

SwiftPack Studio

Package Automation Studio

USER GUIDE AND TUTORIAL

Auto Package

Learn SwiftPack Studio by packaging one real application from start to finish

[Settings](#) → [Add App](#) → [Download](#) → [Automate](#) → [Publish](#)

Product	SwiftPack Studio 1.0.0
Document version	1.0

Contents

Note

This contents list and all page numbers are automatic. To refresh them in Word, select all (Ctrl+A) and press F9, or right-click the list and choose "Update Field". The document is set to update fields automatically the first time you open it.

Contents.....	2
1 The main window.....	3
1.1 How the toolbar is ordered.....	3
1.2 The nine features.....	4
1.3 Buttons on the right.....	4
2 Set up before you start.....	5
2.1 Optional extras.....	5
3 Auto Package: Settings.....	5
3.1 Toolbar.....	6
3.2 What the Settings folders are for.....	9
4 Auto Package: Updates.....	9
5 Auto Package: Add App.....	10
5.1 Step by step.....	10
5.2 What you should see.....	13
5.3 Download the installer.....	14
5.4 Download Information panel.....	14
5.5 The other ways to find an installer.....	18
5.6 How to configure auto schedule.....	18
5.7 How to register SwiftPack Studio in Entra.....	19
5.8 How to configure sending mail to the app owner.....	20
5.9 How to deploy an application via Intune and send mail to the app owner.....	20
5.10 How to configure pre-install, post-install, pre-uninstall, post-uninstall and close apps if required.....	30

1 The main window

SwiftPack Studio opens as one window. Everything is reached from the toolbar along the top. Click a button on the toolbar and the area below changes to that page. There is nothing to open first and no project to create.

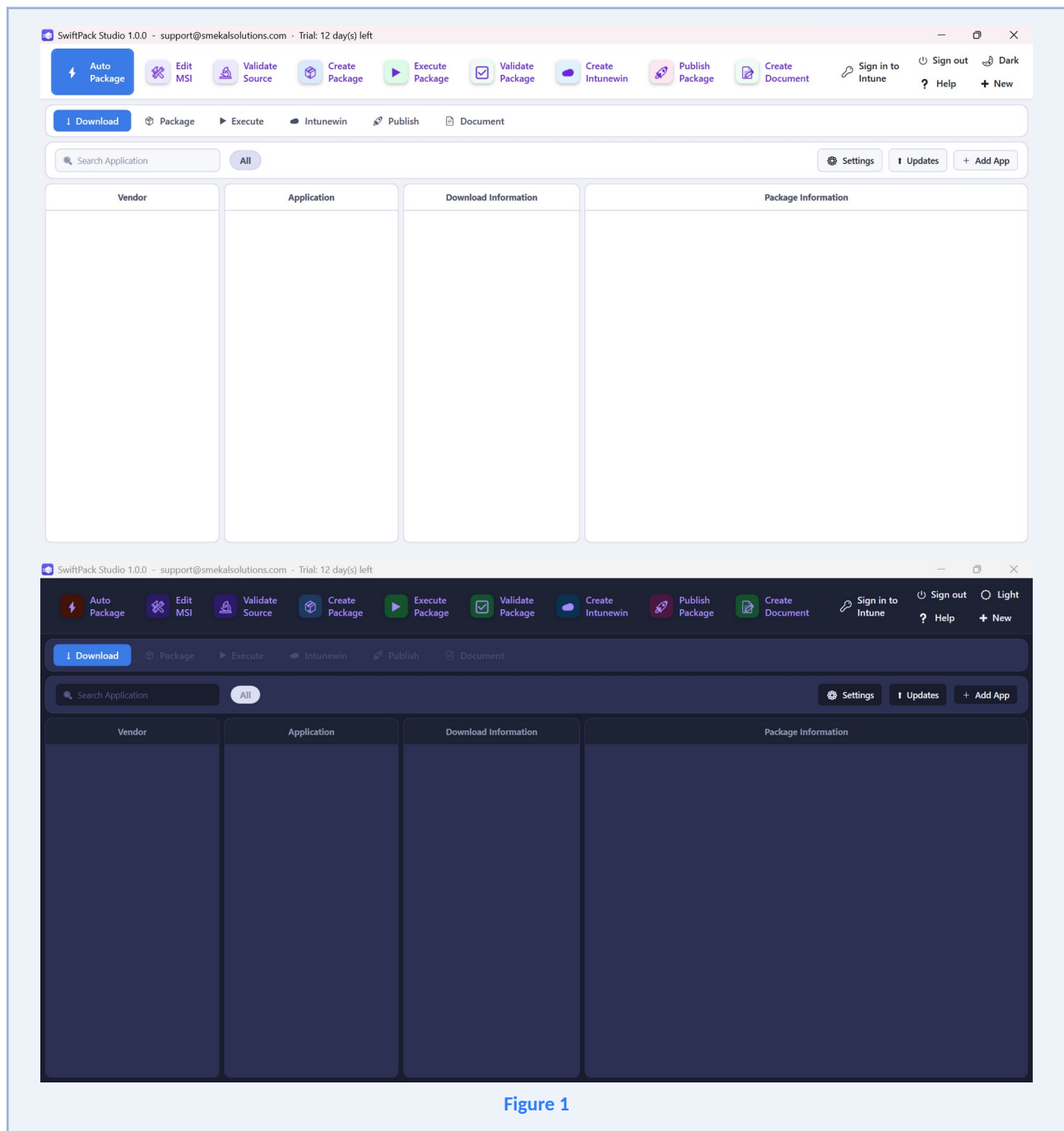


Figure 1

The title bar shows the version, the account you signed in with, and how many trial days are left.

1.1 How the toolbar is ordered

The toolbar is not in alphabetical order. From left to right it follows the order you build a package. This tool has two categories. Auto Package does the whole job for you. Manual Package is the same work done step by step, and contains Edit MSI, Validate Source, Create Package, Execute Package, Create Intunewin, Publish Package and Create Document. Auto Package comes first because it does all of that for you.

1.2 The nine features

1. Auto Package

Does the whole job for you. You give it a list of applications. It downloads each installer, builds the package, tests it, makes the .intunewin file, publishes to Intune and writes the documentation and send mail. Set an application up once and it can keep itself up to date on a schedule.

2. Edit MSI

Opens and changes MSI files. Shows an MSI in a friendly view (product details, properties, files, registry, shortcuts, services etc.) and in a raw table view. You can save your changes as an MST transform instead of editing the vendor's MSI, so the change survives the next vendor update.

3. Validate Source

Records what an installer changes, and can build an MSI. It watches an installer and records every file, registry key, shortcut, service, driver, firewall rule and scheduled task it creates, then saves this as an SVR (Source Validation record) file. You use that file to write detection rules, and Validate Package checks against it later. It can also turn the recording into an MSI, which is useful when an application only ships as an EXE.

4. Create Package

Wraps an installer in a PSADT script. Builds a PSAppDeployToolkit package around an installer. You give it the template folder and the application details, and it writes the deployment script and folders for you.

5. Execute Package

Installs, uninstalls or repairs your package. Runs the package on this machine so you can see it work before it reaches real users. You can run it as Administrator or as SYSTEM, which is how it runs when deployed.

6. Validate Package

Checks your package did the same thing as the installer. Compares what your package installed against the SVR recording from Validate Source, and gives you a pass or fail report you can save as HTML or Excel.

7. Create Intunewin

Makes the file Intune needs. Turns your package folder into a .intunewin file. This step needs IntuneWinAppUtil.exe, which Microsoft provides and you copy in yourself.

8. Publish Package

Uploads the application to Intune. Creates the Win32 app in your Intune tenant and uploads the .intunewin along with the detection rules and settings. You must sign in to Intune first.

9. Create Document

Writes up what you built. Produces a document describing the package: where it came from, the switches used, the detection rules and how it installs. Useful for handover.

1.3 Buttons on the right

These apply to the whole application, not to one page. The three window buttons are at the far right of the title bar.

Button	What it does
Sign in to Intune	Signs in to your Intune tenant. Only needed for Publish Package. In auto package feature, this is required to add dependency applications and to add assignments.
Sign out	Signs you out of SwiftPack Studio and goes back to the sign-in window.
Light / Dark	Changes the colour theme. Your choice is remembered.
Help	Opens the help and user guide.
New	Clears the page you are on so you can start again.
Minimise	Hides the window. SwiftPack Studio keeps running, so a scheduled download still happens while the window is hidden.
Maximise / Restore	Makes the window full the screen, or puts it back.
Close (X)	Closes SwiftPack Studio. It asks you to confirm first, because unsaved work is lost.

2 Set up before you start

Once SwiftPack Studio is installed, the extras below are only needed for particular jobs. Skip any you do not need, and come back later when you do. If you have already done a step, ignore it.

2.1 Optional extras

Each line below is optional. Do it only if you want that feature. If it is already done, ignore this.

If you want to...	Do this
Create PSADT packages (Optional)	Download the PSAppDeployToolkit template and copy the files and folders into C:\SwiftPack\Auto Package\PSADT Template.
Test packages in SYSTEM context (Optional)	Download PsExec.exe from Microsoft Sysinternals and copy it into C:\Program Files\Smekal\SwiftPack Studio\Templates.
Create .intunewin files (Optional)	Download IntuneWinAppUtil.exe (Microsoft Win32 Content Prep Tool) and copy it into C:\Program Files\Smekal\SwiftPack Studio\Templates.
Publish packages to Intune (Optional)	Open Settings and fill in Tenant ID and Client ID. Follow the instructions in the document "5. SwiftPack Studio - How to register an application in intune.pdf" to get the Tenant ID and Client ID. You can do it later too.
Send mail to the application owner (Optional)	Open Settings and fill in the four Mail fields. Follow the instructions in the document "6. SwiftPack Studio - How to configure mail notification in intune.pdf". You can do it later too.
Use GitHub sources often (Optional)	Open Settings and paste a GitHub token. Without one you share a small hourly limit with everyone else, and a long list of applications can run out part way through. You can do it later too.

3 Auto Package: Settings

Settings holds the values shared by every application: the folders, the mail, the GitHub token and the Intune details. Nothing here belongs to one application.

3.1 Toolbar

Control	What it does
Search Application	Filters the Vendor and Application lists by app name or vendor as you type.
Category chips (All, Browser, Communication, Media, PDF, Runtime, Security, Utility, In-House, Other)	Filter the catalogue to one category. Click the active chip again to return to All.
Settings	Opens the global settings editor (AutoPackage.json) - paths, Intune credentials, mail, GitHub token, MST properties.
Updates	Opens the Downloads window listing Updates Available, Auto Schedule Status and Missed Auto Schedules.
Add App	Opens the Add App dialog to register a new application in the catalogue.
Cancel	Appears only while a download (or a silent install / uninstall) is running, to stop it.

1. Go to **Auto Package**, then the **Download** tab. Click **Settings** at the top right.

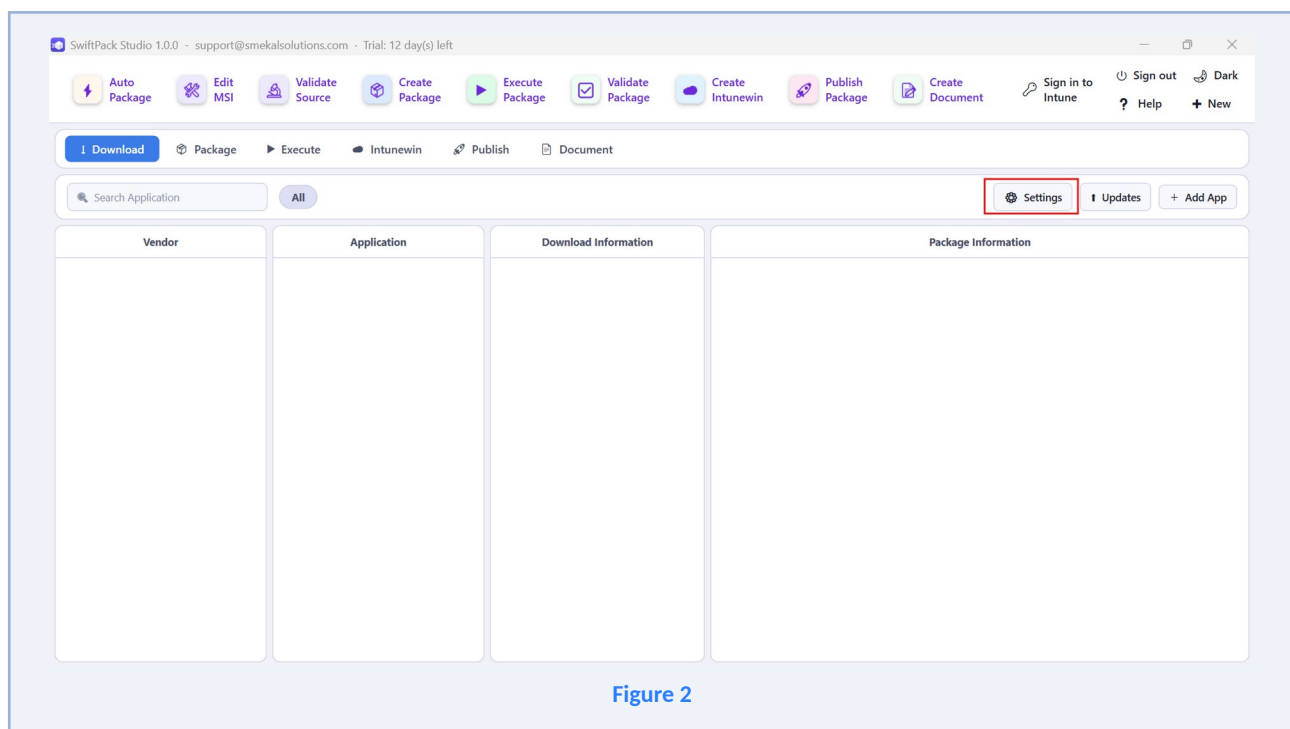
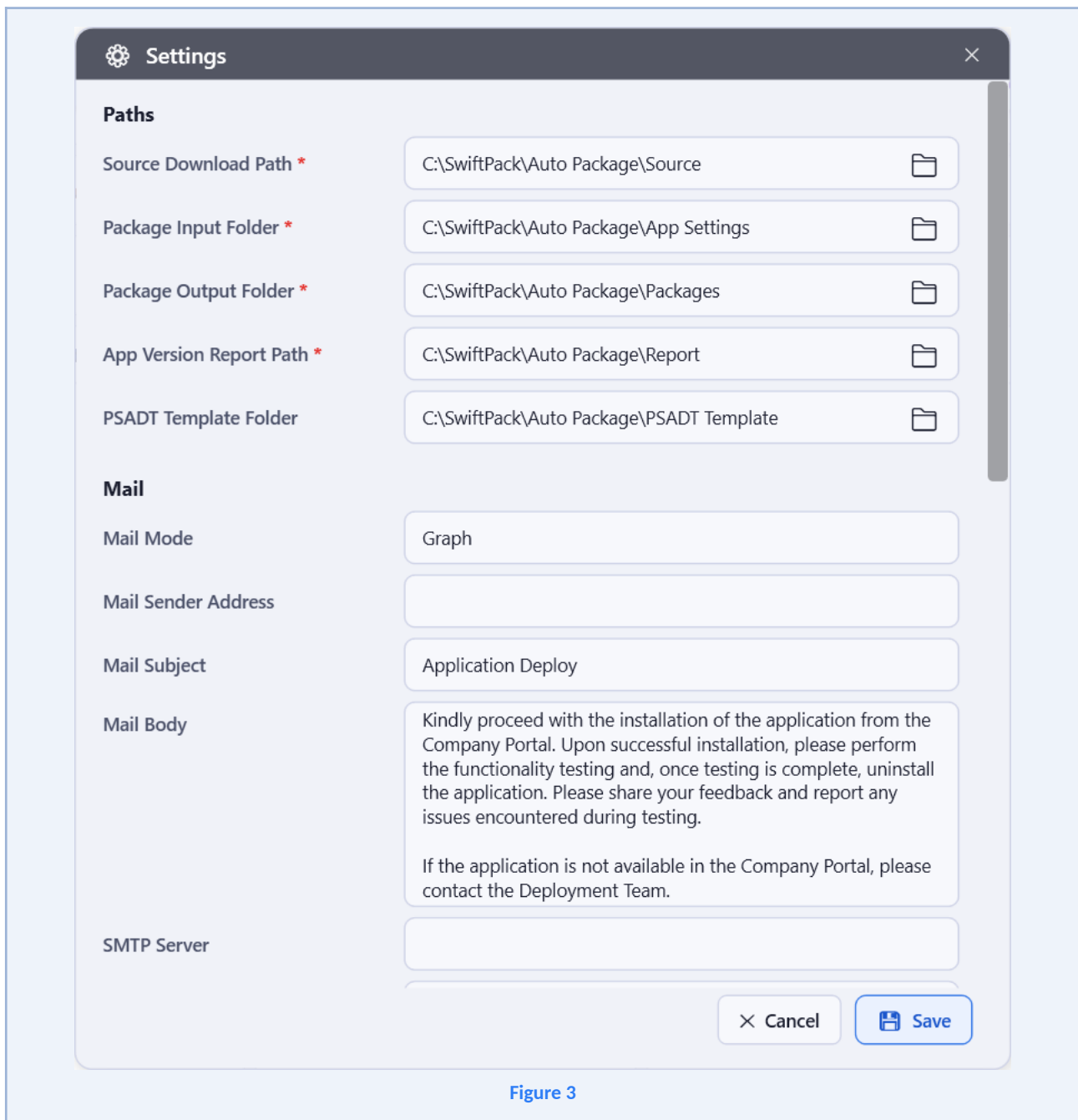


Figure 2

2. Change any folder you want. Type the path, or click the folder button at the right of the box to browse. Boxes with a red star are required and cannot be left empty.



3. Scroll down to **Flags** and set each one to Yes or No. These change what the packaging step produces. For example, if Copy Logs to Documents is Yes, PSADT logs will be copied to package folder\Documents; if Copy .txt and .docx to Documents is Yes, *.txt and *.docx (containing package details) will be copied to package folder\Documents. If Add MSI/EXE Command to PS1 is Yes, it automatically adds MSI and EXE install and uninstall commands to Invoke-AppDeployToolkit.ps1 for every application - check the commands once before deploying the package.
4. In **MST Properties** you can add and remove properties. These are applied whenever a transform (Optional) is created, for every application, so use it for values your organisation always sets.

