LOGIC UNKNOWN APS

AEMS (Application Elevation Management Service)

Administrator documentation - Applies to AEMS 1.0.8.0

This documentation describes AEMS 1.0.8.0, released 2026-11-20.

Contents:

1. Overview
2. Install and deploy
3. Configure
4. Licensing
5. Troubleshooting
6. Release notes
7. Security model

Online version: https://docs.logic-unknown.com/aems/
Contact: https://logic-unknown.com/contact/

AEMS overview

> **Applies to AEMS 1.0.8.0**
>
> This documentation describes AEMS 1.0.8.0 (service 0.1.9.0, driver 1.0.8.0, Allow-list Tool 1.3.0.0), released 2026-11-20. Install 1.0.8.0. Text marked "from 1.0.8.0" does not apply to earlier versions. Where the text says "from 1.0.6.0", it applies to 1.0.7.0 and 1.0.8.0 as well.

AEMS (Application Elevation Management Service) from Logic Unknown ApS lets a standard (non-admin) user on a Windows kiosk device run the specific applications you approve with administrator rights, without giving the user local administrator rights and without a UAC prompt. Every elevation is decided by an allow-list in the registry and written to the Windows event log.

What it does

- You approve applications in an allow-list under `HKLM\SOFTWARE\Logic Unknown\AEMS\Applications`. An entry names an executable by its path or, from 1.0.6.0, by a wildcard path pattern, and can pin it to a SHA-256 hash or to the publisher of its Authenticode signature.
- A path without wildcards is matched as a **prefix** of the launched path. Always write the full path including the file name: a folder path approves every program below it.
- When the kiosk user starts an approved application, AEMS relaunches it elevated in the user's session. There is nothing for the user to click.
- Everything that is not on the allow-list starts as it always does, unelevated.
- Every elevation is logged to the Windows **Application** event log under the source `AEMS`.
- There is no portal, no cloud service and no online dependency. Every decision is made on the device from the allow-list, and license keys are checked offline on the device. Signature checks use only the revocation information already cached on the device; AEMS does not fetch it from the network.

How it works

AEMS is a headless Windows service plus a kernel driver, installed from one MSI:

Component	Role
`AEMSvc` (`AEMSvc.exe`)	The LocalSystem service. Reads the allow-list from the registry, sends it to the driver, decides every caught launch and relaunches approved applications elevated into the active console session. No window, tray icon, shell menu or dialog.
AEMS kernel driver (`AEMSDriver.sys`)	Catches process creation for allow-listed paths and provides the elevation capability. It is signed by Microsoft. Its device can be opened by SYSTEM only; in addition the driver accepts a caller only when its image path ends in `\Program Files\AEMS\AEMSvc.exe` (a path check, not a signature check).

The short form of the decision model: **the driver catches, the service decides, the driver executes.** From 1.0.8.0 the driver holds a caught launch and AEMS suspends it, with the user's own token, until the service has decided; then it is replaced by the elevated relaunch or resumed unchanged. If the service is stopped, top-level launches run unelevated. If the service ends unexpectedly while a launch is suspended, the driver ends that process. Child processes of a governed (elevated) application do not start while the service is stopped. The full model is on the Security model page.

Where to start

1. Install and deploy - the MSI, silent install, Intune and Configuration Manager.
2. Licensing - from 1.0.6.0 AEMS elevates only with a valid license key from Logic Unknown ApS.
3. Configure - approve your first application and read the settings reference.
4. Troubleshooting - common problems and the events you are most likely to see.

What AEMS does and does not do

- It elevates **only the specific applications you allow-list** - not arbitrary programs.
- Child processes are **not** elevated by default: an allow-listed app's children run non-elevated unless that app explicitly opts in with `AllowChildElevation=1` (and even then only children that are themselves opted-in continue the chain). See Approve applications for the opt-in model and its security warning.
- From 1.0.6.0 an opted-in app can start its children elevated **in place**, as real child processes with their output and exit code (`ChildInPlace`: direct children, or the whole tree up to 8 levels). This is meant for wrapper programs that call other programs. The children must lie in a folder that standard users cannot change. See Children in place.

- From 1.0.6.0 elevated programs use the machine's drive letters, not the user's own mapped or `subst` drive letters. See Drive letters of elevated programs.
- Every elevation is logged to the Windows Application event log (source `AEMS`). There is no network, portal or online dependency.

Requirements

Item	Requirement
Windows	64-bit (x64) Windows 10 22H2 (build 19045) or Windows 11; see Supported Windows versions.
Device type	A kiosk with **one interactive user**. AEMS elevates into the active console session; multi-user Remote Desktop Services (several interactive sessions at once) is out of scope.
Driver signing	On stock Windows 10 and 11 the AEMS driver must be signed by Microsoft to load. The AEMS MSI from Logic Unknown ApS contains that driver.
Installation	Administrator rights on the device (interactive or silent MSI install, for example through Intune, Configuration Manager or your RMM tool).
Version	Install AEMS 1.0.8.0 (service 0.1.9.0, driver 1.0.8.0, Allow-list Tool 1.3.0.0). It replaces 1.0.7.0 and earlier; a device on 1.0.5.0 or earlier is upgraded directly to 1.0.8.0 in one run, then restart the device once; see the release notes and Upgrade and repair.
License	From AEMS 1.0.6.0 a license key from Logic Unknown ApS is needed for elevation; see Licensing.
Network	None. AEMS has no cloud, portal or online dependency; license keys are checked offline and against the device clock.

Supported Windows versions

Windows 10 and Windows 11, **x64 only**, from build **10.0.19045** (Windows 10 22H2) and every later build. AEMS 1.0.6.0, including children in place, was measured on Windows 11 build 26200 (25H2) and on build 26300. The driver is validated up to build 26200. On build 26300 AEMSvc logs warning 2231 (the build is newer than the validated ones and the driver's start-up self-check passed), and elevation and children in place worked there; 26300 is not a validated build.

Windows 10 22H2 is supported by the driver's build rule, but children in place were not validated on Windows 10.

AEMS 1.0.8.0 (driver 1.0.8.0, which holds the start of an allow-listed program until AEMS has decided, see Programs that finish very quickly) was measured on Windows 11 build 26200 (25H2) and on build 26300. Windows 11 24H2 (build 26100) has the same kernel code base as build 26200 and is covered by that measurement. Windows 11 23H2 (build 22631) is not validated, and Windows 10 22H2 was not in the scope of this validation. On build 26300 AEMS 1.0.8.0 still logs warning 2231: the driver's validated build limit stays 26200. On every build AEMS checks at every launch that the suspend took effect; if it did not, that launch is handled as in 1.0.7.0 and warning 2235 is logged.

- **Below build 19045** the driver refuses to load. This includes Windows 10 Enterprise LTSC 2021 (build 19044). `sc start AEMSDriver` then fails and AEMSvc logs 3005.
- **There is no upper build limit.** The driver depends on internal Windows kernel structures and checks them itself at every start. On a build newer than the validated ones it loads, and elevation works only if that self-check passes: then AEMSvc logs warning 2231 at start, which names the build and says that the self-check passed. If it does not pass, AEMSvc logs error 3010 and nothing is elevated until an AEMS release supports that build.
- **Guard on a build newer than the validated ones.** On such a build AEMSvc writes the REG_DWORD value `BindAttemptBuild` (the build number) under `HKLM\SOFTWARE\Logic Unknown\AEMS-DriverBindGuard` before it connects to the driver, and removes it when the connection call has returned. If the device stopped during that call (for example a crash), the value is still there at the next start: AEMSvc then does not connect to the driver on that build, logs error **3212**, and elevation stays off while the service keeps running. What to do: install an updated AEMS version. Only after you have checked the cause, set the REG_DWORD value `ClearBindAttempt` = 1 under the same key and restart the AEMS service to try once more (AEMS deletes the value after use). Do not create or edit `BindAttemptBuild` yourself. A value written on another build is ignored and removed, and on builds up to the validated one the guard is not active. If AEMSvc cannot write the value, it does not connect and elevation stays off (error **3214**; retried); if it cannot remove it after the connection, the next start does not connect (error 3214, then 3212): check the permissions of the key. From 1.0.8.0 the uninstall removes the key when it is empty and keeps it when it holds `BindAttemptBuild`; see The driver connection guard key after an uninstall.
- **Policy switch.** The REG_DWORD value `DisableDriverBindOnUnvalidatedBuild` = 1 under `HKLM\SOFTWARE\Policies\Logic Unknown\AEMS` (or `HKLM\SOFTWARE\Logic Unknown\AEMS`) stops AEMS from connecting to its driver on any build newer than the validated ones: error **3213**, elevation off. Use it to keep devices that receive a new Windows feature update before a matching AEMS release unelevated instead of untested. Remove the value to allow the connection again.
- Windows Server is not supported. AEMS targets single-interactive-session kiosk devices; multi-user RDS is out of scope.

Install and deploy AEMS

> **Applies to AEMS 1.0.8.0**
>
> This documentation describes AEMS 1.0.8.0 (service 0.1.9.0, driver 1.0.8.0, Allow-list Tool 1.3.0.0), released 2026-11-20. Install 1.0.8.0. Text marked "from 1.0.8.0" does not apply to earlier versions. Where the text says "from 1.0.6.0", it applies to 1.0.7.0 and 1.0.8.0 as well.

AEMS ships as one MSI (`AEMS.msi`) that contains the service and the kernel driver. This page covers a manual or silent install, deployment with Microsoft Intune or Configuration Manager, baking your allow-list into the MSI, upgrades (including the move from AEMS 1.0.5.0 or earlier), repair and uninstall.

Typical order for a new deployment:

1. Get a license key from Logic Unknown ApS; see Licensing.
2. Install the MSI with the key on a pilot device and approve one application (Configure).
3. Deploy the MSI and your allow-list to the pilot ring, then to all kiosks.

Install the MSI

AEMS ships as a single MSI (`AEMS.msi`) containing the service (`AEMSvc.exe`) and the kernel driver (`AEMSDriver.{inf,cat,sys}`).

```
:: interactive
msiexec /i AEMS.msi

:: silent, for deployment tools (Intune / SCCM / your RMM)
msiexec /i AEMS.msi /qn

:: silent, with the license key (1.0.6.0 and later; see the Licensing page)
msiexec /i AEMS.msi /qn LICENSEKEY=<key>
```

AEMS 1.0.6.0 and later elevate only with a valid license key; see Licensing. A **new installation** without a key does not elevate anything: allow-listed programs start unelevated and AEMSvc logs error 3013 (`missing`). Only a device that ran AEMS 1.0.5.0 or earlier gets the 30-day upgrade grace.

The MSI: - installs `AEMSvc.exe` to `C:\Program Files\AEMS\` (the driver's device can be opened by SYSTEM only; in addition the driver accepts a caller only when its image path ends in `\Program Files\AEMS\AEMSvc.exe` - a path check, not a signature check), - installs + registers the `AEMSDriver` kernel-driver service from the INF, - creates the `AEMSvc` service (LocalSystem, auto-

start) and registers the `AEMS` Windows Event Log source, - seeds `HKLM\SOFTWARE\Logic Unknown\AEMS` with an empty `Applications` subkey.

Publisher: **Logic Unknown ApS** (logic-unknown.com). Apps and features shows that publisher and the AEMS icon; the code-signing certificate subject reads "Logic Unknown Aps". Since 1.0.6.0 the registry key is `HKLM\SOFTWARE\Logic Unknown\AEMS`; upgrading from 1.0.5.0 or earlier is covered in Upgrading from 1.0.5.0 or earlier.

> **Driver signing:** on stock Windows 10/11 the driver must be signed by Microsoft to load. If the `AEMSDriver` service fails to start with `STATUS_INVALID_IMAGE_HASH (0x241)`, the shipped driver wasn't Microsoft-signed - see Driver Diagnostics.

Deploy with Intune or Configuration Manager

Deploy `AEMS.msi` as an Intune **Win32 app** or as a Configuration Manager application with an MSI deployment type. AEMS installs per machine and runs as LocalSystem, so install it in the system context.

Setting	Value
Install command	`msiexec /i AEMS.msi /qn` - deliver the license key as the policy value `License` (Group Policy, or an Intune configuration or script that writes the registry value) or bundle it into a customer MSI (see below). `msiexec /i AEMS.msi /qn LICENSEKEY=<key>` also works, but puts the key in the install command.
Uninstall command	`msiexec /x {ProductCode} /qn` with the ProductCode of the installed AEMS MSI
Install behaviour	System
Device restart	Normally not needed: a new installation starts the driver, and an upgrade stops the old driver and starts the new one. For an upgrade from 1.0.5.0 or earlier, restart the device once after the upgrade (see Rollout order). If AEMSvc logs 2236, 2014, 3011 or 2220 after an upgrade, the old driver is still loaded: restart the device (see Upgrade and repair). If the old driver cannot be unloaded the upgrade fails and asks for a restart. A repair can return 3010 (restart required).
Detection rule	The MSI ProductCode of the AEMS version you deploy, with the MSI product version check "Greater than or equal to" that version. In a script rule or in Configuration Manager: the product version for the AEMS UpgradeCode, optionally combined with "service `AEMSvc` exists".

> **Do not use a file or registry detection rule.** Do not detect AEMS with a file-exists rule (`C:\Program Files\AEMS\AEMSvc.exe`), with an "any AEMS product installed" rule or with a registry rule on `InstallDir` . After a failed upgrade such rules can be satisfied while no AEMS service runs, so the deployment would never be retried. Details are in Detection rule for the deployment.

The license key is confidential: it is not bound to a device. The MSI log masks `LICENSEKEY`, but Intune and other management tools store the install command and log it on the device, and process-creation auditing, Sysmon and EDR telemetry record command lines. For fleets, prefer the policy value `License` or a customer MSI built with `Build-MsiWithSettings.cmd` (provided by Logic Unknown ApS); see Where the key is visible and Apply a key.

Deploy the allow-list separately (a `.reg` import or a policy, see Deploy settings at scale) or bake it into the MSI (Bundle settings into the MSI).

Start every new AEMS version with a pilot ring that contains at least one kiosk of every hardware and image type before you assign it broadly.

Deploy settings at scale

1. `.reg` **import** via your deployment tool (Intune Win32 app, SCCM package, RMM script) into the base or Policies hive. Simple and immediate.
2. **Bake the allow-list into a customer-specific MSI** so a fresh install already has your policy - see Bundle settings into the MSI.

Bundle settings into the MSI

What it does: builds a customer-specific `AEMS.msi` with a chosen allow-list (`.reg` files) baked in, so a fresh install already has your policy - no separate `reg import` step on each endpoint.

Run the tool

`Build-MsiWithSettings.cmd` is provided by Logic Unknown ApS. Start it:

```
Build-MsiWithSettings.cmd
```

A window opens listing every `.reg` file in its `Examples` folder. It flags each as HKLM or HKCU (see the caveat below) and shows a preview when you click a row.

Steps

1. **Check** the allow-list `.reg` files you want in the installer - e.g. `NetworkSettings.reg` , or your own `apps.reg` .
2. Click **Build MSI**.
3. The tool combines your selections into one settings bundle and packages it into `AEMS.msi` .
4. A result box confirms success and the output path.

On install, the MSI applies the bundle with `reg import` (after files land, before the service starts).

Bundle a license key (optional, 1.0.6.0+)

The window has a masked **License key** field. When you fill it in:

- the key is verified first with the `AEMSvc.exe` the MSI packages; a key that build would reject (malformed, bad signature, another product, expired) stops the build and nothing is written. Only licenses issued by Logic Unknown ApS are accepted;
- the key is added to the bundle as REG_SZ `License` under `HKLM\SOFTWARE\Logic Unknown\AEMS` only for this build, and removed from the bundle file again afterwards (also

when the build fails), so the bundle file does not keep a license. The bundle header and the console never show the key;

- the resulting MSI carries the license: treat it as confidential.

If a run is interrupted (window closed, process ended), the key can remain in the bundle file until the tool is started again; start the tool once more, or delete the `License` line from the bundle file.

`LICENSEKEY=<key>` on the msiexec command line overrides a bundled key. See the Licensing for the license states and events.

HKLM vs HKCU - important

- **HKLM settings work perfectly.** The install-time import runs as SYSTEM and writes to `HKLM`. The AEMS allow-list lives entirely under `HKLM\SOFTWARE\Logic Unknown\AEMS`, so this covers everything AEMS reads.
- **HKCU settings do NOT reach the real user.** The SYSTEM import writes HKCU into SYSTEM's profile, not the installing user's. The tool flags HKCU files with a warning. AEMS itself uses no HKCU settings; ship any such file as a per-user logon script instead.

Legacy key in older `.reg` files

From AEMS 1.0.6.0 the configuration lives under `HKLM\SOFTWARE\Logic Unknown\AEMS`. A customer `.reg` file written for 1.0.5.0 or earlier names the legacy key `HKLM\SOFTWARE\Circle Of Bytes\AEMS`, which AEMS 1.0.6.0 no longer reads. The bundler handles such files for you:

- Key lines that name exactly `HKEY_LOCAL_MACHINE\SOFTWARE\Circle Of Bytes\AEMS` or a key below it (also the `Policies` twin and `[-...]` delete lines) are rewritten to `...\Logic Unknown\AEMS` in the bundle. Your source files are not changed; the bundle header and the result dialog list every rewrite.
- Any other non-comment line that still names `Circle Of Bytes` (the bare vendor key, another product, the WOW6432Node view, value data) makes the bundler REFUSE the build; no bundle is written. Fix the file and run it again.
- Check the rules without the GUI: `powershell -NoProfile -ExecutionPolicy Bypass -File Build-MsiWithSettings.ps1 -SelfTest`.

Sign the result

The bundled MSI is unsigned. Sign it with your code-signing certificate before you distribute it.

Notes

- Editing a `.reg` (e.g. tailoring the allow-list in `NetworkSettings.reg`) before bundling is how you produce a per-customer installer.

Verify the installation

- Confirm the `AEMSvc` service is **Running** (`sc query AEMSvc`) and set to auto-start.
- Confirm the `AEMSDriver` driver service is **Running** (`sc query AEMSDriver`).
- In the Application log, source `AEMS` , the service start shows event **1007** (driver connected) and **1008** (allow-list sent to the driver). Their absence, or an error such as 3005 or 3006, points to a driver problem - see Driver Diagnostics.
- Launch an allow-listed app as the kiosk (non-admin) user: the original launch is caught and the app reappears running elevated, with no prompt. Confirm the elevation event 1009 under the `AEMS` source.

Upgrade and repair

- **To 1.0.8.0:** install the AEMS 1.0.8.0 MSI over the installed version in one run (`msiexec /i AEMS.msi /qn`). There is no need to install an intermediate version first.
- **From 1.0.6.0 or 1.0.7.0:** settings, allow-list and license are kept. The upgrade replaces driver 1.0.6.0 with driver 1.0.8.0 and loads it at once (see "Driver after an upgrade" below).
- **From 1.0.5.0 or earlier:** the upgrade also moves the settings from the legacy registry key to `HKLM\SOFTWARE\Logic Unknown\AEMS` and, without a license key, starts the 30-day upgrade grace. Follow Upgrading from 1.0.5.0 or earlier, which applies to the 1.0.8.0 MSI as written for 1.0.6.0, and restart the device once after the upgrade.
- **Upgrade:** install the newer `AEMS.msi` over the old one (`msiexec /i AEMS.msi /qn`). The old version is removed first and the new one installed; the allow-list under `HKLM\SOFTWARE\Logic Unknown\AEMS\Applications` is **kept** (from 1.0.2.0 onwards). An allow-list bundled into the new MSI is imported again on upgrade. Exception: upgrading *from* 1.0.1.0 removes the base allow-list (1.0.1.0's own uninstall logic) - `reg export` the key first if that applies.
- **Driver after an upgrade:** run the MSI once. The old version's uninstall stops and unloads its driver and the new installation starts the new one, so driver 1.0.8.0 runs right after the upgrade; a restart is not needed for it. Children in place, wildcard rules and the hold of programs that finish very quickly need the new driver. If the old driver cannot be unloaded (for example a handle is held open), the upgrade fails and asks for a restart (see Upgrading from 1.0.5.0 or earlier). If the old driver is nevertheless still loaded after a completed upgrade (not expected), AEMSvc logs warning 2236 (the loaded driver does not hold caught programs) or 3011, 2014 or 2220 ("... the loaded AEMS driver ... is not the one installed with this AEMS version. Restart the device ..."): restart the device, and restart it again if the event is still logged after the restart.
- **Upgrading from 1.0.5.0 or earlier (registry move):** 1.0.6.0 and later (also the 1.0.8.0 MSI) move the configuration from the legacy key `HKLM\SOFTWARE\Circle Of Bytes\AEMS` to `HKLM\SOFTWARE\Logic Unknown\AEMS` . The upgrade copies all values and the whole `Applications` subtree and verifies the copy. The legacy key is removed at MSI commit (or by AEMSvc after its driver handshake), never while the old driver may still protect it, and only while it still matches the fingerprint recorded at the copy; a legacy key without a pending

migration is never deleted (warning 2005). On any failure it keeps the legacy key and logs an error event, so the allow-list is never lost. Upgrade the devices BEFORE you move Intune scripts or the Allow-list Tool 1.3.0.0 to the new path, and start with a pilot ring (a failing copy step leaves no AEMS on the device). Intune detection rule: the ProductCode of the MSI you deploy (1.0.8.0) with the MSI product version check >= 1.0.8.0 (or product version >= 1.0.8.0 for the AEMS UpgradeCode); never a file-exists or "any AEMS installed" rule, which a failed upgrade can satisfy without a running service, so the deployment would never be retried. Events 1011, 1012, 2005, 2006, 2007 and 3012 are new in 1.0.6.0 (see New and changed events). The legacy **policy** key `HKLM\SOFTWARE\Policies\Circle Of Bytes\AEMS` is never touched: re-target your GPO / Intune policy to `HKLM\SOFTWARE\Policies\Logic Unknown\AEMS` (AEMSvc logs a warning at each start while the legacy policy key still holds values). Update saved `.reg` files and scripts to the new path. Details, event ids and the `TamperProtection` case: Upgrading from 1.0.5.0 or earlier.

