InstallScout – technical reference

Version **1.9.2**. Short path: quickstart-en.md. Find the silent switches. The user workflow is in guide.md. This file covers engines, the command line, and Microsoft Graph. PDF: InstallScout-reference-en.pdf.

InstallScout does not run the installer. It reads the file, builds a silent command, wraps PSADT 4.1.8 as `.intunewin`, and creates a Win32 app.

1. Engines

An engine is recognized from byte markers in the file, PE section names, and sibling files. The highest score wins. `base_confidence` below is the starting value before extra evidence is added.

Wrappers (7-Zip SFX, WinRAR SFX, IExpress, Self Extractor, unknown EXE) are unpacked locally when the outer file has no reliable silent line. Clean engines such as Inno, NSIS, Wacom, and DDPM are not unpacked. InstallScout ships a bundled 7-Zip Extra `7za` for 7z payloads, so a system 7-Zip install is not required.

`/install` is not added. It is already the default for WiX Burn. `/norestart` is not added either. Inno Setup keeps its own `/NORESTART`.

`{file}` is the quoted installer path. `{product_code}` is the MSI `ProductCode`. `{app}` is the installed folder PSADT knows at uninstall time.

Id	Name	Silent install	Alternative	Uninstall	Base
msi	Windows Installer (MSI)	msiexec /i "{file}" /qn	msiexec /i "{file}" /qb	msiexec /x "{product_code}" /qn	0.97
inno	Inno Setup	"{file}" /VERYSILENT /SUPPRESSMSGBOXES /NORESTART /SP-	"{file}" /SILENT /NORESTART	"{app}\unins000.exe" /VERYSILENT /NORESTART	0.95
wixburn	WiX Burn	"{file}" /quiet	"{file}" /passive	"{file}" /uninstall /quiet	0.93
ddpm	Dell Display and Peripheral Manager	"{file}" /Silent	"{file}" /Silent /CreateDebugLog="%TEMP%\DDPM-install.log"	"{file}" /uninst /Silent	0.93
advanced_installer	Advanced Installer	"{file}" /exenoui /qn	"{file}" /exenoui /passive /qn	"{file}" /exenoui /x // /qn	0.92
nsis	NSIS	"{file}" /S		"{app}\uninstall.exe" /S	0.90
wacom	Wacom Tablet Driver	"{file}" /s	"{file}" /s /opt nowdc	"{file}" /s /u	0.90
msix	MSIX / AppX	Add-AppxPackage -Path "{file}"	Add-AppxProvisionedPackage -Online -PackagePath "{file}" -SkipLicense		0.90
self_extractor	Self Extractor	harvested, otherwise the inner setup		/uninst or /uninstall when present	0.88
bitrock	BitRock InstallBuilder	"{file}" --mode unattended --unattendedmodeui none			0.88
install4j	install4j	"{file}" -q			0.85
installaware	InstallAware	"{file}" /s			0.80
sevenzip_sfx	7-Zip SFX	"{file}" -y			0.80
qt_installer	Qt Installer Framework	"{file}" --silent --accept-licenses			0.80
installshield	InstallShield	"{file}" /s /v"/qn"	"{file}" /s /sms		0.78
winrar_sfx	WinRAR SFX	"{file}" /S			0.75

squirrel	Squirrel	"{file}" --silent			0.72
clickonce	ClickOnce	no reliable silent line			0.70
iexpress	IExpress	"{file}" /Q			0.70
wise	Wise Installer	"{file}" /s			0.65

Rules that differ from the template

- **NSIS.** /S must be capital. /D= must be last and unquoted.
- **Inno.** /S is not silent here. /NORESTART is Inno's own switch and stays.
- **InstallShield.** Newer wrappers pass /qn through /v"/qn". Older InstallScript may need a recorded setup.iss (/r, then /s /f1).
- **WiX Burn.** /quiet is the silent switch. /install is omitted because install is the default.
- **Advanced Installer.** /S and /VERYSILENT typically produce *Invalid command line*.
- **DDPM.** /Silent is not InstallShield /s /v"/qn".
- **Wacom.** The download is a 7-Zip SFX, but silent is /s on the EXE, not -y.
- **Self Extractor.** An outer /sAll, --silent, or /silent is kept. The Adobe-style chain becomes "file" /sAll /rs /rps /msi /quiet when /sAll or /rps is in the file. With no outer silent flag, the inner MSI/EXE is unpacked.
- **Squirrel.** Usually per-user. Context becomes User.
- **ClickOnce.** No silent template. Packaging for Intune is not a reliable target.
- **MSIX.** Detection uses the package name, not a product code.

