not care about its contents, remove the generated md5checksum information in the installs item info. Make sure the provided path is intact.

```
<key>installs</key>
<array>

 <dict>
 <key>path</key>
 <string>/private/var/db/dslocal/nodes/MCX/computergroups/loginwindow.plist
 </string>
 <key>type</key>

  <string>file</string>

 </dict>

</array>
```

Install macOS Applications using the Receipts

When an Apple-style package is installed, you can generate a receipt on the machine. Metapackages generate multiple receipts. The VMware AirWatch Admin Assistant adds the names and versions of those receipts to a receipts array in the `pkginfo` for a package.

Following is a receipts array for the Avid LE QuickTime codecs, version 2.3.4.

```
<key>receipts</key>
<array>

 <dict>

  <key>filename</key>
  <string>AvidCodecsLE.pkg</string>
  <key>installed_size</key>
  <integer>1188</integer>
  <key>name</key>
  <string>AvidCodecsLE</string>
  <key>packageid</key>
  <string>com.avid.avidcodecsle</string>
  <key>version</key>
  <string>2.3.4</string>

 </dict>

</array>
```

If `Munki` is using the receipts array to determine the installation status, it verifies for the existence and the version of each receipt in the array. If any receipt is missing or has a lower version number than the version specified for that receipt in the receipts array, the item is considered not installed. Only if every receipt is present and all versions are the same as the ones in the pkginfo (or higher) is the item considered installed. To troubleshoot issues, use the `pkgutil` tool to examine the installed receipts.

```
# pkgutil --pkg-info com.avid.avidcodecsle
No receipt for 'com.avid.avidcodecsle' found at '/'.
```

In this case, the receipt for the Avid LE QuickTime codecs was not found on this machine. A common complication with receipts is, with many metapackages, the installation logic results in only a subset of the subpackages being installed. Generally, the receipts list contains a receipt for every subpackage in a metapackage (and needs this info if `Munki` is asked to remove the software item based on package receipts). But if it is normal and expected that not every subpackage is installed, `Munki` marks the item as not currently installed and offer to install it again and again. One solution for this issue is to add an optional key with the value of true to the receipts that are optionally installed. `Munki` does not consider these receipts when determining installation status.

```
<key>receipts</key>
<array>

 <dict>

  <key>filename</key>
  <string>mandatory.pkg</string>
  <key>installed_size</key>
  <integer>1188</integer>
  <key>name</key>
  <string>Mandatory</string>
  <key>packageid</key>
  <string>com.foo.mandatory</string>
  <key>version</key>
  <string>1.0</string>

 </dict>
 <dict>

  <key>filename</key>
  <string>optional.pkg</string>
  <key>installed_size</key>
  <integer>1188</integer>
  <key>name</key>
  <string>Optional</string>
  <key>optional</key>
  <true/>
  <key>packageid</key>
```

```
            <string>com.foo.optional</string>
            <key>version</key>
            <string>1.0</string>

           </dict>

          </array>
```

Another solution for this situation is to provide the `intalls` array with the lists items that are installed by the package. `Munki` can use the `installs` array information instead of the receipts to determine the installation status.

Troubleshooting macOS Software Distribution Deployment

This section helps you understand how to troubleshoot problems related to the macOS software distribution process. It also details you on the path to verify the logs.

Troubleshooting Issues

- How to verify on the device locally that an application is assigned?

  All assigned applications are shown in /`Library/Application\ Support/Workspace ONE UEM/Data/Munki/Munki_Repo/manifests/device_manifest.plist` in the `managed_installs` array.

  Furthermore, all assigned applications have their corresponding `pkginfo` stored in the catalog `plist` at `/Library/Application\ Support/Workspace ONE UEM/Data/Munki/Munki_Repo/catalogs/device_catalog.plist`.

- How to verify on the console that an application is assigned?

  In the internal applications **List View** page, select the application to go to the application **Details** page. Then select the **Devices** tab. This page shows the application install statuses for all assigned and enrolled devices.

- How to get direct access to `Munki` logs?

  `Munki` Logs can be directly accessed on the device in the path:

  `/Library/Application Support/Workspace ONE UEM/Data/Munki/Managed\Installs/Logs/`, where they are saved as ManagedSoftwareUpdate.log files.

- Where to look for device report data on the UEM console?

  The UEM console reports data from the device in a few locations.

  - Navigate to **Resources > Apps > Native > Internal**. Select an application and access **Application Details > Devices** tab to view the install statuses for each device.
  - Navigate to **Devices & Users > Devices > List View** and select a device to access **Device Details > Troubleshooting** tab. You can view the activities performed on the device and filtering options to show the information relating to the software distribution.

Uninstalling the Software distribution of your macOS application

There are multiple methods available for the uninstallation of software and the appropriate method is selected by default by the VMware Admin Assistant tool based on the file type. If needed, you can override the default with any of the following methods.

Remove Copied Items

The Remove Copied Items method is primarily used for DMG file types, where it pulls from the items_to_copy array[dicts] array in the `pkginfo` file and deletes all file paths in the array.

Future Console release shows the paths in the items_to_copy array in the UI.

Remove App

The `Remove App` method pulls from the installs array [dicts] in the `pkginfo` file and deletes all file paths in the array.

Future Console release shows the paths in the installs array in the UI.

Remove Packages

The Remove Packages method is used primarily for PKG file types. This method:

- Uses receipts and analyzes the packages to remove.
  - Tries to determine what all files were installed through Bom file
  - Deletes receipt
- Removes non-associated packages only

Future Console release shows the receipts that the `Munki` checks for in the UI.