- **Repair** (`msiexec /fa AEMS.msi /qn`) restores the files and registry values; it can return **3010** (restart required) because the running `AEMSvc.exe` / `AEMSDriver.sys` are replaced at the next reboot. It does not recreate services that were deleted by hand - uninstall and reinstall instead. With `TamperProtection` on, the 1.0.6.0 repair skips its registry write (the driver would deny it with Error 1406); see the Settings reference.

Upgrading from 1.0.5.0 or earlier

The move described here was introduced with AEMS 1.0.6.0. The AEMS 1.0.8.0 MSI does the same migration, so a device on 1.0.5.0 or earlier is upgraded directly to 1.0.8.0; where this section names the 1.0.6.0 MSI or a 1.0.6.0 device, the same applies to 1.0.8.0.

AEMS 1.0.6.0 moves the AEMS registry settings from the historical Circle Of Bytes vendor key to the Logic Unknown vendor key, in the `<Vendor>\<Product>` format. The MSI migrates installed devices automatically. This section describes what moves, what an administrator has to do, and what happens in the edge cases.

What moves

Use	Up to 1.0.5.0 (legacy)	From 1.0.6.0
Settings	`HKLM\SOFTWARE\Circle Of Bytes\AEMS`	`HKLM\SOFTWARE\Logic Unknown\AEMS`
Allow-list	`HKLM\SOFTWARE\Circle Of Bytes\AEMS\Applications\*`	`HKLM\SOFTWARE\Logic Unknown\AEMS\Applications\*`
Policy override	`HKLM\SOFTWARE\Policies\Circle Of Bytes\AEMS`	`HKLM\SOFTWARE\Policies\Logic Unknown\AEMS`

Unchanged: the service name `AEMSvc`, the event source `AEMS`, the install folder `C:\Program Files\AEMS`, the driver service `AEMSDriver` and device `\\.\AEMS`, and every value name and its meaning.

AEMS 1.0.6.0 reads ONLY the new keys. The legacy keys are not read as a fallback.

What the upgrade does

The MSI (1.0.6.0, same UpgradeCode, major upgrade) runs the migration as LocalSystem:

1. **Old product removal, first** (before anything of 1.0.6.0 is installed). The 1.0.5.0 uninstall stops and deletes its service and driver and removes its files. During an upgrade it does NOT delete the settings: only its `InstallDir` value goes; the allow-list, `LogLevel`, `AdminPassthrough`, `TamperProtection` and everything else stay in the legacy key. Its driver is unloaded here, so it can no longer protect the legacy key. This holds for 1.0.2.0 to 1.0.5.0. **Exception: 1.0.1.0.** Its own uninstall deletes the whole legacy settings key, and it runs before the copy, so nothing is left to migrate. Before upgrading a 1.0.1.0 device, export the key (`reg export "HKLM\SOFTWARE\Circle Of Bytes\AEMS" aems-legacy.reg`), and after the upgrade import it again with every key line rewritten to `HKLM\SOFTWARE\Logic Unknown\AEMS` (or upgrade it to 1.0.5.0 first, which keeps the key, then to 1.0.6.0).
2. **Copy** (right after the installer has copied its files, the installer runs the installed, signed `AEMSvc.exe --migrate`). When `HKLM\SOFTWARE\Logic Unknown\AEMS` does not exist yet and the legacy settings key does, the legacy tree is read, a marker value `LegacyMigrationPending` (REG_BINARY, state "copying") is written into the new key first, then every value of the legacy key and the whole `Applications` subtree (all subkeys, all values, exact types and bytes). The copy is read back and compared value by value, the legacy key is re-read to make sure it did not change meanwhile, `TamperProtection` is written last, and the marker is set to "complete". The marker holds a SHA-256 fingerprint of the legacy content that was copied and of the content the new key received. If anything fails, what this run wrote is removed again, the legacy key is left untouched, event 3012 is logged and the install aborts.

3. **New product install.** The new driver service and `AEMSvc` service are created, the installer writes `InstallDir` into the new key, imports any bundled settings, and only THEN starts the new driver and the service.
4. **Removal of the legacy key, after the transaction committed** (`AEMSvc.exe --migrate-cleanup` , run when the installation commits). The legacy key is removed only when the copy is complete and the legacy key STILL matches the fingerprint in the marker. A flag value `LegacyRemovalInProgress` is written into the legacy key first, then its subkeys and the key itself are deleted. Its parent `HKLM\SOFTWARE\Circle Of Bytes` is deleted only when it is then completely empty (other products may live there). Event 1012.
5. **If the install fails after step 1**, the rollback removes what 1.0.6.0 created: it stops and deletes the new `AEMSvc` and `AEMSDriver` services, and the rollback deletes the new settings key again while it is still an unmodified copy of the legacy key. The device is then left WITHOUT an AEMS product (Add/Remove Programs and ProductCode detection report "not installed", so Intune or ConfigMgr retry), with the legacy key intact. Reboot and run the AEMS MSI again. AEMS is not running until then.

Why the old product is removed first: none of the 1.0.x MSIs can restore its own services and driver registration when a removal is rolled back. Removing it inside the 1.0.6.0 transaction would turn a late failure into a 1.0.5.0 that Add/Remove Programs reports as installed but that has no service and no driver, which no deployment tool would repair.

Reading a verbose MSI log (`msiexec /i AEMS.msi /qn /l*vx upgrade.log`). After a failed upgrade (exit code 1603) search the log for these action names; the step that logged "Return value 3" is the one that failed:

Step	Action name in the MSI log
Check that the previous driver is not still loaded and marked for deletion ("Restart the computer" message)	`AppSearch` (registry search `AemsDriverDeleteFlagSearch`) and the launch conditions
Old product removal, first	`RemoveExistingProducts`, right after `InstallValidate`
Copy of the legacy settings	`MigrateLegacySettings`, or `MigrateLegacySettingsMerge` when the new key existed before the upgrade
Rollback of that copy	`RollbackMigrateLegacySettings`
Driver package and driver service	`InstallDriver`, `CreateDriverService` (error 1072 here = the old driver is still marked for deletion: reboot, then run the MSI again)
AEMSvc service	`InstallService`, `SetAemsvcDependency`
Bundled settings and license key	`ImportBundledSettings`, `WriteLicenseKey` (only with `LICENSEKEY`; the key itself is not logged)
Start of the driver and the service	`StartDriverService`, `StartAemsService`
Removal of the legacy key at commit	`CleanupLegacySettings`

The undo steps of a rollback have the same names with the prefix `Rollback` (for example `RollbackCreateDriverService`, `RollbackInstallService`).

The migration is idempotent: when the new key already holds a complete copy nothing is copied again, and a device without a legacy key (fresh install) is a no-op.

The marker and the fingerprint

The legacy key is never removed because of the marker alone. Every run compares the legacy key with the fingerprint recorded at the copy:

Situation at a later run	What happens
Legacy key unchanged since the copy	Removal proceeds (MSI commit, or AEMSvc after its driver handshake).
Legacy key changed (for example a failed upgrade, then further edits under the old path), new key still holds only the migrated content (`InstallDir` ignored)	The new key is refreshed from the legacy key (`InstallDir` kept), event 1011 "the unmodified copy was refreshed". Only the MSI copy step and AEMSvc after its driver handshake (outside an installer transaction) refresh; the MSI commit step leaves it to them.
Legacy key changed AND the new key changed too	Nothing is copied or removed. Warning 2005 explains; AEMS keeps using the new key. Move what you still need from the legacy key, then delete it by hand.
A removal was interrupted (`LegacyRemovalInProgress` in the legacy key)	The removal continues; the missing parts are not treated as an edit.
Marker still "copying" (an earlier copy could not finish or could not be undone)	The copy is verified again: when only `TamperProtection` is missing it is written now; otherwise the MSI copy step or AEMSvc after its handshake copies again. The legacy key is not removed before that.
Marker of an unknown format	Nothing is removed; warning 2005.

A new key that existed before the upgrade (merge)

If something created `HKLM\SOFTWARE\Logic Unknown\AEMS` while the device still ran 1.0.5.0 (an Intune script moved to the new path too early, `AEMSAllowlist.exe` 1.3.0.0 `--apply`), the MSI merges instead of skipping the migration: it runs `AEMSvc.exe --migrate --merge-into-existing` when `AppSearch`, which runs before the old product is removed, found no `InstallDir` in the new key (property `AEMSNEWKEYINSTALLDIR` empty: no 1.0.6.0 install ever completed on the device). Because that is device state, not "an old product is installed", a retry after a failed first attempt (old product already removed) merges as well, and an upgrade from 1.0.6.0 to a later version (`InstallDir` present before the old removal) never merges a stale legacy key. A new key that has neither `InstallDir` nor a marker receives every legacy value and every `Applications\<entry>` it does not have yet; whatever the new key already holds wins (an entry is taken whole or not at all). Event 1011 lists what was kept from the new key, for example

`2 items kept from the new key: LogLevel, Applications\Notepad`. The merge is verified like a copy and the legacy key is then removed like after a copy. AEMSvc itself never merges, and the MSI never merges into a key that has `InstallDir` (an installed 1.0.6.0).

Safety net at service start

`AEMSvc` 0.1.7.0 and later repeat the migration at every start:

- Before its first configuration read: new key missing and legacy key present -> copy (event 1011, or 3012 on failure). This covers devices where the files were replaced by hand instead of through the MSI. It also completes a "copying" marker when only `TamperProtection` is missing.
- After its driver handshake (the registered service process is exempt from the driver's registry protection): refresh an unmodified copy whose legacy key changed, and remove a pending legacy key (event 1012), unless a Windows Installer transaction is running (the MSI's own steps then do it). A legacy key that cannot be removed stays in place and is retried at the next start (event 2005). A legacy key that exists next to the new key WITHOUT the marker is never deleted by AEMS; event 2005 asks the administrator to move whatever still writes it.

What an administrator must do

Rollout order

1. **Pilot ring first.** Assign the AEMS MSI (1.0.8.0) to a small pilot group before broad assignment, including at least one kiosk of every hardware/image type, one with `TamperProtection=1`, one with a WDAC or AppLocker policy and one with pre-written new-path settings, and check events 1011/1012/3012/2005/2007 there. A systematic failure of the copy step (for example a blocked `AEMSvc.exe` or an unreadable legacy subkey) aborts the install AFTER the old product was removed, so it would leave NO AEMS on every targeted device.
2. Upgrade the devices first: run the AEMS 1.0.8.0 MSI once (`msiexec /i AEMS.msi /qn`, optionally with `LICENSEKEY=<key>`), then restart the device. After the restart, events 1007 and 1008 must appear and 3011, 2014, 2220 and 2236 must not; a device without a key shows the upgrade grace (2010, and `--license-status` reports UpgradeGrace).
3. Only then move anything that writes AEMS settings (Intune remediation and platform scripts, configuration profiles, `.reg` files, `AEMSAllowlist.exe` 1.3.0.0) to the new path, scoped to devices that run 1.0.6.0 or later (for example by the MSI ProductVersion, or by the presence of `InstallDir` in `HKLM\SOFTWARE\Logic Unknown\AEMS`). During the transition it is safe to write BOTH paths.

A device on 1.0.5.0 reads only the legacy key; a 1.0.6.0 device only the new one. If a writer reaches a 1.0.5.0 device early, the upgrade merges (see above), so nothing is lost, but the 1.0.5.0 device ignores the new key until it is upgraded.

Group Policy (or any other policy source)

The legacy policy key `HKLM\SOFTWARE\Policies\Circle Of Bytes\AEMS` is GPO-owned. AEMS never reads, changes or deletes it. AEMS 1.0.6.0 ignores it; its settings no longer apply.

1. In the GPO (Group Policy Preferences registry items, or an ADMX/ADML you maintain), change the key path from `SOFTWARE\Policies\Circle Of Bytes\AEMS` to `SOFTWARE\Policies\Logic Unknown\AEMS`. Value names and data stay the same; the `Applications` subtree moves the same way.
2. Remove the old registry items from the GPO (with "Remove this item when it is no longer applied" or a delete action) so the legacy policy key disappears from the clients.
3. Check a client after `gpupdate /force`: `reg query "HKLM\SOFTWARE\Policies\Logic Unknown\AEMS" /s`.

While the legacy policy key still holds any value or subkey, `AEMSvc` logs warning **2006** at every service start.

Do the GPO change together with (or right after) the 1.0.6.0 rollout: a 1.0.5.0 device reads only the legacy policy key, a 1.0.6.0 device only the new one. Deploying both paths during the transition is safe.

Detection rule for the deployment

Use the MSI you deploy (AEMS 1.0.8.0), so that devices still on 1.0.5.0, 1.0.6.0 or 1.0.7.0 are updated. Use one of these for the Intune Win32 app (or the ConfigMgr deployment type):

- **MSI ProductCode** of the 1.0.8.0 package (Intune Win32 app: detection rule type "MSI", the ProductCode from the MSI's `Property` table; tick "MSI product version check", "Greater than or equal to" 1.0.8.0), or
- **product version >= 1.0.8.0 for UpgradeCode** `{7F3C1A28-9D6B-4E51-AE2C-5B0E9A1F7C34}` in a script rule or ConfigMgr (for example Windows Installer `RelatedProducts` of that UpgradeCode, then `ProductVersion` of the product found), optionally combined with "service `AEMSvc` exists".

Do NOT use a file-exists rule (`C:\Program Files\AEMS\AEMSvc.exe`), an "any AEMS product installed" / UpgradeCode-only rule or a registry rule on `InstallDir`: after a failed upgrade (old driver stuck, see the TamperProtection caveats) 1.0.5.0 can stay listed with its files restored while no AEMS service runs, and such rules are satisfied, so the deployment is never retried.

Intune, scripts, detection rules, `.reg` files

Anything that writes or reads `HKLM\SOFTWARE\Circle Of Bytes\AEMS` must move to `HKLM\SOFTWARE\Logic Unknown\AEMS` (in the order above): Intune remediation or platform scripts, Win32-app detection rules, configuration profiles (OMA-URI registry writes), allow-list `.reg` files, monitoring checks. Writes to the legacy key after the upgrade are ignored (warning 2005), and the legacy key is removed at the end of the upgrade.

- The AEMS Allow-list Tool 1.3.0.0 (the tool version for AEMS 1.0.6.0) writes the new path.
- `Build-MsiWithSettings.cmd` (customer MSI bundling, provided by Logic Unknown ApS) rewrites key lines that name exactly `...\Circle Of Bytes\AEMS` or `...\Circle Of Bytes\AEMS\...` (also the Policies twin) to the new key in the bundle, and refuses to build when a selected file still names `Circle Of Bytes` anywhere else outside a comment (the bare vendor key, another product, the WOW6432Node view, value data). Check without the GUI: `powershell -NoProfile -ExecutionPolicy Bypass -File Build-MsiWithSettings.ps1 -SelfTest`.

Kiosks with WDAC or AppLocker

The migration runs `C:\Program Files\AEMS\AEMSvc.exe`, the same signed file the service runs (signed by Logic Unknown Aps); nothing is extracted to and run from `C:\Windows\Installer` or a temp folder. A policy that admits AEMSvc therefore admits the migration. `pnputil`, `sc.exe`, `reg.exe` and `powershell.exe` from `System32` are run as before. Test on one device first.

TamperProtection and the old driver

`TamperProtection` (REG_DWORD, settings key) makes the AEMS driver deny registry changes under the protected key to every process except the registered `AEMSvc`.

Driver	Protected key	Flag read from
1.0.5.0 and older (protocol `0x00020000`)	the WHOLE `HKLM\SOFTWARE\Circle Of Bytes` vendor key	`HKLM\SOFTWARE\Circle Of Bytes\AEMS`
1.0.6.0 (driver 1.0.6.0, protocol `0x00020003`)	exactly `HKLM\SOFTWARE\Logic Unknown\AEMS` and its subkeys (component boundary: `...\Logic Unknown\<other product>`, `...\Logic Unknown\AEMSx` and the vendor key itself are NOT protected)	`HKLM\SOFTWARE\Logic Unknown\AEMS`

The new driver also lets `KeySetHandleTagsInformation` through (the fix for registry opens with a 32-bit or 64-bit view flag), so reads with `KEY_WOW64_64KEY` / `KEY_WOW64_32KEY` (`reg.exe /reg:64`, .NET `Registry64`, MSI `RegistrySearch Win64`, Intune detection rules) work under `TamperProtection`. When AEMSvc 0.1.7.0 or later finds an older driver loaded it logs warning 2007 at every start, because `TamperProtection` then has no effect on the new key.

What happens on a device with `TamperProtection=1` during the upgrade:

1. The old product removal stops and deletes the old driver before anything of 1.0.6.0 runs. If the old driver cannot be unloaded (stuck handle) it stays loaded, marked for deletion; the copy still works (reading the legacy key is allowed and the new key is outside the old driver's

prefix), but creating the new driver service then fails (1072) and the install rolls back: no AEMS product, legacy key intact. Reboot and run the upgrade again (until the reboot the launch condition below refuses the retry). With `TamperProtection=1` the old uninstall usually fails earlier, see the caveat on the old product's own uninstall below.

2. The copy writes `TamperProtection` last, after the read-back check.
3. The new driver is started only after every installer write to the new key (the installer's own registry values and the bundled settings it imports); a migrated `TamperProtection=1` therefore cannot make the new driver deny the installer's own writes.
4. The legacy key is removed at commit. The new driver does not protect the legacy path, so the removal succeeds. The commit step is not the registered service, so its delete of the marker value in the (now protected) new key is denied: the marker stays until the next AEMSvc start, normally the next reboot, which removes it without an event. This is expected.
5. If the removal of the legacy key is denied for any other reason, the legacy key and the marker stay, nothing is lost (the verified copy is in the new key), event 2005 is logged, and `AEMSvc` retries after its next driver handshake, as the registered (exempt) process.

The allow-list is never deleted before a verified copy exists, and never while the legacy key differs from what was copied. (Upgrades from 1.0.1.0 are the exception: that version's own uninstall deletes the legacy key before the copy, see step 1 of "What the upgrade does".)

Caveats:

- **Rollback disabled.** With the `DisableRollback` machine policy or `DISABLEROLLBACK=1`, Windows Installer runs neither rollback nor commit actions. The legacy key and the marker then stay after the upgrade; AEMSvc removes the legacy key at its next start outside an installer transaction (the start right after the install happens while the installer still runs and leaves it), and a failed install is not cleaned up.
- **Repair with `TamperProtection=1`.** A repair (`msiexec /fa`, ARP Repair, `REINSTALL=ALL`) would have to rewrite `InstallDir` under the protected key (Error 1406, repair rolled back). The 1.0.6.0 MSI skips its registry write during a repair while `TamperProtection` is non-zero (found by `AppSearch`); files are still repaired. To let a repair restore a deleted `InstallDir` or `Applications` key, set `TamperProtection` to 0 first.
- **Uninstall with `TamperProtection=1`**: the uninstall stops the driver before it removes the settings key. If the driver cannot be unloaded, the key removal is denied and the key stays; delete it after a reboot.
- **The old product's own uninstall with `TamperProtection=1` and a stuck old driver** (the old product's own script, not fixable from 1.0.6.0): if the 1.0.5.0 driver cannot be unloaded, it still protects the whole legacy vendor key and denies the old uninstall's own delete of `InstallDir`. The old uninstall then fails and rolls back, and the AEMS install stops before it changes anything. The settings are safe (the legacy key is untouched), but **1.0.5.0 stays listed in Add/Remove Programs while its `AEMSvc` service is already removed** (that step cannot be rolled back) and its driver is marked for deletion: AEMS does not run until the

device reboots and the AEMS MSI runs again. A detection rule on the deployed MSI's ProductCode or version (below) reports "not installed" and retries.

- **"Restart first" launch condition.** The SCM writes `DeleteFlag=1` into a service's key when the service is deleted while it is still loaded; the key and the flag go away at the next boot. The 1.0.6.0 MSI reads `HKLM\SYSTEM\CurrentControlSet\Services\AEMSDriver`, value `DeleteFlag` (before anything is removed) and refuses to start while it is non-zero: "The AEMS driver from the previous installation is still loaded and is marked for deletion. Restart the computer, then run this installation again." (exit 1603 in `/qn`). A retry before the reboot therefore fails at once instead of tearing down again. The flag is SCM behaviour, not a documented API contract; if it is ever missing the install still fails safely (error 1072 when the new driver service is created, rolled back), and a stale value cannot outlive a reboot.

License key and upgrade grace

AEMS 1.0.6.0 is also the first licensed version (see Licensing). Without a valid license key it does not elevate. Devices that ran 1.0.5.0 or earlier get a one-time **upgrade grace** of 30 days, so the upgrade itself never stops elevation:

1. Evidence of an upgrade is a COMPLETED settings migration: the new key holds the marker `LegacyMigrationPending` in state complete (copy or merge) and the legacy tree it recorded was not empty. That marker exists between the copy step and the removal of the legacy key, which is exactly when the grace marker is written. A legacy key that merely exists is not evidence: an empty legacy key that the copy step copies, or a legacy key next to an installed 1.0.6.0 (no copy, no marker), gives no grace.
2. After a successful MSI copy step (`AEMSvc.exe --migrate`, right after the files are installed; copy or merge) that evidence is checked and, when present, REG_SZ `LicenseUpgradeGraceStartUtc` = the current UTC time is written into `HKLM\SOFTWARE\Logic Unknown\AEMS`, once, with event 1015 naming the end date. This happens before the new driver starts, so a migrated `TamperProtection=1` cannot deny it. An existing marker is never overwritten.
3. AEMSvc repeats this at its start (safety net, for example after a manual file replacement): after its own copy step it checks the evidence and writes the marker into the existing key (never creating the key: the copy must create it itself), and, when `TamperProtection=1` denied that write, again right after its driver handshake (the evidence seen before the handshake is kept, because the handshake step removes the legacy key and the marker).
4. For 30 days AEMS keeps elevating without a license (state UpgradeGrace, warning 2010 every day naming the end date). Apply a license key within that time: `type aems-license.txt | "C:\Program Files\AEMS\AEMSvc.exe" --set-license`, the MSI property `LICENSEKEY` on the upgrade itself (`msiexec /i AEMS.msi /qn LICENSEKEY=<key>`), or the policy value `License` in `HKLM\SOFTWARE\Policies\Logic Unknown\AEMS`.