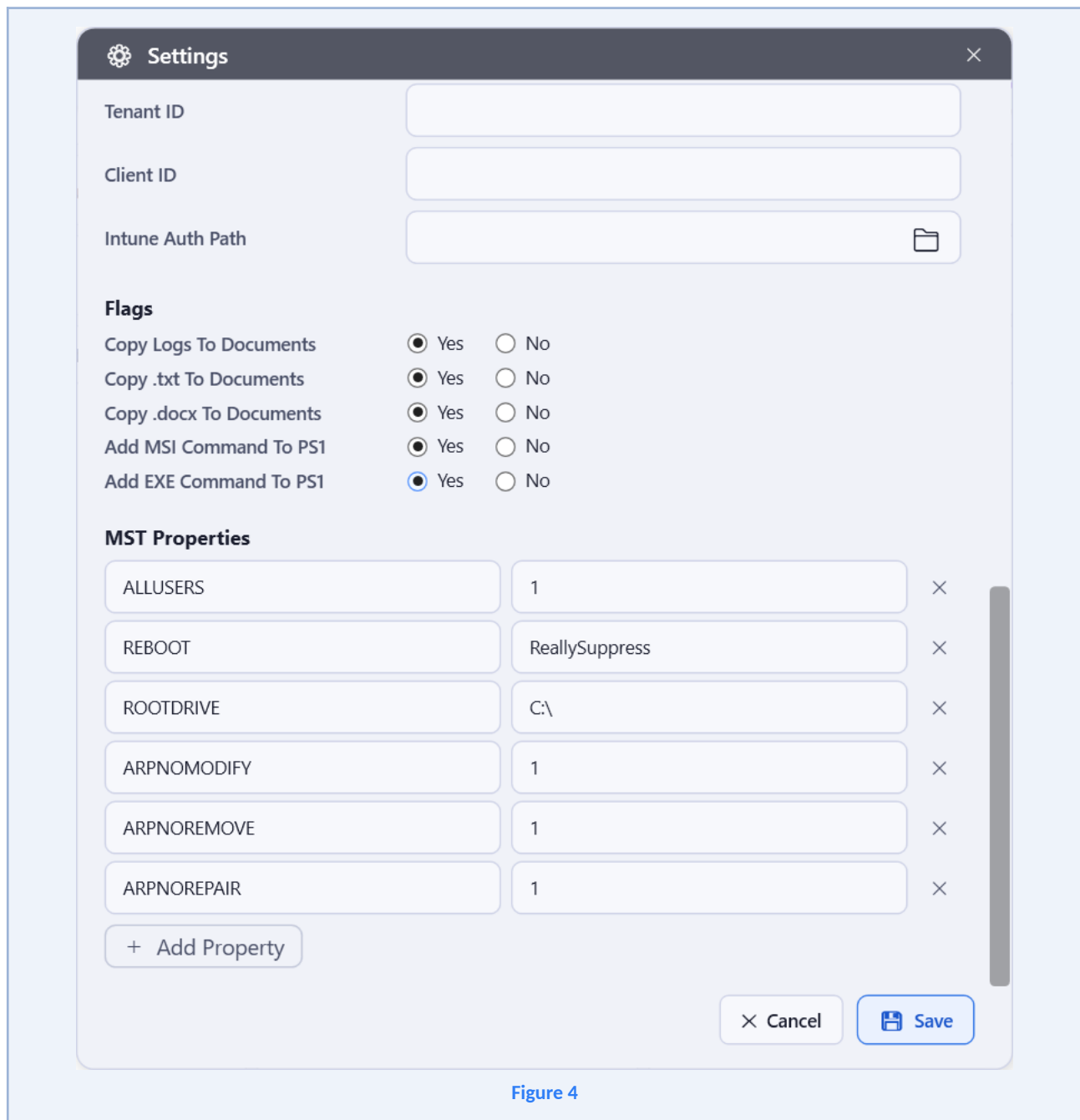


Figure 4

5. Click **Save**. A message shows you the full path of the file that was written.

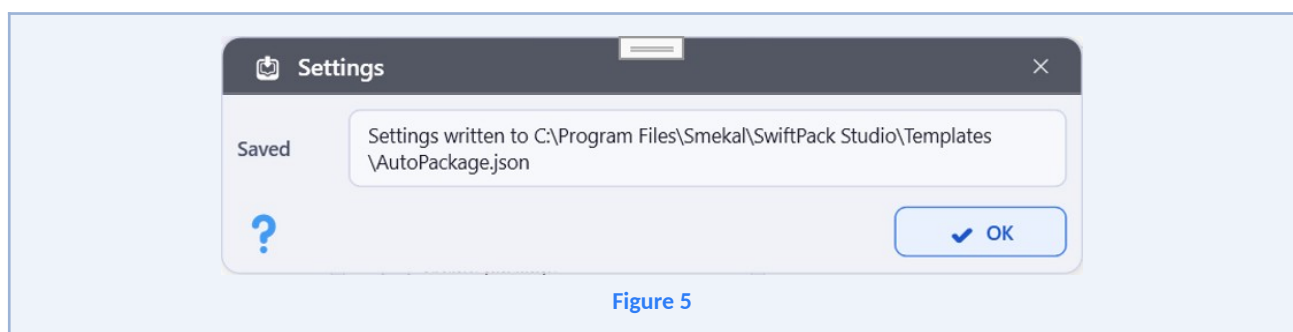


Figure 5

6. If you change your mind, click **Cancel** instead. Nothing is saved and the window just closes.

Settings apply to every application

Everything on this window is saved into one file: C:\Program Files\Smekal\SwiftPack Studio\Templates\AutoPackage.json. If you change a folder here, it changes for every application. Settings that belong to one application only are in the Add App window instead.

3.2 What the Settings folders are for

Folder	What goes there
Source Download Path *	Downloaded installers. Each application gets its own folder: <Source Download Path>\<Vendor>\<App Name>\
Package Input Folder *	The settings file for each application. An application only appears in the Vendor column after its file is here. Add App writes here, and Import reads from here.
Package Output Folder *	Finished packages.
App Version Report Path *	The version report.
PSADT Template Folder	Your PSAppDeployToolkit template. No red star, because it can be empty. But if it is empty, the packaging tick boxes appear ticked and greyed out, and the packaging steps do not run. Set this folder to turn them on.

4 Auto Package: Updates

The Updates button opens the Downloads window. It shows what is out of date. Opening it is instant and free, because it only shows what has already been collected.

- Click **Updates** at the top right of the Download tab.

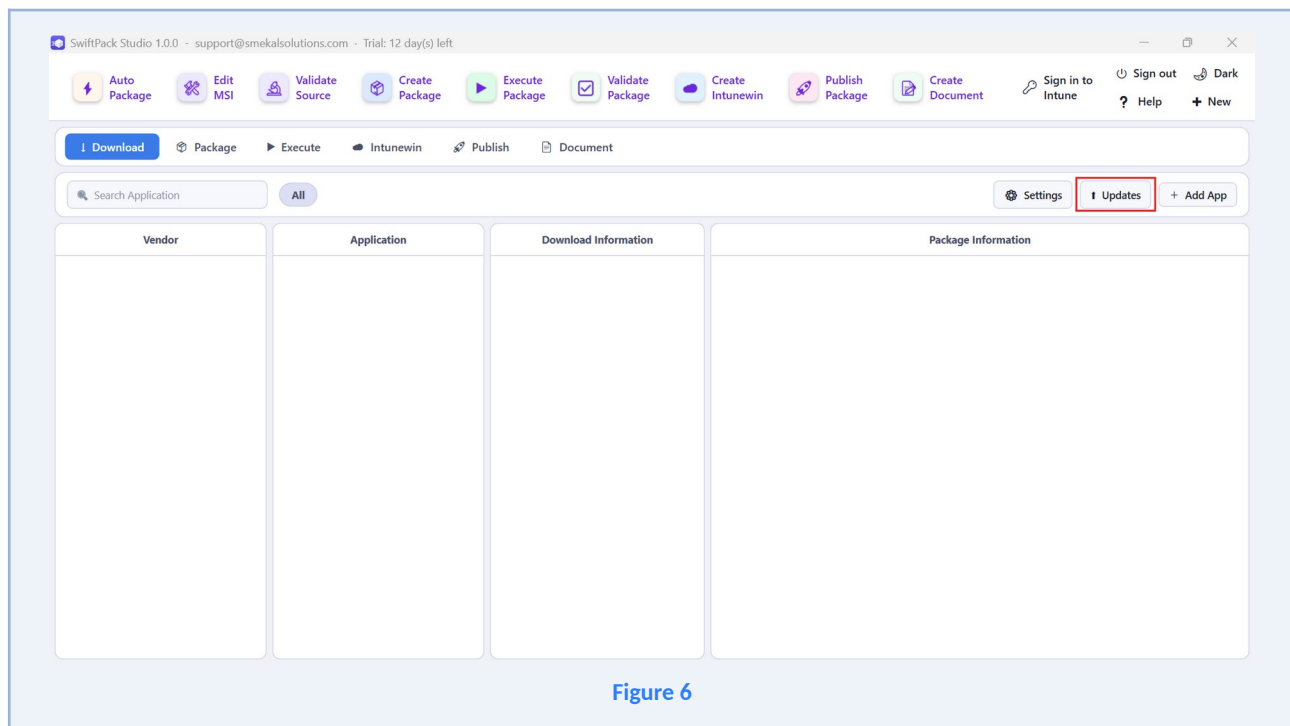


Figure 6

- Look at the three tabs. Each shows a count in brackets.

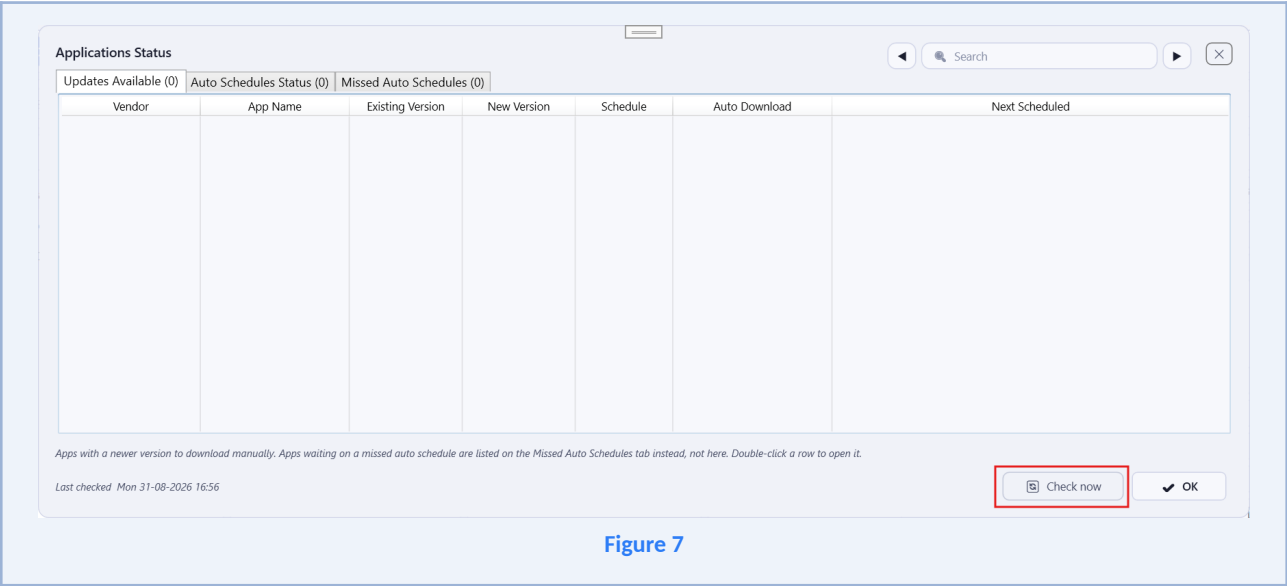


Figure 7

Tab	What it lists
Updates Available	Every application with a newer version, whether it is on a schedule or not.
Auto Schedule Status	Packages added to the catalog since your last check. This shows the status, completed or failed.
Missed Auto Schedules	Applications that were due to download while SwiftPack Studio was closed or missed.

9. Click **Check now** at the bottom right. This is the only button here that goes to the internet. It refreshes the catalog and then checks the current version of every application on your list, one at a time, so it takes longer the more applications you have.
10. Click **OK** to close. Nothing here changes a setting, so there is nothing to save.

i Your first Check now will show no new apps

The first check has nothing to compare against, so it records a starting point and reports nothing as new. This is normal and is not a fault. Run Check now again later and anything added in between appears in New Apps.

5 Auto Package: Add App

This section adds one real application from start to finish. Follow it exactly the first time. It takes about five minutes and nothing here can damage anything: you are only downloading an installer into a folder of your choosing.

5.1 Step by step

11. Click **Auto Package** on the toolbar, then the **Download** tab underneath it.
12. Click **Add App** at the top right.

