Software Distribution Management

VMware Workspace ONE UEM 1907

vmware®

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

https://docs.vmware.com/

If you have comments about this documentation, submit your feedback to

docfeedback@vmware.com

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

Contents

Software Distribution Management

1

Workspace ONE UEM powered by AirWatch offers software distribution that helps you deploy macOS and Win32 applications from the **Apps & Books** 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

Before Workspace ONE UEM console v9.3, most of the macOS applications or software were deployed through Product Provisioning. From v9.3, Workspace ONE UEM also offers a flexible deployment through an integration with Munki, an widely renowned open source tool.

Now all macOS application file types (.dmg, .pkg, .mpkg) can be managed in the Internal Applications section in the UEM console (**Apps & Books > Applications > 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.

For more information on the software distribution configuration and assignment and deployment of applications to macOS devices through the software distribution process, refer the **Software Distribution Management** guide.

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 **Books & Apps > Internal Applications**. 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://getwsone.com/AdminAssistant/VMwareWorkspace ONE UEMAdminAssistant.dmg. 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.

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.

Upload Applications to Deploy to macOS Devices

Deploy internal applications to your mobile network by upload internal applications with local files in the UEM console.

Procedure

1 Navigate to **Apps & Books > Applications > 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. For more information about how to use the VMware AirWatch Admin Assistant Tool, see Generate Metadata Using VMware AirWatch Admin Assistant Tool.

4 Complete the **Images** tab.

Setting	Description
Mobile Images	Upload or drag images of the application to display in the AirWatch Catalog for mobile devices.
Tablet Images	Upload or drag and drop images of the application to display in the AirWatch Catalog for tablets.
Icon	Upload or drag the images of the application 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. For more information on the exit behavior of each script type, see Software Distribution Scripts.

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 unistall 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 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	Enable **Blocking Applications** to define any applications or processes that might block the clean installation of a managed macOS applications. . 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. Disable **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 /Applications/ folder, it can be defined as just the app name and the path will be 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 have the ability to easily quit the app. Thus you should not block any faceless background apps or helper apps.
Restart Action	Select the restart action for the application. The available actions are: ■ 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 have an option toenable or disable the Desired State Management settings based on the business needs. Desired State Management is enabled by default to enforce application management during macOS software installation. If enabled, and if the end-user deletes the app, the application is automatically reinstalled on the next Hub sync. If disabled, and if the end-user deletes the app, the application is not automatically reinstalled, unless pushed from the 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**.

Uninstallation Methods

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 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 Scripts

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
- Customise 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

Software Distribution Conditions

Conditions are a set of attributes provided by the integrated open source Munki library for determining 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 Format

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	This 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 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.

Procedure

1 Navigate to **Apps & Books > Applications > 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 Hub automatically 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 Software Distribution Updates

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

Procedure

1 Navigate to **Apps & Books > 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**.

Methods used by Munki to Install Applications

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 items, understand the methods used by Munki to check the list of software items.

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

Methods

In the order of precedence, listed below are the methods used by Munki in determining whether the given item should be installed (or removed):

- Install Check Script
- Install Items
- 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 will be ignored for purposes of determining 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 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
```

Uninstall Check Script

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 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 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 makepkginfo,

```
/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 just 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 would 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 version information.