A standard user cannot write or delete the marker (HKLM). With `TamperProtection=1` the driver of AEMS 1.0.6.0 denies changes to it to everyone but AEMSvc. Licensing is enforced against

standard users; it relies on administrators not tampering with the device.

Ordering: the grace marker is always written BEFORE the legacy key and the migration marker are removed (they are the evidence), and the removal is held back until it is written. This matters when AEMSvc itself does the copy with `TamperProtection=1` and the driver of AEMS 1.0.6.0 loaded: the copy then cannot finalise its migration marker and the grace write is denied until the driver handshake; right after the handshake AEMSvc writes the grace marker first and only then removes the legacy key. A crash at any point neither loses the grace nor grants it twice. A legacy key with any value (for example `InstallDir`) or any subkey (also an empty `Applications` key) counts as an upgrade; a completely empty one does not.

Rollout note: plan the license keys with the upgrade. A fresh 1.0.6.0 install has no grace, and an upgraded device stops elevating 30 days after its upgrade unless a key was applied.

Wildcard path rules need the new driver

AEMS 1.0.6.0 adds wildcard path rules (`DataPath` with `*` or `?`, plus the new values `Publisher` and `PublisherThumbprint`; see Wildcard path rules). The matching happens in the driver, so wildcard rules work only with driver 1.0.3.0 or later (protocol `0x00020002` or later; AEMS 1.0.6.0 ships driver 1.0.6.0, protocol `0x00020003`). The upgrade normally unloads the old driver (the old version's uninstall stops it) and starts the new one, so the new driver runs right after the upgrade. If the old driver cannot be unloaded, the upgrade fails and asks for a restart (see "TamperProtection and the old driver"). If the old driver is nevertheless still loaded after an upgrade:

- AEMSvc 0.1.7.0 and later do not send wildcard rules to it and logs warning 2014 once at start ("wildcard rules are inactive until the driver is updated"); exact entries keep working;
- reboot the device; the wildcard rules become active without any other step.

Two behaviour changes in 1.0.6.0 that also affect existing exact entries:

- an entry whose `HashSHA` is not exactly 64 hex digits (or whose `PublisherThumbprint` is not 40 hex digits) logs warning 2012 at service start and refuses every launch it matches (2013 `[MalformedRule]`; administrators: 2016); an entry with an empty `DataPath` is dropped (2012). Check the event log after the upgrade, correct such entries and restart the AEMSvc service;
- when the driver catches a launch for a standard user but AEMSvc finds no matching entry, the program is no longer elevated: it keeps running unelevated (warning 2003, then 2004).

Do not deploy wildcard rules to devices that still run 1.0.5.0 or earlier: AEMSvc 0.1.4.0 and earlier read only the legacy key, and even a hand-written legacy entry would be treated as a literal path that never matches; `Publisher` and `PublisherThumbprint` are ignored there, so an exact entry that relies on them for its identity check is NOT checked on an old version. Use `AEMSAllowlist.exe` 1.3.0.0 (or the example file `WildcardRules.reg`) only after the device runs 1.0.6.0.

Example file WildcardRules.reg (provided by Logic Unknown ApS)

```
Windows Registry Editor Version 5.00

; ============================================================================
; WildcardRules.reg
;
; Example AEMS wildcard path rules (AEMS 1.0.6.0 and later).
; Publisher: Logic Unknown ApS - https://logic-unknown.com
; Documentation: https://docs.logic-unknown.com/aems/
;
; A wildcard rule approves an executable by a path PATTERN instead of one exact
; path. An entry is a wildcard rule when DataPath contains * or ?.
;   ?  matches exactly one character that is not a backslash
;   *  matches zero or more characters that are not a backslash
;   Wildcards never cross a folder boundary; the pattern must match the WHOLE
;   path (case-insensitive).
;
; Rules (an entry that breaks one is rejected at service start, with a warning):
;   - local path X:\... with a literal drive letter (no UNC, \\?\ or devices)
;   - the drive and the first folder are literal; no file directly in X:\
;   - no / " < > |, control characters or a second colon; no empty, . or ..
;     folder component
;   - the file name ends with the literal .exe
;   - at most 251 characters (refused, never truncated)
;   - Publisher or HashSHA is set, and GlobalApplication = 1
; A malformed HashSHA (not 64 hex digits) or PublisherThumbprint (not 40 hex
; digits) does not drop the entry: it is kept as a refusing entry (warning
; 2012) and every launch it matches is refused (2013 [MalformedRule]).
; Use a folder under C:\Program Files: a new folder directly in the drive root
; (for example C:\Tools) normally lets users write and fails the check below.
;
; At launch AEMS elevates a matching file only when BOTH guards pass:
;   1. trusted location: a fixed drive on an accepted storage bus (removable
;      media and USB, FireWire, SD/eMMC, iSCSI, Fibre Channel and mounted
;      virtual disks are refused; the check is by bus, so an external NVMe or
;      eSATA enclosure is NOT detected - use the system drive) with a global
;      drive letter
;      (a drive letter created or redirected for one user, for example with
;      subst or a mapped drive, is refused), no junction/symbolic link in the
;      path, the file and every folder above it owned by and writable only by
;      Administrators, SYSTEM or TrustedInstaller;
;   2. identity: valid EMBEDDED Authenticode signature whose leaf certificate
;      CN equals Publisher (and SHA-1 thumbprint equals PublisherThumbprint
;      when set), and/or SHA-256 equals HashSHA.
; Otherwise the program runs unelevated and AEMS logs a warning with the
; reason. Elevation happens when a user launches the file, never when a file
; merely appears in the folder.
;
; New per-app values:
;   Publisher            REG_SZ - leaf certificate CN of the embedded signature
;   PublisherThumbprint  REG_SZ - optional SHA-1 thumbprint (40 hex) of that
;                                 certificate, checked in addition to Publisher
;
; Requires the AEMS 1.0.6.0 driver (protocol 0x00020002), started by the
; upgrade (reboot if warning 2014 says the old driver is still loaded). Contoso and
Fabrikam are placeholder vendors: replace the
; paths, the publisher names and the thumbprint with the real values
; (Get-AuthenticodeSignature, or "Read from file..." in AEMSAllowlist.exe).
;
; Apply (elevated):   reg import WildcardRules.reg
;                     Restart-Service AEMSvc
```

```
; Details: https://docs.logic-unknown.com/aems/configure/ (Wildcard path rules).
; ============================================================================

; Example 1 - a vendor setup program with a changing version in the file name,
; pinned by publisher. Matches C:\Program Files\Contoso\Tools\setup_2.4.1.exe
; and setup_x64.exe; does not match C:\Program Files\Contoso\Tools\old\setup_1.exe.
[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Contoso setup]
"FriendlyName"="Contoso setup (any version)"
"DataPath"="C:\\Program Files\\Contoso\\Tools\\setup_*.exe"
"Publisher"="Contoso Ltd"
"GlobalApplication"=dword:00000001

; Example 2 - one program in a versioned folder, pinned by publisher and
; certificate thumbprint. Matches ...\Console 7.2\console.exe and
; ...\Console 8.0\console.exe; not ...\Console 10.1\console.exe (? is exactly
; one character). The thumbprint changes when the vendor renews its
; certificate; leave PublisherThumbprint out to survive renewals.
[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Fabrikam console]
"FriendlyName"="Fabrikam console"
"DataPath"="C:\\Program Files\\Fabrikam\\Console ?.?\\console.exe"
"Publisher"="Fabrikam, Inc."
"PublisherThumbprint"="0123456789ABCDEF0123456789ABCDEF01234567"
"GlobalApplication"=dword:00000001

; ----------------------------------------------------------------------------
; Example 3 (commented out) - a file that is only catalog-signed or unsigned
; cannot satisfy Publisher; pin its SHA-256 instead. The rule then elevates
; only files with exactly this content.
; ----------------------------------------------------------------------------
;[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Contoso updater]
;"FriendlyName"="Contoso updater"
;"DataPath"="C:\\Program Files\\Contoso\\Updater\\update*.exe"
;"HashSHA"="0000000000000000000000000000000000000000000000000000000000000000"
;"GlobalApplication"=dword:00000001

; ----------------------------------------------------------------------------
; To remove these entries:
;   reg delete "HKLM\SOFTWARE\Logic Unknown\AEMS\Applications\Contoso setup" /f
;   reg delete "HKLM\SOFTWARE\Logic Unknown\AEMS\Applications\Fabrikam console" /f
; ----------------------------------------------------------------------------
```

New and changed events

The licensing events (1013, 1014, 1015, 2008, 2009, 2010, 2011, 2015, 3013, 3014) are listed in the Licensing events table.

Id	Level	When	Text
1011	Information	The legacy settings key was copied to the new key (MSI copy step, or service start safety net)	`Settings migration: migrated (<n> keys, <m> values copied from HKLM\SOFTWARE\Circle Of Bytes\AEMS to HKLM\SOFTWARE\Logic Unknown\AEMS, verified); legacy key kept, removal pending.` (the tail reads `legacy key removed` when the copy and the removal happened in one service start)
1011	Information	An unmodified copy was refreshed because the legacy key changed after it	`Settings migration: migrated (the legacy key changed after the earlier copy, the unmodified copy was refreshed: <n> keys, <m> values copied from ... to ..., verified); ...`
1011	Information	Merge into a new key that existed before the upgrade (MSI only)	`Settings migration: migrated (merged into the existing HKLM\SOFTWARE\Logic Unknown\AEMS: <n> keys and <m> values added from HKLM\SOFTWARE\Circle Of Bytes\AEMS, verified; <k> items kept from the new key: <names>); legacy key kept, removal pending.`
1012	Information	The legacy settings key was removed after a verified copy	`Settings migration: already migrated; legacy key removed.` or `... legacy key removed (and the then-empty HKLM\SOFTWARE\Circle Of Bytes).`
2005	Warning	The legacy settings key could not be removed	`The legacy settings key HKLM\SOFTWARE\Circle Of Bytes\AEMS could not be`

Id	Level	When	Text
			`removed (error <n>). It is ignored: AEMS reads only HKLM\SOFTWARE\Logic Unknown\AEMS. The removal is retried at the next service start.` (from the MSI commit step: `Settings migration: ...; legacy key could NOT be removed (error <n>), it is kept and ignored. The service retries the removal at its next start.`)
2005	Warning	A legacy settings key exists next to the new key without a pending migration (something still writes the old path)	`The legacy settings key HKLM\SOFTWARE\Circle Of Bytes\AEMS exists next to HKLM\SOFTWARE\Logic Unknown\AEMS and is ignored: AEMS 1.0.6.0 and later read only the new key. Move anything that still writes to the old path (scripts, Intune, .reg files) to the new path, then delete the old key.`
2005	Warning	The legacy key changed after the copy and the new key changed too (or after a merge)	`The legacy settings key ... changed after it was copied to ..., and ... was changed too. Both are kept and nothing is removed; AEMS reads only .... Move the changes you still want from the legacy key into the new key, then delete the legacy key by hand.` (merge: `... changed after it was merged into ...`)
2005	Warning	A refresh is needed but not done by this step, or was denied	`... which still holds only the copied content. The new key is refreshed from the legacy key by the next MSI copy step or by AEMSvc after its driver`

Id	Level	When	Text
			`handshake ...` / `... refreshing the new key was denied (error <n>; with TamperProtection=1 only the registered AEMSvc may change it) ...`
2005	Warning	A copy whose marker still says "copying" cannot be completed here; TamperProtection could not be completed; unknown marker format; legacy key unreadable	`The copy of ... in ... was never verified (its marker still says 'copying'). The legacy key is kept. ...` / `... TamperProtection could not be completed in ... (error <n>) ...` / `... the migration marker LegacyMigrationPending in ... has an unknown format ...` / `... could not be read to check it against the migration marker (error <n>) ...`
2006	Warning	Every service start while the legacy policy key holds any value or subkey	`The legacy policy key HKLM\SOFTWARE\Policies\Circle Of Bytes\AEMS holds settings, but AEMS 1.0.6.0 and later read policy only from HKLM\SOFTWARE\Policies\Logic Unknown\AEMS. The old policy is IGNORED. Move the GPO (or other policy source) to the new path; AEMS never changes or deletes policy keys.`
2007	Warning	Every service start while the loaded driver is older than 1.0.3.0 (protocol minor 0)	`The loaded AEMS driver speaks protocol 0x00020000; this AEMSvc 0.1.9 expects minor 1 (driver 1.0.3.0 or later). That older driver protects and reads TamperProtection from the pre-1.0.6.0 settings key, not from HKLM\SOFTWARE\Logic Unknown\AEMS, and lacks the fix that lets registry opens with a 32-bit or 64-bit view`

Id	Level	When	Text
			`flag pass under TamperProtection: TamperProtection is NOT enforced[ although TamperProtection=1 is set in the new key]. Elevation works. Install an AEMS MSI that ships driver 1.0.3.0 or later.`
3012	Error	The copy, refresh or merge failed	`Migration of HKLM\SOFTWARE\Circle Of Bytes\AEMS to HKLM\SOFTWARE\Logic Unknown\AEMS FAILED at step '<step>' (error <n>).` followed by exactly one of: `Nothing was changed; the legacy key is untouched.` / `What this run wrote to the new key was removed again; the legacy key is untouched.` / `What this run wrote to the new key could NOT be removed again (error <n>). [The new key holds a verified copy of everything except TamperProtection.] It keeps its migration marker in state 'copying', so it is not treated as a completed migration and the legacy key is not removed; the next MSI copy step or AEMSvc start completes or redoes the copy. The legacy key is untouched.`
2000	Warning	changed text	`Config key HKLM\SOFTWARE\Logic Unknown\AEMS not found; running with defaults.` or `Config key HKLM\SOFTWARE\Logic Unknown\AEMS could not be opened (error <n>); running with defaults.`

Id	Level	When	Text
1002	Information	changed text (registry view-flag fix)	When the first configuration read found no key or failed to open it, AEMSvc reads again once after its driver registration and logs `AEMSvc 0.1.9 config loaded (reloaded after the driver registration): ...`. In that case 2000/2001/1002 are logged after event 1007 instead of before it.

Uninstall and downgrade

- A real uninstall (from 1.0.6.0) removes `HKLM\SOFTWARE\Logic Unknown\AEMS` and any leftover `HKLM\SOFTWARE\Circle Of Bytes\AEMS`, then `HKLM\SOFTWARE\Logic Unknown` and `HKLM\SOFTWARE\Circle Of Bytes` only when they are completely empty (other Logic Unknown products can live there). Policies keys are never touched.
- A downgrade over 1.0.6.0 is blocked by the MSI. To go back to 1.0.5.0, export the allow-list (`reg export "HKLM\SOFTWARE\Logic Unknown\AEMS" aems.reg`), uninstall 1.0.6.0, install 1.0.5.0 and import the file after replacing `Logic Unknown` with `Circle Of Bytes` in its key lines.

Uninstall

```
msiexec /x AEMS.msi /qn
```

Uninstall stops and deletes the `AEMSvc` service and removes its event source (`AEMSvc.exe --uninstall`), stops and deletes the `AEMSDriver` driver service, removes the driver package from the driver store, and removes the `HKLM\SOFTWARE\Logic Unknown\AEMS` subtree (including the allow-list) plus any leftover legacy key from 1.0.5.0 or earlier. The Policies hive is GPO-owned and is not removed. Other Logic Unknown products under `HKLM\SOFTWARE\Logic Unknown` are kept. What stays afterwards: the AEMS entries in the Application log. On a device that ran on a Windows build newer than the validated ones, the uninstall of 1.0.8.0 also removes the empty key `HKLM\SOFTWARE\Logic Unknown\AEMS-DriverBindGuard` (and then the `Logic Unknown` vendor key when nothing else is in it); the key stays when it holds the value `BindAttemptBuild` (see the guard key after an uninstall). The uninstall of 1.0.7.0 or earlier leaves the empty key; it is harmless and may be deleted. If `TamperProtection` is set, read the uninstall caveat in the Settings reference.

Configure AEMS

> **Applies to AEMS 1.0.8.0**
>
> This documentation describes AEMS 1.0.8.0 (service 0.1.9.0, driver 1.0.8.0, Allow-list Tool 1.3.0.0), released 2026-11-20. Install 1.0.8.0. Text marked "from 1.0.8.0" does not apply to earlier versions. Where the text says "from 1.0.6.0", it applies to 1.0.7.0 and 1.0.8.0 as well.

All AEMS configuration is registry values under `HKLM\SOFTWARE\Logic Unknown\AEMS`, or under `HKLM\SOFTWARE\Policies\Logic Unknown\AEMS` when you push it as policy. AEMS reads its configuration when the service starts, so restart the service (`Restart-Service AEMSvc`) or the device after a change. The license is the exception: it is re-read every 10 minutes.

Common tasks:

- Approve an application - the core task.
- Let an approved program start other programs elevated as real children, for example a wrapper program that calls other programs.
- Approve a program by path pattern, for example a vendor setup program whose file name changes with every version.
- Decide how administrators are treated.
- Push a locked-down allow-list as policy.
- Create allow-list entries with the Allow-list Tool.
- Look up a value in the settings reference.

Approve applications

AEMS only elevates apps you explicitly allow-list. Each approved app is a subkey under `HKLM\SOFTWARE\Logic Unknown\AEMS\Applications\<Name>`.

Minimal example - approve an elevated Registry Editor:

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Registry Editor]
"FriendlyName"="Registry Editor (elevated)"
"DataPath"="C:\\Windows\\regedit.exe"
"GlobalApplication"=dword:00000001
```

Save as `apps.reg`, then `reg import apps.reg` (elevated), then restart the service (`Restart-Service AEMSvc`). AEMS reads its configuration only when the service starts (and at every boot).

Per-app values:

Value	Type	Meaning
`FriendlyName`	SZ	Label used in audit/logging.
`DataPath`	SZ	Full path to the `.exe` to launch elevated. Without wildcards it is matched as a case-insensitive **prefix** of the launched path: always include the file name, because a folder path approves every program below it. From 1.0.6.0 it may be a wildcard pattern (`*`, `?`); see Wildcard path rules.
`Publisher`	SZ	1.0.6.0+. Leaf certificate CN of the file's embedded Authenticode signature. Required for a wildcard rule unless `HashSHA` is set; enforced on any entry that sets it.
`PublisherThumbprint`	SZ	1.0.6.0+. Optional SHA-1 thumbprint (40 hex) of that certificate, checked in addition to `Publisher`.
`Arguments`	SZ	Optional args. For a standard user the elevated launch line is `"<DataPath>" <Arguments>` and the user's own arguments are discarded, unless `AllowUserArguments=1`. See "Command line and arguments" below.
`AllowUserArguments`	DWORD	Absent or `0` (**default**, locked) = a standard user cannot choose the arguments of the elevated app; only `Arguments` is used. `1` = the user's own command line is passed to the elevated app.
`HashSHA`	SZ	Optional 64-hex SHA-256 of the `.exe`. When present, the file is hash-checked before launch (tamper protection).
`AllowChildElevation`	DWORD	Child-elevation opt-in. Absent or `0` (**default**) = this app's child processes are relaunched **non-elevated** (default-deny; an elevated app is not a springboard for arbitrary elevated children). `1` = AEMS relaunches this app's **direct** children elevated. A child that is not itself an opt-in app is never elevated.
`ChildInPlace`	DWORD	1.0.6.0+. Only together with `AllowChildElevation=1`. Absent or `0` (**default**) = children are blocked and relaunched as a separate copy, as in 1.0.5.0. `1` = the app's direct children start elevated **in place**, as real

Value	Type	Meaning
		children. `2` = the whole tree below the app starts in place, up to 8 levels. See Children in place.
`DoNotAllowChildElevateProcesses`	DWORD	**Deprecated** for the child-elevation decision; `AllowChildElevation` now governs it. Still accepted and sent on the wire for back-compat.
`GlobalApplication`	DWORD	`1` = applies to all users on the machine.

A process is elevated-relaunched only if it is a top-level allow-listed app, or a direct child of a registered-ELEVATE parent. A child that is not itself an opt-in app is never elevated. Elevation only continues down a chain where **every** link is explicitly opted in (X opt-in launches C; C is elevated only if C is itself an allow-listed app with `AllowChildElevation=1`, and so on). A single non-opted-in app stops the chain. This describes entries without `ChildInPlace`. With `ChildInPlace` the children start as real children instead of copies; see Children in place.

> **Security warning.** `AllowChildElevation=1` makes an application a springboard: what it starts runs with administrator rights. Opt in only specific, trusted parent applications, and pin them with `HashSHA` or `Publisher`. A general shell or interpreter (`cmd.exe`, `powershell.exe`, `pwsh.exe`, `wscript.exe`, `python.exe`, ...), a terminal, or a program with an Open dialog or plugins can start any program. Opting such a program in is allowed, but it makes the user an administrator for as long as it runs: with `AllowChildElevation=1` every program it starts directly is elevated, and with "Children in place" set to "Whole tree" every command the user types in an elevated shell is elevated too, up to 8 levels deep. There is no per-child allow-list or hash check for such children. Do this only where giving the user administrator rights for that session is acceptable (for example an elevated PowerShell for a support technician). Where a trusted program only needs specific tools, opt in that program and keep those tools in a folder that standard users cannot change.

The example file `NetworkSettings.reg` (provided by Logic Unknown ApS) holds ready-made elevated launchers (`ncpa.cpl`, `services.msc`, `gpedit.msc`, ...).

Example file NetworkSettings.reg (provided by Logic Unknown ApS)