Uninstall Script

Uninstall scripts are written in a shell script. This method is:

- Used for any installer type
- Used to perform a custom uninstall operation. If you have a customized deployment for an application, then write a corresponding uninstall script to remove the custom configurations.

Software Distribution of Win32 Applications

With software distribution, Workspace ONE UEM can deploy Win32 applications from the **Resources** section so that you can use the application flow that exists for all internal applications.

If you have scripting needs, use the product provisioning feature described in the *VMware Workspace ONE UEM Product Provisioning for Windows Desktop Guide* on the VMware Docs site at https://docs.vmware.com/en/VMware-Workspace-ONE-UEM/index.html.

Software Distribution for Win32 Application Deployment Requirements

To deploy Win32 applications with the software distribution, use supported file types, operating systems, and platforms.

Supported Platforms

The supported platform to deploy Win32 Applications is Windows Desktop.

Supported File Types

- MSI
- EXE
- ZIP

> **Note** If using a ZIP file, compress application packages that are 4GB or larger using 7-Zip. Workspace ONE UEM does not decompress ZIP packages containing application packages of 4GB or larger when compressed using the native Windows zip compressor.

File Storage and CDN

File storage needs to be enabled for the distribution of applications larger than 200MB. Configuring file storage manually is only applicable to on-premises customers. It is configured automatically for SaaS customers. It also includes certain reports, internal application deployment, and Workspace ONE UEM managed content. When you enable file storage for any of these functionalities, it is applied to the others automatically. For more information, see Installation / File Path in the System Settings guide.

Workspace ONE supports integration with Akamai CDN for distributing app packages. Configuring CDN manually is only applicable to on-premises customers. It is configured automatically for SaaS customers. CDN integration enables fast downloads for devices distributed across different geographic locations as well as reduces the load on your Workspace ONE Device Services servers. For more information, see Akamai CDN Integration Guide.

Software Package Deployment Enabled for All

Software distribution is now turned on by default in the Workspace ONE UEM console for all customers. By default, customers get up to 50 GB of storage for applications in the database. On-premises customers can adjust the **App Capacity** for **SFD** in **Groups & Settings** > **All Settings** > **Admins** > **Storage**.

File Storage for your Win32 Applications

Configuring file storage manually is only applicable to on-premises customers. It is configured automatically for SaaS customers. It also includes certain reports, internal application deployment, and Workspace ONE UEM-managed content. When you enable file storage for any of these functionalities, it is applied to the others automatically. For more information, see Installation / File Path in the System Settings guide.

Troubleshooting your Win32 Application using Patch Updates

You can use patches to update and fix Win32 applications. Workspace ONE UEM supports additive and cumulative patches. In certain cases, a cumulative patch might trigger the system to create a version of an application.

Cumulative Patches and System Deployment Behavior: When you apply a cumulative patch by editing an application, the system creates a version of the application with the new patch applied. It makes the non-patched version inactive and creates and deploys the patched version of the application to devices.

Patch Restrictions: Workspace ONE UEM does not support patches that do not update the version, and the upgrade code must match the Win32 MSI application.

Win32 Application Installation Behavior

Workspace ONE UEM console includes different ways you can deploy Win32 applications. Select various installation combinations for software distribution or use product provisioning.

Product Provisioning Alternative

It is a best practice to deploy Win 32 applications from **Resources**. However, if you have tried deploying the application with **Resources** and you are not able to meet your needs, as an alternative method you can complete the deployment onto your devices using **Product Provisioning**.

Note Users do not receive User Account Control (UAC) prompts for all the applications that only require standard permissions.

Win32 Application Installation Behavior Using Resources

Refer the table to understand Win 32 Application Installation Behavior for all the apps that require admin privileges.

Configuring Win32 Application from Resources	Install Context Settings In the Workspace ONE UEM console	User is an Admin	User is a standard user
Navigate to **Resources > Apps > Native > Internal** select **Add Application**	Navigate to **Deployment options > How To Install** and set ■ **Install Context** = **Device** ■ **Admin Privileges** = **Yes** The settings indicate that the app is configured for all the users on each of your devices and the user account has an elevated access token to install the application.	■ **Install Context** set to **Device** ■ **Admin Privileges** set to **Yes** ■ User is an admin The installation completes without any prompt.	■ **Install Context** set to **Device** ■ **Admin Privileges** set to **Yes** ■ User is a standard user The installation completes without any prompt.
Navigate to **Resources > Apps > Native > Internal** select **Add Application**	Navigate to **Deployment options > How To Install** and set ■ **Install Context** = **Device** ■ **Admin Privileges** = **No** The settings indicate that the app is configured for all the users on each of your devices and the user account need not have an elevated access token to install the application.	■ **Install Context** set to **Device** ■ **Admin Privileges** set to **No** ■ User is an admin The installation completes without any prompt.	■ **Install Context** set to **Device** ■ **Admin Privileges** set to **No** ■ User is a standard user The installation completes without any prompt.
Navigate to **Resources > Apps > Native > Internal** select **Add Application**	Navigate to **Deployment options > How To Install** and set ■ **Install Context** = **User** ■ **Admin Privileges** = **Yes** The settings indicate that the app is configured for all the users on each of your devices and the user account has an elevated access token to install the application.	■ **Install Context** set to **User** ■ **Admin Privileges** set to **Yes** ■ User is an admin The installation completes without any prompt.	■ **Install Context** set to **User** ■ **Admin Privileges** set to **Yes** ■ User is a standard user The installation fails.
Navigate to **Resources > Apps > Native > Internal** select **Add Application**	Navigate to **Deployment options > How To Install** and set ■ **Install Context** = **User** ■ **Admin Privileges** = **No** The settings indicate that the app is configured for all the users on each of your devices and the user account need not have an elevated access token to install the application.	■ **Install Context** set to **User** ■ **Admin Privileges** set to **No** ■ User is an admin The installation completes with prompt.	■ **Install Context** set to **User** ■ **Admin Privileges** set to **No** ■ User is a standard user The installation fails.