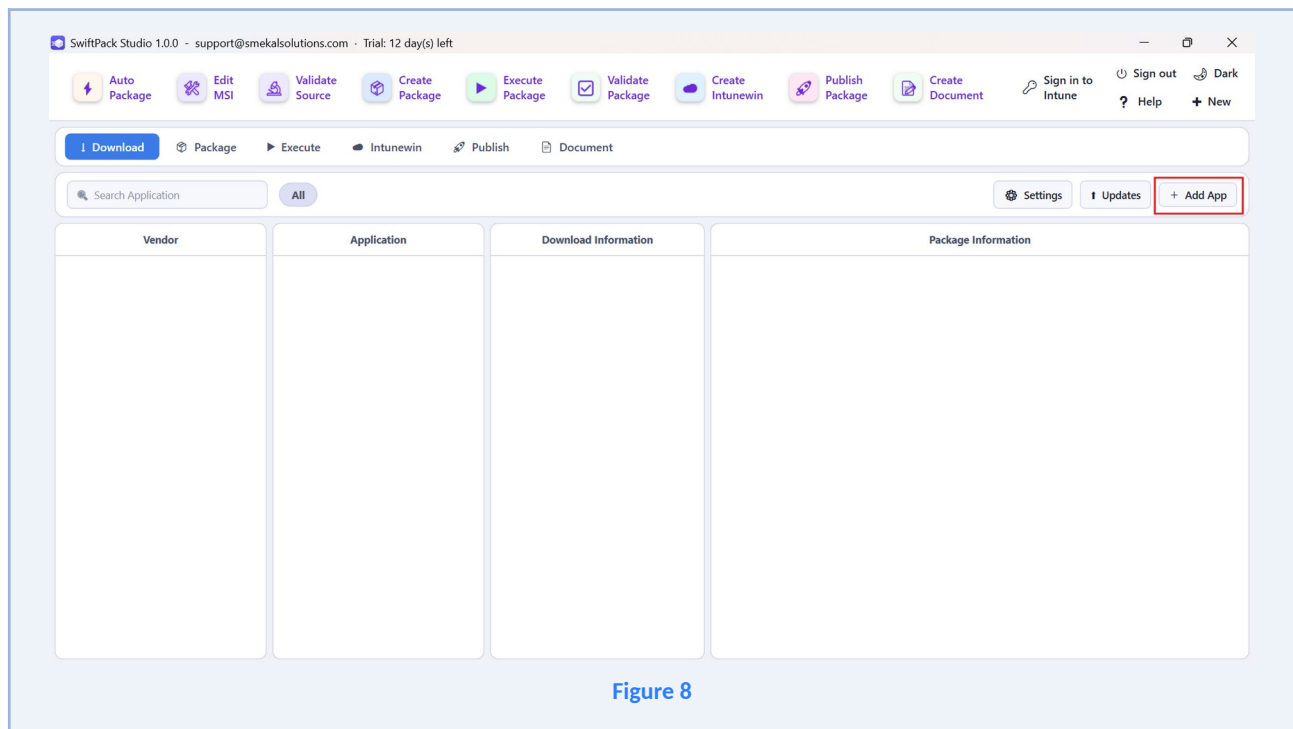


Figure 8

13. The Add App window opens.

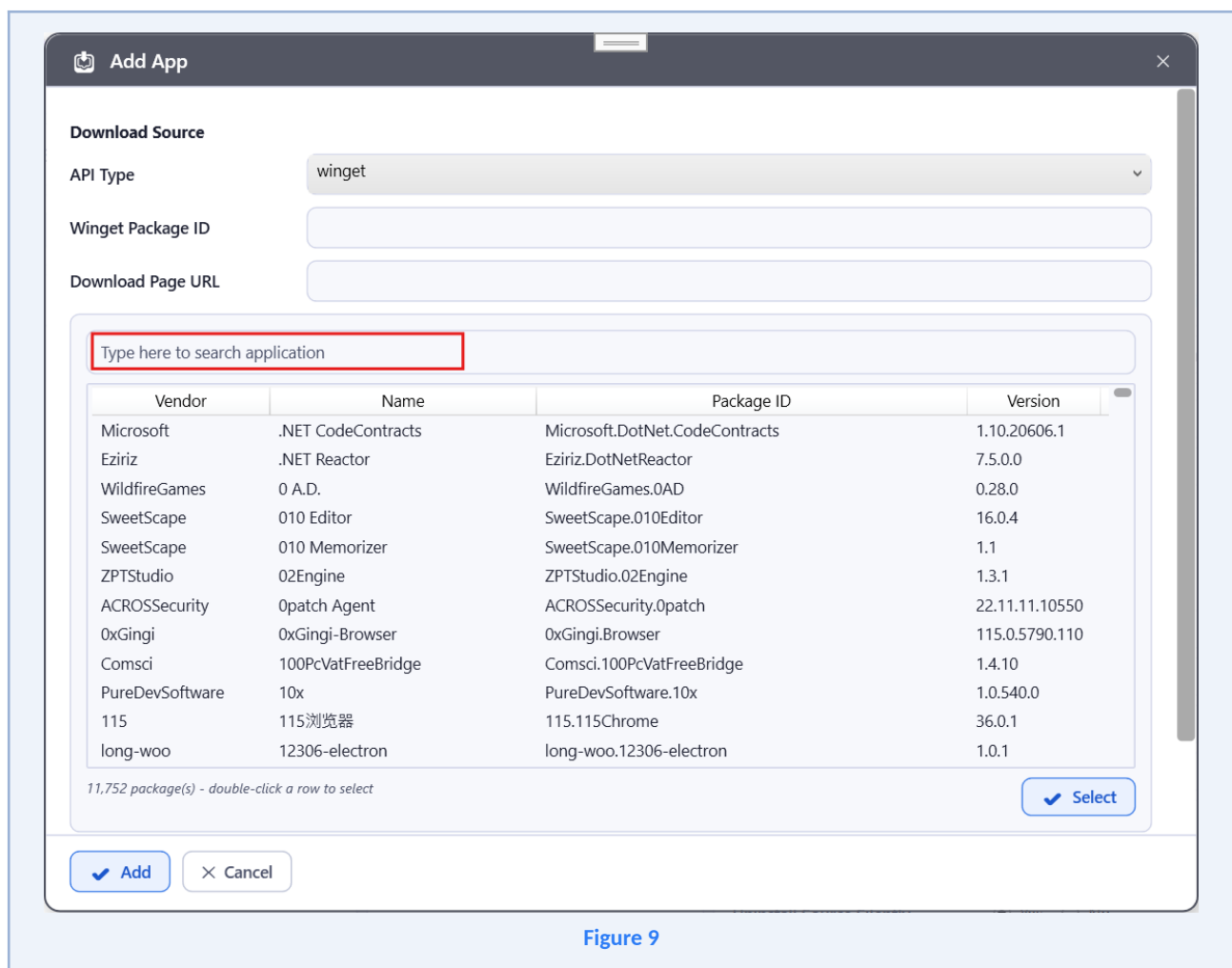
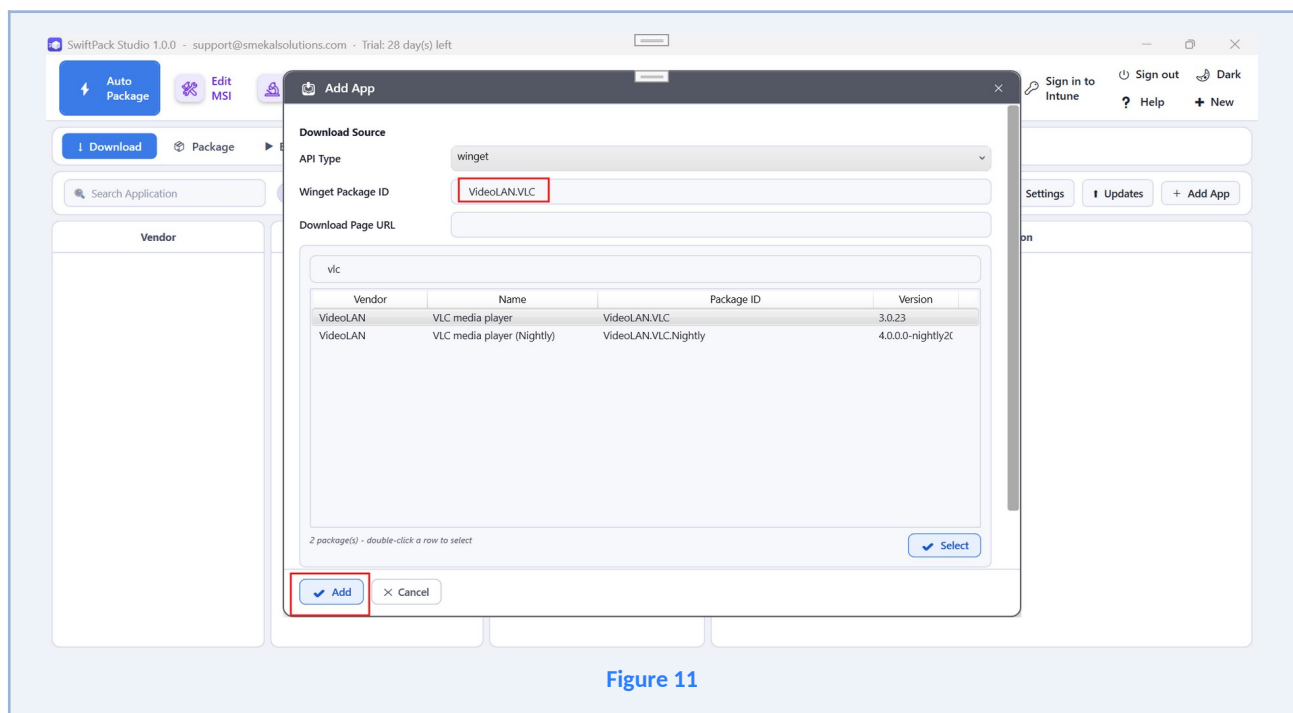
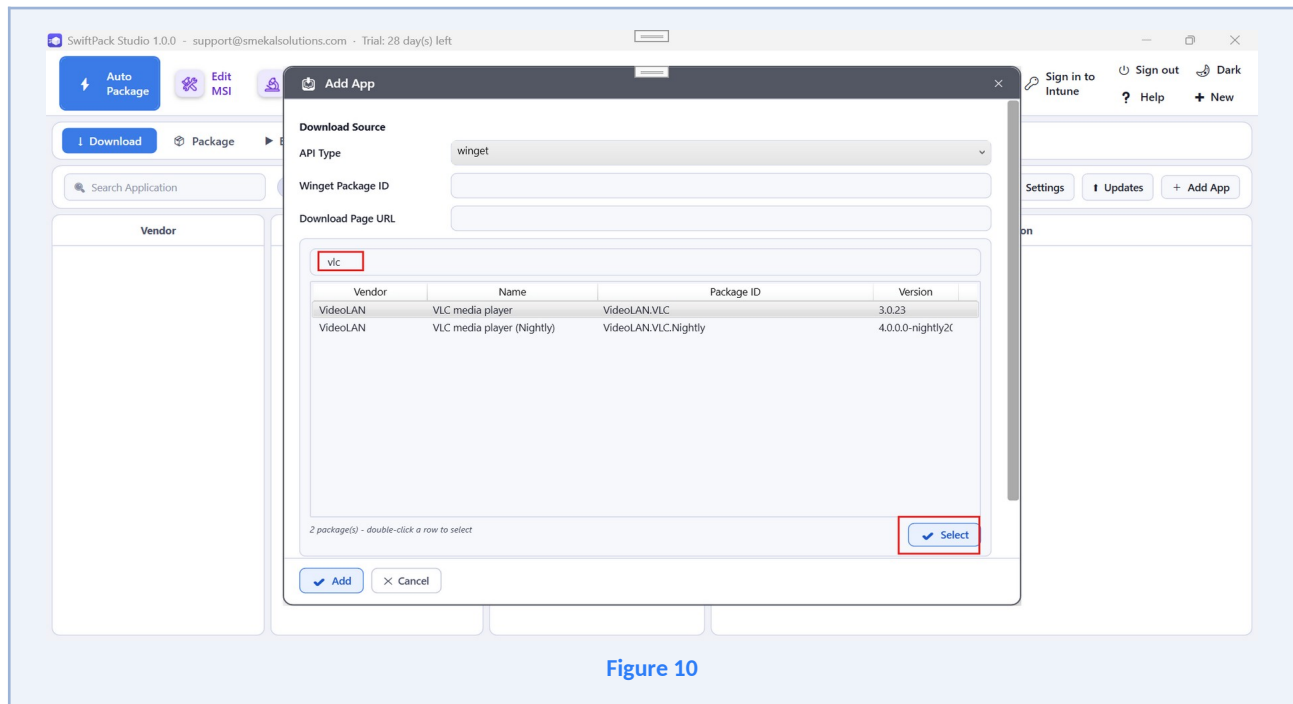


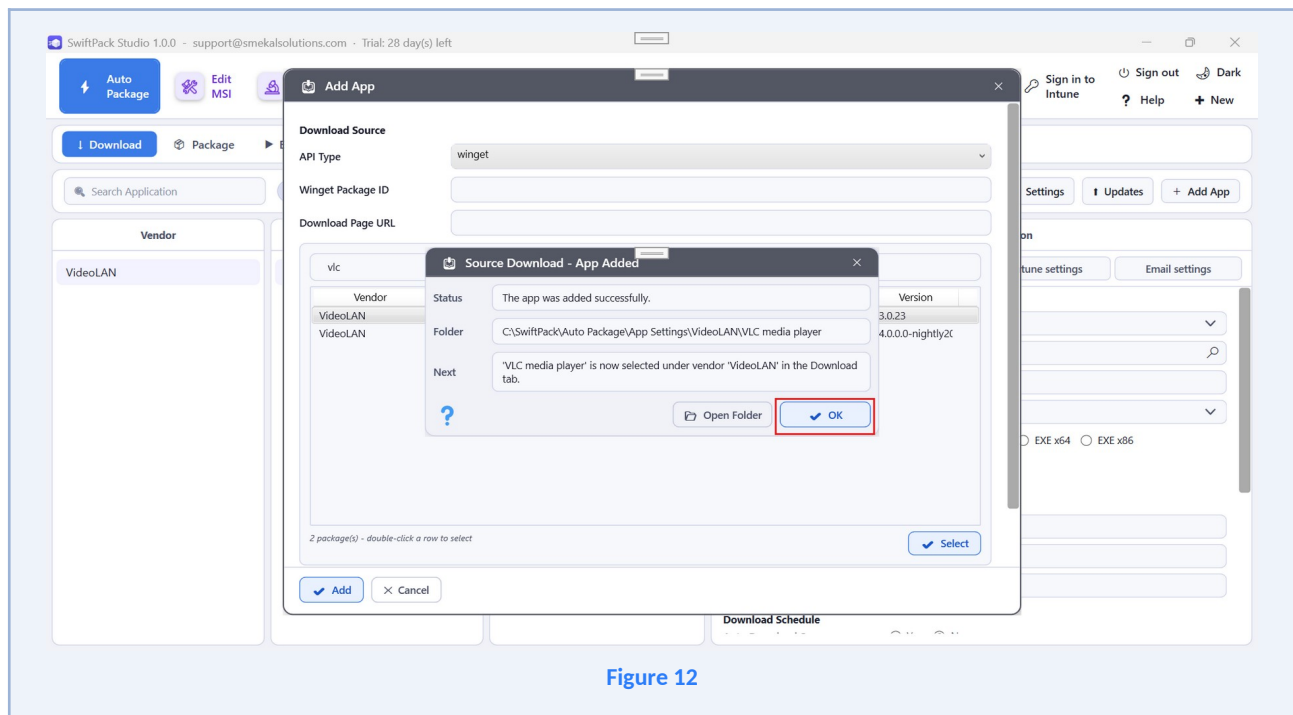
Figure 9

14. Under **Download Source**, leave **API Type** set to **winget**.

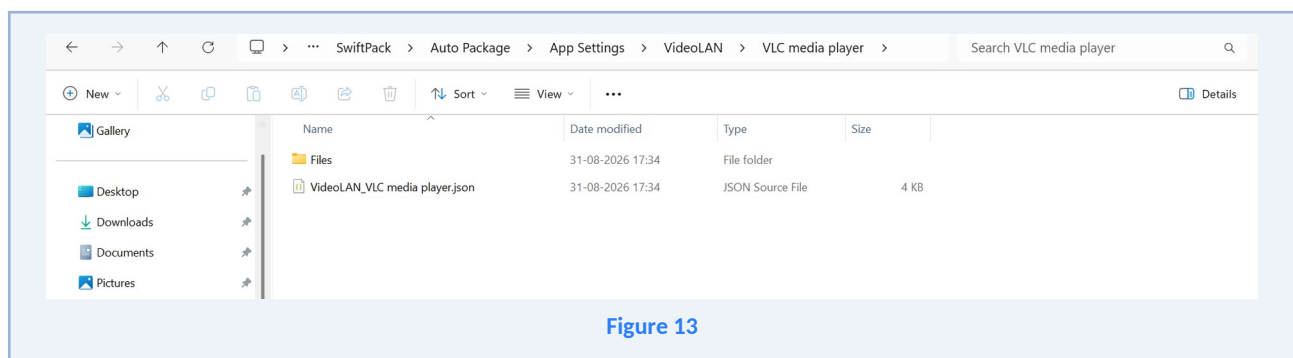
15. Search the application name in the “**Type here to search application**” box.
16. Type part of the application name, select it from the list, and click **Select**. The package ID is filled in for you.



17. Click **Add**, then click **OK** on the confirmation.



18. The folder structure is created in <Package Input Folder>\<Vendor>\<App Name>. With the default settings that is C:\SwiftPack\Auto Package\App Settings\<Vendor>\<App Name>.



5.2 What you should see

The Add App window closes and the application appears in the list. Three of the four columns fill in.

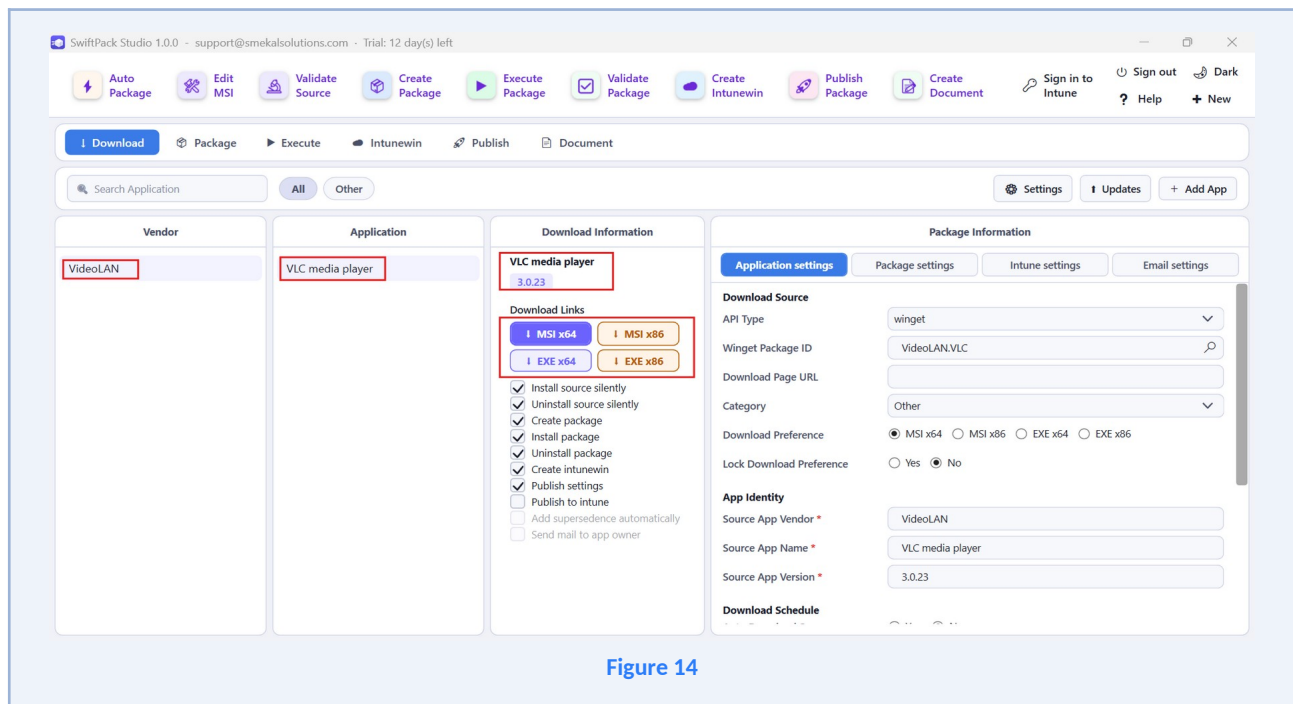


Figure 14

Column	What appears
Vendor	The Source App Vendor.
Application	The Source App Name.
Download Information	The version found online, the download size, and one button for each file the source offers, such as MSI x64 or EXE x64. Below that is the list of steps, with tick boxes.
Package Information	The settings you entered, across the Application, Package and Intune tabs. You can change them here without reopening Add App.

If the Application column stays empty, the settings file has not reached the Package Input Folder yet. If not, close and reopen SwiftPack Studio.

5.3 Download the installer

- Click the application in the **Application** column so the **Download Information** panel fills in.
- Now you tick one after another. If you want install source and uninstall source silently, tick only those. If you want install source and uninstall source silently, create package, install and uninstall package, create intunewin file and publish settings, tick only those.

Note

Publish to intune and Send mail to app owner need additional configuration, as discussed earlier.

5.4 Download Information panel

The centre panel shows the selected app's name, the latest version SwiftPack found (as a coloured badge), and a set of download-link buttons - one per installer variant that is actually available, such as EXE x64, MSI x64 or EXE x86. The button matching the app's Download Preference is highlighted. An Open Download Page link opens the vendor page when one is configured.

Below the download links is the action checklist - the switches that turn on unattended automation:

Action checkbox	Effect when ticked
Install source silently	After download, silently installs the downloaded installer on this machine.
Uninstall source silently	Silently uninstalls the source after the package has been created.
Create package	Builds the PSADT package from the downloaded installer.
Install package	Installs the newly-created package (requires a PSADT template folder).
Uninstall package	Uninstalls the package after installing it (requires a PSADT template folder).
Create intunewin	Produces the .intunewin file. Requires Create package.
Publish settings	Prepares and writes the Intune settings for the package. Requires Create package + Create intunewin.
Publish to intune	Uploads the app to Microsoft Intune. Requires Publish settings.
Add supersedence automatically	Adds all previous versions with the same app name as a supersedence.
Send mail to app owner	Emails the application owner after publishing. Requires an App Owner Email and Publish to intune.