```
Windows Registry Editor Version 5.00

; ============================================================================
; NetworkSettings.reg
; Publisher: Logic Unknown ApS - https://logic-unknown.com
; Documentation: https://docs.logic-unknown.com/aems/
;
; Example AEMS allow-list entries for running network configuration elevated.
;
; AEMS is a headless service: it transparently and silently elevates the
; applications listed here when a standard (non-admin) user on a kiosk device
; launches them. There is no GUI, tray menu, right-click item or confirmation
; prompt - elevation is governed solely by this allow-list and the hash check
; AEMSvc runs before each elevation.
;
; Allow-list hive (base):    HKLM\SOFTWARE\Logic Unknown\AEMS\Applications\<Name>
; Policy override hive:      HKLM\SOFTWARE\Policies\Logic Unknown\AEMS\Applications\
<Name>
;   (a complete Applications subtree under the Policies hive wins over the base)
; Since AEMS 1.0.6.0 the key is SOFTWARE\Logic Unknown\AEMS. For devices still on
1.0.5.0
; or earlier see "Upgrading from 1.0.5.0 or earlier" in the documentation (the upgrade
; migrates the legacy key).
;
; Per-app values:
;   FriendlyName                    REG_SZ     - label for audit/logging
;   DataPath                        REG_SZ     - full path to a PE .exe
;   Arguments                       REG_SZ     - command-line args (optional)
;   HashSHA                         REG_SZ     - 64-char hex SHA-256 of the .exe
(optional tamper check)
;   AllowChildElevation             REG_DWORD  - 1 = relaunch this app's direct
;                                                children elevated; absent/0 = deny
(default)
;   DoNotAllowChildElevateProccesses REG_DWORD  - deprecated; AllowChildElevation now
decides
;   GlobalApplication               REG_DWORD  - 1 = applies to all users
;
; When Arguments is non-empty, AEMS launches:  "<DataPath>" <Arguments>
; This is what lets you elevate .cpl applets through control.exe and .msc
; snap-ins through mmc.exe.
;
; Apply (elevated):   reg import NetworkSettings.reg
;                     Restart-Service AEMSvc
; AEMS reads its configuration only when the service starts (the license is
; the only value it re-reads while running). At scale, push via
; Intune/SCCM/RMM, or bake into a
; customer MSI with Build-MsiWithSettings.cmd.
; ============================================================================


; Network Connections (ncpa.cpl) -- the classic adapter list, fully elevated.
; control.exe loads ncpa.cpl in-process so the applet inherits the elevated
; token. Adapter IPs, DNS, protocols, etc. can be changed without UAC prompts.
[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Network Connections]
"FriendlyName"="Network Connections (elevated)"
"DataPath"="C:\\Windows\\System32\\control.exe"
"Arguments"="ncpa.cpl"
"GlobalApplication"=dword:00000001
```

```
; Network and Sharing Center -- the higher-level networking page.
[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Network and Sharing
Center]
"FriendlyName"="Network and Sharing Center (elevated)"
"DataPath"="C:\\Windows\\System32\\control.exe"
"Arguments"="/name Microsoft.NetworkAndSharingCenter"
"GlobalApplication"=dword:00000001


; Control Panel home -- general fallback.
[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Control Panel]
"FriendlyName"="Control Panel (elevated)"
"DataPath"="C:\\Windows\\System32\\control.exe"
"GlobalApplication"=dword:00000001


; ------------------------------------------------------------------------
; Other useful elevated launchers using the same Arguments mechanism.
; Uncomment the ones you want and run reg import again.
; ------------------------------------------------------------------------
;
;[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Programs and Features]
;"FriendlyName"="Programs and Features (elevated)"
;"DataPath"="C:\\Windows\\System32\\control.exe"
;"Arguments"="appwiz.cpl"
;"GlobalApplication"=dword:00000001
;
;[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Sound]
;"FriendlyName"="Sound (elevated)"
;"DataPath"="C:\\Windows\\System32\\control.exe"
;"Arguments"="mmsys.cpl"
;"GlobalApplication"=dword:00000001
;
;[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Internet Properties]
;"FriendlyName"="Internet Properties (elevated)"
;"DataPath"="C:\\Windows\\System32\\control.exe"
;"Arguments"="inetcpl.cpl"
;"GlobalApplication"=dword:00000001
;
;[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Services]
;"FriendlyName"="Services (elevated)"
;"DataPath"="C:\\Windows\\System32\\mmc.exe"
;"Arguments"="services.msc"
;"GlobalApplication"=dword:00000001
;
;[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Computer Management]
;"FriendlyName"="Computer Management (elevated)"
;"DataPath"="C:\\Windows\\System32\\mmc.exe"
;"Arguments"="compmgmt.msc"
;"GlobalApplication"=dword:00000001
;
; Note: a few classic .cpl applets redirect to the modern Settings UWP app on
; Win10/11 (e.g. desk.cpl, intl.cpl). Those lose elevation when the redirect
; happens, because UWP runs in AppContainer regardless of the launching token.
; ncpa.cpl, appwiz.cpl, mmsys.cpl, inetcpl.cpl, sysdm.cpl are still classic and
; elevate cleanly.


; ------------------------------------------------------------------------
; To remove these entries:
;   reg delete "HKLM\SOFTWARE\Logic Unknown\AEMS\Applications\Network Connections" /f
```

```
;   reg delete "HKLM\SOFTWARE\Logic Unknown\AEMS\Applications\Network and Sharing
Center" /f
;   reg delete "HKLM\SOFTWARE\Logic Unknown\AEMS\Applications\Control Panel" /f
; ------------------------------------------------------------------------
```

Children in place

From AEMS 1.0.6.0 an approved application can start other programs elevated **in place**: as real child processes of the application, with their output, exit code and handle, so the application can wait for them. This is meant for wrapper programs that call other programs, setup programs that run a helper and wait for it, and scripts that run tools.

Without it (the behaviour of 1.0.5.0, still the default), AEMS stops every child of an elevated application when it is created (the application gets "Access is denied") and starts a separate copy of it: elevated with `AllowChildElevation=1`, unelevated otherwise. The application never sees the copy's output or exit code and cannot wait for it, and a console tool opens in a new window. Wrappers then fail or report a failure.

With `ChildInPlace`, AEMS checks each child while Windows creates it. A child that passes starts in place. A child that fails a check is not started at all: the application gets "Access is denied", and **no copy starts**.

Programs that finish very quickly

From 1.0.8.0 the user's own start of an allow-listed program no longer runs without administrator rights while AEMS decides. The driver holds the start for a moment (normally a few milliseconds, at most 1 second) until AEMS has suspended the new process, before its first instruction runs. Then the start completes normally for the program that started it, and the process stays suspended until AEMS has decided: it is ended and started again elevated, or resumed unchanged when it is not to be elevated (for example another user's start, or no valid license). A program that finishes very quickly therefore runs once, elevated. Starts that the event alone decides (a non-interactive session, a command line that is too long, a program that already runs elevated, no valid license) are not suspended at all.

One visible change: AEMS decides one launch at a time. When several allow-listed programs start at the same moment (for example a logon script that starts three tools), the later ones wait their turn suspended instead of running without administrator rights first. A program that waits for the started program to become ready (`WaitForInputIdle`) waits until AEMS has decided, as it waited for the elevated start before.

When a quick program can still run twice (from 1.0.8.0)

- **When the old driver is still loaded after the upgrade**: the upgrade stops the old driver 1.0.6.0 and starts driver 1.0.8.0 at once. If the old driver is nevertheless still loaded after a completed upgrade (not expected: when the old driver cannot be unloaded, the upgrade fails and asks for a restart), AEMS behaves as 1.0.7.0 and logs warning **2236** once per service start. Restart the device.
- **When the hold timed out**: AEMS did not suspend the new process within 1 second (for example the service was very busy or suspended). The start then continues without administrator rights as in 1.0.7.0, and warning **2235** names the program. After two such timeouts within 30 seconds the driver holds nothing for 30 seconds (the hold breaker), so a hung service delays launches by at most 2 seconds per 30 seconds.
- **When the suspend could not be checked** (no thread found in the new process, or the process could not be opened): handled as in 1.0.7.0, warning 2235.
- **When AEMS is busy with one decision for more than 30 seconds** (for example a hash check of a file on a slow disk): new launches are not suspended until that decision is done, so they do not wait behind it; warning 2235 ("the decision worker is busy") once for that period.

In these cases the 1.0.7.0 behaviour applies: the original start runs without administrator rights for a fraction of a second, and a wrapper that does its whole work in that time runs twice (measured with 1.0.6.0: a quick chain of four programs ran twice in 10 of 10 starts). If AEMSvc ends unexpectedly while a program is suspended, the driver ends that program; start it again.

The first fraction of a second (1.0.7.0, and from 1.0.8.0 the cases in the box above). When a standard user starts the allow-listed application, the user's own start runs un-elevated for the short time AEMS needs to decide, and is then ended and replaced by the elevated start. A wrapper can already start programs in that time. AEMS ends those un-elevated programs together with the original (event **1224**), so they do not run twice; if it cannot check or end all of them it logs warning **2232**. AEMS looks once (one snapshot of the running processes) and ends only processes in the original's process chain, in the same session and of the same user, not elevated, created after their parent and after the original, up to 8 levels and 64 processes. What remains: a program that already did its work in that fraction of a second, a program started after that one look, and programs that broke away from the original's process chain (for example started through another process, or after their own parent ended) are not ended. When the original had already exited before AEMS decided (event 1010), nothing is ended.

The machine-wide value `EndOriginalDescendants` (REG_DWORD, see the settings reference) switches this off: `0` = AEMS never ends those programs, as in 1.0.5.0, and logs event **1225** at service start. Set it to `0` only when a program that the application starts must not be interrupted in its first fraction of a second (for example a tool that changes something it cannot undo halfway). The cost: such a program then runs twice, once un-elevated from the user's original start and once from the elevated start.

The folder rule: read this first

A program is started in place only when it lies in a folder that standard users cannot change. The program file, its folder and every folder above it must be writable only by SYSTEM, Administrators or TrustedInstaller. Otherwise any user could replace the program and have it run with administrator rights.

- `C:\Program Files\...` and `C:\Windows\...` qualify.
- **A folder created directly in the drive root does NOT qualify by default**, for example `C:\Contoso`, even when an administrator or an installer created it. On a default Windows installation `C:\` passes Modify for Authenticated Users on to every new folder below it. Wrapper programs and the programs they call often lie in exactly such a folder.

Harden such a folder once, from an elevated prompt. These are the same three commands as in "Check that the folder is trusted" under Wildcard path rules, with the folder name of the example:

```
icacls C:\Contoso
icacls C:\Contoso /inheritance:r /grant:r "*S-1-5-18:(OI)(CI)F" "*S-1-5-32-544:(OI)
(CI)F" "*S-1-5-32-545:(OI)(CI)RX"
icacls C:\Contoso /setowner "*S-1-5-32-544" /T
```

The first command shows the current permissions. The second removes the inherited entries and grants SYSTEM and Administrators full control and Users read and execute. The third makes Administrators the owner of the folder and of everything in it. Subfolders and files inherit the hardened permissions, also the ones you add later; a file or subfolder that was given its own explicit permissions keeps them, so check it separately.

What this does to the folder:

- **Standard users can no longer write in `C:\Contoso`.** If a program there writes its own log, settings or data files while it runs without elevation, give it a separate data subfolder that contains no programs. Only the folders that contain programs, and the folders above them, must be closed to users.
- **Other explicit entries stay.** The second command replaces the entries of SYSTEM, Administrators and Users only. An explicit entry for any other account or group on the folder stays; the check below shows it. Remove it with `icacls C:\Contoso /remove:g "<account>"` and list the whole folder with `icacls C:\Contoso /T` (not measured).
- **Copy new programs into the folder, do not move them.** A file moved from another folder on the same drive can keep its old owner and permissions. If you moved files, run the second and third command again.
- **Subfolders inherit.** Subfolders and files created later in `C:\Contoso` get the hardened permissions automatically.

Check. Run `icacls C:\Contoso` again. Every line must name SYSTEM, Administrators or Users, and Users must have `(RX)` only:

```
C:\Contoso NT AUTHORITY\SYSTEM:(OI)(CI)(F)
           BUILTIN\Administrators:(OI)(CI)(F)
           BUILTIN\Users:(OI)(CI)(RX)
```

A subfolder shows the same entries marked `(I)` (inherited). An entry such as `NT AUTHORITY\Authenticated Users:(OI)(CI)(M)` or `Everyone:(F)` means the folder is still writable by users. The account names are shown in the language of Windows. To see the owners too, use the PowerShell loop under Wildcard path rules.

What you see when the folder is wrong. The program that should start in place does not start. The application that started it gets "Access is denied" (a console shows `Access is denied.`), no copy starts, and the Application log has a warning from source `AEMS` **2213** that reads `Start of a child refused [PathGuard]: '<program>' ...` and names the reason (for example `WritableByOther` with the account and the right). Harden the folder and start the application again. The folder is checked at every start, so a folder change needs no service restart.

An exact-path entry is not protected by the folder rule. AEMS checks the folder only for wildcard rules and for programs started in place. An application with an exact-path entry (`DataPath` without `*` or `?`) starts elevated from any folder, also from one that standard users can change. In such a folder any standard user can replace the program, or put a library (DLL) next to it, and have their own code run with administrator rights. `HashSHA` on the entry stops a replaced program file, but not a planted library. Hardening the folder is therefore not only what makes the called programs start: it is what makes the entry safe at all. Harden the folder before you add the entry (see Folders of elevated programs).

The typical sign of a folder that is still open (measured): the wrapper starts elevated (event 1009), and every program it calls is refused with warning **2213**, for example:

```
Start of a child refused [PathGuard]: 'C:\Contoso\tool.exe' (pid <n>), started by
elevated pid <n>
in session 1, request <id>: [WritableByOther] the file 'C:\Contoso\tool.exe' grants
FILE_WRITE_DATA, FILE_APPEND_DATA, FILE_WRITE_EA, FILE_WRITE_ATTRIBUTES, DELETE to S-1-
5-11
(NT AUTHORITY\Authenticated Users); only SYSTEM, BUILTIN\Administrators and NT
SERVICE\TrustedInstaller
may modify it. Standard users can change the folder or file named above (typical for a
folder
directly under C:\, such as C:\Contoso), so programs in it are not started elevated ...
```

The event names the file, the rights and the group. In addition AEMSvc logs warning **2233** once at every service start that names the exact-path entries on local fixed disks whose program or folder standard users can change. The entries keep working; the warning tells you which folders to harden. After the three commands above, the called program, a chain of three levels and a program in a subfolder start in place (measured).

Turn it on

Verify it with a standard user account. A launch by an administrator uses the administrator passthrough: the application and its children run with the administrator's own rights, AEMS does not check the children and writes no "started in place" events. Trying it with an administrator account therefore shows nothing about children in place.

1. Put the application and the programs it calls in a folder that standard users cannot change (see the folder rule above).
2. In the application's entry set `AllowChildElevation=1` and `ChildInPlace`. In `AEMSAllowlist.exe` 1.3.0.0: check **Allow child elevation**, then choose **Children in place**: "Off", "Direct children start in place" or "Whole tree starts in place (up to 8 levels)".
3. Restart the AEMS service (`Restart-Service AEMSvc`): settings are read at service start, so a changed entry takes effect at the next service start. Children in place need the driver from AEMS 1.0.6.0 or later; an upgrade loads the new driver at once. If warning 2220 says that the loaded driver is the old one, restart the device.
4. Start the application as a **standard user** (see above). For each program started in place the Application log has information event **1200** that reads `Child '<program>' (pid <n>) of pid <n> started in place, elevated ...`; many starts of one application within a minute are summarised in event **1202**. How to read the event text: Reading AEMS events.

Example: a wrapper in `C:\Contoso` that calls other programs, which may call further programs:

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Contoso wrapper]
"FriendlyName"="Contoso wrapper"
"DataPath"="C:\\Contoso\\wrapper.exe"
"HashSHA"="0000000000000000000000000000000000000000000000000000000000000000"
"AllowChildElevation"=dword:00000001
"ChildInPlace"=dword:00000002
"GlobalApplication"=dword:00000001
```

Replace the hash with the SHA-256 of the real `wrapper.exe` (or use `Publisher` instead when the wrapper is signed).

Which setting.

Setting	Value	What starts in place
Off	`0` or absent	Nothing. Children are blocked and relaunched as a copy, exactly as in 1.0.5.0.
Direct children	`1`	The programs the application starts itself. What those programs start is refused (no copy), unless that program has its own entry with `AllowChildElevation=1` and `ChildInPlace`.
Whole tree	`2`	Every program below the application, up to 8 levels deep. None of them needs its own allow-list entry. A program deeper than 8 levels is refused.

Use "Whole tree" for a wrapper that calls programs which may call further programs (for example a wrapper that runs `cmd /c`, a script, or a setup program that runs its own helpers). Use "Direct children" when the application only starts programs that start nothing themselves. `ChildInPlace` has no effect without `AllowChildElevation=1`; AEMS then treats it as off and logs warning **2230** at service start that names the entry. The same happens for a value other than 0, 1 or 2, or a value that is not a DWORD.

A program in the tree that has its own allow-list entry follows that entry's rules. The refusing rows win, in this order: a refusing entry, then `HashSHA`, then locked arguments.

The child's own entry	Result
none	Starts in place under the application's setting.
without `AllowChildElevation=1`, or with a `Publisher` or `PublisherThumbprint` pin	Starts in place; what it starts is refused (the chain ends there). The pin itself is not checked for an in-place start; the folder rule is what protects the file.
`AllowChildElevation=1` without `ChildInPlace`	Starts in place; what it starts is refused.
`AllowChildElevation=1` and `ChildInPlace`, no publisher pin	Starts in place; from there its own setting applies, and the 8 levels count again from it.
locked arguments (`AllowUserArguments` absent or `0`)	Starts only when its command line after the program name equals the entry's `Arguments` exactly (leading and trailing blanks ignored; both empty counts as equal). Otherwise refused `[ArgumentsLocked]`. Locked arguments stay locked in place.
with `HashSHA`	Refused in place (from 1.0.6.0) `[HashedEntryNoPrecompute]`. Remove the hash from the child's entry, or leave the child without an entry of its own.
a refusing entry (malformed pin, inactive wildcard rule)	Refused `[PathGuard]`; the event text ends with `own entry '<name>'` and names no folder. Fix that entry (see warning 2012).

What is refused in place. Each refusal means: the program is not started, the application that started it gets "Access is denied", no copy starts, and AEMS logs a warning that names the program and the reason in square brackets.

Situation	What the user sees	What you can do
The program lies in a folder that standard users can change: the user's `%TEMP%`, Downloads, Desktop or profile, a folder under `C:\ProgramData` that users can write, or a folder directly in the drive root with its default permissions `[PathGuard]`	The application reports a failed start or "Access is denied".	Harden the folder (above) or install the program under `C:\Program Files`.
An installer or setup program unpacks a helper into the user's `%TEMP%` and starts it (self-extracting installers, many bootstrappers) `[PathGuard]`	The setup fails or reports an error.	Documented limit (from 1.0.6.0): leave "Children in place" off for that entry. The helper is then blocked and relaunched as a copy, as in 1.0.5.0.
The program is started through a drive letter created with `subst` or another alias letter, through a symbolic link or a junction (for example a Java `javapath` folder, the winget `Links` folder or a Scoop `current` folder) `[PathAlias]`	The application reports a failed start or "Access is denied".	Start the program by its real long path on a local drive.
The program is started by its 8.3 short name (`C:\PROGRA~1\...`) `[PathGuard]`, detail `[NotCanonical]` (not measured)	The application reports a failed start or "Access is denied".	Start the program by its long path.
The program lies on a mapped network drive, on a UNC path (`\\server\share\...`), on removable media such as a USB stick, or in a mounted ISO `[VolumeNotFixed]`	The application reports a failed start or "Access is denied".	Copy the program to a protected folder on a local fixed drive.
The program lies on a volume that is not NTFS (ReFS, Dev Drive, FAT, exFAT) `[FileIdentity]`	The application reports a failed start or "Access is denied".	Move the program to an NTFS volume.

Situation	What the user sees	What you can do
The program is on a drive letter that the user mapped or created with `subst`	Not found. An elevated program does not see the user's own mapped network drives or `subst` letters at all.	Use a real local path, or a drive letter that is defined for the whole machine.
A Chromium-based browser, an Electron program or a program that hosts Edge WebView2 starts its sandboxed helper processes	The program starts, but its helpers do not: blank windows, or the program exits.	Do not use "Children in place" for such programs.
The application has more than 6 child starts waiting for AEMS at the same moment	The extra start gets "Access is denied".	Starts are refused, not queued; the application can retry. Most wrappers start one program at a time and are not affected.
The tree is deeper than 8 levels (with "Direct children": deeper than 1 level)	The deeper program does not start.	Use "Whole tree", or give the program at the deepest working level its own entry with `ChildInPlace`.
The application was started before the AEMS service was last restarted `[CreatorNotHeld]`	The programs it starts are refused.	Close the application and start it again.
The AEMS service is stopped or restarting	Programs that in-place applications start do not start.	Wait for the service; start the application again after a restart.