Win32 Application Installation Behavior Using Product Provisioning

It is a best practice to deploy Win 32 applications from **Resources**. However, if you have tried deploying the application with **Resources** and you are not able to meet your needs, as an alternative method you can complete the deployment onto your devices using **Product Provisioning**.

If you are configuring Win32 applications using product provisioning, you can use the following table to understand the combinations of **Install** and **Run** manifest and the context of the command. You can select install or run at the system level, user level, or admin account level. Based on the selections made, your installation can vary.

Refer the table to understand the Win32 Application Installation Behavior Using Product Provisioning

Table 1-1. Win32 Application Installation Behavior Using Product Provisioning

Configuring Win32 Application	Install/ Run Settings in the Products Provisioning in the UEM console	User is an Admin	User is a standard user
Navigate to **Devices > Provisioning > Components > Files/ Actions and select Add Files/Actions**.	Navigate to**Manifest** tab and set ■ **Action(s) To Perform** = **Install/ Run** ■ **Execution Context** = **System**	■ **Action(s) To Perform** = **Install/ Run** ■ **Execution Context** = **System** ■ User is an admin The installation completes without any prompt.	■ **Action(s) To Perform** = **Install/ Run** ■ **Execution Context** = **System** ■ User is a standard user The installation completes without any prompt.
Navigate to **Devices > Provisioning > Components > Files/ Actions and select Add Files/Actions**	Navigate to **Manifest** tab and set ■ **Action(s) To Perform** = **Install/ Run** ■ **Execution Context** = **Admin**	■ **Action(s) To Perform** = **Install/ Run** ■ **Execution Context** = **Admin** ■ The user is an admin The installation completes without any prompt.	■ **Action(s) To Perform** = **Install/ Run** ■ **Execution Context** = **Admin** ■ User is a standard user The installation completes with prompt.
Navigate to **Devices > Provisioning > Components > Files/ Actions and select Add Files/Actions**	Navigate to **Manifest** tab and set ■ **Action(s) To Perform** = **Install/ Run** ■ **Execution Context** = **User**	■ **Action(s) To Perform** = **Install/ Run** ■ **Execution Context** = **User** ■ User is an admin The installation completes without any prompt.	■ **Action(s) To Perform** = **Install/ Run** ■ **Execution Context** = **User** ■ User is a standard user The installation fails.

Considerations for Retry Count, Retry Interval, Install Timeout and Device Restart for your Win32 Applications

The values for **Retry Count**, **Retry Interval**, and **Install Timeout** options for Win32 applications affect the length the system takes to report a failed installation process. You can change the default values to decrease deployment times.

The default values for the options

- **Retry Count** - three times
- **Retry Interval** - five minutes
- **Install Timeout** - 60 minutes

work in the following sequence for a single failed installation process.

Table 1-2. Time to Install and Failure Reported

60 minutes (one hour)	65 minutes (one hour and five min)	125 minutes (two hours and five min)	130 minutes (two hours and 10 min)	190 minutes (three hours and 10 min)	195 minutes (three hours 15 min)
Win32 app fails to install and reaches install the timeout of 60 minutes.	System retries the installation (retry count #1) at a retry interval of 5 minutes.	Win32 app fails to install and reaches install timeout of 60 minutes.	System retries the installation (retry count #2) at a retry interval of 5 minutes.	Win32 app fails to install and reaches install the timeout of 60 minutes.	System retries the installation (retry count #3) at a retry interval of 5 minutes.

After 3 hours and 15 minutes, the system reports a single application installation as failed. Then, the system installs the next application.

Configure Options Depending on the Application

Configure values that compliment the application.

Fast Installation Example

A browser application installs on a device in four minutes. Consider setting these values for this application.

- Retry Count - two times
- Retry Interval - five minutes
- Install Timeout - five minutes

The system reports the failure of this application within 20 minutes. Then, it installs the next application.

Slow Installation Example

A large productivity application installs on a device in 30 minutes. Consider these values for these applications.

- Retry Count - three times
- Retry Interval - five minutes
- Install Timeout - 35 minutes

The system might report the failure of this application within 120 minutes. Then, it installs the next application.

Considerations for Device Restart for your Win32 Applications

The values for **Device Restart** for Win32 applications allow the user to defer the device reboot and associate a deadline time up to which the user can defer the reboot. The values allow admins and end users to have more control over restarts to prevent a user from losing work. Admins can choose to force restart a PC after application installation or allow the user to postpone the restart for a more convenient time.

Device restart helps you configure the following settings:

- Prompt the user before the device restart so that they can save their file and close applications.
- Prompt the user when a device restart is required.
- Allows the user to snooze the device restart as per their convenience.
- Allows administrators to force the device restart.
- Allows administrators to set a deferral deadline after which the users cannot snooze the restart.

The Workspace ONE Intelligent Hub shows notifications regarding device reboots at various stages. The deferral notification allows the user to restart or snooze. Workspace ONE Intelligent Hub updates the reboot data for snooze or restart and shows that notification according to the time selected by the user.

The following table shows notifications that are displayed at various stages:

System Alert	Description
During the application installation	Notifies the user to save the files and close the application.
After application installation	Displays the first alert and notifies the user about the system restart.
48 Hours before the restart deadline	Displays the second alert and warns the user about the force restart.
15 minutes before the restart deadline	Displays the third alert and warns the user about the force restart.
5 minutes before the restart deadline	Clear system prompts that indicate the date and time of the planned force restart.