21. Click on MSI x64 and observe the process on the screen.

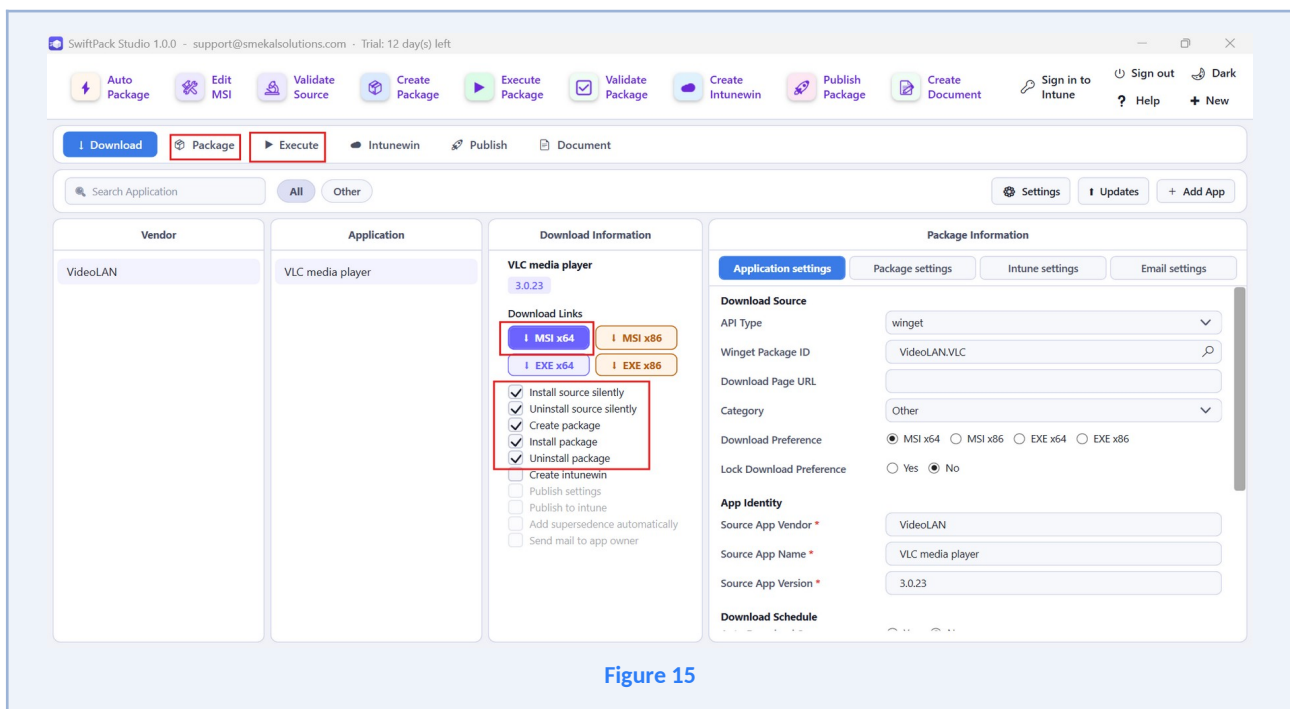


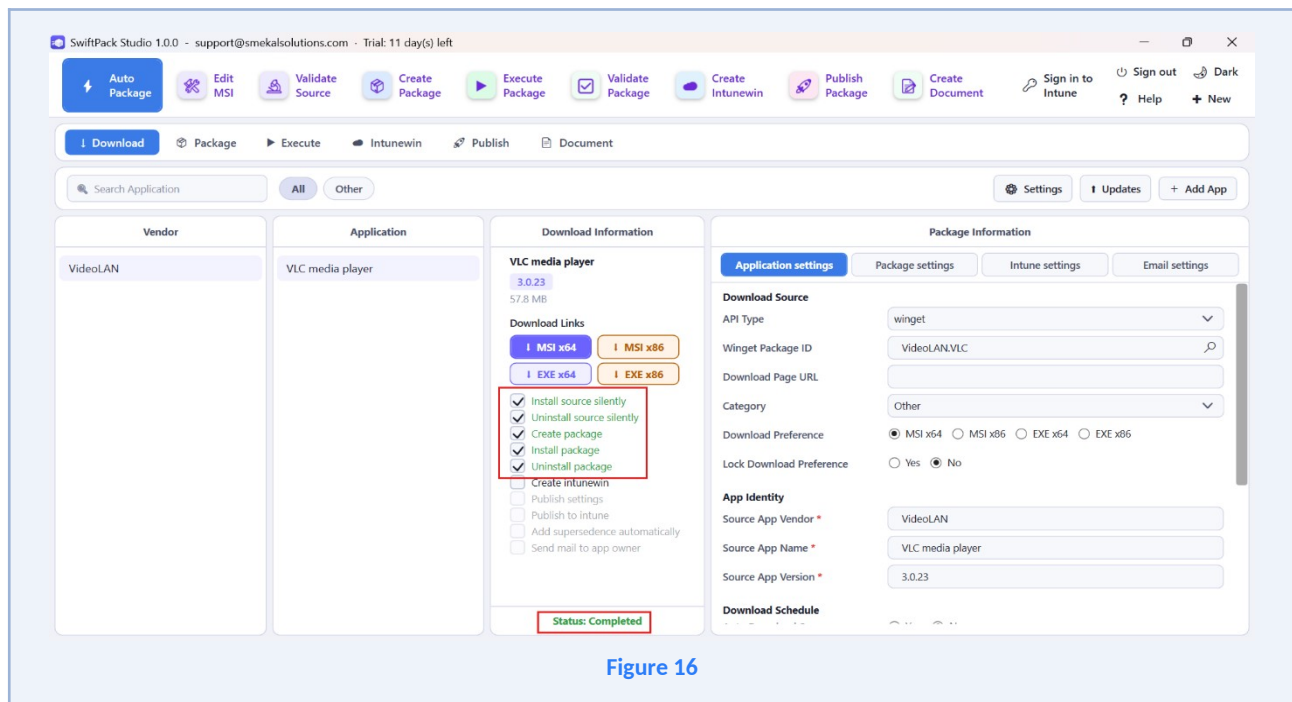
Figure 15

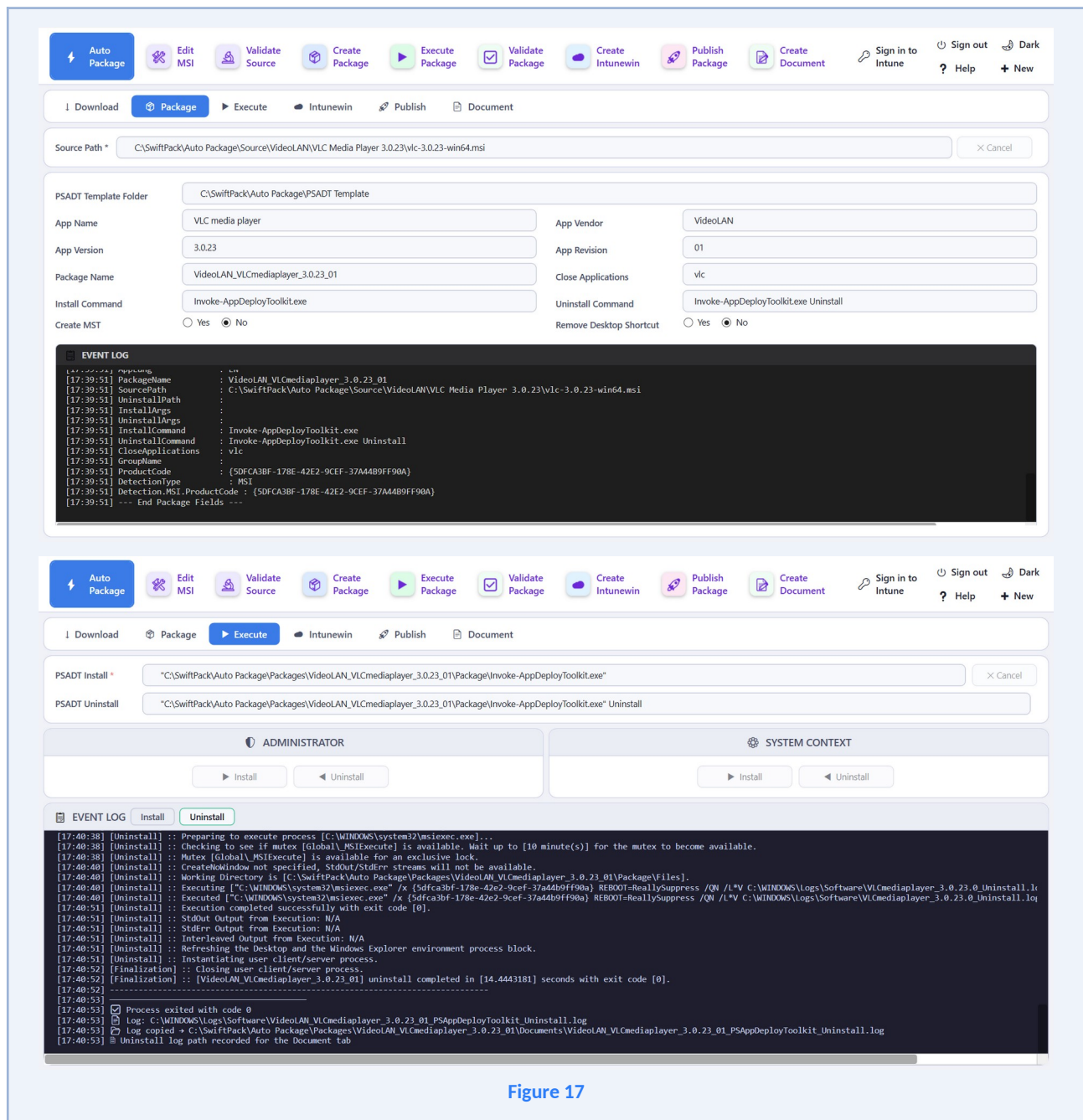
22. You can see the status in the Download Information column; also click the Package and Execute tabs to see the result.

The workflow tab strip

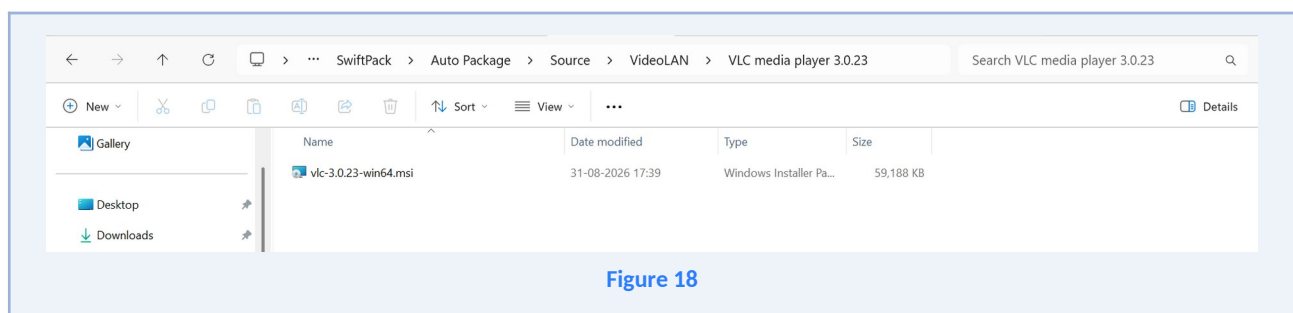
Tab	Icon	Purpose (summary)
Download	↓	Find, configure and download the installer; the control centre for unattended runs.
Package	📦	Build the PSADT package from the installer.

Execute		Install / uninstall the package locally to test it.
Intunewin		Create the .intunewin deployment file.
Publish		Upload the app to Microsoft Intune.
Document		Generate the delivery document.





23. The downloaded file is saved to <Source Download Path>\<Vendor>\<App Name>, e.g. C:\SwiftPack\Auto Package\Source\VideoLAN\VLC media player 3.0.23\



24. The package folder is created in <Package Output Folder>\<Vendor>_<App Name>_<Version>_<ReleaseVersion>, e.g. C:\SwiftPack\Auto Package\Packages\VideoLAN_VLCmediaplayer_3.0.23_01

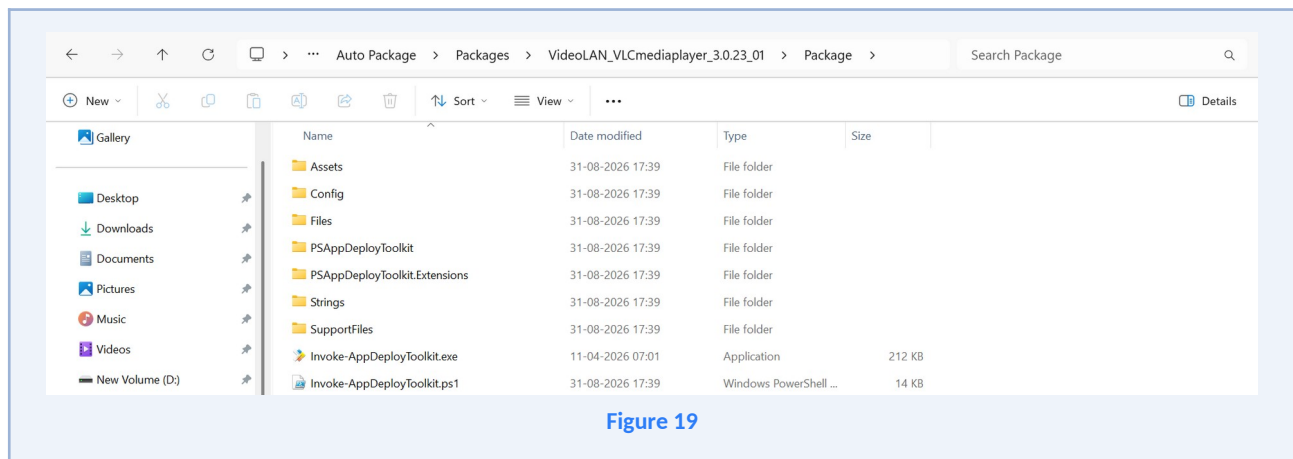


Figure 19

That is one complete application. Everything else in this guide is either another way of finding the installer, or another step to switch on after the download.

5.5 The other ways to find an installer

API Type tells SwiftPack Studio where to look for the installer. It has four settings. The tutorial used winget. The other three work the same way, with a different box below.

API Type	How you use it
winget	For anything in the winget catalog, which is most common software. Click the search icon next to Winget Package ID, search by name, and select. SwiftPack reads the version and the download links from the catalog.
github	For applications released on GitHub. Click the search icon next to the repository box, search, and select. SwiftPack reads the newest release and lists the files attached to it. Paste a GitHub token in Settings if you use this often, or you will hit the hourly limit.
Vendor API	For a vendor that publishes its own version feed. Enter the Vendor API ID. Use this when the application is not in winget and not on GitHub, but the vendor offers a version check.
none	For everything else. Enter the Download Page URL yourself. Nothing is looked up, so there is no automatic version check: you tell SwiftPack where the file is and it downloads it.

Everything after this point is identical whichever you choose. The application still appears in the Vendor and Application columns, the Download Information panel still shows the download buttons and the step tick boxes, and the file still goes to the same folder.

Which should you choose?

Try winget first, because it needs the least work and keeps the version check accurate. Use github if the application lives there. Use Vendor API only if the vendor documents one. Use none as a last resort, and remember that it cannot tell you when a new version appears.

5.6 How to configure auto schedule

What happens during an automated run

While an unattended chain is running, Auto Package locks the input fields and most buttons so you cannot accidentally change something mid-flight. The tool still drives the hidden buttons itself. During this time:

- The tab strip switches automatically as each stage runs; you can watch but should not fight it.
- Editable fields become read-only and action buttons are greyed out.

- A Cancel button remains active on each stage so you can stop the run at any point.
 - If any step fails, the chain stops immediately, the remaining steps are skipped, and the workspace unlocks so you can investigate.
25. Select **Yes** in Auto Download Source, then select **"Hourly"** in Download frequency. Select Schedule Hour, Year, Month, Date and Time.
 26. On the particular date and time, download triggers automatically if a higher version is available, and completes all the steps selected in the Download Information column.
 27. You can also choose Daily, Weekly, Every 2 weeks, Monthly, Quarterly, Half yearly, Yearly and Every, as per your requirement, in Download frequency.

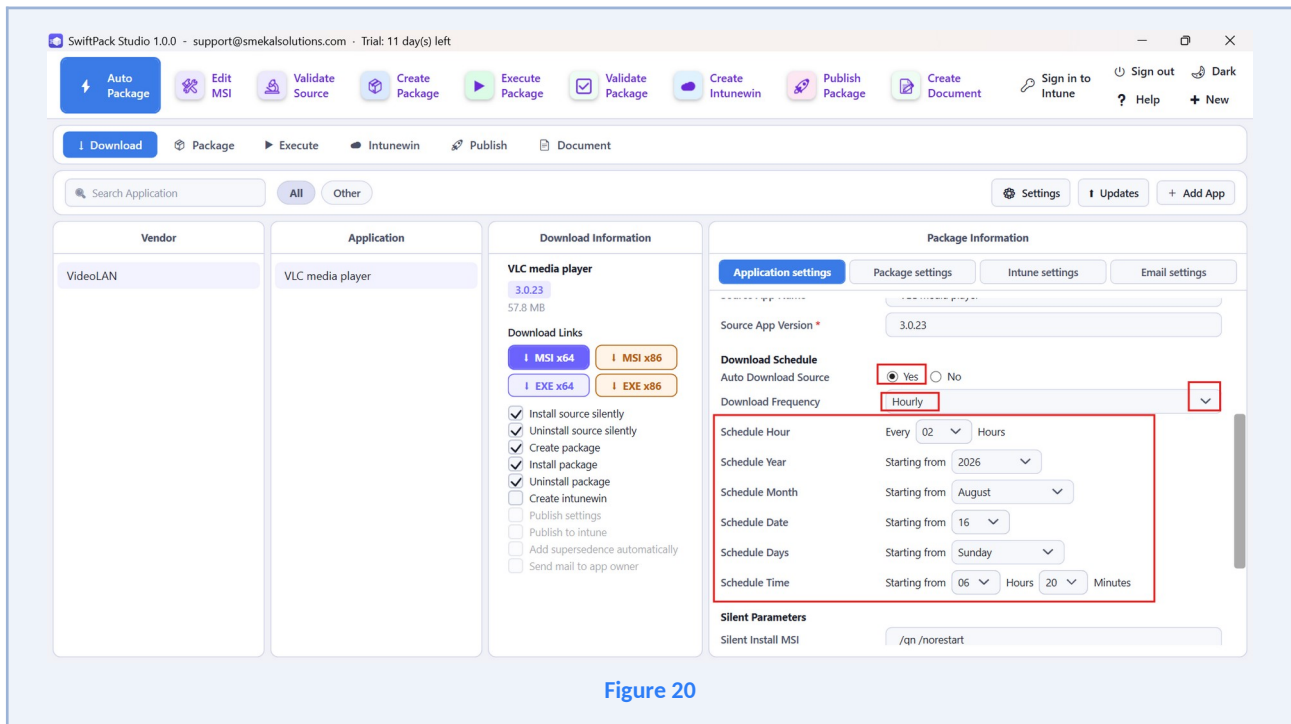


Figure 20

5.7 How to register SwiftPack Studio in Entra

Please follow the instructions in the document **"5. SwiftPack Studio - How to register an application in intune.pdf"**.

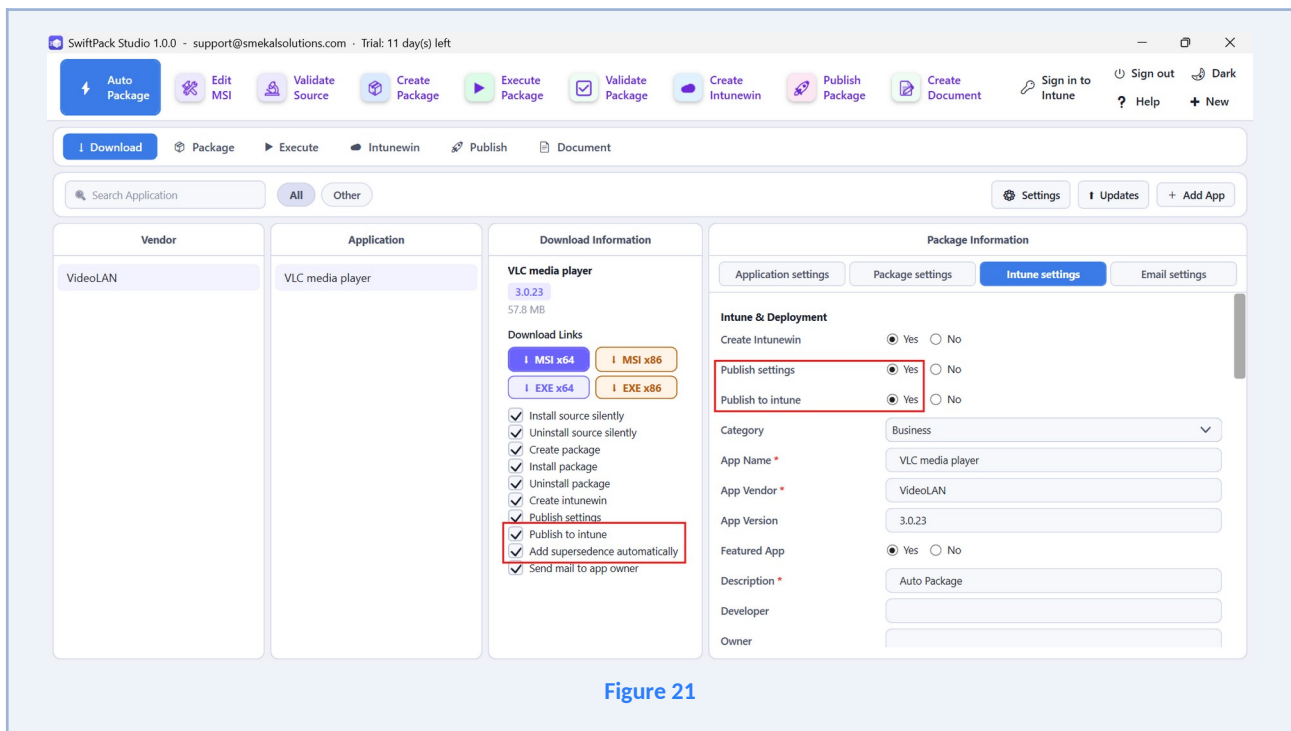


Figure 21

5.8 How to configure sending mail to the app owner

Please follow the instructions in the document “[6. SwiftPack Studio - How to configure mail notification in intune.pdf](#)”. If you add a mail subject and mail body here, it is used instead of the mail subject and mail body in Settings.