Packaged apps and tools that a program finds by name on the user's own `PATH` (for example the `WindowsApps` aliases in the user's profile) lie in user folders or are reached through links, so they are expected to be refused in place (not measured).

When "Children in place" is configured but cannot work. If at least one entry has `ChildInPlace` but in-place children cannot start, AEMSvc logs warning **2220** "Children in place (ChildInPlace) is configured for ... but is not active: ..." once at service start, and again when the state changes (event **1222** when it is active again). The reason is one of: the loaded AEMS driver is the old one (restart the device; the upgrade normally loads the new driver at once), AEMS has

no valid license (event 3013), or the service could not connect to the driver. For the reasons old driver and connection not ready, those entries behave like `AllowChildElevation=1` without `ChildInPlace`: children are blocked and relaunched as an elevated copy, and AEMS logs warning **2225** for each launch. When AEMS has no valid license nothing is elevated at all (event 2011).

What does not change. Entries without `ChildInPlace` (also with `AllowChildElevation=1`) handle child processes exactly as in 1.0.5.0. Top-level elevation, administrator passthrough, wildcard rules, licensing and `TamperProtection` are not affected by the setting.

Speed. Each child waits while AEMS checks it, at most 2 seconds. In a measurement on a virtual machine, a short console command (`cmd /c exit 0`) started in place took as long as without the check (median about 33 ms), and 50 starts in a row ran without a timeout.

Wrapper patterns measured with "Whole tree": a program called directly, through `cmd /c`, through `powershell -File`, a GUI program that starts a console program, and a chain of three levels. Each called program ran exactly once, elevated (for the quick-program limit see the note at the top of this section).

Security consequences and the limits that cannot be closed: Children in place. Problems: A child program does not start in place.

Command line and arguments

From driver protocol 2 (AEMS.msi 1.0.3.0) the driver forwards the full command line of every caught launch. What the elevated relaunch gets depends on who launched it:

Relaunch	Command line used	Why
Standard user, entry locked (default)	`"<DataPath>" <Arguments>`, or no arguments - exactly as 1.0.2.0	A standard user must not choose what an elevated app runs.
Standard user, entry has `AllowUserArguments=1`	the user's command line verbatim; `Arguments` only when the user gave none	Explicit administrator decision per app.
Administrator (passthrough, incl. runas with an admin account)	the original command line verbatim; `Arguments` only when none	The administrator already owns the machine.
Governed child, default-deny (relaunched non-elevated)	the child's own command line	The child runs unelevated, as the user, so its arguments grant nothing.
Governed child of an opt-in parent (`AllowChildElevation=1`, relaunched elevated)	the child's own command line	Opting a parent in is already the explicit administrator decision to elevate what it launches, with the arguments it launches it with.
Child started in place (`ChildInPlace`)	the command line, working directory and environment the parent chose; nothing is relaunched	The child is the parent's real child. If the child has its own entry with locked arguments, it starts only with exactly those arguments (see Children in place).

Event 1009 for a top-level relaunch ends with `(arguments: configured)` or `(arguments: user)`. A command line longer than 2047 characters is never relaunched truncated: a top-level launch is left running unelevated (event 2004), a governed child is not relaunched (event 3009). The working directory is the application's own folder.

> **Security warning for `AllowUserArguments=1`.** Set it only for applications whose command line cannot be turned into arbitrary elevation. Never set it on a host that executes what its arguments name (`cmd.exe`, `powershell.exe`, `mmc.exe`, `rundll32.exe`, `regedit.exe`, script hosts, ...): an allow-listed `mmc.exe` with user arguments elevates any `.msc` the user names, and `cmd.exe` elevates `cmd /c <anything>`. Prefer a locked entry with fixed `Arguments`, pinned with `HashSHA`. On an unlocked

> entry `argv[0]` (the program name inside the command line) is also chosen by the user, so the application must not trust `argv[0]` to locate itself, its plugins or its DLLs (some Python, Java and Electron launchers and installers do). Locked entries are safe: AEMS always builds `"<DataPath>" <Arguments>` itself.

Wildcard path rules

From AEMS 1.0.6.0 an entry can approve an executable by a **path pattern** instead of one exact path, so you do not need to know the exact file name (or its hash) in advance, for example a vendor's versioned setup program. The exact rules are in the Settings reference - Wildcard path rules; the short version:

- An entry is a wildcard rule when `DataPath` contains `*` or `?`.
- `?` is one character, `*` is any number of characters, and **neither ever matches a** `\`: a wildcard stays inside one folder name or the file name. The pattern must match the whole path (case-insensitive).
- The pattern is a local path (`C:\...`), the drive and the first folder are literal, the file name ends with `.exe`, the whole pattern is at most 251 characters, `GlobalApplication` is `1`, and `Publisher` or `HashSHA` is set. A rule that breaks one of these is rejected at service start (warning 2012) and does nothing; so is any entry with a missing or empty `DataPath`. A malformed `HashSHA` (not 64 hex digits) or `PublisherThumbprint` (not 40 hex digits), on any entry, is different: the entry is kept in its place as a refusing entry (2012 once at start), and every launch it matches is refused with 2013 `[MalformedRule]`, so a later, broader entry cannot elevate the same file without the pin. Correct the value and restart the AEMSvc service.
- At launch AEMS checks two guards on the concrete file before it elevates:
- **trusted location**: a fixed drive on an accepted storage bus, a global drive letter, no junctions or symbolic links in the path, and the file and every folder above it owned by and writable only by Administrators, SYSTEM or TrustedInstaller;
- **identity**: a valid embedded Authenticode signature from `Publisher` (plus `PublisherThumbprint` when set), and/or the SHA-256 in `HashSHA`.

If a guard fails, the user's program runs **unelevated** and AEMS logs warning 2013 with the reason. The guards protect top-level launches and a child's own rule, not the children of a parent with `AllowChildElevation=1` and no `ChildInPlace` (children started in place always get the folder check, see Children in place): such a parent elevates its direct children whatever rule they match (a child that fails a `Publisher`, thumbprint or location check loses only its own entry's settings). A child whose file fails its own entry's hash, or whose entry is malformed or an inactive wildcard rule, is not started at all. When the launching user is an administrator, a failed check does not block: AEMS relaunches the app with the administrator's own real token and logs warning 2016. - Limits you should know before you rely on wildcard rules: - the storage check is by **bus**. Removable media and disks on the USB, FireWire (1394), SD, MMC (eMMC), SSA, iSCSI, Fibre Channel, virtual or unknown buses (including mounted VHD/VHDX files) are refused, and so are

a volume that cannot be queried and drive letters created for one user (`subst` , mapped drives). Disks on an accepted bus (SATA, SAS, SCSI, ATA/ATAPI, NVMe, RAID, Storage Spaces, SCM, UFS) are accepted even when they are attached externally, for example an eSATA or Thunderbolt/USB4 NVMe enclosure, external SAS, or a Storage Spaces pool built on USB disks: they are not detected as external. Point wildcard rules at the system drive; - a `DELETE` right on a volume root is ignored, so the default root of a data volume such as `D:\` does not block a rule; every other right on the root still counts; - certificate revocation is not checked online (local cache only), so a revoked certificate may still be accepted. `PublisherThumbprint` or `HashSHA` is the strict option; - `Publisher` compares the certificate's common name only; - a launch can be delayed, or refused, while another process holds the file open. - AEMS elevates a matching file when a user **launches** it. Nothing happens when a file merely appears in the folder. - Wildcard rules need the driver from AEMS 1.0.6.0 (driver 1.0.6.0, protocol `0x00020003` ; wildcard matching needs protocol `0x00020002` or later). The upgrade loads the new driver at once; if 2014 or 3011 shows that the old driver is still loaded, restart the device, and again if it is still logged. With an older driver AEMS logs warning 2014 once at start and never elevates through wildcard rules; a launch that a wildcard rule matches first is refused (2013 `[WildcardInactive]` , administrators 2016), even when a later entry would match it; other exact entries keep working. If 2014 is still logged after the reboot, the installed package does not contain driver 1.0.3.0 or later: install the current AEMS MSI (1.0.8.0) from Logic Unknown ApS.

Example 1 - a vendor setup program with a changing version in the name, pinned by publisher. Elevates `setup_2.4.1.exe` , `setup_x64.exe` , and so on, in that one folder only:

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Contoso setup]
"FriendlyName"="Contoso setup (any version)"
"DataPath"="C:\\Program Files\\Contoso\\Tools\\setup_*.exe"
"Publisher"="Contoso Ltd"
"GlobalApplication"=dword:00000001
```

Example 2 - one program in a versioned folder, pinned by publisher and certificate thumbprint. Matches `C:\Program Files\Fabrikam\Console 7.2\console.exe` and `...\Console 8.0\console.exe` , but not `...\Console 10.1\console.exe` (`?` is exactly one character) and not `...\Console 7.2\bin\console.exe` (no folder crossing):

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Fabrikam console]
"FriendlyName"="Fabrikam console"
"DataPath"="C:\\Program Files\\Fabrikam\\Console ?.?\\console.exe"
"Publisher"="Fabrikam, Inc."
"PublisherThumbprint"="0123456789ABCDEF0123456789ABCDEF01234567"
"GlobalApplication"=dword:00000001
```

Both are in the example file `WildcardRules.reg` (provided by Logic Unknown ApS). Replace the vendor names and the thumbprint with the real values. `AEMSAllowlist.exe` 1.3.0.0 (the tool for

AEMS 1.0.6.0) writes such entries for you and validates them with the same rules.

Example file WildcardRules.reg (provided by Logic Unknown ApS)

```
Windows Registry Editor Version 5.00

; ============================================================================
; WildcardRules.reg
;
; Example AEMS wildcard path rules (AEMS 1.0.6.0 and later).
; Publisher: Logic Unknown ApS - https://logic-unknown.com
; Documentation: https://docs.logic-unknown.com/aems/
;
; A wildcard rule approves an executable by a path PATTERN instead of one exact
; path. An entry is a wildcard rule when DataPath contains * or ?.
;   ?  matches exactly one character that is not a backslash
;   *  matches zero or more characters that are not a backslash
;   Wildcards never cross a folder boundary; the pattern must match the WHOLE
;   path (case-insensitive).
;
; Rules (an entry that breaks one is rejected at service start, with a warning):
;   - local path X:\... with a literal drive letter (no UNC, \\?\ or devices)
;   - the drive and the first folder are literal; no file directly in X:\
;   - no / " < > |, control characters or a second colon; no empty, . or ..
;     folder component
;   - the file name ends with the literal .exe
;   - at most 251 characters (refused, never truncated)
;   - Publisher or HashSHA is set, and GlobalApplication = 1
; A malformed HashSHA (not 64 hex digits) or PublisherThumbprint (not 40 hex
; digits) does not drop the entry: it is kept as a refusing entry (warning
; 2012) and every launch it matches is refused (2013 [MalformedRule]).
; Use a folder under C:\Program Files: a new folder directly in the drive root
; (for example C:\Tools) normally lets users write and fails the check below.
;
; At launch AEMS elevates a matching file only when BOTH guards pass:
;   1. trusted location: a fixed drive on an accepted storage bus (removable
;      media and USB, FireWire, SD/eMMC, iSCSI, Fibre Channel and mounted
;      virtual disks are refused; the check is by bus, so an external NVMe or
;      eSATA enclosure is NOT detected - use the system drive) with a global
;      drive letter
;      (a drive letter created or redirected for one user, for example with
;      subst or a mapped drive, is refused), no junction/symbolic link in the
;      path, the file and every folder above it owned by and writable only by
;      Administrators, SYSTEM or TrustedInstaller;
;   2. identity: valid EMBEDDED Authenticode signature whose leaf certificate
;      CN equals Publisher (and SHA-1 thumbprint equals PublisherThumbprint
;      when set), and/or SHA-256 equals HashSHA.
; Otherwise the program runs unelevated and AEMS logs a warning with the
; reason. Elevation happens when a user launches the file, never when a file
; merely appears in the folder.
;
; New per-app values:
;   Publisher            REG_SZ - leaf certificate CN of the embedded signature
;   PublisherThumbprint  REG_SZ - optional SHA-1 thumbprint (40 hex) of that
;                                 certificate, checked in addition to Publisher
;
; Requires the AEMS 1.0.6.0 driver (protocol 0x00020002), started by the
; upgrade (reboot if warning 2014 says the old driver is still loaded). Contoso and
Fabrikam are placeholder vendors: replace the
; paths, the publisher names and the thumbprint with the real values
; (Get-AuthenticodeSignature, or "Read from file..." in AEMSAllowlist.exe).
;
; Apply (elevated):   reg import WildcardRules.reg
;                     Restart-Service AEMSvc
```

```
; Details: https://docs.logic-unknown.com/aems/configure/ (Wildcard path rules).
; ============================================================================

; Example 1 - a vendor setup program with a changing version in the file name,
; pinned by publisher. Matches C:\Program Files\Contoso\Tools\setup_2.4.1.exe
; and setup_x64.exe; does not match C:\Program Files\Contoso\Tools\old\setup_1.exe.
[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Contoso setup]
"FriendlyName"="Contoso setup (any version)"
"DataPath"="C:\\Program Files\\Contoso\\Tools\\setup_*.exe"
"Publisher"="Contoso Ltd"
"GlobalApplication"=dword:00000001

; Example 2 - one program in a versioned folder, pinned by publisher and
; certificate thumbprint. Matches ...\Console 7.2\console.exe and
; ...\Console 8.0\console.exe; not ...\Console 10.1\console.exe (? is exactly
; one character). The thumbprint changes when the vendor renews its
; certificate; leave PublisherThumbprint out to survive renewals.
[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Fabrikam console]
"FriendlyName"="Fabrikam console"
"DataPath"="C:\\Program Files\\Fabrikam\\Console ?.?\\console.exe"
"Publisher"="Fabrikam, Inc."
"PublisherThumbprint"="0123456789ABCDEF0123456789ABCDEF01234567"
"GlobalApplication"=dword:00000001

; ----------------------------------------------------------------------------
; Example 3 (commented out) - a file that is only catalog-signed or unsigned
; cannot satisfy Publisher; pin its SHA-256 instead. The rule then elevates
; only files with exactly this content.
; ----------------------------------------------------------------------------
;[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Contoso updater]
;"FriendlyName"="Contoso updater"
;"DataPath"="C:\\Program Files\\Contoso\\Updater\\update*.exe"
;"HashSHA"="0000000000000000000000000000000000000000000000000000000000000000"
;"GlobalApplication"=dword:00000001

; ----------------------------------------------------------------------------
; To remove these entries:
;   reg delete "HKLM\SOFTWARE\Logic Unknown\AEMS\Applications\Contoso setup" /f
;   reg delete "HKLM\SOFTWARE\Logic Unknown\AEMS\Applications\Fabrikam console" /f
; ----------------------------------------------------------------------------
```

Find the publisher name and thumbprint. Use a sample of the signed file:

- Explorer: right-click the file -> Properties -> **Digital Signatures**. The "Name of signer" is the `Publisher` value. Select it -> Details -> View Certificate -> Details -> Thumbprint for `PublisherThumbprint` (remove the spaces). If there is no Digital Signatures tab, the file has no embedded signature.
- PowerShell:

```
$s = Get-AuthenticodeSignature 'C:\Program Files\Contoso\Tools\setup_2.4.1.exe'
$s.Status                                          # must be Valid
$s.SignatureType                                   # Authenticode = embedded signature;
if it shows
                                                   # Catalog, check the Digital
Signatures tab
$s.SignerCertificate.GetNameInfo('SimpleName', $false)   # -> Publisher
$s.SignerCertificate.Thumbprint                    # -> PublisherThumbprint
```

- `AEMSAllowlist.exe`: **Read from file...** next to Publisher fills both from the file's embedded signature.

An embedded Authenticode signature is required. A file that is only catalog-signed (no Digital Signatures tab) does not qualify for `Publisher`; pin its `HashSHA` instead. PowerShell reports `Catalog` for some files that have both kinds of signature; such a file qualifies.

A thumbprint changes when the vendor renews its certificate; `Publisher` alone survives a renewal. Pin the thumbprint only when you accept updating the rule at each renewal.

Check that the folder is trusted. The file and every folder from the drive root down to it must be writable only by administrators. `C:\Program Files\...` and `C:\Windows\...` normally pass; folders under `C:\Users\...`, `C:\ProgramData\...` (by default users can create files there) and `%TEMP%` fail.

A new folder directly in the drive root fails too, even when an administrator or an installer created it: on a default Windows installation `C:\` passes Modify for Authenticated Users on to every new folder below it (for example `C:\Contoso`). Put the files in a folder under `C:\Program Files`. If a folder in the drive root is required, inspect it and then remove the inherited entries and grant only administrators and SYSTEM full control and users read and execute:

```
icacls C:\Contoso
icacls C:\Contoso /inheritance:r /grant:r "*S-1-5-18:(OI)(CI)F" "*S-1-5-32-544:(OI)
(CI)F" "*S-1-5-32-545:(OI)(CI)RX"
icacls C:\Contoso /setowner "*S-1-5-32-544" /T
```

(`S-1-5-18` = SYSTEM, `S-1-5-32-544` = Administrators, `S-1-5-32-545` = Users; the SIDs work on every Windows language. The owner step is needed when a user created the folder or a file in it.)

List the owner and allow entries of the file and of each folder above it:

```
$f = 'C:\Program Files\Contoso\Tools\setup_2.4.1.exe'
$acl = Get-Acl -LiteralPath $f
'{0}  owner={1}' -f $f, $acl.Owner
$acl.Access | Where-Object { $_.AccessControlType -eq 'Allow' } |
    Format-Table IdentityReference, FileSystemRights, PropagationFlags -AutoSize
$p = Split-Path -Path $f -Parent
while ($p) {
    $acl = Get-Acl -LiteralPath $p
    '{0}  owner={1}' -f $p, $acl.Owner
    $acl.Access | Where-Object { $_.AccessControlType -eq 'Allow' } |
        Format-Table IdentityReference, FileSystemRights, PropagationFlags -AutoSize
    $p = Split-Path -Path $p -Parent
}
```

Any allow entry for a SID other than `NT AUTHORITY\SYSTEM`, `BUILTIN\Administrators` or `NT SERVICE\TrustedInstaller` that grants Write, Modify, FullControl, Delete, ChangePermissions or TakeOwnership on the file or its folder (on higher folders: Delete, delete-subfolders-and-files, ChangePermissions, TakeOwnership) makes the check fail. `icacls <folder>` shows the same information. Rows with PropagationFlags `InheritOnly` do not count for the folder they are listed on (this is the normal state of `C:\`). Junctions and symbolic links in the path also fail; use the real path.

Troubleshooting by event ID (Application log, source `AEMS`):

Event	What it means	What to do
2012 (warning, at start)	A wildcard rule that breaks a rule, or an entry with a missing or empty `DataPath`, was dropped; or an entry with a malformed `HashSHA` or `PublisherThumbprint` was kept as a refusing entry. The text names the entry and the rule it breaks.	Fix the pattern (literal drive and first folder, no `/ " < > |` or empty, `.` or `..` component, `.exe` at the end, at most 251 characters), add `Publisher` / `HashSHA`, correct a malformed `HashSHA` (64 hex) or `PublisherThumbprint` (40 hex), set `GlobalApplication=1`, then `Restart-Service AEMSvc`.
2013 (warning, at launch)	A launch matched the rule but a guard failed, so the rule did not elevate it. A top-level program ran unelevated. For a child: with `HashMismatch`, `HashFailed`, `MalformedRule` or `WildcardInactive` the child was NOT started at all; otherwise the text ends with "relaunched UNELEVATED" (its parent does not allow child elevation) or "IS still relaunched elevated ... by design" (its parent has `AllowChildElevation=1`). `WildcardInactive` = the loaded driver is older than the one shipped in 1.0.6.0 (see 2014). Codes: `MalformedRule` = the entry's `HashSHA` or `PublisherThumbprint` is malformed (see 2012); `RemovableVolume` = the device reports removable media, or the disk is on a refused storage bus (the text names it: USB, 1394 (FireWire), SD, MMC, SSA, Fibre Channel, iSCSI, virtual or file-backed virtual such as a mounted VHD, VHDX or ISO, unknown); `NotFixedVolume` naming the "original's logon session" or the "logon session of the elevated process" = the drive letter is redirected for that user (for example by `subst` or a per-user mapping). The text names the image, the entry and the reason: which folder, SID and right; owner; reparse point; drive type or alias; file open	Fix the folder permissions (see "Check that the folder is trusted"), use the real (non-junction) long path instead of an 8.3 short path, retry after a copy into the folder has finished, move the file to the system drive (`RemovableVolume`), use a real local path instead of a redirected drive letter (`NotFixedVolume`), or correct `Publisher` / `PublisherThumbprint` / `HashSHA` and restart AEMSvc (`MalformedRule`).

Event	What it means	What to do
	for writing; short or non-canonical path; or the signature problem (no embedded signature, untrusted, wrong publisher, wrong thumbprint, hash mismatch).	
2016 (warning, at launch)	ADMINISTRATOR OVERRIDE: a guard failed, but the user is an administrator, so the app was relaunched elevated with the administrator's own real token.	Fix the rule or the folder as for 2013; a standard user would have been refused.
2003 (warning, at launch)	The driver caught the launch but AEMS found no matching entry (or, for an administrator, the file does not match `HashSHA`). Standard user: NOT elevated, the program keeps running unelevated (followed by 2004). Administrator: relaunched with the administrator's own real token (passthrough). Child of a governed parent: relaunched with child elevation denied.	Use the AEMS driver and service from the same MSI, and check that `DataPath` matches the path.
2014 (warning, at start)	The loaded driver predates wildcard support; wildcard rules are not active: they never elevate, and launches they match first are refused (2013 `[WildcardInactive]`).	Install the current AEMS MSI; its upgrade normally unloads the old driver and starts the new one. If 2014 is still logged after that, the old driver is still loaded: reboot. If it stays after the reboot, the installed package does not contain driver 1.0.3.0 or later: install the current AEMS MSI (1.0.8.0) from Logic Unknown ApS.
5000 (error)	Unexpected error; AEMSvc may have stopped itself after an internal safety failure. If AEMSvc ends unexpectedly, Windows restarts it automatically after 5 seconds, twice; after a third failure within a day it stays stopped. Event 5000	If AEMSvc stays stopped, start it manually (`Start-Service AEMSvc`) and contact support.