Suggested switches

At most 10. File flags come first (/analytics no, /sso no, and other name yes|no), then optional switches for the engine. /install and /norestart are not offered. A tick adds the switch to the silent line and invalidates the local .intunewin, so the package must be built again.

Engine	Offered in addition to flags found in the file
MSI	ALLUSERS=1, REBOOT=ReallySuppress, /l*v "%TEMP%\install.log"
Inno	/NORESTARTAPPLICATIONS, /LOG="%TEMP%\install.log", /MERGETASKS=!desktopicon
NSIS	/NCRC
InstallShield	/f2"%TEMP%\install.log"
WiX Burn	/log "%TEMP%\install.log"
Advanced Installer	/exenoupdates
Wise	/l="%TEMP%\install.log"
DDPM	/TelemetryConsent=false, /HeadlessMode=true, /CreateDebugLog=%TEMP%\DDPM.log, /TurnOffCA
Wacom	/opt nowdc

Detection and the engine

Detection.ps1 writes to STDOUT and exits 0 when the app is installed. Empty STDOUT means not installed. The script does not throw.

Source	Match
MSI ProductCode, or a GUID in the uninstall command	That uninstall key, and version >=
EXE without a product code	DisplayName, Publisher, and version >=
MSIX	Package name, and version when known

A trailing Installer, Setup, Bootstrapper, or Driver word may be absent from DisplayName. Logi Options+ Installer matches Logi Options+, and Wacom Tablet Driver matches Wacom Tablet. The uninstall

key name counts too. A key named as the product still counts when DisplayName, publisher, and version are missing, as long as the key has some other value. A value that is present must still match. The publisher may differ when DisplayName is the product name plus a version and the name is not a short word. `WinRAR` matches `WinRAR 6.00 (64-bit)` even when the file says Alexander Roshal and the key says win.rar GmbH. `Git` still requires the right publisher. A leftover generic word such as Advanced is not used. Legal suffixes are ignored, so `Logitech, Inc.` matches `Logitech`. Versions are compared numerically. The same version and newer count. Older does not. A hyphen revision such as `6.4.14-1` is satisfied by both `6.4.14-1` and `6.4.14`.

Keys read: `HKLM` and `HKCU` under `SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall`, in both the 32-bit and 64-bit views. That still holds when the script itself runs as 32-bit.

2. CLI

`InstallScoutPortable.exe` with no arguments opens the GUI. A frozen EXE also opens the GUI unless a CLI flag is set. From source, `python -m installscout` behaves the same way.

Any of these flags forces the CLI even without `--cli`: `--json`, `--out`, `--psadt`, `--intune`, `--online`, `--sihq`, `--publish-intune`, `--switches`, `--icon`, `--supersede`, `--supersede-id`, `--outdated`, `--winget`.

Flag	Meaning
`paths`	Files or folders. Suffixes: `.exe`, `.msi`, `.com`, `.msix`, `.appx`, `.msixbundle`, `.appxbundle`
`--gui`	Open the GUI even when paths are present
`--cli`	Force text output
`--json`	Write the analysis as JSON on stdout
`--out PATH`	Write the same JSON to a file
`--psadt DIR`	Write the PSADT source, without `.intunewin`
`--intune DIR`	Write `.intunewin`, PSADT source, `Intune.txt`, and `Detection.ps1`
`--no-copy-installer`	Leave the installer in place. The package points at the original path
`--publish-intune`	Create the Win32 app. Requires `--intune`
`--icon PATH`	PNG, JPEG, GIF, ICO, or BMP for `largeIcon`. Without the flag, the icon comes from the EXE, the MSI, or a sibling file
`--supersede update|replace`	Supersede older Win32 apps with a related name. At most 10
`--supersede-id GUID`	Repeatable. Use known app IDs instead of a name search
`--no-supersede`	With `--winget --publish-intune`: do not supersede earlier versions. Otherwise `update` is the default there
`--outdated`	Compare the tenant's Win32 apps with the newest WinGet or Chocolatey version. Signs in. Changes nothing
`--only TEXT`	With `--outdated`: products whose name contains the text
`--map "Name=Publisher.Package"`	With `--outdated`: confirm a match. Use `Name=choco:id` for Chocolatey. Repeatable. An empty ID forgets the match
`--winget ID`	Download the newest version of a WinGet package (or `choco:id`), verify its SHA256, analyze it, and build the package. Pair with `--intune` and `--publish-intune`
`--winget-version V`	With `--winget`: this version instead of the newest
`--arch x64|x86|arm64`	With `--winget`: installer architecture. Default `x64`
`--close-apps a,b`	With `--winget`: programs to ask the user to close. The install command becomes `-DeployMode Auto`
`--upgrade-test`	With `--winget`: install the previous version and then the new one in Windows Sandbox, and check detection. Stops with exit code 5 unless the upgrade passes
`--from-version V`	With `--upgrade-test`: start from this version
`--tenant ID`	Empty means `organizations`
`--client-id ID`	Empty means Microsoft Graph PowerShell
`--online`	WinGet and Chocolatey. Only the product name and vendor are sent
`--sihq`	Open a name search on silentinstallhq.com. The command is