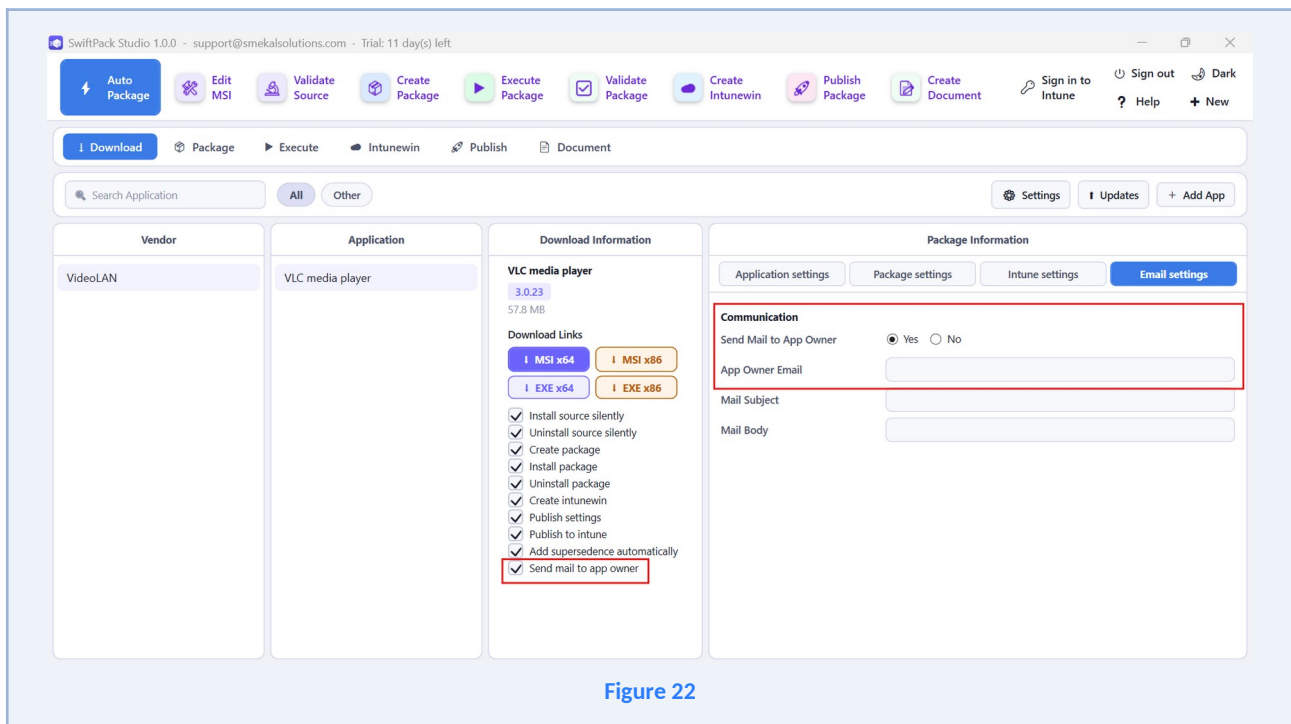


Figure 22

5.9 How to deploy an application via Intune and send mail to the app owner

1. Click on Sign in to Intune. Enter the Intune user credentials. Wait for 10 seconds.

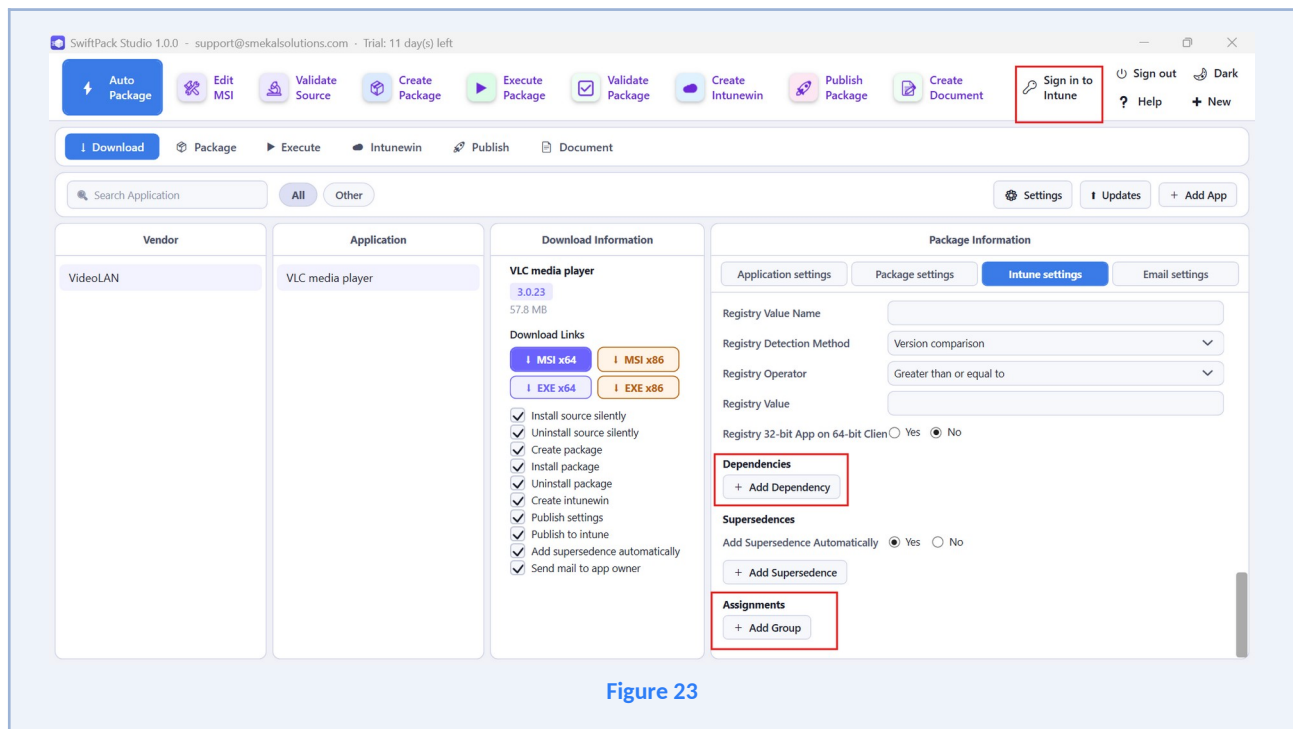


Figure 23

2. If you want to add a dependency, click “**Add Dependency**” (this is just an example). You can search for the particular app name, tick the application you want to add as a dependency, then click **Select**.

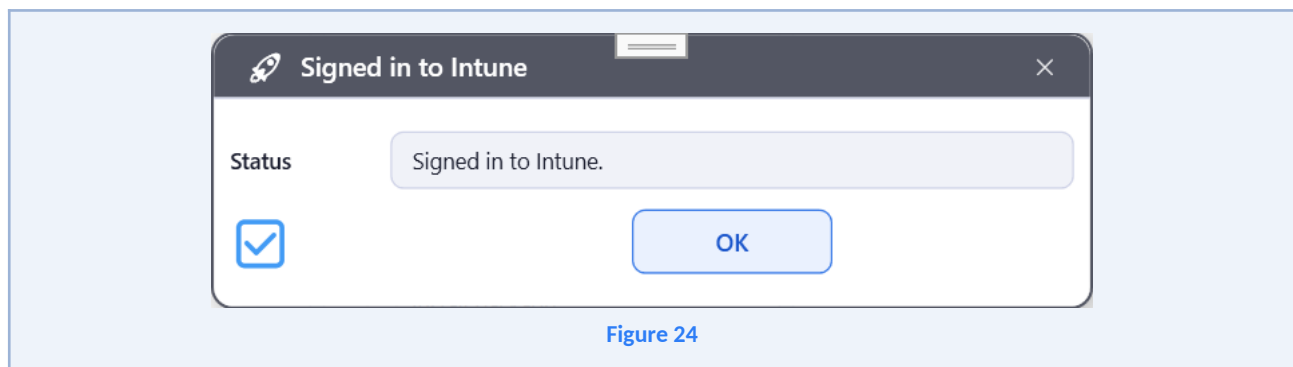


Figure 24

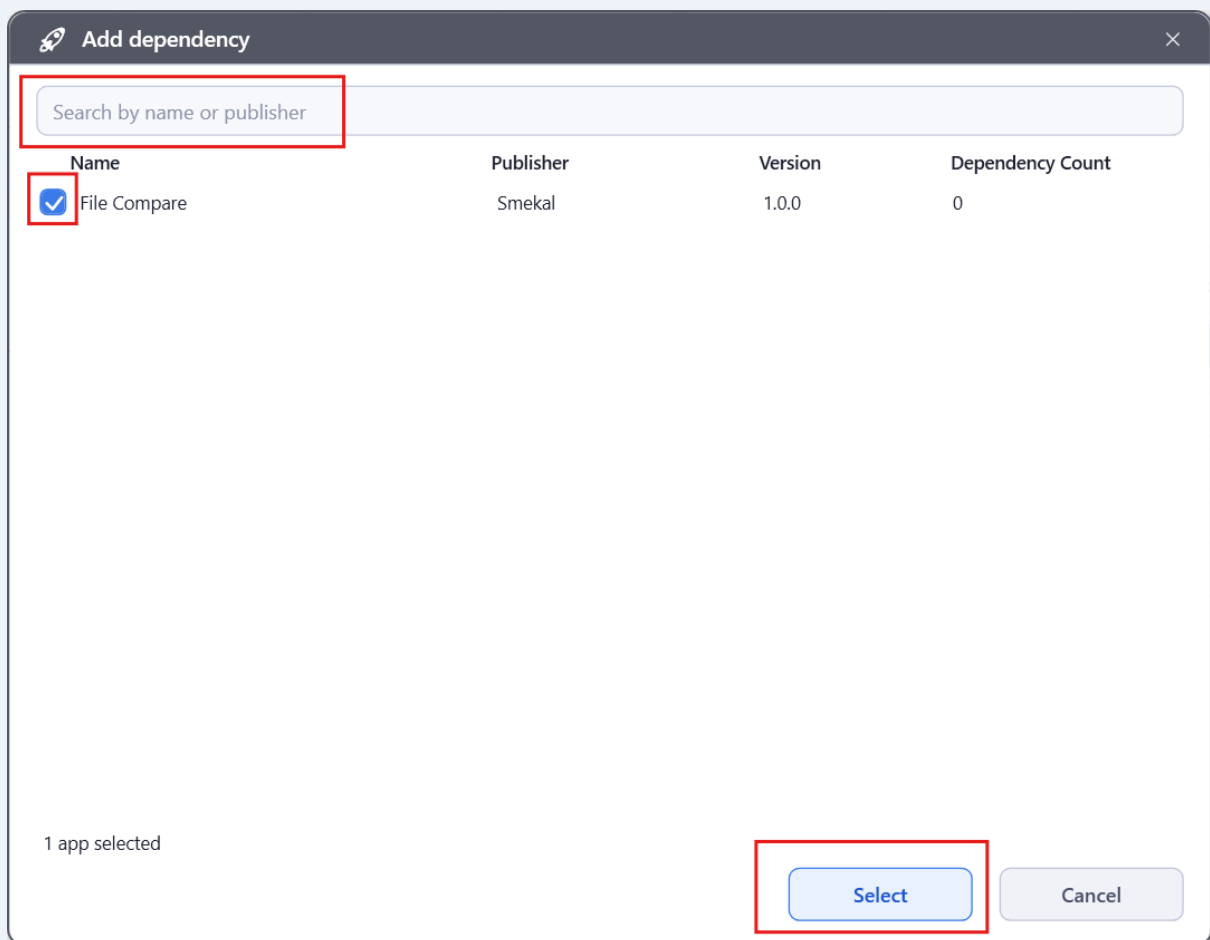


Figure 25

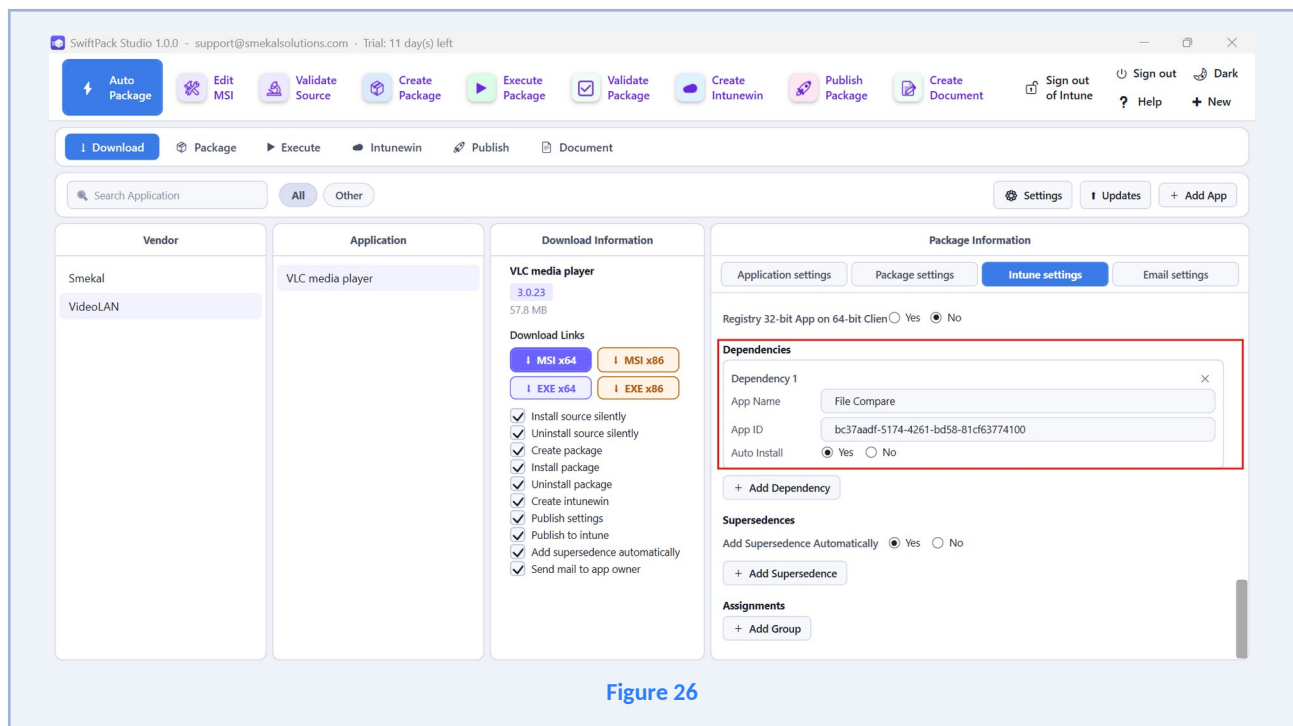


Figure 26

- Similarly, if you want to add supersedence, in the Download Information column, if you tick “**Add supersedence automatically**”, any previous versions available will automatically be added as a supersedence if the App Name matches.

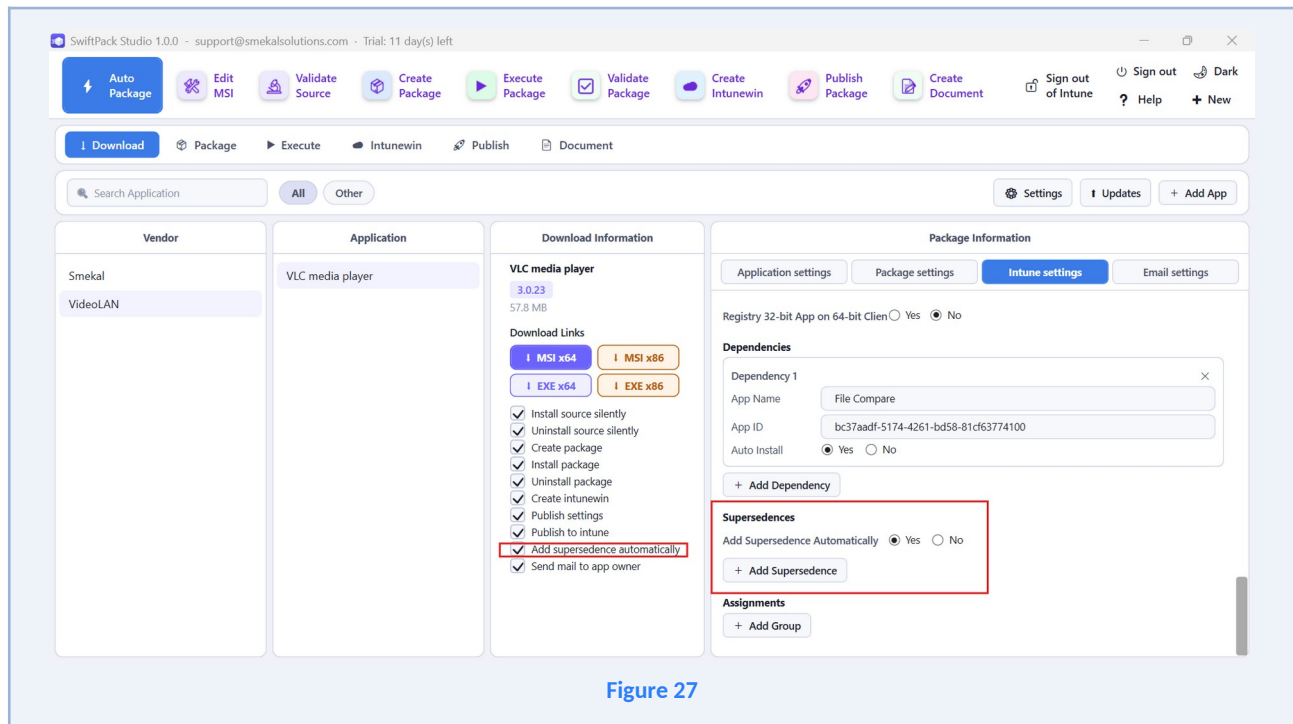


Figure 27

- Click “**Add Group**”, tick the group name you want, then click **Select**.