Add Windows applications from the Enterprise Application Repository

Enterprise application Repository in Workspace ONE UEM console allows you to easily set up common applications. You can add internal applications on your network with an external application repository and manage the applications with Workspace ONE UEM. Once the applications are added to the application Repository, they can then be distributed and installed on devices.

Complete the following steps to add an application using Enterprise Application Repository.

Procedure

1 Navigate to **Resources > Apps > Internal > From Enterprise App Repository** .

2 Search and select the internal application.

3 Select **Next**.

4 Edit the **Application Name** or **Managed By** if necessary, and select **Next**.

Note You can get email and console notifications when a new version of an existing app in your catalog becomes available. You can click on add application from the console notification and it automatically takes you through the steps to update and distribute the new version of your application. You can also enable notifications for the existing EAR apps by editing them from your **Apps and Books** section.

You can review the summary before adding the application. You can edit some fields later through the application list view.

5 Select **Save**.

Upload and Configure Win32 Files for Software Distribution

Software Distribution in Workspace ONE UEM allows you to deliver Windows applications and adhere to the application life-cycle. Software Distribution enables your organization to deploy applications, track the installation statuses, debug and troubleshoot installation issues, and maintain applications with ease.You can upload Win32 applications as either main files or dependency files. You can use the same process for EXE, MSI, and ZIP files. If you are using a ZIP file, you can compress the application packages that are 4GB or larger using 7-Zip. Workspace ONE UEM does not decompress ZIP packages containing application packages of 4GB or larger when compressed using the native Windows zip compressor.Software Distribution engine (SFD) requires an MSI file or an EXE file inside the ZIP file to deploy a PowerShell script file.

Complete the following steps to upload Win32 Files for Software Distribution:

Procedure

1 Navigate to **Resources > Apps > Native > Internal** and select **Add Application**.

2 Select **Upload**.

3 Select **Local File** and select the application to upload.

4 Select an answer to **Is this a dependency file**.

- Select **Yes** to tag a dependency file and associate it to Win32 applications. Examples of dependency files are libraries and frameworks.
- Select **Continue** to go to the next phase in the life cycle.

5 Configure the **Details** tab options.

Workspace ONE UEM cannot parse data from an EXE or ZIP file. Enter the information for the EXE and ZIP files on this tab. The system parses the listed information for MSI files.

Details Setting	Details Description
Name	Enter a name for the application.
Managed By	View the organization group (OG) that the application belongs to in your Workspace ONE UEM OG hierarchy.

Details Setting	Details Description
Application ID	Represents the application with a unique string. This option is pre-populated and was created with the application. Workspace ONE UEM uses the string to identify the application in systems like application whitelists and blacklists.
Actual File Version	If you are using Workspace ONE UEM console 2005 or above, you can enter the actual file version of the application. If you are using a Workspace ONE UEM console version prior to 2005, the version number displays the coded version of the application set by the application's developer. When you add an application version, the version field appears as read only for the newer version that is added.
Build Version	Displays an alternate "File Version" for some applications. This entry ensures Workspace ONE UEM records all version numbers coded for applications because developers have two places within some applications they can code a version number.
Version	If you are using Workspace ONE UEMconsole 2005 or later, you can enter the actual file version of the application. **Note** ■ If you want the existing applications in the Workspace ONE UEM catalog to display version information that is consistent with the version you are deploying, you must create a new application in the catalog and provide the appropriate actual version information, and then choose to retire the existing application. ■ You can edit the application version only when you initially upload the zip or exe file. Once you save, the application version is not editable unless you add a new application version. If you are using a Workspace ONE UEM console version prior to 2005, the version number displays the internal version of the application set by the Workspace ONE UEMconsole.
Supported Processor Architecture	Select the bit-architecture value for applicable Windows applications.
Is Beta	Tags the application as still under development and testing, a BETA version.
Update Notifications	Enable this to get notifications when a new version of an existing app in your catalog becomes available.
Change Log	Enter notes in this text box to provide comments and notes to other admins concerning the application.
Categories	Provide a category type in the text box to help identify how the application can help users. You can configure custom application categories or keep the application's pre-coded category.
Minimum OS	Select the oldest OS that you want to run this application.
Supported Models	Select all the models that you want to run this application.

Details Setting	Details Description
Default Scheme	Indicates the URL scheme for supported applications. The application is packaged with the scheme, so Workspace ONE UEM parses the scheme and displays the value in this field. A default scheme offers many integration features for your internal applications, including but not limited to the following options: ■ Use the scheme to integrate with other platform and web applications. ■ Use the scheme to receive messages from other applications and to initiate specific requests. ■ Use the scheme to start Apple iOS applications in the AirWatch Container.
Description	Describe the purpose of the application. Do not use '<' + String in the Description, as you might encounter an Invalid HTML content error.
Keywords	Enter words that might describe features or uses for the application. These entries are like tags and are specific to your organization.
URL	Enter the URL from where you can download the application and get information about it.
Support Email	Enter an email to receive suggestions, comments, or issues concerning the application.
Support Phone	Enter a number to receive suggestions, comments, or issues concerning the application.
Internal ID	Enter an identification string, if one exists, that the organization uses to catalog or manage the application.
Copyright	Enter the publication date for the application.