	not overwritten
`--switches TEXT`	Append arguments to the silent line for every file in the run
`--lang en|da`	Language of CLI text. The GUI keeps its own choice
`-r` / `--recursive`	Search folders recursively. This is the default
`--no-recursive`	Only the given folder
`--version`	Print the version and exit

Exit code `0` is success. Exit code `2` is used when no supported file was found, `--publish-intune` is missing `--intune`, `--icon` cannot be read, a WinGet package cannot be read or packaged, or Windows Sandbox is missing. Exit code `3` means the download did not match the manifest's SHA256. Exit code `4` means the sandbox closed without a result. Exit code `5` means the upgrade test did not pass.

On startup, one anonymous call goes to Aptabase in the EU (`eu.aptabase.com`) with the version and language. File names and paths are not sent. `INSTALLSCOUT_USAGE=0` turns the call off.

```
.\InstallScoutPortable.exe --cli .\setup.exe
.\InstallScoutPortable.exe --cli .\setup.exe --online
.\InstallScoutPortable.exe --cli .\setup.exe --switches "/LOG=C:\logs\app.log" --json --out
.\analysis.json
.\InstallScoutPortable.exe --cli .\setup.exe --intune C:\out\intune
.\InstallScoutPortable.exe --cli .\setup.exe --intune C:\out\intune --publish-intune --icon
.\logo.png --supersede update
.\InstallScoutPortable.exe --cli .\setup.exe --intune C:\out\intune --publish-intune --
supersede replace --supersede-id <app-guid>
```

`--json` is a list of analysis objects: path, engine, confidence, commands, harvested switches, `option_flags`, VersionInfo, MSI properties, and notes. The first command with purpose `silent_install` is the recommended line.

The install line written to the portal is always PSADT, not the raw silent command:

```
.\Invoke-AppDeployToolkit.exe -DeploymentType Install -DeployMode Silent
.\Invoke-AppDeployToolkit.exe -DeploymentType Uninstall -DeployMode Silent
```

When a package lists programs to close (`--close-apps` or the Update check window), the install line is `-DeployMode Auto` instead, and the app also gets return code `1602` as `retry`:

```
.\Invoke-AppDeployToolkit.exe -DeploymentType Install -DeployMode Auto
```

`setupFilePath` is `Invoke-AppDeployToolkit.exe`.

3. Graph

The Win32 fields `detectionRules`, `displayVersion`, and `maxRunTimeInMinutes` exist on Graph **beta**, not on v1.0. App calls therefore go to `https://graph.microsoft.com/beta`. `GET /me` for whoami goes to `https://graph.microsoft.com/v1.0`.

Sign-in

OAuth 2.0 authorization code with PKCE. Device code is not used. A local HTTP listener on `http://localhost:<port>/` receives the code. The browser opens in the Edge work profile when that profile can be recognized.

Authority	`https://login.microsoftonline.com/{tenant}/oauth2/v2.0/authorize and /token`
Default tenant	`organizations`
Default client	`14d82eec-204b-4c2f-b7e8-296a70dab67e` (Microsoft Graph PowerShell, public client)

Scopes	`DeviceManagementApps.ReadWrite.All`, `User.Read`, `offline_access`, `openid`, `profile`
Token file	`%LOCALAPPDATA%\InstallScout\graph.json`, Windows DPAPI for the current user (`ISC1`). Bound to the tenant and Client ID. Plaintext is deleted
Refresh	Refresh token. A 401 is retried once after refresh. An expired session deletes the token file. A different tenant or Client ID does not reuse the session

A custom Client ID must be a public client with redirect `http://localhost` and the same permission. InstallScout does not create an app registration. The signed-in user must be able to create Win32 apps. The Entra role Intune Administrator, or the Intune role Application Manager, includes that. The tenant still needs admin consent for `DeviceManagementApps.ReadWrite.All`.