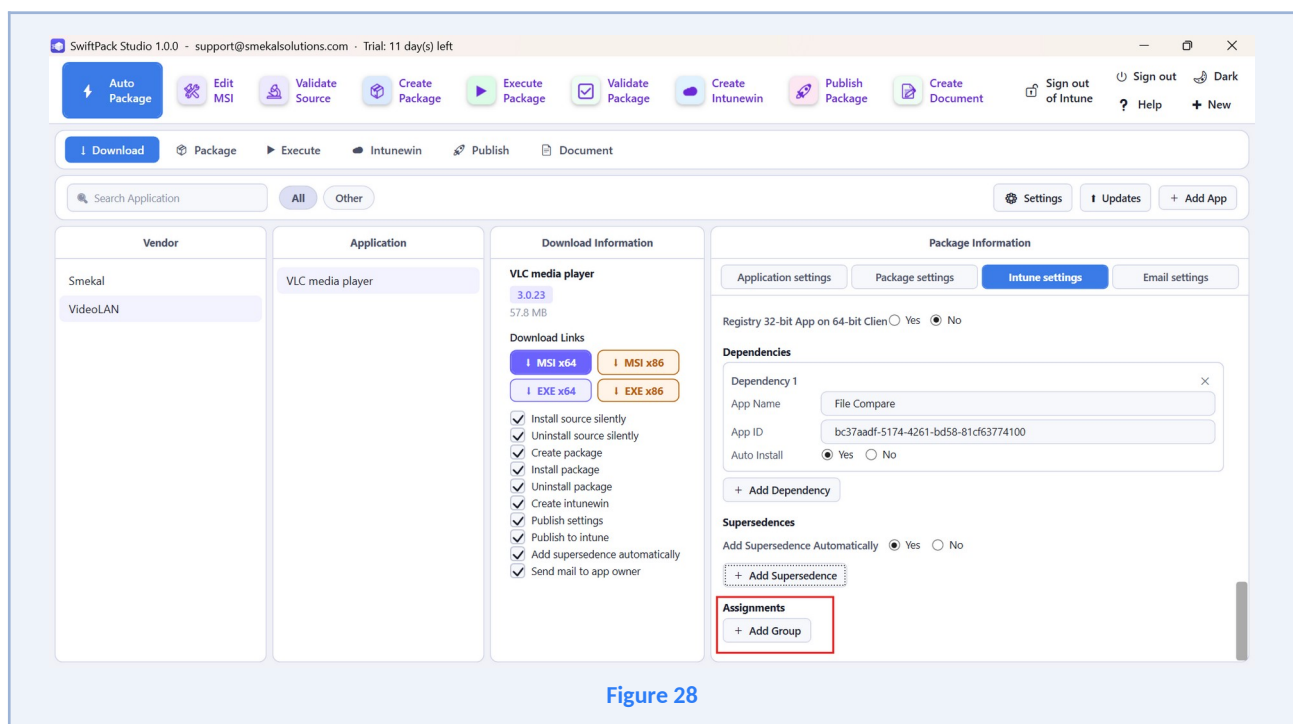


Figure 28

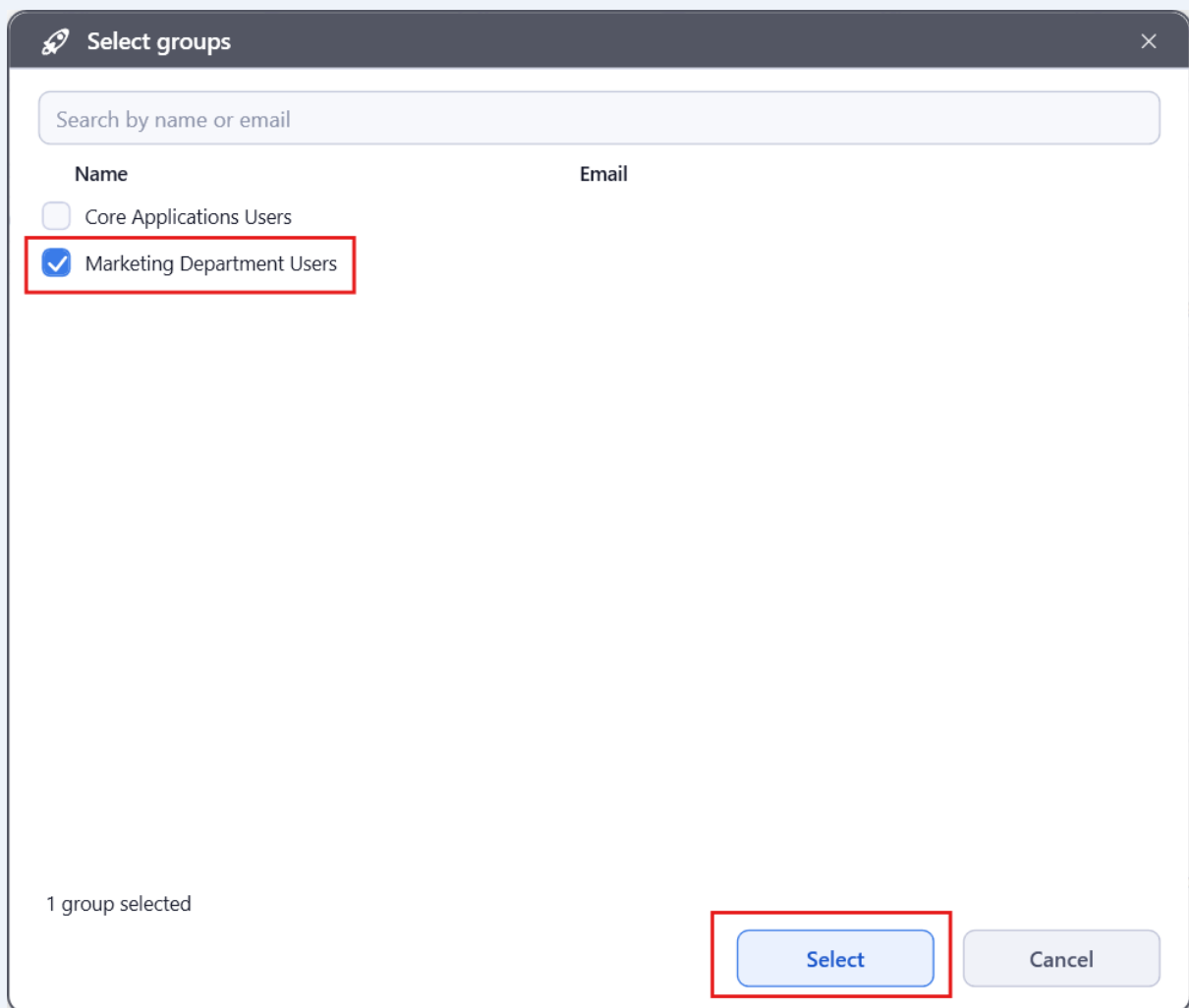


Figure 29

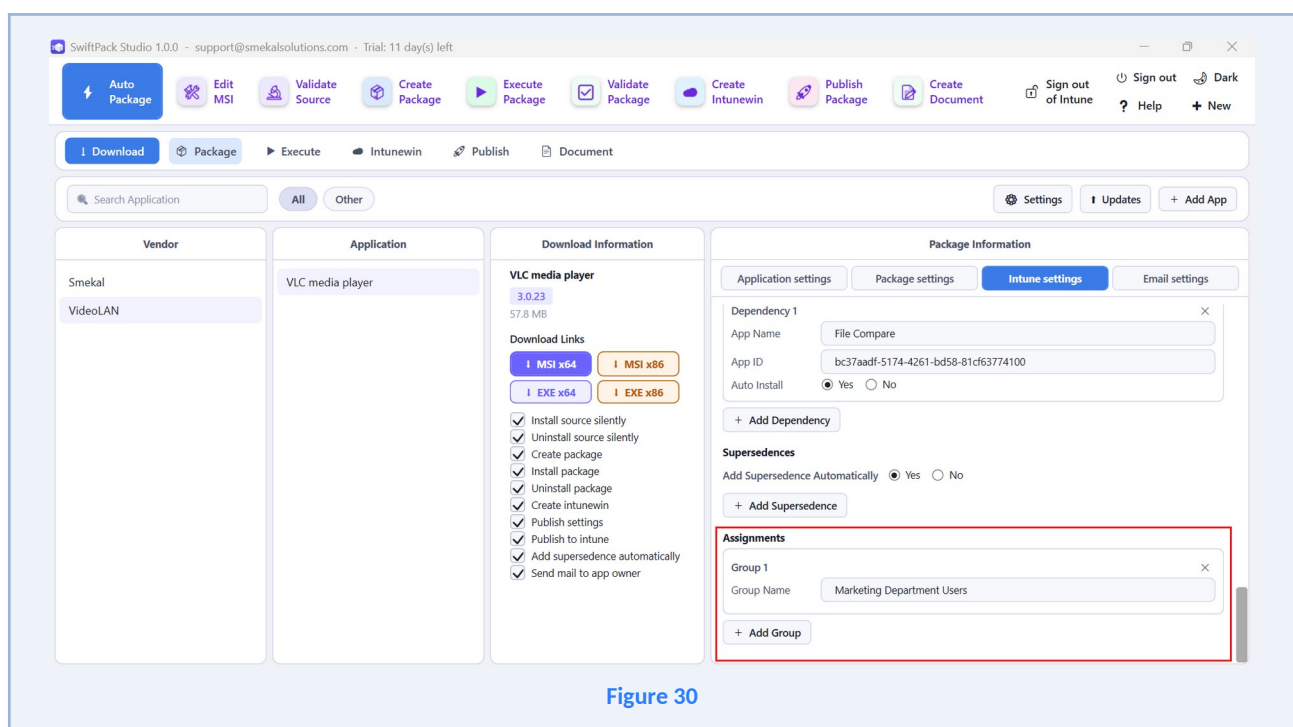


Figure 30

5. In Email settings, add the app owner email.

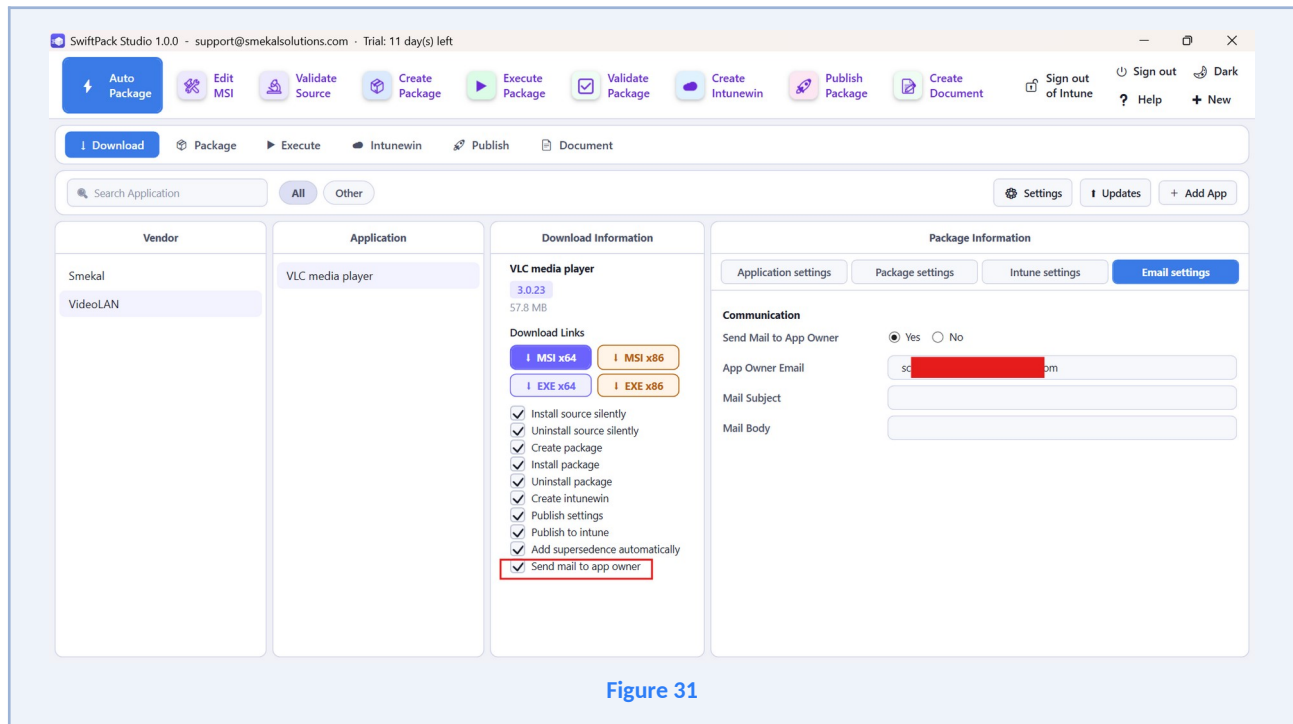


Figure 31

6. If you want to test now, click on MSI x64, then see the results.

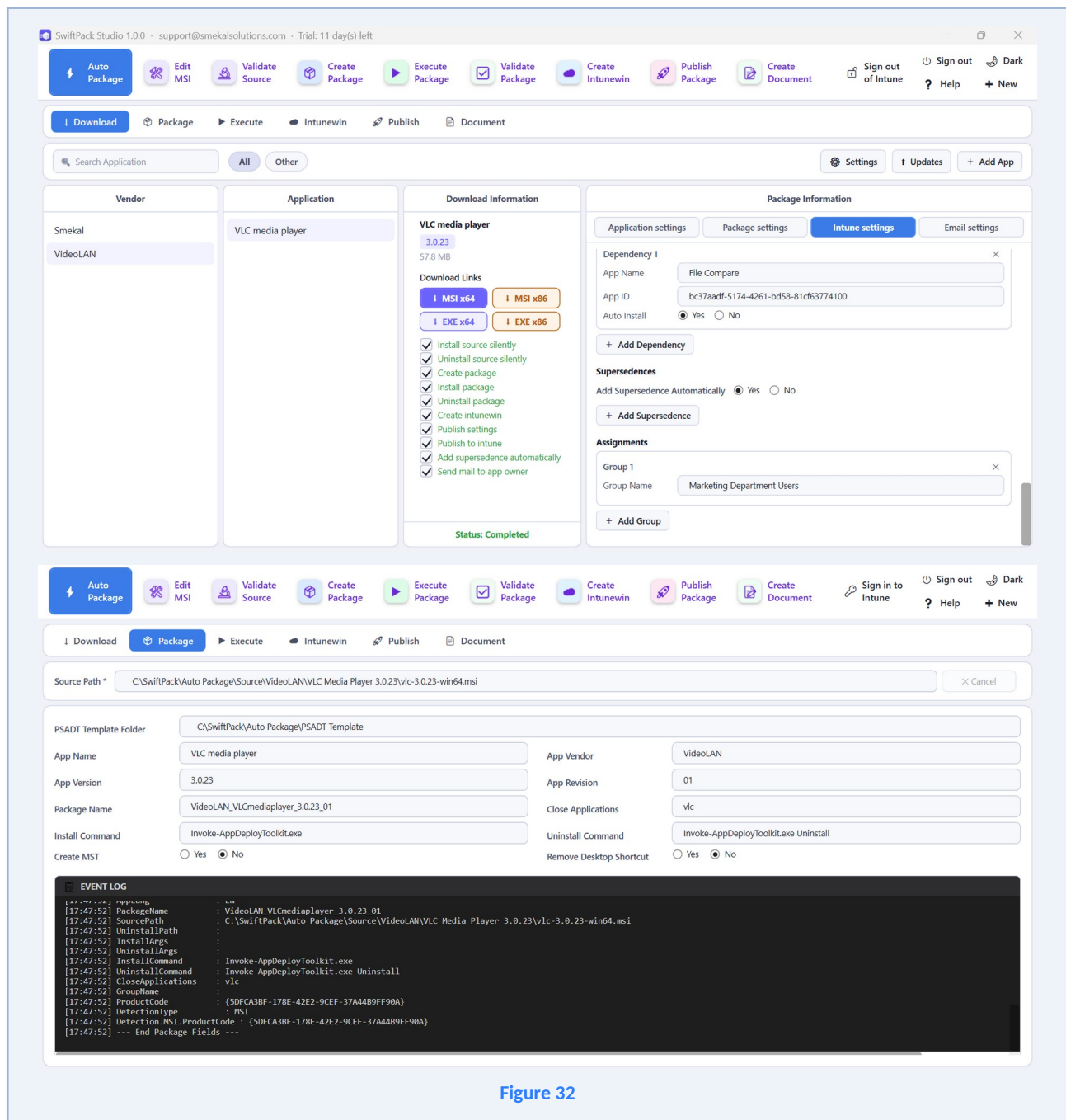


Figure 32

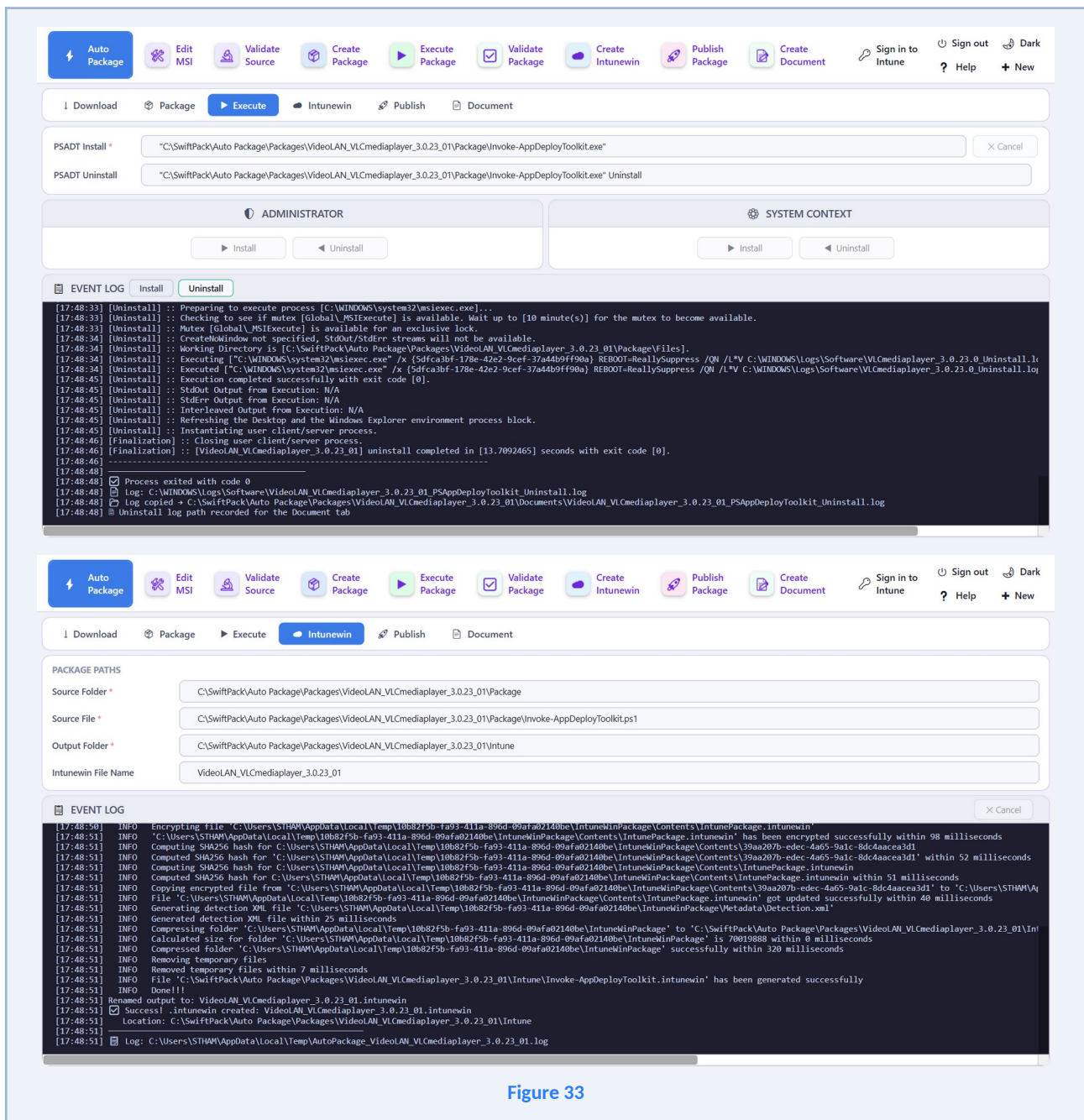


Figure 33

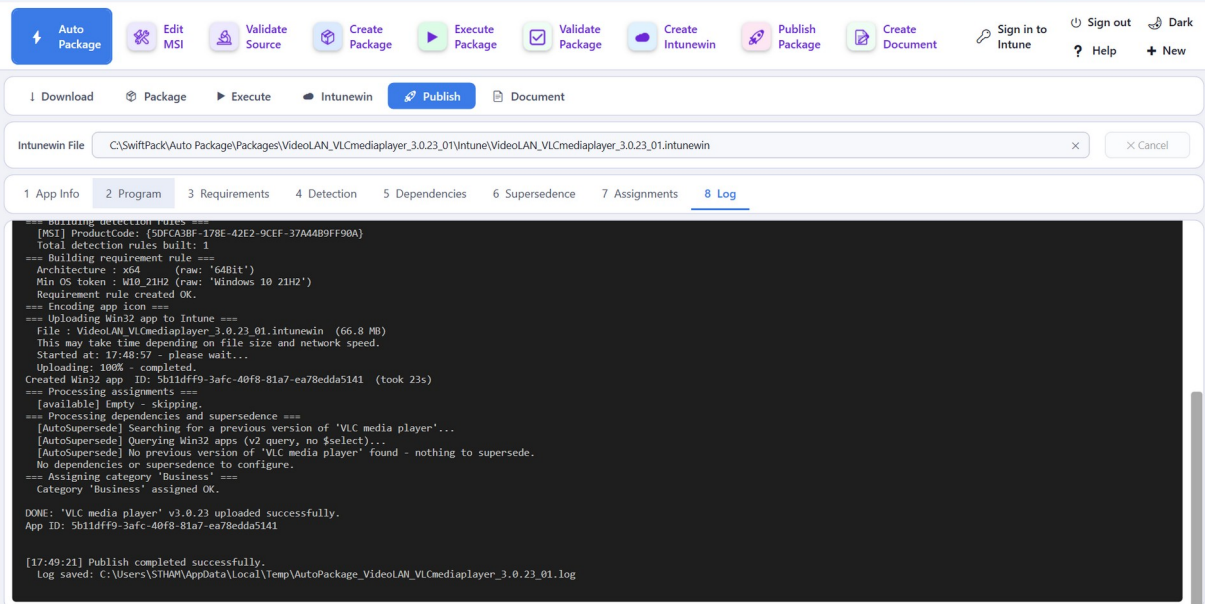