6 Complete the **Files** tab options by uploading dependencies, transforms, patches, and uninstallation processes.

File	Description	Configurations
App Dependencies MSI, EXE, ZIP	The environment and devices need these applications to run the Win32 application.	1 Select dependency files in the **Select Dependent Applications** option. 2 Enable the system to apply dependencies in a specified order. The system works from top to bottom.
App Transforms MST file type	These files control the installation of the application and can add or prevent components, configurations, and processes during the process.	Select **Add** to browse to the MST file on the network.

File	Description	Configurations
App Patches MSP file type	These files add changes that are fixes, updates, or new features to applications. The two types are additive and cumulative. ■ **Additive** – Includes only changes developed after the latest version of the application or the last additive patch. ■ **Cumulative** – Includes the entire application including any changes since the latest version of the application or the last patches.	1 Select **Add**. 2 Identify the patch as cumulative or additive. 3 Select**File** to browse to the MSP file on the network.
App Uninstall Process	These scripts instruct the system to uninstall an application under specific circumstances. Customized scripts are optional for MSI files.	1 Select the **Use Custom Script** option. 2 Select to upload or enter a script to the system for **Custom Script Type**. ■ Select **Upload** and browse to the script file on the network. ■ Select **Input** and enter the custom script.

7 Complete the settings in **Deployment Options > When To Install**.

This tab instructs the system to install the application with specific criteria. The system can parse information for MSI files. However, for EXE and ZIP files, the system requires you to enter this information.

a Select **Data Contingecies > Add** and complete the options that depend on the criteria type you select.

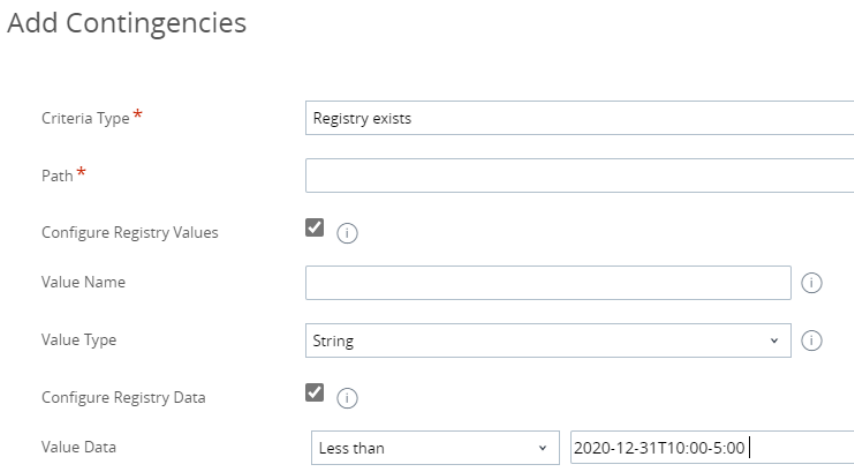
Set contingencies for instruction and completion scenarios.

- Instruction – Contingencies instruct the system to install applications when the device meets specific criteria.
- Completion – Contingencies identify when an installation is complete.

Setting - App	Description - App
Criteria Type **App exists** **App does not exist**	■ Instruction – Configure the system to install the application when a specific application is or is not on devices. ■ Completion – Configure the system to identify the installation is complete when a specific application is or is not on devices. Workspace ONE UEM checks for the existence of the application but it does not deploy the application to devices.
Application Identifier	Enter the application identifier so the system can recognize the existence or non-existence of the auxiliary application. This value is also known as the product code of the application.
Version	Enter the specific version. Use a comparison operator to specifically target an app version or to target a range of app versions.

Setting - File	Description - File
Criteria Type **File exists** **File does not exist**	■ Instruction – Configure the system to install the application when a specific file is or is not on devices. ■ Completion – Configure the system to identify the installation is complete when a specific file is or is not on devices.
Path	Enter the path on the device where you want the system to look for the file and include the filename.
Version	Enter the specific version. Use a comparison operator to target a specific file version or a range of file versions.
Modified On	Enter the date the file was last modified.

Setting - Registry	Description - Registry
Criteria Type **Registry exists** **Registry does not exist**	■ Instruction – Configure the system to install the application when a specific registry value is or is not on devices.

Setting - Registry	Description - Registry
	■ Completion – Configure the system to identify the installation is complete when a specific registry value is or is not on devices.
Path	Enter the path on the device where the system can find the keys and values. Include the entire path, beginning with HKLM\ or HKCU\.
Configure Registry Values	■ **Value Name**- Enter the name of the key. This container object stores the value and it displays in the file structure of the device. ■ **Valut Type**- Select the type that corresponds to the registry value. ■ **Value Data** - Enter the value of the key. You can use comparison operators to target registry values by date and time. However, you must use the ISO Date and Time standard. Access information about the standard, ISO 8601, on the W3C (W3 Consortium) website at Date and Time Formats. Here is an example of targeting registgries exsisting before 10 am Eastern Standard Time on December 31, 2020. 

b Set the disk space devices must have available for the system to install the application for **Disk Space Required**.

c Set the battery power devices must have available for the system to install the application for **Device Power Required**.

d Set the random access memory devices must have available for the system to install the application for **RAM Required**.

8 Complete the settings in **Deployment Options > How To Install**.

Define the installation behavior on devices. While configuring the Win32 applications in the Workspace ONE UEM console, you have different combinations that you can select while setting the **Install Context** and **Admin Privileges** under the **Deployment** tab. Your installation process might vary based on the settings.

Setting	Description
Install Context	Select how the system applies the installation. ■ Device- Define the installation by the device and all the users of that device. ■ User- Define the installation by particular user accounts (enrolled).