Event	What it means	What to do
	is in the Application log (source `AEMS`); the restarts and the final stop appear as Service Control Manager events in the System log.	
3009 (error, at launch)	Elevation failed; the text names the step. For a standard user and a hash-only entry: the file no longer matches `HashSHA` ("refusing to relaunch", followed by 2004).	After a vendor update, set the new `HashSHA` and restart AEMSvc; otherwise investigate the file. For other steps see the event reference.
1009 (information)	The launch was elevated; the text names the rule.	Nothing.
no event	The launch did not match any rule (the driver did not catch it).	Compare the full path with the pattern: `*` and `?` never match `\`, and the whole path must match. Use **Test against a file...** in `AEMSAllowlist.exe`.

Administrator users

When the console user is a **local administrator** (directly, or through a nested local or domain group that is a member of the local `Administrators` group), AEMS behaves like "Run as administrator" without the consent prompt:

- the allow-listed app is relaunched with the user's **real** full token (the UAC linked token, or the user's own token when UAC is off), not a token built by AEMS;
- the relaunch is **not** registered for child governance, so every child process inherits that real token untouched (no deny, no de-elevation, no loss of arguments). Exception: if the child's own image path is itself an allow-listed path, the driver still intercepts it as a top-level launch and the relaunch gets only that entry's configured `Arguments`.

The decision is made only from what Windows put in the user's logon token (`BUILTIN\Administrators` enabled and the token elevated). A standard user cannot opt in, and an administrator gets nothing they could not already get by accepting a UAC prompt. Standard

users are unaffected: they keep the kiosk model (AEMS-built elevated token + default-deny child governance).

The `HashSHA` pin is informational for administrators and enforced for standard users: if an administrator has updated an allow-listed app, the relaunch still happens with the administrator's real token (warning 2003 asks you to update `HashSHA`; for an entry with `Publisher`, `PublisherThumbprint` or a wildcard rule the warning is 2016), while a standard user is refused on any hash mismatch: for a hash-only entry that is error 3009 ("On-disk hash ... no longer matches the configured HashSHA; refusing to relaunch") followed by 2004, for an entry with `Publisher`, `PublisherThumbprint` or a wildcard rule it is 2013 `[HashMismatch]`.

The event log shows the path taken. Event **1009** for an administrator ends with `[administrator passthrough: real token, child governance off]`.

Setting (service-wide, read at service start, Policies hive wins):

Value	Type	Meaning
`AdminPassthrough`	DWORD	Under `HKLM\SOFTWARE\Logic Unknown\AEMS` or `HKLM\SOFTWARE\Policies\Logic Unknown\AEMS`. Absent or any non-zero value (**default** `1`) = administrator passthrough on. `0` = off: administrators are treated exactly like standard users.

Other accounts, "Run as different user" and "Run as administrator" (driver protocol 2)

From AEMS.msi 1.0.3.0 the decision is made from the token of the process that was actually launched, not from whoever is signed in at the console:

How the allow-listed app was started	What AEMS does
"Run as administrator" (any admin account), or by an administrator with UAC off, or by SYSTEM	Nothing. The process already runs elevated; the driver does not catch it, no event.
"Run as different user" / `runas` with a **local administrator** account	Relaunched with **that** account's real full token in the same session, child governance off (event 1009 `[administrator passthrough ...]`).
By the console user (standard user)	Kiosk path: AEMS-built elevated token, default-deny child governance, `HashSHA` enforced (event 1009).
"Run as different user" with another **standard** account, or from a non-interactive session	Left running **unelevated** with its own token, untouched (event 2004 "left running unelevated"). AEMS never elevates an identity it cannot vouch for.

The original is no longer killed before AEMS has decided: it runs with the launching user's own token for the few milliseconds the decision takes, and is terminated only when an elevated relaunch replaces it (the relaunch waits up to 2 seconds for the original to exit, so single-instance apps do not see their predecessor). If AEMSvc is stopped or crashes, every launch it has not decided keeps running unelevated.

With `AdminPassthrough=0`, "Run as different user" with an administrator account is treated like any other non-console identity: it is **left running unelevated** (event 2004), not relaunched. The administrator relaunch also loads that account's user profile so `HKCU` stays available after the original is replaced. Note that a "Run as different user" administrator window left open on the kiosk desktop lets whoever is at the keyboard start allow-listed apps elevated without a prompt; sign such windows out after use.

Audit

Every elevation and service lifecycle event is written to the Windows **Application** event log under source `AEMS` (Event Viewer -> Windows Logs -> Application). There is no network, cloud, remote-logging or portal dependency. Collect these centrally with your existing event forwarding / SIEM pipeline if required.

Allow-list Tool (AEMSAllowlist.exe)

`AEMSAllowlist.exe` is a small, portable Windows tool that creates AEMS allow-list registry entries one application at a time. It writes a regedit `.reg` file and, if you choose, imports it into the registry.

It is provided by Logic Unknown ApS separately from the MSI. It is **not** part of `AEMS.msi`, is not installed under `C:\Program Files\AEMS`, and AEMSvc has no knowledge of it. Copy the single exe anywhere (USB stick, Downloads) and run it; no installer and no files beside it are needed.

Publisher: Logic Unknown ApS (https://logic-unknown.com). The "Documentation" link at the bottom left of the window opens this documentation (https://docs.logic-unknown.com/aems/).

Version 1.3.0.0 (AEMS 1.0.6.0) adds wildcard path rules, the `Publisher` / `PublisherThumbprint` values (see "Wildcard path rules" below) and the "Children in place" choice (`ChildInPlace`, see Children in place). These need the AEMS 1.0.6.0 service and driver or later (AEMS 1.0.8.0: AEMSvc 0.1.9.0, driver 1.0.8.0) and, for children in place, a valid license or the upgrade grace; an older service ignores the new values and treats a pattern as a literal path that never matches.

Version 1.3.0.0 is the tool that ships with AEMS 1.0.6.0. It writes the Logic Unknown key path below. It is for AEMS 1.0.6.0 and later; AEMS 1.0.5.0 and earlier read a different (legacy) key, so use AEMSAllowlist 1.0.x for those devices or upgrade them first (see Upgrading from 1.0.5.0 or earlier). A `.reg` file saved by AEMSAllowlist 1.0.x names the legacy key and is not accepted by `--apply`; recreate the entries with this version.

Rollout order: upgrade a device to AEMS 1.0.6.0 FIRST, then use this version on it. `--apply` accepts the new key whatever AEMS version is installed, so running it on a device that still runs 1.0.5.0 creates `HKLM\SOFTWARE\Logic Unknown\AEMS` early. The 1.0.6.0 MSI then merges the legacy settings into that pre-created key (values and allow-list entries it lacks; the entries the tool wrote win), but a 1.0.5.0 device ignores the new key until it is upgraded.

What the tool writes

One key per entry under the hive you select:

- Base (default): `HKLM\SOFTWARE\Logic Unknown\AEMS\Applications\<Name>`
- Policy: `HKLM\SOFTWARE\Policies\Logic Unknown\AEMS\Applications\<Name>`

Value	Type	Written when
`FriendlyName`	REG_SZ	always (defaults to the file description, else the file name)
`DataPath`	REG_SZ	always; full path to the exe, or a wildcard pattern; never empty
`Arguments`	REG_SZ	always (`""` when empty)
`HashSHA`	REG_SZ	only when "Pin SHA-256 hash" is checked (64 uppercase hex)
`Publisher`	REG_SZ	only when the Publisher box is filled (leaf certificate CN, trimmed)
`PublisherThumbprint`	REG_SZ	only when the Thumbprint box is filled (40 uppercase hex, SHA-1)
`GlobalApplication`	REG_DWORD	always; 1 when "Applies to all users" is checked (default)
`AllowChildElevation`	REG_DWORD	always; 1 when "Allow child elevation" is checked, else 0
`AllowUserArguments`	REG_DWORD	always; 1 when "Allow user arguments" is checked, else 0
`ChildInPlace`	REG_DWORD	always; 0 = Off, 1 = Direct children, 2 = Whole tree. The choice is available only while "Allow child elevation" is checked; clearing that check sets it back to Off.

Because `Arguments`, `AllowChildElevation`, `AllowUserArguments` and `ChildInPlace` are always written, importing an entry over an existing key of the same name replaces a stale value. Pins (`HashSHA`, `Publisher`, `PublisherThumbprint`) are written only when set, so an import never removes a pin that is already in the key; delete the entry first to remove a pin. The tool never lists or deletes existing entries, and never writes service-wide values (`LogLevel`, `AdminPassthrough`) or the deprecated `DoNotAllowChildElevateProccesses`. The default is path-only matching; the hash is opt-in.

When "Children in place" is set, **Add to list** shows warnings before it adds the entry: helpers started from the user's TEMP folder are refused and no copy starts; what "Direct children" or "Whole tree" covers; locked arguments stay locked; and the setting needs AEMS 1.0.6.0 with its driver active and a valid license (or the upgrade grace). For a children-in-place entry the warning dialog defaults to **No**.

The `.reg` file is UTF-16 LE with BOM and CRLF line endings, starts with `Windows Registry Editor Version 5.00`, and has the same shape as the example file `ElevatedApps.reg` (provided by Logic Unknown ApS).

Example file ElevatedApps.reg (provided by Logic Unknown ApS)

```
Windows Registry Editor Version 5.00

; ============================================================================
; ElevatedApps.reg
; Publisher: Logic Unknown ApS - https://logic-unknown.com
; Documentation: https://docs.logic-unknown.com/aems/
;
; Template: how to add applications to the AEMS allow-list so a standard
; (non-admin) user gets them elevated silently, with no UAC prompt and no UI.
;
; Allow-list hive (base):    HKLM\SOFTWARE\Logic Unknown\AEMS\Applications\<Name>
; Policy override hive:      HKLM\SOFTWARE\Policies\Logic Unknown\AEMS\Applications\
<Name>
;   (a COMPLETE Applications subtree under the Policies hive wins over the base)
; Since AEMS 1.0.6.0 the key is SOFTWARE\Logic Unknown\AEMS. For devices still on
1.0.5.0
; or earlier see "Upgrading from 1.0.5.0 or earlier" in the documentation (the upgrade
; migrates the legacy key).
;
; Per-app values:
;   DataPath                          REG_SZ     - full path to a PE .exe (never empty:
;                                                  an empty path prefix-matches
everything)
;   FriendlyName                      REG_SZ     - label used in the event log
;   Arguments                         REG_SZ     - command line (optional)
;   HashSHA                           REG_SZ     - exactly 64 hex chars, SHA-256 of the
.exe.
;                                                  Enforced for standard users; for
admins it
;                                                  is informational only (event 2003 +
relaunch)
;   GlobalApplication                 REG_DWORD  - 1 = applies to all users/sessions
;   AllowChildElevation               REG_DWORD  - 1 = this app's DIRECT children are
relaunched
;                                                  elevated; absent/0 = deny (default)
;   AllowUserArguments                REG_DWORD  - 1 = forward the standard user's own
command
;                                                  line on a top-level relaunch;
;                                                  absent/0 = configured Arguments only
(default)
;   DoNotAllowChildElevateProccesses REG_DWORD  - deprecated (spelling is part of the
wire
;                                                  contract); AllowChildElevation now
decides
;
; Launch form when Arguments is non-empty:   "<DataPath>" <Arguments>
;
; THREE THINGS THAT COST TIME:
;   1. AEMSvc reads the configuration ONLY at service start.
;      After reg import, run:  Restart-Service AEMSvc
;   2. Allow-list the REAL installed executable, never a copy. A copied Windows
;      exe without its <lang>\*.mui satellite fails resource loading.
;   3. mmc.exe or control.exe on the list WITHOUT Arguments always opens empty,
;      because an entry is locked by default: the user's own arguments are
;      discarded and only the configured Arguments are used.
;
; Get a hash:   (Get-FileHash "<path>" -Algorithm SHA256).Hash
; Apply (elevated):   reg import ElevatedApps.reg  ->  Restart-Service AEMSvc
; At scale: push via Intune/SCCM/RMM, or bake into a customer MSI with
; Build-MsiWithSettings.cmd (provided by Logic Unknown ApS).
```

```
; ============================================================================


; --- 1) Plain GUI admin tool - the common case ----------------------------
[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Example GUI Tool]
"FriendlyName"="Example GUI Tool (elevated)"
"DataPath"="C:\\Program Files\\Vendor\\Tool\\tool.exe"
"GlobalApplication"=dword:00000001


; --- 2) MMC snap-in - Arguments is REQUIRED -------------------------------
[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Disk Management]
"FriendlyName"="Disk Management (elevated)"
"DataPath"="C:\\Windows\\System32\\mmc.exe"
"Arguments"="diskmgmt.msc"
"GlobalApplication"=dword:00000001


; --- 3) Control Panel applet via control.exe ------------------------------
[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Network Connections]
"FriendlyName"="Network Connections (elevated)"
"DataPath"="C:\\Windows\\System32\\control.exe"
"Arguments"="ncpa.cpl"
"GlobalApplication"=dword:00000001


; --- 4) App that spawns helper processes + hash-pinned --------------------
; AllowChildElevation=1 makes the app a deliberate elevation springboard for
; its direct children, so pin the binary with HashSHA when you enable it.
; Replace the placeholder hash with the real one before importing. A malformed
; hash (not 64 hex digits) keeps the entry as a refusing entry: warning 2012 at
; service start and every launch it matches is refused. A well-formed hash that
; does not match the file refuses a standard user's launch (error 3009).
[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Example Launcher]
"FriendlyName"="Example Launcher (elevated)"
"DataPath"="C:\\Program Files\\Vendor\\Launcher\\launcher.exe"
"HashSHA"="0000000000000000000000000000000000000000000000000000000000000000"
"AllowChildElevation"=dword:00000001
"GlobalApplication"=dword:00000001


; --- 5) App launched with the user's own command line ---------------------
[HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS\Applications\Example Arg Tool]
"FriendlyName"="Example Arg Tool (elevated)"
"DataPath"="C:\\Program Files\\Vendor\\Calib\\calib.exe"
"AllowUserArguments"=dword:00000001
"GlobalApplication"=dword:00000001


; ----------------------------------------------------------------------------
; Service-wide values live on the parent key, not per app:
;   [HKEY_LOCAL_MACHINE\SOFTWARE\Logic Unknown\AEMS]
;   "AdminPassthrough"=dword:00000001   ; default 1; 0 = administrators are
;                                       ; elevated the same way as standard
;                                       ; users and their child processes are
;                                       ; governed (default-deny)
;   "LogLevel"=dword:00000000
;
; To remove an entry:
;   reg delete "HKLM\SOFTWARE\Logic Unknown\AEMS\Applications\Example GUI Tool" /f
```

```
;   Restart-Service AEMSvc
; ------------------------------------------------------------------------
```

Use the tool

1. **Browse...** to the exe. Name and Friendly name are filled in from the file unless you already typed them; the line below the path shows the file's company and product version so you can confirm it is the right binary. The path stays editable, so you can author entries for another machine (you get a warning, not a block, when the file does not exist here).
2. Set Arguments if needed. They are **required** for `mmc.exe` (`.msc`) and `control.exe` (`.cpl`): an entry is locked by default, so the user's own arguments are discarded and the elevated program gets only the configured Arguments.
3. Choose options, then **Add to list**. Repeat for more applications.
4. Choose the target hive.
5. **Save .reg file...** writes the file, or **Add to registry...** imports it now.

Add to registry writes the same `.reg` content to `%TEMP%\AEMSAllowlist-<guid>.reg` and imports it with `System32\reg.exe import` . As a standard user the tool relaunches itself elevated (one UAC prompt) with `--apply "<file>" [--restart-service] --delete-after` ; that instance has no window, only a result message box. The elevated instance reads the file once, checks it, writes the checked content to a new file in its own temp folder and imports that copy, so the launching user cannot swap the file between the check and the import. After a successful import the list is cleared. When "Restart AEMSvc after import" is checked (and AEMSvc is installed), the service is stopped and started again with a 30 second timeout each. AEMSvc reads its configuration **only at service start**, so without the restart run `Restart-Service AEMSvc` yourself.

`--apply` is an internal switch for that relaunch, not a general import command:

- It accepts only a file named `AEMSAllowlist-<guid>.reg` ; any other file is refused (exit 2) and left untouched. To import a `.reg` file you saved, run `reg import "<file>"` elevated.
- The content must look exactly like this tool's output: only the AEMS base or policy root (`HKLM\SOFTWARE\Logic Unknown\AEMS` or `HKLM\SOFTWARE\Policies\Logic Unknown\AEMS` , nothing else under the `Logic Unknown` vendor key, so the keys of other Logic Unknown products are never touched), its `Applications` key and `Applications\<Name>` keys, values only on the `<Name>` keys, only the ten value names above, strings or `dword:00000000` / `dword:00000001` (`ChildInPlace` also `dword:00000002` , and not 1 or 2 without `AllowChildElevation=1`), `HashSHA` exactly 64 hex, `PublisherThumbprint` exactly 40 hex, `Publisher` and `DataPath` non-empty, no deletions. Keys under the legacy pre-1.0.6.0 vendor key are refused.
- The incoming file is deleted only when `--delete-after` is given (the GUI relaunch passes it).

`--apply` exit codes: 0 imported (and restarted, if asked), 1 import failed, 2 bad arguments, wrong file name or refused content, 3 imported but the restart failed, 4 not 64-bit.

Wildcard path rules in the tool

Type a pattern in the Executable box, for example `C:\Program Files\Contoso\Tools\setup_*.exe` or `C:\Program Files\Contoso\App ?.?\app.exe`. **Browse...** picks a sample file; then edit its name into a pattern. A path that contains `*` or `?` is a wildcard rule. The tool applies the same validity rules as AEMSvc at configuration load (see settings reference):

- local path with a literal drive letter (`X:\...`); no UNC, `\\?\` or device paths;
- the drive and the first folder are literal; a file directly in the drive root is refused;
- the file name ends with the literal `.exe` ;
- no `/` , `"` , `<` , `>` , `|` , control characters or a second `:` ; no empty, `.` or `..` folder component;
- at most 251 characters (refused, never truncated);
- `Publisher` or a SHA-256 hash is set, and "Applies to all users" is checked;
- the hash, when set, is 64 hex digits; the thumbprint, when set, 40 hex digits.

The tool applies the same rules as the service, in the same order (it additionally treats U+007F to U+009F as control characters, and asks for Publisher whenever a thumbprint is set). `*` and `?` never match a `\` , and the pattern must match the whole path. A folder created directly in the drive root (for example `C:\Contoso`) normally lets users write and fails the service's trusted-location check; use a folder under `C:\Program Files` .

Every string value has a maximum length, which the `--apply` guard enforces as well: path 251 characters for an exact path and a wildcard pattern alike (the driver's allow-list record holds 255 characters including the 4-character `\??\` prefix, so a longer path could never match), friendly name 512, arguments 2047, publisher 256, key name 200. Line breaks, control characters and the Unicode line and paragraph separators are refused in every value.

What the service does with the rules (events 2012, 2013, 2016), where wildcard rules work (fixed drives on an accepted storage bus; the check is by bus, so point rules at the system drive) and why the folder of every elevated program must not be writable by standard users: Wildcard path rules and Folders of elevated programs.

- **Test against a file...** picks a file and shows MATCH or NO MATCH for the typed pattern (and, when Publisher is set, whether the file's publisher matches). The preview is advisory; AEMSvc decides on the launched file.
- **Read from file...** fills Publisher (subject CN of the leaf signing certificate) and Thumbprint (its SHA-1 thumbprint) from a file's **embedded** Authenticode signature. A file that is only catalog-signed (many Windows binaries) has no embedded signature and cannot be pinned by Publisher; pin its hash instead. A signature that is present but not valid on this machine is reported.
- **Pin SHA-256 hash** on a wildcard rule asks for the file whose hash the rule must require.
- Thumbprint is an extra pin and needs Publisher. Spaces and the invisible mark that the Windows certificate dialog adds when you copy a thumbprint are removed.

The tool does not check whether the folders are writable by standard users; AEMSvc does that at launch time and refuses the elevation when they are (see Wildcard path rules).

64-bit note for the tool

AEMSvc reads the 64-bit registry view. A 32-bit process that starts `System32\reg.exe` gets the WOW64 `reg.exe` and writes `SOFTWARE\WOW6432Node\...`, which AEMS never reads. The tool is therefore built x64 only and refuses to run as a 32-bit process.

Settings reference

Every AEMS setting lives under the base hive

```
HKLM\SOFTWARE\Logic Unknown\AEMS
```

or, to override it, under the policy hive

```
HKLM\SOFTWARE\Policies\Logic Unknown\AEMS
```

These are the paths from AEMS 1.0.6.0. AEMS 1.0.5.0 and earlier used a legacy vendor key; see Upgrading from 1.0.5.0 or earlier. Value names and meanings did not change.

AEMS (`AEMSvc`) runs as a native x64 service and reads the **64-bit** registry view throughout. Author values in the 64-bit view (plain `reg import` from a 64-bit process, or an elevated `regedit`, does this).

The service reads configuration once, when it starts (install, boot, or `Restart-Service AEMSvc`) - a change takes effect after a service restart. Values documented here are exactly the ones AEMS reads; AEMS ignores anything else under the key.