Create and upload

1. `POST /deviceAppManagement/mobileApps` with `#microsoft.graph.win32LobApp`.
2. `POST .../microsoft.graph.win32LobApp/contentVersions`.
3. `POST .../contentVersions/{version}/files` with the encrypted size and the unencrypted size.
4. Poll the file until `azureStorageUri` is present.
5. `PUT` the encrypted blob in 6 MiB blocks.
6. `POST .../files/{id}/commit` with `fileEncryptionInfo`.
7. `PATCH` the app with `committedContentVersion`.

If upload or commit fails, the new app is deleted. Supersedence runs afterwards and does not delete the app if it fails.

Encryption matches Microsoft Win32 Content Prep: AES-CBC, a 256-bit key, HMAC-SHA256, `ProfileVersion1`, digest `SHA256`. The key, IV, MAC, and digest are sent at commit. The installer binary is not sent during the online lookup, only inside the encrypted `.intunewin`.

App payload

Field	Value
`displayName`	The name from the dialog, otherwise `{product} {version}`. At most 500 characters
`publisher`	Vendor, or the product name when the vendor is Unknown. At most 500
`displayVersion`	Analyzed version. At most 50 characters
`description`	Engine, context, and detection summary. At most 4000
`installCommandLine` / `uninstallCommandLine`	PSADT Silent, as above
`setupFilePath`	`Invoke-AppDeployToolkit.exe`
`applicableArchitectures`	`x64`, `x86`, or `arm64`
`minimumSupportedWindowsRelease`	`Windows10_22H2`
`installExperience.runAsAccount`	`system` or `user`
`installExperience.deviceRestartBehavior`	`suppress` (No specific action)
`installExperience.maxRunTimeInMinutes`	`60`
`allowAvailableUninstall`	`true`
`detectionRules`	One PowerShell script, `enforceSignatureCheck` false, `runAs32Bit` false, body base64 UTF-8
`largeIcon`	Omitted when there is no logo. Otherwise `mimeContent` as PNG, JPEG, or GIF
`returnCodes`	`0` success, `1707` success, `3010` softReboot, `1641` hardReboot, `1618` retry

The logo comes from an EXE/DLL/COM resource, from the MSI Icon table (ARPPRODUCTICON, otherwise a shortcut, otherwise the only icon), from `logo.png` / `icon.ico` next to the file, or from the file the user picks. Several icons with no ARPPRODUCTICON and no shortcut are left unused.

Supersedence

Search is `GET /deviceAppManagement/mobileApps` with `$filter=startswith(displayName,'…')` and `$top=50`, at most 5 pages. The whole catalogue is not downloaded. The search term is the product name, not the edited Intune display name.

`POST /deviceAppManagement/mobileApps/{id}/updateRelationships` with `#microsoft.graph.mobileAppSupersedence`:

supersedenceType	Meaning
`update`	Install the new app without uninstalling the old one first
`replace`	Uninstall the old app, then install the new one

At most 10 targets. The GUI selects nothing automatically. The CLI with `--supersede` selects apps whose names look like the product.

Portal link after success:

```
https://intune.microsoft.com/#view/Microsoft_Intune_Apps/SettingsMenu/~/0/appId/{appId}
```

4. Windows Sandbox

Try in Sandbox is an optional GUI button. There is no CLI flag. It appears after **Analyze**, and only when `WindowsSandbox.exe` is installed. Packaging and upload stay available while the test runs. The result comes back in this window when the test finishes.

The button runs the current silent command inside a new Windows Sandbox, then the same `Detection.ps1` that upload sends to Intune. Empty detection output means not installed. The sandbox shows a message with the install exit code and that detection result. InstallScout reads the same text back into this window.

The command must contain the installer path. A package folder (the setup plus MSI, CAB, prerequisites, or a subfolder the installer needs) is mapped read-only at `C:\InstallScout\Installer`. A file directly in Downloads, Desktop, Documents, the OneDrive root, Temp, or the user profile is copied into the sandbox work folder, so the parent folder is not shared. That still applies when the folder has been redirected. A subfolder there, for example `Downloads\PSADT - CutePDF`, is still a package folder. The work folder is writable. Networking is enabled. On a PC with free RAM, the sandbox gets up to 8 GB of memory instead of the 4 GB default ceiling. Only one Windows Sandbox can run at a time. Closing the sandbox discards it. If you close it before a result comes back, the button turns back on so you can change the command and run it again. Nothing is installed on this PC.

The sandbox also lists the programs the installer added: new `.exe` files under Program Files and `%LOCALAPPDATA%\Programs`, written to `processes.txt`. The Update check window uses that list as suggestions for programs to close.