Install Command	Enter a command to control the installation of the application.
	■ **MSI**- The system automatically populates the installation commands, and the commands include patches and transforms.
	  ■ Patches- To update the order in which the patches install on devices, update their listed order in the install command.
	  ■ Transforms- The order in which the system applies transforms is set when you assign the application. You see a placeholder name for the transform until you associate the transform during the assignment process.
	■ **EXE and ZIP**- Populate the install command and specify the patch names and their order of application in the command. You must also enter the install command that triggers the installation of the Win32 application.
	If you do not package the patches and transforms in the EXE or ZIP file and you add them separately, ensure to add the patch filenames and the transform lookup text boxes in the install command.

Admin Privileges	Set the installation to bypass admin privilege requirements.
Device Restart	The values for **Device Restart** options for Win32 applications allow the user to defer the device reboot for "x" days with a maximum limit of 7 days. The values allow admins to have a much better control over the application management lifecycle. Admins can choose to force restart applications or create application restarts that require the user engagement and provides a friendly user experience while installing the admin mandated applications. By default, **Device Restart** is set to **User engaged Restart** that allows you to configure the **Restart Deadline** for each application. To set the **Restart Deadline**, enter the deadline date. You can also choose **Do Not Restart** or **Force Restart** options if you choose not to the restart or force the device restart. **Note** Restart deadline has a maximum limit of 7 days.
Retry Count	Enter the number of times the system attempts to install the application after an unsuccessful attempt.
Retry Interval	Enter the time, in minutes, the system waits when it tries to install the application after an unsuccessful attempt.
Install Timeout	Enter the maximum time, in minutes, the system allows the installation process to run without success.

Setting	Description
Installer Reboot Exit Code	Enter the code the installer outputs to identify a reboot action. Review the entry for **Device Restart**. If you selected to **Do not restart** but you enter a reboot exit code, the system considers the installation a success after the reboot completes even though the Device Restart settings do not require a restart for success.
Installer Success Exit Code	Enter the code the installer outputs to identify a successful installation.

9 Complete the settings in **Deployment Options > When To Call Install Complete**.

Configure Workspace ONE UEM to identify the successful installation of Win32 applications. The system requires this information for EXE and ZIP files.

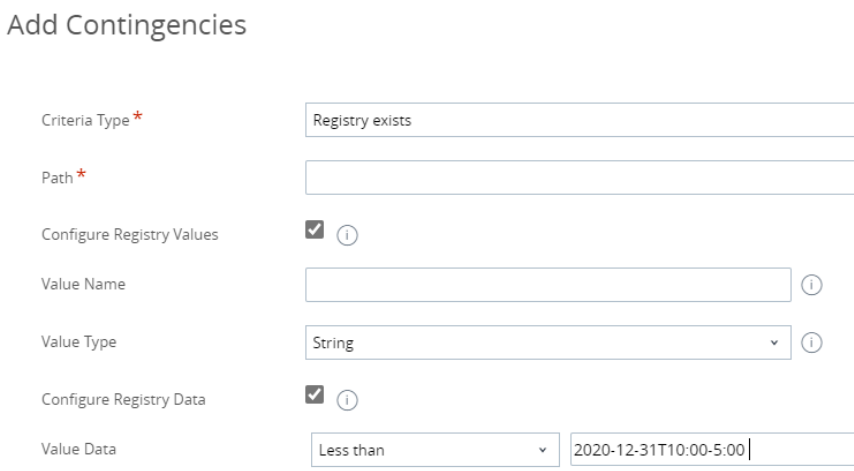
a Configure the system to use specific criteria to recognize the completion of the installation process for **Use Additional Criteria**.

b To identify the installation completion or use custom scripts, add a specific criteria for **Identify Application By**.

Setting - Defining Criteria - App	Description - Defining Criteria - App
Criteria Type **App exists** **App does not exist**	■ Instruction – Configure the system to install the application when a specific application is or is not on devices. ■ Completion – Configure the system to identify the installation is complete when a specific application is or is not on devices. Workspace ONE UEM checks for the existence of the application but it does not deploy the application to devices.
Application Identifier	Enter the application identifier so the system can recognize the existence or non-existence of the auxiliary application. This value is also known as the product code of the application.
Version	Enter the specific version. Use a comparison operator to specifically target an app version or to target a range of app versions.

Setting - Defining Criteria - File	Description - Defining Criteria - File
Criteria Type **File exists** **File does not exist**	■ Instruction – Configure the system to install the application when a specific file is or is not on devices. ■ Completion – Configure the system to identify the installation is complete when a specific file is or is not on devices.
Path	Enter the path on the device where you want the system to look for the file and include the filename.
Version	Enter the specific version. Use a comparison operator to target a specific file version or a range of file versions.
Modified On	Enter the date the file was last modified.

Setting - Defining Criteria - Registry	Description - Defining Criteria - Registry
Criteria Type **Registry exists** **Registry does not exist**	■ Instruction – Configure the system to install the application when a specific registry value is or is not on devices.

Setting - Defining Criteria - Registry	Description - Defining Criteria - Registry
	■ Completion – Configure the system to identify the installation is complete when a specific registry value is or is not on devices.
Path	Enter the path on the device where the system can find the keys and values. Include the entire path, beginning with HKLM\ or HKCU\.
Configure Registry Values	■ **Value Name**- Enter the name of the key. This container object stores the value and it displays in the file structure of the device. ■ **Value Type**-Select the type that corresponds to the registry value. ■ **Value Data** - Enter the value of the key. You can use comparison operators to target registry values by date and time. However, you must use the ISO Date and Time standard. Access information about the standard, ISO 8601, on the W3C (W3 Consortium) website at Date and Time Formats. Here is an example of targeting registgries exsisting before 10 am Eastern Standard Time on December 31, 2020. 