Policy override (precedence)

When the **policy hive** is present, it wins over the base hive:

- Scalar values (below) are read from the policy hive first, then the base hive.
- The **allow-list** is all-or-nothing: if the policy hive has an `Applications` subkey, AEMS uses the policy allow-list **in full** and ignores the base allow-list. If it does not, AEMS uses the base allow-list.

Use the policy hive to push a locked-down allow-list from Intune/SCCM/RMM that a local edit of the base hive cannot weaken.

Top-level values

Key: `HKLM\SOFTWARE\Logic Unknown\AEMS` (or the policy hive)

Value	Type	Default	Meaning
`LogLevel`	REG_DWORD	`0`	Verbosity of the service's Windows Event Log output (source `AEMS`). `0` = normal; higher = more detail.
`AdminPassthrough`	REG_DWORD	`1`	Administrator passthrough. When the console user is a local administrator (directly or through nested groups), an allow-listed app is relaunched with the user's real full token and is not registered for child governance. `0` = off (administrators are treated like standard users); any non-zero value = on. Policy hive wins. Read at service start. See Administrator users.
`InstallDir`	REG_SZ	*(set by MSI)*	Informational install path, written by the installer (`C:\Program Files\AEMS`). AEMS itself runs from the gate-required fixed path regardless.
`License`	REG_SZ	absent = unlicensed	1.0.6.0+: the signed license key (`<payload>.<signature>`). The policy value wins when it exists (of any type; only REG_SZ is valid). Without a valid license AEMS does not elevate. Set by `LICENSEKEY` (MSI), `AEMSvc.exe --set-license` or policy; re-read at start, every 10 minutes and after `--set-license`. Never logged. See Licensing.
`EndOriginalDescendants`	REG_DWORD	absent = on	1.0.6.0+. Absent or any non-zero value (**default**) = when AEMS ends the un-elevated original start of an allow-listed program, it also ends the un-elevated programs that original had already started, so they do not run twice (event 1224). `0` = never, as in 1.0.5.0: those programs keep

Value	Type	Default	Meaning
			running un-elevated next to the elevated start (event 1225 at service start). Policy hive wins. Read at service start. See Children in place.
`DisableDriverBindOnUnvalidatedBuild`	REG_DWORD	absent = `0`	1.0.6.0+. `1` = on a Windows build newer than the builds this AEMS version was validated on, AEMSvc does not connect to its driver: error 3213, no elevation. Policy hive or base hive. See Supported Windows versions.
`LicenseUpgradeGraceStartUtc`	REG_SZ	*(written once by AEMS)*	1.0.6.0+: UTC start (`YYYY-MM-DDTHH:MM:SSZ`) of the one-time 30-day upgrade grace on a device that had AEMS 1.0.5.0 or earlier. Base key only (never read from the policy key). Written once, never renewed; do not create or edit it.

Driver-read value: `TamperProtection`

Key: `HKLM\SOFTWARE\Logic Unknown\AEMS` only (the base hive; the driver does not read the policy hive for this value, and AEMSvc does not read it at all).

Value	Type	Default	Meaning
`TamperProtection`	REG_DWORD	absent = `0` (off)	Read by the kernel driver on every registry change under the AEMS key (driver 1.0.3.0 and later check the path first, so other registry writes on the system never read it). Non-zero = the driver denies registry changes under `HKLM\SOFTWARE\Logic Unknown\AEMS` (the key itself and everything below it) for every process except the registered `AEMSvc`: set value, delete value, delete key, set key security, replace key and set key information. Reads are always allowed. `0` or absent = no protection.

Notes and caveats (read before setting it):

- **Not set by the MSI, by a bundled settings file or by any example.** It is an opt-in hardening switch; leave it absent unless you need it.
- **Scope.** Only the base key `...\Logic Unknown\AEMS` is protected, with a key boundary: other products under the same vendor key (`HKLM\SOFTWARE\Logic Unknown\<product>`) are never affected. The policy hive `HKLM\SOFTWARE\Policies\Logic Unknown\AEMS` is not protected by this switch; protect it with your management tool (GPO/Intune own it).
- **It locks itself.** The value lives under the protected key, so once it is non-zero no administrator tool (`reg.exe` , `regedit` , Intune, a `.reg` import, `AEMSAllowlist.exe`) can change the allow-list, `LogLevel` , `AdminPassthrough` or `TamperProtection` itself while the driver is loaded. To change settings, stop `AEMSvc` and the `AEMSDriver` service (or uninstall), edit, then start them again (or reboot).
- **Registry view flags.** From the driver shipped with AEMS 1.0.6.0 the harmless `KeySetHandleTagsInformation` call (information class 5, a tag on the handle, not data) is let through, so 64-bit reads with an explicit registry view (`reg.exe /reg:64` , .NET `RegistryView.Registry64` , Intune detection rules, MSI `RegistrySearch Win64`) work under the flag. Every other set-information class stays denied. Drivers in AEMS 1.0.5.0 and earlier denied class 5 as well, which made such view-flag reads fail with access denied and, combined with AEMSvc 0.1.4.0 and earlier, could start the service with an empty allow-list.
- **Uninstall caveat.** With the 1.0.5.0 and earlier MSI, the uninstall's `RemoveConfigKey` step runs under `msiexec` , not the registered service, and was denied under the flag, leaving the AEMS key behind. For the behaviour of the 1.0.6.0 uninstall and upgrade with `TamperProtection` set (including an upgrade while the old driver is still loaded), see Upgrading from 1.0.5.0 or earlier. If a key is left behind after an uninstall, reboot (the driver is then gone) and delete it.

- **Repair.** A repair (`msiexec /fa` , ARP Repair, `REINSTALL=ALL`) would rewrite `InstallDir` under the protected key and fail with Error 1406. From 1.0.6.0 the MSI skips its registry write during a repair while `TamperProtection` is non-zero (files are still repaired). To restore a deleted `InstallDir` or `Applications` key by repair, set `TamperProtection` to 0 first (stop `AEMSvc` and `AEMSDriver` , edit, start them).
- **What it is and is not.** `TamperProtection` is tamper evidence against casual edits (a script, `regedit` , a stray `.reg` import), not a barrier against a determined local administrator. Not covered by the driver, unchanged in 1.0.6.0: renaming the `AEMS` key (or the `Logic Unknown` vendor key) away and creating a fresh unflagged key, `RegRestoreKey` / `RegLoadKey` over the vendor key, the Policies hive (a policy `Applications` subtree replaces the base allow-list) and the `WOW6432Node` view. An administrator can also stop the driver service.
- **Requires driver 1.0.3.0 or later** (AEMS 1.0.6.0 ships driver 1.0.6.0). An AEMS MSI that packages an older driver (protocol `0x00020000`) leaves `TamperProtection` without effect on this key; AEMSvc 0.1.7.0 and later report that at every start (warning 2007).

Approved applications (the allow-list)

Key: `HKLM\SOFTWARE\Logic Unknown\AEMS\Applications\<Name>` (one subkey per approved app; `<Name>` is any label).

Value	Type	Meaning
`FriendlyName`	REG_SZ	Label used in audit/logging.
`DataPath`	REG_SZ	Full path to the `.exe` launched elevated. Without wildcards it is matched as a **case-insensitive prefix** of the launched path: always write the full path including the file name, because a folder path (for example `C:\Apps\`) approves every program below it. 1.0.6.0+: a value that contains `*` or `?` is a **wildcard path rule**, see Wildcard path rules.
`Publisher`	REG_SZ	1.0.6.0+. Optional; **required for a wildcard rule unless `HashSHA` is set**. Exact, case-insensitive (trimmed) match against the subject common name (CN) of the leaf signing certificate of the file's **embedded** Authenticode signature. Also enforced on an exact-path entry when set.
`PublisherThumbprint`	REG_SZ	1.0.6.0+. Optional extra pin: SHA-1 thumbprint (40 hex) of that leaf certificate. When present it must match, in addition to `Publisher` when that is set. AEMSvc also enforces a thumbprint without `Publisher` (a thumbprint-only pin in a hand-written entry); `AEMSAllowlist.exe` always asks for `Publisher` too. A value that is not exactly 40 hex digits (on any entry) is not dropped: AEMSvc logs warning 2012 once at service start and keeps the entry in its place as a refusing entry, so every launch it matches is refused with 2013 `[MalformedRule]` (an administrator gets 2016 and the relaunch with the own token; a child is not started at all). It is kept so that a later, broader entry cannot elevate the same file without the pin. Correct the value and restart the AEMSvc service.
`Arguments`	REG_SZ	Optional. Launch line = `"<DataPath>" <Arguments>` for a standard user's elevated relaunch (the user's own arguments are discarded unless `AllowUserArguments=1`). Enables elevating `.cpl` via `control.exe`, `.msc` via `mmc.exe`.

Value	Type	Meaning
`AllowUserArguments`	REG_DWORD	Absent/`0` (**default**, locked) = a standard user's top-level elevated relaunch gets only `Arguments`, never the user's own command line. `1` = the user's command line is forwarded verbatim (and `Arguments` is used only when the launch has no arguments). Administrator passthrough always forwards the original command line; governed children always keep their own command line. Event 1009 ends with `(arguments: configured)` or `(arguments: user)`. Needs driver protocol 2 (AEMS.msi 1.0.3.0).
`HashSHA`	REG_SZ	Optional 64-hex SHA-256 of the `.exe`. When present, the file is hash-verified before launch (tamper check). 1.0.6.0+: a value that is not exactly 64 hex digits (on any entry) is not dropped: AEMSvc logs warning 2012 once at service start and keeps the entry in its place as a refusing entry, so every launch it matches is refused with 2013 `[MalformedRule]` (an administrator gets 2016 and the relaunch with the own token; a child is not started at all). It is kept so that a later, broader entry cannot elevate the same file without the pin. Correct the value and restart the AEMSvc service.
`AllowChildElevation`	REG_DWORD	Child-elevation opt-in. Absent/`0` (**default**) = this app's children are relaunched non-elevated (default-deny). `1` = direct children are relaunched elevated. A child is elevated only if it is itself a top-level allow-listed app or a direct child of a registered-ELEVATE parent; a non-opted-in app is never elevated and stops the chain. Opt in only specific trusted tools; opting in a general shell or interpreter makes the user an administrator while it runs (see the security warning under Approve applications).
`ChildInPlace`	REG_DWORD	1.0.6.0+. Valid only with `AllowChildElevation=1`. Absent/`0` (**default**) = children are blocked and relaunched as a copy, as in 1.0.5.0. `1` = direct children start elevated in place (as real children); what they start is refused unless it has its own opted-in entry. `2` = whole tree: every program below the app, up to 8 levels, starts in place without an entry of its own. Programs started in place must lie in a folder standard users cannot change; a child that fails a check is not

Value	Type	Meaning
		started and no copy starts. Without `AllowChildElevation=1`, with another value or with another type the value counts as off and AEMSvc logs warning 2230 at service start that names the entry. Needs the 1.0.6.0 driver and a valid license or the upgrade grace; otherwise warning 2220. Read at service start. See Children in place.
`DoNotAllowChildElevateProcesses`	REG_DWORD	**Deprecated** for the decision (superseded by `AllowChildElevation`). Still read and sent on the wire for back-compat.
`GlobalApplication`	REG_DWORD	`1` = applies to all users on the machine.

> The value name `DoNotAllowChildElevateProccesses` is spelled exactly as AEMS reads it (including the double-c `Proccesses`); it is part of the wire contract with the driver and must not be "corrected".

Worked example: the example file `NetworkSettings.reg` (elevated `ncpa.cpl`, Network and Sharing Center, Control Panel, with more launchers commented out).

Wildcard path rules

From AEMS 1.0.6.0 (AEMSvc 0.1.7.0 or later with driver 1.0.6.0; wildcard matching needs driver protocol `0x00020002` or later) an allow-list entry can name an executable by a **path pattern** instead of one exact path, for example `C:\Program Files\Contoso\Tools\setup_*.exe` or `C:\Program Files\Contoso\App ?.?\app.exe`. A launched file whose full path matches the pattern is elevated like any other entry, but only after two mandatory guards pass. Worked example: the example file `WildcardRules.reg`; how-to: Wildcard path rules.

An entry is a wildcard rule when its `DataPath` contains `*` or `?` (neither is legal in a real path, so no extra mode value exists). Entries without those characters keep their behaviour unchanged (case-insensitive prefix match).

Pattern semantics (driver and service match identically)

- The pattern must match the **whole** image path, not a prefix. Case-insensitive.
- `?` matches exactly one character that is not `\`.
- `*` matches zero or more characters that are not `\`. **Wildcards never cross a folder boundary**: `C:\Program Files\Contoso\*.exe` does not match `C:\Program`

`Files\Contoso\sub\x.exe`, and there is no recursive `**`.

- Wildcards may appear in the file name and in folder components (`C:\Program Files\Contoso\*\app.exe` matches exactly one folder level).
- When several entries match, the first one in registry order wins, as for exact entries.

Validity rules (checked by AEMSvc at service start)

An entry that breaks a rule is **dropped** (not sent to the driver, never matched) or, for a malformed `HashSHA` or `PublisherThumbprint`, **kept as a refusing entry** (see below). Either way AEMSvc logs warning 2012 with the entry name and the reason. The order of the checks: an empty `DataPath` drops any entry; a wildcard entry with an invalid pattern or `GlobalApplication` not 1 is dropped; a malformed pin on any remaining entry keeps it as a refusing entry; a wildcard entry without `Publisher` and `HashSHA` is dropped.

- Absolute local path `X:\...` with a literal drive letter. No UNC (`\\server\share`), no `\\?\`, no device paths.
- The drive and the first folder component are literal (no wildcard before the second `\`). A file directly in the drive root (`C:\setup*.exe`) is rejected.
- No `/`, `"`, `<`, `>`, `|`, control characters or a second `:`; no empty, `.` or `..` folder component.
- The last component ends with the literal `.exe` (case-insensitive).
- At most **251 characters** (the driver's 256-character field minus `\??\` and the terminator). A longer pattern is rejected, never truncated, because truncation would change what it matches.
- `Publisher` or `HashSHA` is set.
- `GlobalApplication` is `1`. An entry of any kind whose `DataPath` is missing or empty is dropped as well (2012, code `[path]`).

A malformed `HashSHA` (not exactly 64 hex digits) or `PublisherThumbprint` (not exactly 40 hex digits), on any entry, exact or wildcard, is different: the entry is **kept** in its place as a refusing entry. AEMSvc logs 2012 once at start (`[hash]` or `[thumbprint]`, "KEPT in its place as a REFUSING entry"), and every launch the entry matches is refused with 2013 `[MalformedRule]` (an administrator gets 2016 and the relaunch with the own token; a child is handled as not started at all). Keeping it in place means a later, broader entry cannot elevate the same file without the pin. Correct the value and restart the AEMSvc service.

The reason in warning 2012 is one of `[length]`, `[drive]`, `[character]`, `[component]`, `[root]`, `[first-folder-wildcard]`, `[extension]`, `[global]`, `[signer]`, `[path]` (entry dropped), or `[hash]` / `[thumbprint]` (entry kept as a refusing entry). `AEMSAllowlist.exe` 1.3.0.0 applies the same rules before it writes a `.reg` file.

Guards, limitations and events

At launch AEMS checks two guards on the concrete file before a wildcard rule (or a `Publisher` / `PublisherThumbprint` pin) elevates anything. The guards, their outcomes and the limitations of

wildcard rules are described on the security page: Wildcard rule guards and Wildcard rule limitations. The events are listed in Wildcard rule events.

Applying settings

- **Manually:** edit one of the example files from the Configure page (or write your own), `reg import <file>` from an elevated prompt, then `Restart-Service AEMSvc`.
- **At scale:** push the `.reg` via Intune/SCCM/RMM into the base or policy hive, or bake the allow-list into the MSI - see Bundle settings into the MSI.

AEMS licensing

> **Applies to AEMS 1.0.8.0**
>
> This documentation describes AEMS 1.0.8.0 (service 0.1.9.0, driver 1.0.8.0, Allow-list Tool 1.3.0.0), released 2026-11-20. Install 1.0.8.0. Text marked "from 1.0.8.0" does not apply to earlier versions. Where the text says "from 1.0.6.0", it applies to 1.0.7.0 and 1.0.8.0 as well.

From 1.0.6.0 AEMS needs a license key from Logic Unknown ApS. Without a valid license AEMS **does not elevate**: every allow-listed application starts unelevated, as if AEMS were not installed, and each refused launch is logged (event 2011). Everything else keeps working: the service runs, logs, reads its configuration, and uninstall and upgrades work as before. A fresh install without a key is unlicensed at once (event 3013, reason `missing`); there is no silent free period. If the license stops being valid while an application runs, the programs it then starts in place (Children in place) are refused (event 2204), and no copy starts.

The license check runs entirely on the device: the key is verified offline against the signature of Logic Unknown ApS and the device clock. AEMS never contacts a license server and has no portal or online dependency.

Get a key

- **Trial (30 days):** trial keys are not available yet; ask Logic Unknown ApS through the contact page.
- **Paid (standard) license:** contact Logic Unknown ApS through the contact page. A standard license has a 14-day grace after its expiry date; a trial has none.

The key is one line of text (`<payload>.<signature>` , letters, digits, `-` , `_` and one `.`). Treat it as confidential: it is not bound to a device, so whoever has it can use it until it expires.

Where the key is visible once applied (plan for it):

- **Registry read access.** `HKLM\SOFTWARE\Logic Unknown\AEMS\License` (and the policy value, and a staged `AEMS-LicenseRequest`) have the default `HKLM\SOFTWARE` ACL: every local user, also a standard user, can READ the key on a device. Only administrators (and, with `TamperProtection=1` , only AEMSvc) can change it.
- **Command lines.** `msiexec ... LICENSEKEY=<key>` puts the key on the msiexec command line, and the MSI passes it on the command line of `AEMSvc.exe --msi-write-license` . The MSI log masks it, but process-creation auditing (event 4688 with command line), Sysmon, EDR telemetry and deployment tools record command lines, and Intune and other management tools store the install command and log it on the device.

- `--set-license` reads the key from standard input, so it is in no command line; a key pasted at the console prompt stays in the console scroll-back until the window closes.

For fleets, prefer the Group Policy / Intune policy value or a customer MSI built with `Build-MsiWithSettings.cmd`, which is provided by Logic Unknown ApS (the key is then inside the MSI, not in a command line) over `LICENSEKEY` in an RMM command line.

Apply a key

Use any one of these (the policy value wins when both exist):

Way	How
At install (MSI property)	`msiexec /i AEMS.msi /qn LICENSEKEY=<key>`. The property is hidden: the MSI log shows `**********`, but the key is on the command line (see above). AEMSvc verifies the key first and writes only a valid one to `HKLM\SOFTWARE\Logic Unknown\AEMS\License`, so a mistyped key never replaces a working license (event 3014). On a repair this works only while `TamperProtection=0` or the driver is not loaded: with `TamperProtection=1` the write is denied, only event 3014 says so and the MSI still succeeds; use `--set-license` instead.
On a running device	`type aems-license.txt | "C:\Program Files\AEMS\AEMSvc.exe" --set-license` from an elevated prompt (PowerShell: `Get-Content aems-license.txt | & 'C:\Program Files\AEMS\AEMSvc.exe' --set-license`). Without a pipe it asks you to paste the key. The key is read from standard input, so it never appears in a command line or the process list. An invalid key is refused (exit 1) and nothing changes. A valid key takes effect at once (event 1014).
Group Policy / Intune	REG_SZ `License` in `HKLM\SOFTWARE\Policies\Logic Unknown\AEMS`. This value wins over the base value, except when it is empty or holds only blanks: such a value is ignored, so clearing the policy field never hides a valid base license (`--license-status` and event 1013 / 3013 then name the base key as the source and say that the policy value is ignored). A policy value of the wrong type (not REG_SZ) still wins and fails closed (`malformed`). AEMS re-reads it every 10 minutes.
Customer MSI	`Build-MsiWithSettings.cmd` has an optional license field; the key is verified with the AEMSvc.exe the MSI packages and bundled as the base value (bundler guide). `LICENSEKEY` at install time overrides a bundled key.

With `TamperProtection=1` nobody but AEMSvc may write the settings key, so `--set-license` hands the verified key to the running service (staged under `HKLM\SOFTWARE\Logic Unknown\AEMS-LicenseRequest`, administrators only, plus a service control), which verifies it again and writes it. AEMSvc opens that key without following registry links and only ever deletes the key itself: a request key that is a registry link (or is reached through one) is refused, removed as a link and logged as 3014, and nothing is read or changed. The hand-off writes only valid keys, but it does REPLACE the current license with the staged one, also with a key that expires earlier (for example a 30-day trial over a running standard license): `--set-license` is an administrator action, also under `TamperProtection=1`.

`--set-license` exit codes:

Exit	Meaning
0	Written (directly, or applied by AEMSvc through the hand-off).
1	Refused or nothing done: the key is invalid (or AEMSvc refused it), standard input could not be read (or is over 16 KiB), the caller is not elevated, or the key could not be staged. Nothing changed.
2	The key is valid and staged, but not written yet: AEMSvc is not running, did not answer within 15 seconds, or could not write it (event 3014 names the error). AEMSvc applies it at its next start.
3	The key was staged, but the hand-off key disappeared before AEMSvc answered (for example AEMSvc restarted and processed it). The outcome is not known to the command: check `--license-status` and events 1014 / 3014.

Check the state at any time: `"C:\Program Files\AEMS\AEMSvc.exe" --license-status` (exit 0 = elevation on, 1 = off). It prints the state, licenseId, customer, edition and expiry, the source key (and whether an empty policy value was ignored) and the upgrade grace window with its status (active, ended, not needed, or not applied because the marker is ahead of the device clock), never the key.