For other filesystem items, Munki can only determine existence (in the case of a non-bundle directory), or can calculate a checksum (for files). For files with checksums, the test fails (and therefore the item will be 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>
```

Receipts

When an Apple-style package is installed, generates 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 installation status, it checks 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 will actually be installed, Munki will continually mark 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 will then 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 an **installs** array that lists items that are installed by the package. Munki can use installs array information instead of the receipts to determine installation status.

Troubleshooting macOS Software Distribution

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 the /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 also 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 **Apps & Books > Applications > 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.

Software Distribution of Win32 Applications

With software distribution, Workspace ONE UEM can deploy Win32 applications from the **Apps & Books** 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.

Note For more information on software distribution and how to troubleshoot the system, see the following Knowledge Base article, https://support.workspaceone.com/articles/115001674888.

Requirements to Deploy Win32 Applications for Software Distribution

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.

CDNs and File Storage Systems

It is considered to be a best practice to use content delivery network (CDN) to deploy applications. This option has the advantage of sending content to devices in the network and to remote devices. It also offers increased download speeds and reduces bandwidth on Workspace ONE UEM servers. However, in some scenarios, a CDN is not a viable option. For these instances, use a file storage system.

Enable Software Package Deployment - SaaS Environments

Configure Workspace ONE UEM to recognize the deployment of Win32 applications through the software distribution method.

For the **Software Package Deployment** option to display, Workspace ONE UEM enables the CDN for the environment. Go to **Groups & Settings > All Settings > Device & Users > Windows > Windows Desktop > App Deployments** and enable **Software Package Deployment**.

> **Note** If your deployment whitelists Workspace ONE UEM IP addresses, the CDN does not work.

Enable Software Package Deployment - On-premises Environments

Software distribution is now turned on by default in the Workspace ONE UEM console for all on-premises customers. By default, customers get up to 5 GB of storage for applications in the database. For storing large Win32 applications, you can use a file storage system.

It is considered to be a best practice to use content delivery network (CDN) to deploy applications. This option has the advantage of reducing the bandwidth on other servers.

File Storage

Certain Workspace ONE UEM functionality uses a dedicated file storage service to handle processing and downloads, which reduces the overall burden on your Workspace ONE UEM database and increases its performance. Configuring file storage manually is only applicable to on-premises customers. It is automatically configured for SaaS customers.

It also includes certain Workspace ONE UEM 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. Setting up file storage causes all reports, all internal applications, and all managed content to be stored there.

Workspace ONE UEM Reports

As of console version 9.0.2, three new reports were added that appear the same as existing reports but use a revamped back-end framework. This new framework generates reports with greater reliability and faster download times. To take advantage of these benefits, you must set up file storage.

For more information about the new reports, see Workspace ONE UEM Reports Overview.

Internal Applications

When file storage is enabled, all internal application packages that you upload through the UEM console are stored in a file storage location.

File storage is required to deploy Win32 applications (IPA, PAK, APPX, MSI, EXE, and so on) and macOS applications (.dmg, .pkg, .mpkg, and so on) from the Apps & Books area of the UEM console. This feature is called software distribution.

For more information about software distribution for Win32, see Software Distribution of Win32 Applications.

For more information about software distribution for macOS, see Software Distribution for macOS applications.

Workspace ONE UEM Managed Content

You can separate the managed content from the Workspace ONE UEM database by storing it in a dedicated file storage location. Uploading large amounts of managed content might cause issues with database performance. In this case, on-premises customers can free up space in the database by moving the managed content to an integrated local file storage solution.

Personal content also moves to the file storage solution is enabled. By default, personal content is stored in the SQL database. If you have a Remote File Storage enabled, personal content is stored in the RFS and not in the file storage or SQL database.

For more information about the managed content, see AirWatch Managed Content Repository .

File Storage Requirements

Separate the managed content from the Workspace ONE UEM database by storing it in a dedicated File Storage. To set up a file storage, you must determine the location and storage capacity for your file storage, configure the network requirements, and create an impersonation account.

Important File Storage is required for Windows 10 Software Distribution.

Create the Shared Folder on a Server in Your Internal Network

- File storage can reside on a separate server or the same server as one of the other Workspace ONE UEM application servers in your internal network. It is only accessible to components that require access to it, such as the Console and Device Services servers.
- If the Device Services server, Console server, and the server hosting the shared folder are not in the same domain, then supply the domain during service account configuration in the format <domain \username>. Domain Trust can also be established to avoid authentication failure.

Configure the Network Requirements

- **If using Samba/SMB** – TCP: 445, 137, 139. UDP: 137, 138
- **If using NFS** – TCP and UDP: 111 and 2049

Allocate Sufficient Hard Disk Capacity

Your specific storage requirements may vary depending on how you plan to use file storage. The file storage location should have enough space to accommodate the internal apps, managed content, or reports you intend to use. Take into the account the following considerations.

- If you enable caching for internal apps or content, then a best practice is to size the Device Services server for 120 percent of the cumulative size of all the apps/content you need to publish.
- For storing reports, your storage requirements depend on the number of devices, the daily amount of reports, and the frequency with which you purge them. As a starting point, you should plan to allocate at least 50 GB for deployment sizes up to 250,000 devices running about 200 daily reports. Adjust these numbers based on the actual amount you observe in your deployment. Apply this sizing to your Console server as well if you enable caching.

Create a Service Account with Correct Permissions

- Create an account in the domain of the shared storage directory.
- Give the local user read/write/modify permissions to the file share that is being used for the File Storage Path.
- Configure File Storage Impersonation User in Workspace ONE UEM with the domain account in the format <domain\username>.
- If the shared storage directory is not on a domain, create an identical local user and password on the server being used for File Storage, Console, and Device Services server. In this case, supply the local user account in the format <username>.

You can also use a domain service account instead of a local user account.

Configure File Storage at the Global Organization Group

Configure file storage settings at the Global organization group level in the UEM Console.

Enable File Storage for Applications

Configure file storage for internal applications using the procedure below. This is required if you are deploying Win32 apps using software distribution, but will apply to all internal apps once configured.

Procedure

1 At the Global organization group level, navigate to **Groups & Settings > All Settings > Installation > File Path** and scroll to the bottom of the page.

2 Select the **File Storage Enabled** slider and configure the settings.

When file storage is enabled, you can configure an external repository in which files are stored. A disabled setting means that files are stored as binary large objects in the database.

Setting	Description