Setting - Using Custom Script	Description - Using Custom Script
Script Type	Select the type of script.
Command to Run the Script	Enter the value that triggers the script.Custom Script Type. For example, you can run a script such as **`powershell -executionpolicy bypass -file script.ps1`**
Custom Script File	Select **Upload** and navigate to the custom script file on the network.
Success Exit	Enter the code that the script outputs to identify the successful installation.

Note If you are required to edit the **When to Call Installation Complete** criteria after an app has been deployed, select the **Edit** button in this section. Enter the Admin PIN to complete the action. If you change the criteria to a value that is invalid, the app will be removed from all the systems where it is deployed. Be cautious while editing the **Call Installation Complete** criteria.

10 Select **Save & Assign** to configure flexible deployment options.

What to do next

- You can now monitor your Win32 applications deployed through software distribution with the statistics on the **Details View** and by reviewing installation status codes. View the reasons in the Details View to track the progression of an installation. The reason codes help identify the status of an installation and if there is an issue with an installation, so that you can easily track and troubleshoot application deployments.
- You can remove Win3 applications from your devices. Workspace ONE UEM includes several methods to remove Win32 applications from your devices. Choose from deleting, the application, devices, organization group, assignment group, or user. Several admin functions impact multiple assets. It is important for you to understand the impact before making any changes.

Table 1-3. Win32 Application Deletion Methods

Deletion Method	Description
Details View	Select the **Delete Application** function in the details view of the application. This action removes the Win32 application off devices in smart groups assigned to the application.
Device	Delete the applicable device from the console.
Organization Group	Delete the organization group. This action impacts all assets and devices in the organization group.
Assignment Group	Delete the smart or user group assigned to the Win32 application. This action impacts every device in the group.
User	Delete the applicable user account from the console.

Deploy your Office 365 ProPlus with Workspace ONE UEM using Software Distribution

You can deploy Office 365 ProPlus with Workspace ONE UEM using the Office Configuration Service Provider (CSP), perform a full office install using software distribution or use a light office install using software distribution. The following use case explains deploying Office 365 ProPlus as a full office install using software distribution. The use case covers prerequisites such as downloading the files ahead of time per your XML, creating a Office.zip file, and then uploading the file to the Workspace ONE UEM console.

Complete the following steps to deploy Office 365 ProPlus in Workspace ONE UEM console using software distribution.

Prerequisites

Complete the following steps to create a Office.zip file.

1 Create a working folder. For example, create `c:\Office`.

2 Go to config.office.com to create your XML file.

   a Here is an example. Copy and paste the following sample configuration into a text file:

```
<Configuration ID="9cc840fd-62b9-49b0-9295-d6310d26df31">
<Add OfficeClientEdition="64" Channel="Monthly" ForceUpgrade="TRUE">
<Product ID="O365ProPlusRetail">
<Language ID="en-us" />
<ExcludeApp ID="Groove" />
</Product>
</Add>
<Property Name="SharedComputerLicensing" Value="0" />
<Property Name="PinIconsToTaskbar" Value="TRUE" />
<Property Name="SCLCacheOverride" Value="0" />
<Updates Enabled="TRUE" />
<RemoveMSI />
</Configuration>
```

   b Save the XML file as `configuration.xml` in a folder.

3 Download the Office Deployment Tool and run officedeploymenttool.exe.

4 Extract files to your working directory. For example, `c:\Office`. The tool downloads several sample XMLs you can edit or you can use the one that you have already created from the web tool.

5 You can choose to automate the next step by creating a simple command file and name it as `Download.cmd`. Edit the command file and add the following text:

```
@echo off
pushd %~dp0
echo Downloading Office 365 Pro Plus Retail x64 source files
setup.exe /download configuration.xml
```

6 Double-click `Download.cmd` to begin the download. A new `Office` is created and the command window closes after the download is complete.

7 (Optional) If you want to have an uninstall command in the Workspace ONE UEM console, create a `uninstall.xml` file, and paste the following text in the configuration XML file.

```
<Configuration>
<Remove>
<Product ID="O365ProPlusRetail">
<Language ID="en-us"/>
```

```
</Product>
</Remove>
<Display Level="None" AcceptEULA="TRUE"/>
</Configuration>
```

8 Once the download is complete, select all your files and add them to a compressed zip folder.

Procedure

1 In the upper-right corner of the Workspace ONE UEM console, select **Add**.

2 Select **Internal Application**.

3 Select **Upload**.

4 Browse for the Office.zip file and click **Save**.

5 Select **No** for **Is this a dependency app?**.

6 Select **Continue**.

7 Configure the application details.

   a Navigate to the **Details** tab.

   b Enter a name for your application as **`Office 365 Pro Plus`**.

   c Enter the uploaded by **`[name]`** on **`[date]`** in the **Change Log**.

   d Select **64-bit** for the **Supported Processor Architecture** if you are running on a 64-bit operating system.

8 Configure the application files.

   a Navigate to the **Files** tab.

   b Scroll down to find the **App Uninstall Process** section.

   c Select **Input** for the **Custom Script Type**.

   d Enter **`setup.exe /CONFIGURE uninstall.xml`** in the **Uninstall Command** text box.

9 Complete the application deployment process.

   a Navigate to **Deployment Options** .

   b Scroll down until you see the option for **Install Command**.

   c In the **Install Command** text box, enter the following command : **`setup.exe /CONFIGURE configuration.xml`**.

10 Scroll down to find the **When To Call Install Complete** section.

   a Select **Defining Criteria** for **Identity Application By**.

   b Select **Add**.

   c In the **Criteria Type** drop-down , select **File Exists**.

d Enter the **Path**.