Figure 34

In Intune

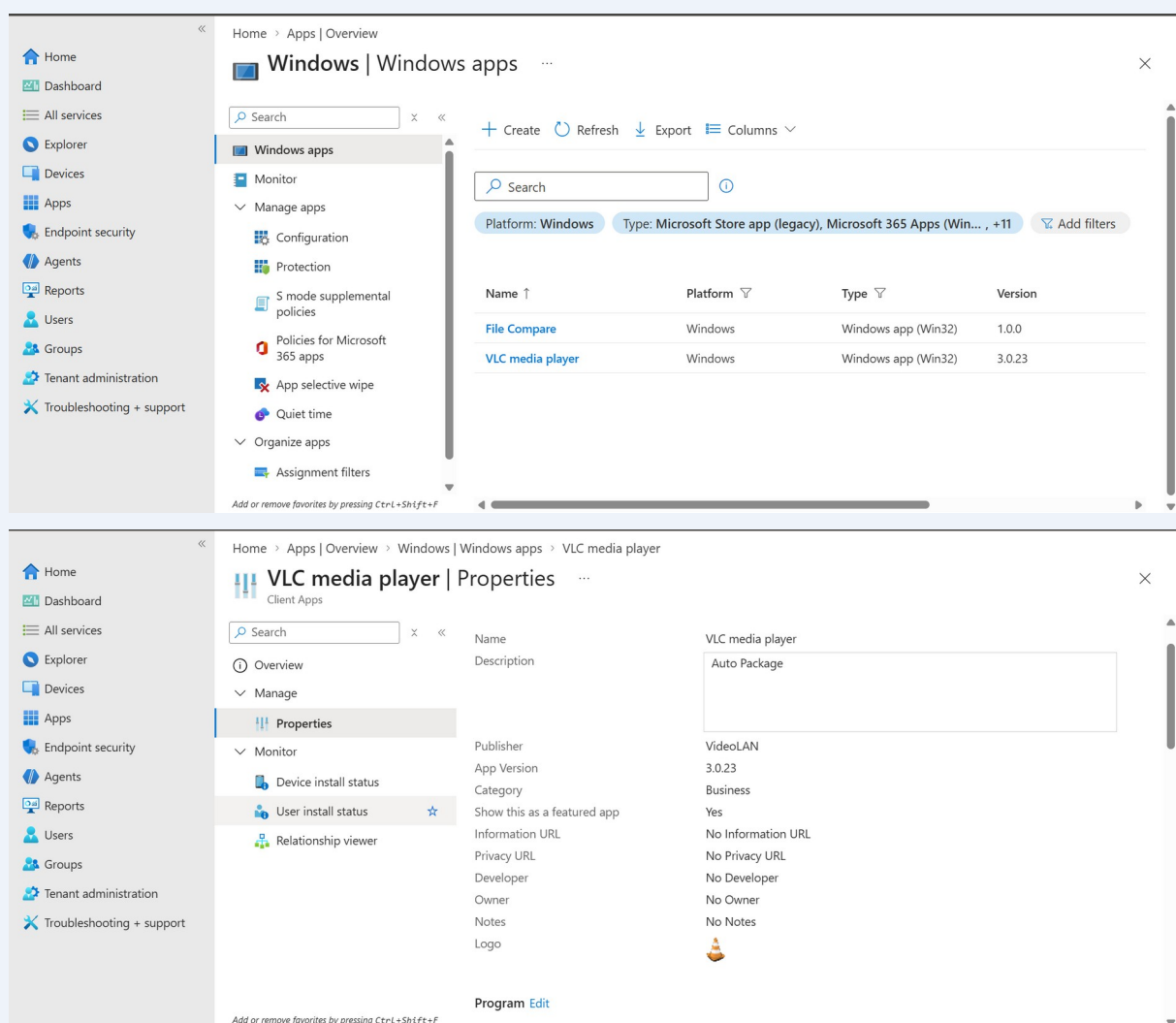
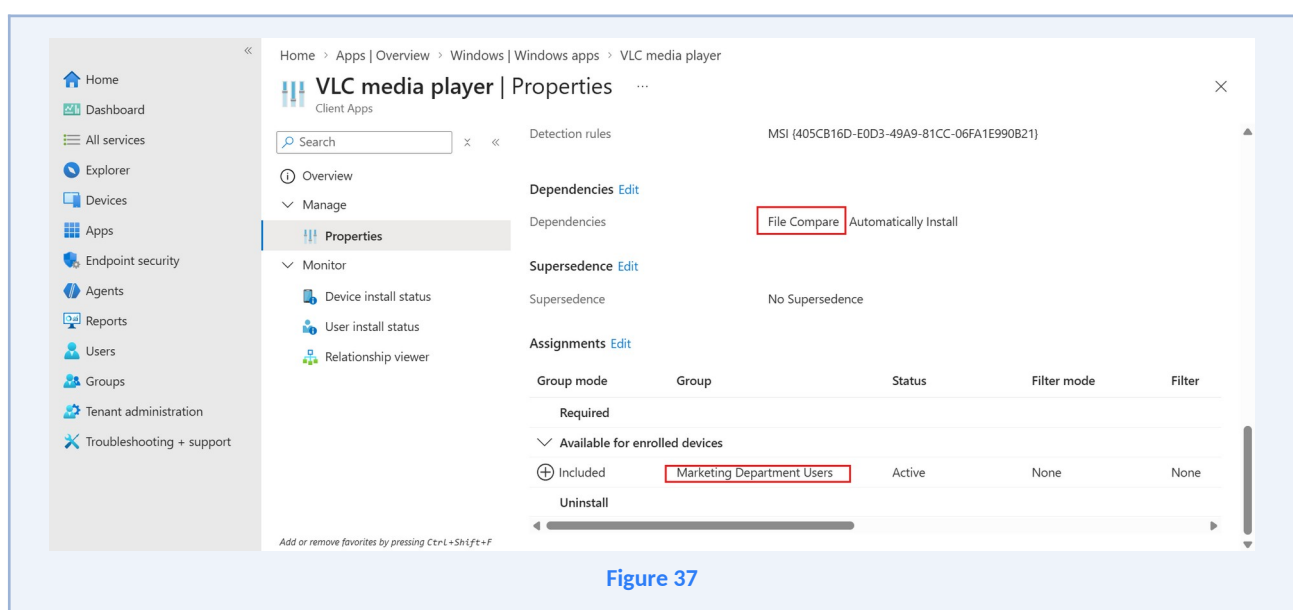
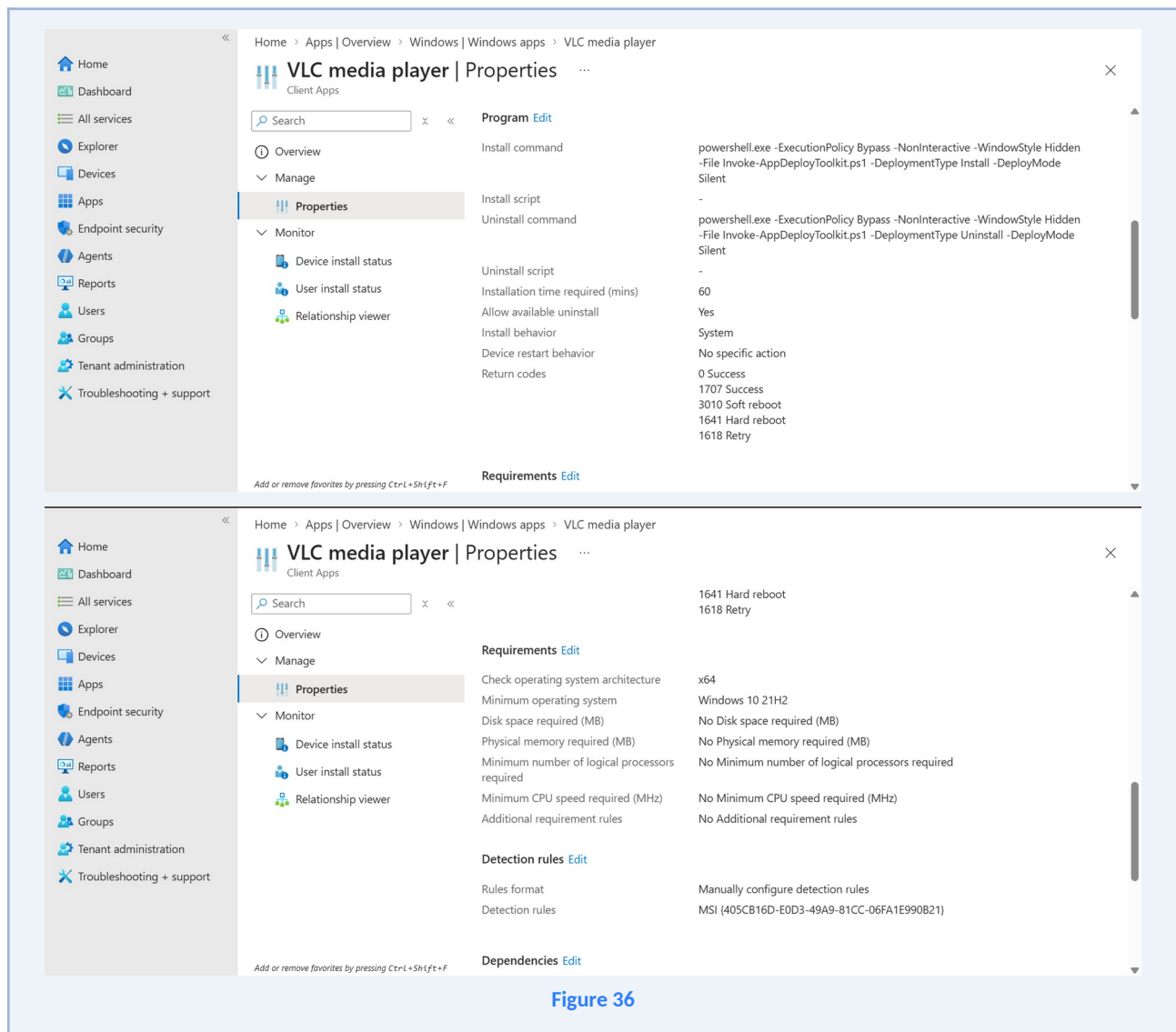


Figure 35



In mail

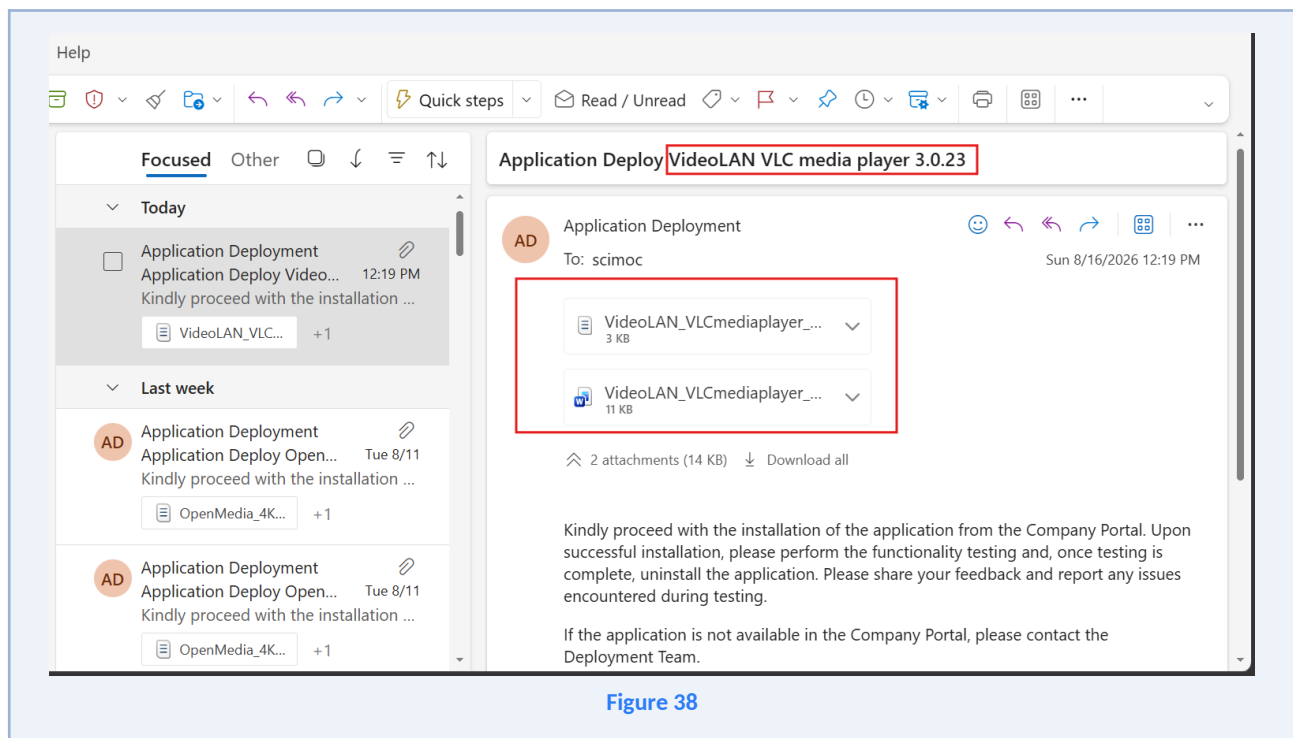


Figure 38

5.10 How to configure pre-install, post-install, pre-uninstall, post-uninstall and close apps if required

1. Once the application is added via the Add App window, the folder structure is created, e.g. C:\SwiftPack\Auto Package\App Settings\Smekal\Folder Compare.