Upgrade test

The Update check window and `--upgrade-test` use a second report script. It installs the old version (`install_old.cmd`, from `payload_old`), runs the new `Detection.ps1`, installs the new version (`install_new.cmd`, from `payload_new`), and runs detection again. The verdict:

Verdict	Meaning
`ok`	Detection said not installed with the old version and installed with the new one
`old_failed`	The old version did not install
`detection_lax`	Detection already said installed with the old version. Intune

	would never update
`new_failed`	The upgrade failed
`detection_missing`	Detection does not find the new version. Intune would reinstall it forever
`incomplete`	The sandbox closed or the report is partial

5. Third-party updates (beta)

Source: the WinGet community repository `microsoft/winget-pkgs`. Nothing but the product name, the publisher, and a package ID leaves the PC.

- **Versions.** `GET api.github.com/repos/microsoft/winget-pkgs/contents/manifests/<letter>/<Id parts>`. A package ID such as `Git.Git` maps to `manifests/g/Git/Git`. The list is cached for six hours. `GITHUB_TOKEN` is sent when set.
- **Manifests.** `raw.githubusercontent.com/.../<version>/<Id>.installer.yaml` and `<Id>.locale.en-US.yaml`, with the single-file `<Id>.yaml` as fallback. Versions sort by numeric parts, so 2.9 is older than 2.10, and a leading `v` is ignored.
- **Match, WinGet.** `https://cdn.winget.microsoft.com/cache/source.msix` is the index `winget` itself uses: a zip with a SQLite database of every package ID, name, and publisher. It is downloaded at most once a day (about 20 MB) and a short list is kept in `winget_catalog.json` next to the preferences. The Intune app's name and publisher are matched on this PC. A stale copy is used when the download fails. Without the catalogue, `winget.exe search` is the fallback when installed.
- **Match, Chocolatey.** Only when WinGet has nothing. `GET community.chocolatey.org/api/v2/Search()` with the product name, at most one call every 0.4 s. A hit counts only when the title has the same words as the product. Newest version: `Packages()` with `IsLatestVersion`. Older versions: `FindPackagesById()`, stable ones only, 40 at most.
- **Confirming.** A match stays a guess until confirmed with **Set package ID…** or `--map`. Confirmed IDs are stored in the preferences under `winget_map`. Chocolatey IDs are written `choco:<id>`.
- **Chocolatey installer.** The `.nupkg` is downloaded (at most 8 MB). Its `tools/chocolateyInstall.ps1` must call `Install-ChocolateyPackage` with `fileType` exe, msi, or msix, an `https` `url` or `url64bit`, and a 64 hex digit `checksum`. Simple variables (`$version`, `${version}`, `$env:ChocolateyPackageVersion`) are resolved. Anything computed in code is not. A package with only a dependency (`7zip` needs `7zip.install`) is followed one step. A larger package, a zip, or an installer inside the package is refused with the reason.
- **ZIP.** A manifest with `InstallerType: zip` is used when `NestedInstallerType` is an installable type and `NestedInstallerFiles` names the setup (`RelativeFilePath`, no `..`, no drive). The ZIP is downloaded and its SHA256 checked first. It is then unpacked to `<version>\package\` (at most 100,000 files and 8 GB, and no path may leave that folder), the ZIP is deleted, and the named setup is the installer. Its folder is copied whole into `Files`. A ZIP with a portable setup is refused.
- **Installer.** The first `https` installer that fits architecture and scope, of type MSI, WiX, Burn, Inno, NSIS, EXE, MSIX, or APPX. The download is refused without a SHA256 and deleted when the hash does not match. Files go to `<folder>\_winget\<Id>\<version>\`.
- **Package.** The manifest overrides the display name, publisher, version, silent switches, and MSI product code. Detection uses the Apps & features name from the manifest. A product code is used only when it is a GUID.
- **Inventory.** `GET /deviceAppManagement/mobileApps` with a Win32 filter, without `$select`, so Graph returns `displayVersion`. Read only. Versions are compared by their leading dotted number. A product is *outdated*, *current*, *ahead*, *unknown*, *unmatched*, or *error*.

- **Supersedence.** The tenant's apps are grouped by name without version. Everything older than the new app is added as `update`, at most 10, newest first.
- **Close programs.** With names set, PSADT's `Show-ADTInstallationWelcome` gets the names in `AppProcessesToClose`, `DeferDays = 3`, `DeferRunInterval = 4h`, and `CloseProcessesCountdown = 600`. The install line is `-DeployMode Auto`. PSADT exits with `1602` on a deferral. For these packages, Graph `returnCodes` has `1602` as `retry`.