Here is an example. Enter the path of Outlook.exe : `C:\Program Files\Microsoft Office\root\Office16\OUTLOOK.EXE`.

e Select **Add** to save the detection criteria.

11 Select **Save and Assign**.

12 Select **Add Assignment** to add the assignment group.

a Select the **Select Assignment Groups** search box and select **All Devices**.

b Select **Auto** for the **App Delivery Method.**.

c Select **Show** for **Display in App Catalog**.

d Select **Enabled** for **Make App MDM Managed if User Installed.**.

e Select **Add**.

To control the deployment of applications, you can either add a single assignment or multiple assignments. If you add multiple assignments, prioritize the importance of the assignment by moving its place in the list up for most important or down for least important.

13 Click **Save and Publish**.

Working with Win32 App Dependency Files

Dependency files in the software distribution are applications that are necessary for a Win32 application to function. Examples include framework packages and libraries. Although you upload them like a file and you can view them in the **List View**, they have reduced features.

Dependency File Features

- Dependency file does not have assignments of their own. The applications to which they are associated give the dependency files their assignments.
- Every dependency file is a separate file and the system does not create versions for the file.
- The system cannot parse information from dependency files so you must enter details such as uninstallation processes.
- Dependency files have reduced options on the Deployment Options tab.
- You cannot associate patches or transforms to dependency files.
- You can view the dependency files by setting the filters as **Windows Desktop** apps and **Dependencies**.
- If you modify the application manifest to add or remove the application dependencies, the software distribution agent installs or uninstalls the dependencies. Currently, the application must first be uninstalled and then reinstalled for the changes to take effect. If the reinstalled application is User Installed, it becomes Managed, and it is removed upon unenrollment.

Delete Considerations

Before you delete a dependency, ensure that other applications are not associate to it. When you delete the dependency file, the system removes its association from all applications. Devices newly assigned to the application do not get the dependency. Deletion does not remove the dependency from devices that had the application previous to deletion.

User-Installed Win32 Applications

When you enable **Make App MDM Managed if User Installed** for Win32 applications, the system processes the command to assume management of the Win32 application. Assuming management of Win32 applications includes certain caveats to work.

Considerations for User-Installed Win32 Applications

When you enable **Make App MDM Managed if User Installed** for Win32 applications, the system processes the command to assume management of the Win32 application. Assuming management of Win32 applications includes certain caveats to work.

This feature works for devices that meet these caveats.

- Devices that enrolled or were assigned after you enabled this option and did not have the application installed.
- Devices that enrolled or were assigned after you enabled this option and did have the application installed with a status of user-installed.

This feature does not support the management assumption process on devices that meet these caveats.

- Devices that enrolled or were assigned before you enabled this option and have the application installed with a status of user-installed.
- Devices that are employee owned. If users have BYODs, you cannot assume management of Win32 applications on these devices.

User-Installed Win32 Application Workflow

When you activate **Make App MDM Managed if User Installed** for Win32 applications, the system processes the command to assume management of the Win32 application.

If you activate **Make App MDM Managed if User Installed**, the management assumption process begins with an install command.

Note If you deactivate the option and the user installs the application, the system marks the application as user-installed.

1 Workspace ONE UEM sends install commands to all the enrolled devices after publication.

2 The device responds and confirms the acceptance of the app installation command.

3 The system checks if the **Make App MDM Managed if User Installed** for the Win32 application is enabled and the administrator is assuming management.

4 The system looks for the application on the device. If the application is already installed, the system downloads the installer to the app management cache.This step is required for management tasks such as app updates and app removal.

5 If the application is not installed, the system installs the application as per the configured deployment options.

6 The device reports the status of the application as managed to the console.

Technical Preview: Manage Software Distribution applications used in Workflows

You can now create a workflow for any of your Windows and macOS software distribution applications.

Note Workspace ONE UEM offers Freestyle Orchestrator Workflows as a tech preview feature for our SaaS customers. For more information, see What is Freestyle Orchestrator.

Technical preview features are not fully tested and some functionality may not work as expected. However, these previews help Workspace ONE UEM improve current functionality and develop future enhancements. The content in this section applies only to customers who are participating in the technical preview. If you are participating in the tech preview feature and you intend to use an application or a specific version of an app in workflows, consider the following properties:

Newly added applications and the default policy

The app assignment section of your software distribution application, displays a **Workflow** assignment tab. If you are uploading an application for the first time, the workflow tab is empty as there are no workflows associated with the application. You must create at least one assignment rule for the application to be picked as **default policy** to use the application in the workflows. The properties specified in the default policy are used by all the devices that receive this application version through any workflow. If you want to override the policy for a group of users, you can create additional assignments and prioritize them over the default policy.

Note Default policy ignores the deployment properties such as **deployment mode**, **deployment begins on**, as Workflow drives some of these properties.

Considerations while adding new version of the application

If you are using **Latest version** option for a particular app in your workflow and you add a version of this app to your inventory, direct app assignments, default policies, and workflow assignments is copied over from the previous version. Canceling the assignment without publishing the changes to the end-user devices will not recognize the newly added version as the **latest**. If and when the assignments for the newly added version are published, a new version of the workflows that contains this application is published to the devices.

Considerations while deleting or retiring the application

Retiring or deleting a version of your application that is used in workflows is not allowed if you are using a specific version in your workflows. These actions are allowed only if you use the option **latest version**.

Note When you retire the latest version (V2) of your app, and make its lower version (V1) as the latest, all the devices that receive the highest version (V2) is not impacted. However, the devices that receive the workflow after the action, will receive the older version (V1) as its latest.