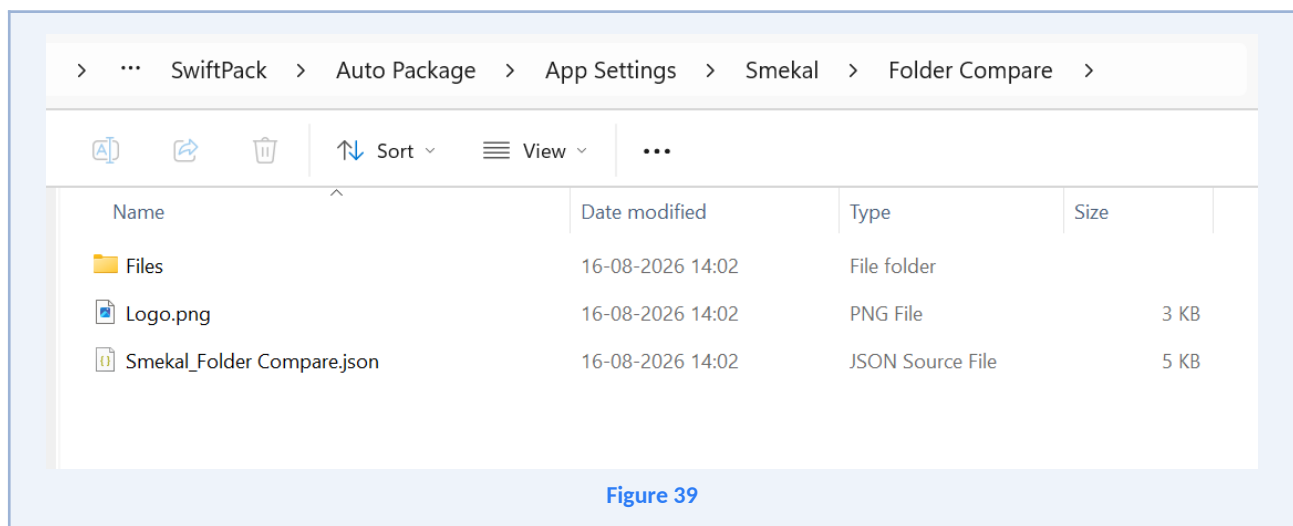


Figure 39

2. If you open the "Files" folder, it is empty. You can keep any files here, e.g. "Test.lic".

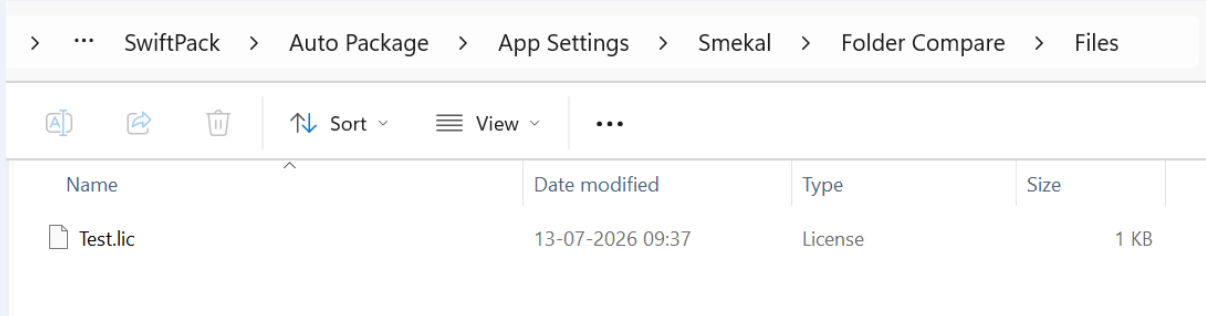


Figure 40

3. Copy Invoke-AppDeployToolkit.ps1 from the PSADT template to C:\SwiftPack\Auto Package\App Settings\Smekal\Folder Compare.

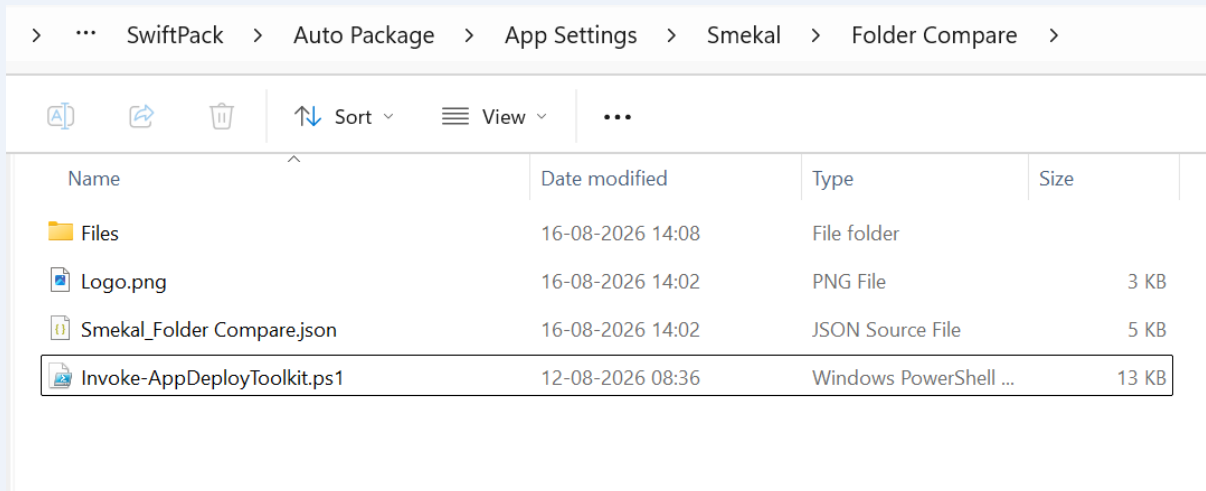


Figure 41

4. Edit Invoke-AppDeployToolkit.ps1 - if you want to close Notepad, add it here.

```

73 [System.Management.Automation.SwitchParameter]$SuppressRebootPassThru,
74 [Parameter(Mandatory = $false)]
75 [System.Management.Automation.SwitchParameter]$TerminalServerMode,
76 [Parameter(Mandatory = $false)]
77 [System.Management.Automation.SwitchParameter]$DisableLogging
78 )
79
80
81
82
83 #####
84 ## MARK: Variables
85 #####
86
87 # Zero-Config MSI support is provided when "AppName" is null or empty.
88 # By setting the "AppName" property, Zero-Config MSI will be disabled.
89 $adtSession = @{
90     # App variables.
91     AppVendor = ''
92     AppName = ''
93     AppVersion = ''
94     AppArch = ''
95     AppLang = 'EN'
96     AppRevision = '01'
97     AppSuccessExitCodes = @(0)
98     AppRebootExitCodes = @(0) # Example: @('excel', @{ Name = 'winword'; Description = 'Microsoft Word' })
99     AppProcessesToClose = @('notepad')
100     AppScriptVersion = '1.0.0'
101     AppScriptDate = '2026-01-14'
102     AppScriptAuthor = '<author name>'
103     RequireAdmin = $true
104
105     # Install Titles (Only set here to override defaults set by the toolkit).
106     InstallName = ''
107     InstallTitle = ''
108
109     # Script variables.
110     DeployAppScriptFriendlyName = $MyInvocation.MyCommand.Name
111     DeployAppScriptParameters = $PSBoundParameters
112     DeployAppScriptVersion = '4.1.8'

```

Figure 42

5. If you want to copy Test.lic to C:\Program Files\Smekal\Smekal Folder Compare after the installation, add the copy command, then save Invoke-AppDeployToolkit.ps1.

```

145
146 #####
147 ## MARK: Install
148 #####
149 $adtSession.InstallPhase = $adtSession.DeploymentType
150
151 ## Handle Zero-Config MSI installations.
152 if ($adtSession.UseDefaultMsi)
153 {
154     $ExecuteDefaultMSIplat = @{ Action = $adtSession.DeploymentType; FilePath = $adtSession.DefaultMsiFile }
155     if ($adtSession.DefaultMstFile)
156     {
157         $ExecuteDefaultMSIplat.Add('Transforms', $adtSession.DefaultMstFile)
158     }
159     Start-ADTMSIProcess @ExecuteDefaultMSIplat
160     if ($adtSession.DefaultMspFiles)
161     {
162         $adtSession.DefaultMspFiles | Start-ADTMSIProcess -Action Patch
163     }
164 }
165
166 ## <Perform Installation tasks here>
167
168 #####
169 ## MARK: Post-Install
170 #####
171 $adtSession.InstallPhase = "Post-$(($adtSession.DeploymentType))"
172
173 ## <Perform Post-Installation tasks here>
174 Copy-ADTFile -Path "$($adtSession.DirFiles)\Test.lic" -Destination "$env:SystemDrive\Program Files\Smekal\Smekal Folder Compare"
175
176 ## Display a message at the end of the install.
177 if (!$adtSession.UseDefaultMsi)
178 {
179     Show-ADTInstallationPrompt -Message 'You can customize text to appear at the end of an install or remove it completely for unattended installations.' -ButtonR
180 }
181
182
183
184 function Uninstall-ADTDeployment

```

Figure 43

6. During the package creation, it will copy Invoke-AppDeployToolkit.ps1 and Test.lic to the package folder, then append the install and uninstall command.
7. To test this, click on **EXE x64**.

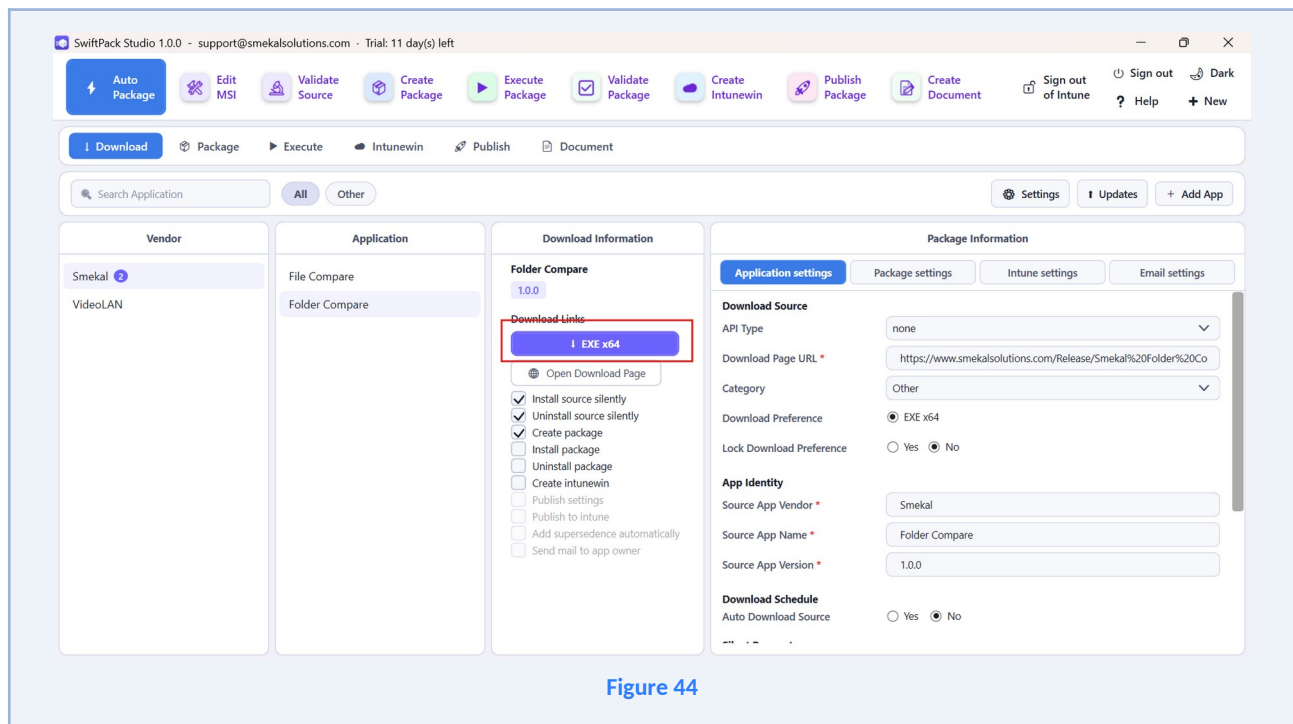


Figure 44

- Go to C:\SwiftPack\Auto Package\Packages\Smekal_FolderCompare_1.0.0_01\Package, open the "Files" folder, and check whether Test.lic is copied.

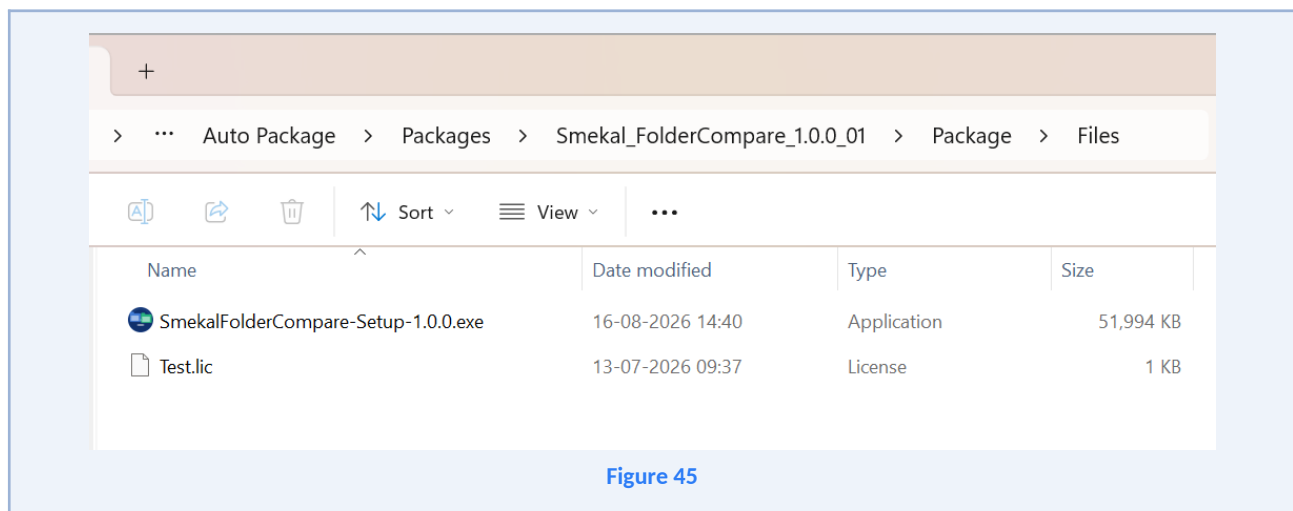


Figure 45

- Edit "Invoke-AppDeployToolkit.ps1".

```

85 #####
86
87 # Zero-Config MSI support is provided when "AppName" is null or empty.
88 # By setting the "AppName" property, Zero-Config MSI will be disabled.
89 $adtSession = @{
90     # App variables.
91     AppVendor = 'Smekal'
92     AppName = 'Folder Compare'
93     AppVersion = '1.0.0'
94     AppArch = ''
95     AppLang = 'EN'
96     AppRevision = '01'
97     AppSuccessExitCodes = @(0)
98     AppRebootExitCodes = @(1641, 3010)
99     AppProcessesToClose = @('notepad')
100     AppScriptVersion = '1.0.0'
101     AppScriptDate = '2026-08-16'
102     AppScriptAuthor = '<author name>'
103     RequireAdmin = $true
104
105     # Install Titles (Only set here to override defaults set by the toolkit).
106     InstallName = 'Smekal_FolderCompare_1.0.0_01'
107     InstallTitle = 'Smekal_FolderCompare_1.0.0_01'
108
109     # Script variables.
110     DeployAppScriptFriendlyName = $MyInvocation.MyCommand.Name
111     DeployAppScriptParameters = $PSBoundParameters
112     DeployAppScriptVersion = '4.1.8'
113 }
114
115 function Install-ADTDeployment
116 {
117     [CmdletBinding()]
118     param
119     (
120     )
121
122     #####
123     ## MARK: Pre-Install
124     #####
125
126 }
127
128 }
129
130 ## <Perform Installation tasks here>
131
132 # Install Smekal Solutions Smekal Folder Compare 1.0.0
133 Start-ADTProcess -FilePath "$($adtSession.DirFiles)\Smekal\FolderCompare-Setup-1.0.0.exe" -ArgumentList '/VERYSILENT' -WindowStyle "Hidden"
134
135 #####
136 ## MARK: Post-Install
137 #####
138 $adtSession.InstallPhase = "Post-$(($adtSession.DeploymentType))"
139
140 ## <Perform Post-Installation tasks here>
141
142 # ARP flags - Smekal Solutions Smekal Folder Compare 1.0.0
143 Set-ADTRegistryKey -LiteralPath "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\{82359FB4-AF01-4B5C-847D-C798F08C2B7A}_is1" -Name "NoRemov
144 Copy-ADTFile -Path "$($adtSession.DirFiles)\Test.Tic" -Destination "$env:SystemDrive\Program Files\Smekal\Smekal Folder Compare"
145
146 ## Display a message at the end of the install.
147 if (!$adtSession.UseDefaultMsg)
148 {
149     Show-ADTInstallationPrompt -Message 'You can customize text to appear at the end of an install or remove it completely for unattended installations.' -ButtonR
150 }
151
152 function Uninstall-ADTDeployment
153 {
154     [CmdletBinding()]
155     param
156     (
157     )
158
159     #####
160     ## MARK: Pre-Uninstall
161     #####
162     $adtSession.InstallPhase = "Pre-$(($adtSession.DeploymentType))"
163
164 }
165
166 }
167
168 ## <Perform Installation tasks here>
169
170 # Install Smekal Solutions Smekal Folder Compare 1.0.0
171 Start-ADTProcess -FilePath "$($adtSession.DirFiles)\Smekal\FolderCompare-Setup-1.0.0.exe" -ArgumentList '/VERYSILENT' -WindowStyle "Hidden"
172
173 #####
174 ## MARK: Post-Install
175 #####
176 $adtSession.InstallPhase = "Post-$(($adtSession.DeploymentType))"
177
178 ## <Perform Post-Installation tasks here>
179
180 # ARP flags - Smekal Solutions Smekal Folder Compare 1.0.0
181 Set-ADTRegistryKey -LiteralPath "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\{82359FB4-AF01-4B5C-847D-C798F08C2B7A}_is1" -Name "NoRemov
182 Copy-ADTFile -Path "$($adtSession.DirFiles)\Test.Tic" -Destination "$env:SystemDrive\Program Files\Smekal\Smekal Folder Compare"
183
184 ## Display a message at the end of the install.
185 if (!$adtSession.UseDefaultMsg)
186 {
187     Show-ADTInstallationPrompt -Message 'You can customize text to appear at the end of an install or remove it completely for unattended installations.' -ButtonR
188 }
189
190 }
191
192 function Uninstall-ADTDeployment
193 {
194     [CmdletBinding()]
195     param
196     (
197     )
198
199     #####
200     ## MARK: Pre-Uninstall
201     #####
202     $adtSession.InstallPhase = "Pre-$(($adtSession.DeploymentType))"
203
204 }
205
206 }
  
```

Figure 46

Note

For more details, please refer to the document **"SwiftPack Studio - Auto Package - User Guide.docx"**.

End of guide - SwiftPack Studio, Auto Package User Guide and Tutorial.