File Storage Path	Enter the path files are to be stored in the following format: \\{Server Name}\{Folder Name}, where Folder Name is the name of the shared folder you create on the server.
File Storage Caching Enabled	When enabled, a local copy of files requested for download is stored on the Device Services server as a cache copy. Subsequent downloads of the same file retrieve it from the Device Services server as opposed to file storage. When enabled, files are cached locally on the DS server when accessed for the first time. Subsequent requests are served using the file cached on the DS server instead of streaming from the file storage location. If you enable caching, consider accommodating for the amount of space needed on the server. If you integrate with a CDN, then apps and files are distributed through the CDN provider, and a local copy is not stored on the Device Services server. For more information, refer to the **VMware Workspace ONE UEM CDN Integration Guide** (https://resources.air-watch.com/view/8cr52j4hm6xfvt4v2wgg/en).
File Storage Impersonation Enabled	Select to add a service account with the correct permissions.
File Storage Impersonation Username	Provide a valid service account user name to obtain both read and write permissions to the shared storage directory.
Password	Provide a valid service account password to obtain both read and write permissions to the shared storage directory.

3 Select the **Test Connection** button to test the configuration.

Application Lifecycle for Software Distribution

Workspace ONE UEM can help manage Win32 applications with its lifecycle features, so that you can know their installation statuses, keep them current, and delete them.

To manage the deployment of your Win32 applications, use the life cycle of internal application.

- Upload Win32 Files for Software Distribution - Add the Win32 application and define if it is a dependency file.

- Configure Win32 Files for Software Distribution - Enter details for the Win32 application, add supporting files, and enter deployment criteria. You use the flexible deployment feature to assign to devices.
- Monitor the deployment of your Win32 Applications - Track the installation progress of Win32 applications.
- Internal App Versions.
- Methods to Delete Win32 Files - Delete applications with several options.

Upload Win32 Files for Software Distribution

Upload Win32 applications as either main files or dependency files. Use the same process for EXE, MSI, and ZIP files.

Prerequisites

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.

Procedure

1. Navigate to **Apps & Books > Applications > Native > Internal** and select **Add Application**.
2. Select **Upload**, and then select **Local File** and select the application to upload.
3. 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.

Configure Win32 Files for Software Distribution

Configure details about the Win32 application, which include to define when to install it, how to install it, and when to identify the installation is complete.

Procedure

1. Configure the **Details** tab options.

   The Workspace ONE UEM system 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.

   - Application name
   - Application version
   - Application identifier (also called a product code)

2 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.
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.

3 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 instrction 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.

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.
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 is or is not on devices.

Setting - Registry	Description - Registry
	■ Completion – Configure the system to identify the installation is complete when a specific registry 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 of key displayed in the file structure of the device. ■ **Value Data** - Enter the value of key. The name-data pairs stored in the key display in the file structure of the device.

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**.

4 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 could choose while setting the **Install Context** and **Admin Privileges** under the **Deployment** tab. Your installation process may vary based on the settings. To understand more about Win 32 application installation behavior see Win32 Application Installation Behavior, Software Distribution or Product Provisioning.

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.

Setting	Description
Admin Privileges	Set the installation to bypass admin privilege requirements.
Device Restart	Require the device to restart after the application installs successfully, require the device to restart only if necessary for the application to function, or do not require the device to restart.
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.
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.

5 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.

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.
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 is or is not on devices. ■ Completion – Configure the system to identify the installation is complete when a specific registry 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 of key displayed in the file structure of the device.

Setting - Defining Criteria - Registry	Description - Defining Criteria - Registry
	■ **Value Data** - Enter the value of key. The name-data pairs stored in the key display in the file structure of the device.

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
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.

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

What to do next

Assign flexible deployment schedules to the Win32 application. See Add Assignments and Exclusions to ApplicationsVMware Workspace ONE UEM 1904.

Win32 Application Installation Behavior, Software Distribution or Product Provisioning

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 **Apps & Books**. However, if you have tried deploying the application with **Apps & Books** 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 Apps & Books

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

Configuring Win32 Application from Apps & Books	Install Context Settings In the Workspace ONE UEM console	User is an Admin	User is a standard user
Navigate to **Apps & Books > Applications > 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 **Apps & Books > Applications > 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 **Apps & Books > Applications > 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 **Apps & Books > Applications > 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 **Apps & Books**. However, if you have tried deploying the application with **Apps & Books** 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, and Install Timeout Options

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.

Default Values and Time to Installation Failure Reported

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.

Dependency Files in Software Distribution

Dependency files in 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.

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.

Supported Scenarios to Assume Management of Win32 Applications

Assuming management of Win32 applications includes certain caveats to work.

Supported and Unsupported Scenarios

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.

Assuming Management of Win32 Applications for Software Distribution

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

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

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

Procedure

1 Workspace ONE UEM sends install commands to devices that enroll after publication.

2 The device responds that it received the command.

3 The next check depends if the admin is assuming management.

4 The system looks for the application on the device.

   If the application is already installed, the system re-downloads and reinstalls the application. If the application is not installed yet, it insalls following the regular configured flexible deployment configurations.

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

Monitor the deployment of your Win32 Applications

Monitor your Win32 applications deployed through software distribution with the statistics on the Details View and by reviewing installation status codes.

Use the Details View of internal applications to view the progress and status of installations. See Track Internal Applications With Details View .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.

Methods to Delete Win32 Files

Workspace ONE UEM includes several methods to remove Win32 applications off devices. Choose from deleting, the application, devices, organiztion group, assignment group, or user. Several admin functions impact multiple assets, so understand the changes before you take action.

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.

Patches in Software Distribution

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.