If a registry read fails (for example while the value is being replaced), AEMS keeps the last good state, logs warning 2015 once and re-reads every 60 seconds; after 3 failed reads in a row it treats the license as unreadable (3013 `malformed`, elevation off) until a read succeeds again.

A new or replaced license takes effect without a restart: at once after `--set-license`, otherwise within 10 minutes (AEMSvc re-reads the license every 10 minutes and at every start).

States

State	When	Elevation	Events
Licensed	signature valid, issued at most 1 day in the future, not expired	on	1013 at start and when the license changes
ExpiringSoon	licensed, expires within 14 days (trial: 5 days)	on	2008 once per day
Grace	standard license expired less than 14 days ago (a trial has no grace)	on	2009 once per day
UpgradeGrace	no valid license, the device ran AEMS 1.0.5.0 or earlier, within 30 days of the first 1.0.6.0 start (below)	on	2010 once per day, naming the end date
Unlicensed	everything else: no key, malformed, bad signature, a key for another Logic Unknown product, not yet valid, expired	**off**	3013 at start, on the change and once per day, with a reason code

Reason codes in 3013: `missing`, `malformed`, `bad-signature`, `wrong-product`, `not-yet-valid`, `expired`.

Upgrade grace (devices that ran AEMS 1.0.5.0 or earlier)

So that an upgrade does not stop elevation on devices that already use AEMS, the first licensing-aware AEMS on a device whose settings from 1.0.5.0 or earlier are migrated writes a marker once: REG_SZ `LicenseUpgradeGraceStartUtc` (UTC date and time) in `HKLM\SOFTWARE\Logic Unknown\AEMS` (event 1015). For 30 days from that moment AEMS keeps elevating without a license (state UpgradeGrace, warning 2010 every day with the end date). Apply a license within those 30 days; after them the device is Unlicensed.

- Evidence of an upgrade is a settings migration: the 1.0.6.0 copy (or merge) of a non-empty legacy settings key (see Upgrading from 1.0.5.0 or earlier) into the new key, recorded by the migration marker. Non-empty means any value (for example the `InstallDir` the old installer wrote) or any subkey (also an empty `Applications` key), so an unconfigured 1.0.5.0 device counts as well. A legacy key that merely exists is not enough: an empty legacy key, or a legacy key created next to an installed 1.0.6.0, gives no grace.

- The marker is written by the MSI upgrade step (before the new driver starts) or, as a safety net, by AEMSvc at its start after its own copy (with `TamperProtection=1` right after its driver handshake). It is always written before the old settings key is removed, and the old key is kept until it is written. It is written once and never renewed or overwritten. A fresh install gets no grace.
- It lives in HKLM, so a standard user can neither write nor delete it. With `TamperProtection=1` even administrators cannot change it (only AEMSvc can). An uninstall removes it with the rest of the settings key; a later fresh install then gets no grace.
- Licensing is enforced against standard users; it relies on administrators not tampering with the device.
- The upgrade grace also covers a key that is present but not valid (for example an expired trial) while the 30 days last; event 2010 names the reason.
- Details for the upgrade itself: Upgrading from 1.0.5.0 or earlier.

Clock

AEMS uses the device's own UTC clock (no network time). A wrong clock fails towards Grace or Unlicensed, never towards more elevation:

- A clock that runs ahead reaches the expiry (and the end of a grace) early.
- A clock that is behind: a key issued more than 1 day after the device clock is `not-yet-valid`, and `--set-license` refuses it (nothing is written). An upgrade-grace marker more than 1 day after the device clock (for example a kiosk with a dead clock battery that boots in the past until time sync) is not applied: the device is Unlicensed until the clock is right; event 3013 and `--license-status` then say "check the device clock". The grace still ends 30 days after the marker.

Keep the kiosk clock synchronised (domain or Windows time service).

Events

The license string and the requester e-mail address in it are never logged. Events name licenseId, customer, edition, trial, the expiry and which trusted key verified it.

Id	Level	Meaning
1013	Information	Valid license (state Licensed, ExpiringSoon or Grace) at start and when it changes: licenseId, customer, edition, expiry, verifying key, source key.
1014	Information	A license was set by `--set-license`, by the TamperProtection hand-off or by `LICENSEKEY`.
1015	Information	The one-time upgrade-grace marker was written; names the end date.
2008	Warning	Once per day: the license expires within 14 days (trial: 5 days).
2009	Warning	Once per day: the license expired; AEMS elevates until the grace ends (date).
2010	Warning	Once per day: no valid license, upgrade grace until the named date, with the license reason.
2011	Warning	A caught allow-listed launch (or an opted-in child) was left unelevated because AEMS is not licensed. At most 20 per hour; the first one of the next hour says how many more were not logged.
2015	Warning	The license value or the upgrade-grace marker could not be read; the last good state is kept and the read is retried every 60 seconds (3 failures in a row: 3013).
3013	Error	AEMS is not licensed (reason code and detail, the source key, a clock hint when the upgrade-grace marker is ahead of the clock); elevation is off. At start, on the change, then once per day.
3014	Error	A key given to `--set-license` (hand-off) or `LICENSEKEY` was refused or could not be written, or the hand-off key was a registry link and was removed; the current license is unchanged.

Troubleshoot AEMS

> **Applies to AEMS 1.0.8.0**
>
> This documentation describes AEMS 1.0.8.0 (service 0.1.9.0, driver 1.0.8.0, Allow-list Tool 1.3.0.0), released 2026-11-20. Install 1.0.8.0. Text marked "from 1.0.8.0" does not apply to earlier versions. Where the text says "from 1.0.6.0", it applies to 1.0.7.0 and 1.0.8.0 as well.

Start with the event log: AEMS writes every elevation, every refusal and every configuration problem to **Event Viewer -> Windows Logs -> Application**, source `AEMS`. The event reference at the end of this page lists every event and where each one is explained.

Reading AEMS events

AEMS 1.0.6.0 has no message table for its events. Event Viewer therefore shows a line such as "The description for Event ID 1009 from source AEMS cannot be found" before the actual text, or reports that it cannot retrieve the message text. The text AEMS wrote follows under "The following information was included with the event", and is on the **Details** tab (event data). In PowerShell, read the text from the event data:

```
Get-WinEvent -FilterHashtable @{ LogName = 'Application'; ProviderName = 'AEMS' } -
MaxEvents 50 |
    Select-Object TimeCreated, Id, LevelDisplayName, @{ n = 'Text'; e = {
$_.Properties[0].Value } } |
    Format-List
```

Quick checks:

- `sc query AEMSvc` - the service must be **Running** and set to start automatically.
- `sc query AEMSDriver` - the driver service must show `STATE : 4 RUNNING`.
- `"C:\Program Files\AEMS\AEMSvc.exe" --license-status` - exit code 0 means elevation is on, 1 means it is off (AEMS 1.0.6.0 and later).

Common problems

Symptom	Likely cause / fix
App launches but stays non-elevated	App not on the allow-list, or its `DataPath` / `HashSHA` doesn't match (a hash mismatch shows error 3009, then 2004). Re-check the `Applications\<Name>` subkey.
Warning 2003, then 2004	The driver caught the launch but the service found no matching entry, so it was not elevated. Install the driver and service from the same AEMS MSI and check `DataPath`.
A wildcard rule does not elevate	Look for the AEMS warning events (2012 at start, 2013 at launch, 2016 for an administrator); they name the reason. Wildcard rules work only from fixed drives on an accepted storage bus (removable media and USB, FireWire, SD/eMMC, iSCSI or virtual-disk storage are refused), from folders that only administrators can write; point them at the system drive. Warning 2014 means the current driver is not loaded yet, so wildcard rules never elevate: install the current AEMS MSI, and reboot only if 2014 is still logged afterwards. See Wildcard path rules.
Error 5000 and the AEMSvc service stopped	AEMSvc stopped after an internal failure. Windows restarts it automatically after 5 seconds, twice; after a third failure within a day it stays stopped. Event 5000 is in the Application log (source `AEMS`); the restarts appear as Service Control Manager events in the System log. Then start the AEMSvc service manually and contact support.
From 1.0.8.0: after AEMSvc stopped unexpectedly (or was ended), an allow-listed program that was started at that moment does not appear and shows in Task Manager as suspended for up to about 10 seconds, then disappears	Expected: AEMS keeps a started program suspended until it has decided how to start it. When the service ends unexpectedly, the driver ends those programs once the service connection is closed, which takes up to about 10 seconds; they never run without the decision. Start the program again after the AEMSvc service is running (Windows restarts it after 5 seconds).
Nothing happens / service not running	Check the `AEMSvc` service is running; check the `AEMSDriver` driver with driver diagnostics.

Symptom	Likely cause / fix
`AEMSDriver` won't start (0x241)	The driver is not Microsoft-signed. Install the AEMS MSI from Logic Unknown ApS, which ships the driver signed by Microsoft.
Elevation on a second interactive session	Out of scope - AEMS targets the active console session only.
A program worked elevated before 1.0.6.0 and fails now (a tool "not found", no proxy, a missing setting or license variable)	From 1.0.6.0 elevated programs get the machine environment, not the user's own variables: per-user `PATH` entries and every value under `HKCU\Environment` are not passed. Set the variable the program needs machine-wide (System Properties > Environment Variables > System variables, or Group Policy / Intune) and start the program again. See Compatibility effects.
Error 3009 "The environment for the elevated start of ... could not be built"	The user's profile folder or account name could not be determined, so the program was not elevated (a top-level program keeps running unelevated, 2004; a child is not started). Check that the user has a profile under `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList` and that the machine environment is readable; sign the user out and in.
Error 3009 "Building the owner (Administrators) and the default DACL of the elevated token failed"	The elevated token could not be prepared; nothing was elevated. Restart AEMSvc; contact support if it repeats.
Since 1.0.6.0 a non-elevated program can no longer change a file that an elevated program created (for example under `C:\ProgramData`)	Files that elevated programs create are owned by Administrators. Where the folder gives users write access only as creator, grant the Users group the needed rights on that folder explicitly. See The elevated token.
A tool that an application with `AllowChildElevation=1` starts through `conhost.exe --headless <tool>` (or `conhost.exe <tool>`) runs non-elevated since 1.0.6.0	Expected: a program that the console host process itself starts is never elevated. Start the tool directly from the application (not through `conhost.exe`): a direct child of a parent with `AllowChildElevation=1` is still elevated. Commands typed in the console window of an elevated, allow-listed console program such as `cmd.exe` are not

Symptom	Likely cause / fix
	affected: they follow that entry's `AllowChildElevation` as before.
An allow-listed browser, Electron or WebView2 application does not work (blank windows, crashes, helper processes in their own windows)	Not supported as an elevated application in AEMS (from 1.0.6.0): such programs start sandboxed helper processes. Without "Children in place" AEMS governs them like any other child and starts them again outside the sandbox; with "Children in place" they are refused, so the program starts but its helpers do not. Remove the entry and run the program without elevation. See Limits.
A program that an allow-listed application starts does not start; the application reports "Access is denied"	With "Children in place" the child was refused. Most often its folder can be changed by standard users (a folder directly under `C:\` with default permissions, or the user's `%TEMP%`). See A child program does not start in place.
An elevated program does not find a network drive or a `subst` drive that the user sees	Expected from 1.0.6.0: every program AEMS elevates resolves drive letters through the machine's drive letters, not the user's own. Use a local path, or a drive letter that is defined for the whole machine. See Drive letters of elevated programs.
An elevated console program (for example `cmd.exe`) closes at once without output, exit code `0xC0000142`	Its console host could not be started. This can happen when AEMS refused the console host (see the warnings around that time) or could not give it the machine's drive letters. Restart AEMSvc; contact support if it repeats.
Error 3009 "Child ... is not relaunched: its creator runs below High integrity"	A process that is not an AEMS-elevated application was reported as the creator of a governed child (for example a process that lowered its own rights); the child was not started, in either child mode. Not expected in normal operation; if it repeats, collect the events and contact support.

Launchers and programs that start allow-listed programs

- **A launcher that gets its target as a command-line argument** needs "Allow user arguments" (`AllowUserArguments=1`) on its entry. Without it the entry is locked: the elevated start gets only the configured `Arguments`, the user's argument is dropped, and the launcher

starts without its target. Set it only for a launcher whose command line cannot be turned into arbitrary elevation (see the security warning under Command line and arguments).

- **A program that itself starts another allow-listed program without elevation** sees AEMS elevate that program again through its own entry: the start is caught like any start of an allow-listed program. If that program must run without elevation when it is started this way, remove its entry.

Driver diagnostics

What it does: checks the AEMS kernel driver's health and helps fix the common "driver not running / not loading" problems.

Quick check

Confirm the kernel-driver service is up:

```
sc query AEMSDriver
```

A healthy driver shows `STATE : 4  RUNNING`. (The `AEMSvc` user-mode service is separate - `sc query AEMSvc`.) If the driver is not running, start it from an elevated prompt with `sc start AEMSDriver` and compare the error with the table below.

Reading the results

Symptom	Meaning / fix
Service **STATE : RUNNING**	Driver is healthy. The device `\\.\AEMS` now exists (SYSTEM-only).
`sc start` -> **577 / 0x241 STATUS_INVALID_IMAGE_HASH**	The `.sys` / `.cat` isn't signed by Microsoft, and the machine enforces driver signing. Install the AEMS MSI from Logic Unknown ApS, which ships the driver signed by Microsoft.
`sc start` -> **1275 / 0x4FB ERROR_DRIVER_BLOCKED**	Microsoft's vulnerable driver blocklist blocked the driver. Ask Logic Unknown ApS through the contact page.
`sc start` -> **50 / 0x32 ERROR_NOT_SUPPORTED**	The **Windows-version gate**: the driver refuses to load on Windows builds below 10.0.19045 (Windows 10 22H2); there is no upper build limit (see Supported Windows versions). Check the build number (`winver`).
`sc start` -> **31 / 0x1F ERROR_GEN_FAILURE**	`DriverEntry` failed for another reason. Reinstall with the MSI and reboot; contact support if it repeats.
`sc start` -> **2 / 0x2 ERROR_FILE_NOT_FOUND**	The `.sys` isn't where the service's ImagePath points. Reinstall.

Driver signing

On stock Windows 10/11, kernel drivers must be Microsoft-signed (WHQL or attestation) to load. A driver that is not Microsoft-signed installs fine but won't start (`0x241`). The AEMS MSI from Logic Unknown ApS ships the driver signed by Microsoft; reinstall from that MSI.

A child program does not start in place

First check the folder. A program is started in place only when it lies in a folder that standard users cannot change. A folder created directly in the drive root (for example `C:\Contoso`) does not qualify by default, because Windows gives Authenticated Users Modify on every new folder under `C:\` . Run `icacls <folder>` and harden the folder with the three commands in The folder rule.

Then go through the table from the top:

Check	What you see	What to do
Is the device on AEMS 1.0.6.0 and does the user have a split token?	Every child is refused with warning **2224** `[PRE_TOKEN]`, also programs in `C:\Program Files`. In the user's normal session `whoami /groups` shows a group as "Group used for deny only" (for example Network Configuration Operators).	Update to AEMS 1.0.7.0 or later (install 1.0.8.0), which builds the elevated token from the user's full token (event 1226). See Users with a split token.
Did you try it as a standard user?	Children run, but there is no "started in place" event at all.	An administrator's launch uses the administrator passthrough: AEMS does not check the children and logs nothing for them. Verify with a standard user account.
Is the setting on?	The entry has `AllowChildElevation=1` and `ChildInPlace` 1 or 2 (`reg query "HKLM\SOFTWARE\Logic Unknown\AEMS\Applications\<Name>"` ; with a policy allow-list, under `HKLM\SOFTWARE\Policies\Logic Unknown\AEMS\Applications`).	Set both values. `ChildInPlace` without `AllowChildElevation=1` counts as off (warning 2230 at service start naming the entry).
Was the service restarted after the change?	The child is still relaunched as a separate copy (a new window, the application does not get its output).	`Restart-Service AEMSvc`. Settings are read at service start.
Warning **2220** "Children in place (ChildInPlace) is configured for ... but is not active: ..." (and warning **2225** for each launch)	The reason is in the text: the loaded driver is the old one, AEMS has no valid license, or the service could not connect to the driver. With the old driver or no connection, children are relaunched as a copy; without a license nothing is elevated.	Old driver: restart the device (the upgrade normally loads the new driver at once). Not licensed: apply a license key (event 3013, Licensing). Connection not ready:

Check	What you see	What to do
		restart AEMSvc; contact support if 2220 stays.
Warning **2213** `Start of a child refused [PathGuard]`	The program lies in a folder that standard users can change, for example the user's `%TEMP%` or a folder directly under `C:\` with default permissions. The text names the folder, the account and the right. If the text ends with `own entry '<name>'` and names no folder, the child's own allow-list entry is refusing (see warning 2012): fix that entry. A detail `[NotCanonical]` means the program was started by its 8.3 short name: start it by its long path.	Harden the folder (The folder rule) or install the program under `C:\Program Files`. For an installer that unpacks a helper to `%TEMP%`: leave "Children in place" off for that entry (documented limit, from 1.0.6.0).
Warning **3200** `Start of a child refused [PathAlias]`	The program was started through an alias drive letter (`subst`), a symbolic link or a junction.	Start it by its real path on a local drive.
Warning **3201** `Start of a child refused [VolumeNotFixed]`	The program lies on a network drive, a UNC path, removable media or a mounted ISO.	Copy it to a protected folder on a local fixed drive.
Warning **2211** `Start of a child refused [FileIdentity]`	The program lies on a volume that is not NTFS (ReFS, Dev Drive, FAT, exFAT).	Move it to an NTFS volume.
Warning **2224** "The AEMS driver refused ... child start(s) [PRE_DEPTH]"	The program is deeper than the setting allows: a grandchild with "Direct children", or deeper than 8 levels with "Whole tree".	Use "Whole tree", or give the program its own entry with `ChildInPlace`.
Warning **2203** `Start of a child refused [CreatorNotHeld]`	The application was started before the AEMS service was last restarted.	Close the application and start it again.
Warning **2216** `Start of a child refused`	The child has its own entry with locked arguments, and its	Set the child entry's `Arguments` to the

Check	What you see	What to do
`[ArgumentsLocked]`	command line does not equal that entry's `Arguments` (leading and trailing blanks are ignored).	arguments the application passes, or set `AllowUserArguments=1` on the child entry if that is safe.
Warning **2215** `Start of a child refused [HashedEntryNoPrecompute]`	The child has its own entry with `HashSHA`; such entries are refused in place (from 1.0.6.0).	Remove `HashSHA` from the child's entry, or remove the child's entry.
Warning **2224** "The AEMS driver refused ... child start(s) [PRE_TOKEN]"	The child runs with a reduced or sandboxed token, typically a helper process of a Chromium, Edge WebView2 or Electron program.	Not supported in place; leave "Children in place" off for such programs.
Warning **2224** "The AEMS driver refused ... child start(s) [LIMIT_CREATOR]"	The application had more than 6 child starts waiting at once.	Starts are refused, not queued; let the application retry or start fewer programs at once.
Error **3240** "... result=notApplied ..."	AEMS could not give the program the machine's drive letters, so it did not start it (fail closed).	Restart AEMSvc; contact support if it repeats.
No AEMS event at all for the child	The application was started by an administrator (administrator passthrough), or the child is not started through the allow-listed application (for example a scheduled task, a COM server or Windows Installer starts it), so AEMS does not see it.	Such programs are outside the elevated tree; see Children in place.

Every refusal means: the program is not started, the application gets "Access is denied", and no copy starts. Similar refusals of the same application are logged at most once per 10 seconds per application and reason; the next line says how many were not logged. Events 3200 and 3201 are logged at Warning level although their number is in the 3000 range.

Wildcard rule events

ID	Level	Meaning
1009	Information	Caught app relaunched elevated; for a wildcard rule the text names the rule.
2003	Warning	For a standard user (from 1.0.6.0): the driver caught the launch but AEMSvc found no matching entry; NOT elevated, the original keeps running (followed by 2004). For an administrator (no matching entry, or a file that does not match a `HashSHA`-only pin): relaunched with the administrator's own real token (passthrough; for a no-match with the own command line). For a child of a governed parent: relaunched with child elevation denied.
2012	Warning	At configuration load: a wildcard entry that breaks a validity rule, or any entry with a missing or empty `DataPath` (`[path]`), was dropped; or an entry with a malformed `HashSHA` or `PublisherThumbprint` was KEPT as a refusing entry (every match refused with 2013 `[MalformedRule]`). Names the entry and the rule (code in brackets).

ID	Level	Meaning
2013	Warning	A wildcard match (or a publisher-pinned exact entry) was refused at launch; names the image, the entry and the reason (folder, SID and right; owner; reparse point; drive type or alias; file open for writing; short or non-canonical path; or the signature or hash problem). 2013 always means the rule did not elevate via its own entry; for a child the text ends with "relaunched UNELEVATED" (parent without `AllowChildElevation`) or "IS still relaunched elevated ... by design" (parent with `AllowChildElevation=1`). Codes include `MalformedRule` (the entry has a malformed `HashSHA` or `PublisherThumbprint`; correct it and restart AEMSvc), `WildcardInactive` (the loaded driver is older than the one shipped in 1.0.6.0; install the current MSI, and reboot only if 2014 is still logged), `RemovableVolume` (the device reports removable media, or the disk is attached over a bus wildcard rules refuse; the text names the bus, for example "is attached over the USB bus"; move the file to the system drive), `NotFixedVolume` (network, RAM disk, or a drive letter redirected for the user, for example by `subst` or a per-user mapping; use a real local path), `OpenFailed`, `NotCanonical`, `ReparsePoint`, `UntrustedOwner`, `WritableByOther`, `NotSigned`, `SignatureInvalid`, `PublisherMismatch`, `ThumbprintMismatch` and `HashMismatch`.
2016	Warning	ADMINISTRATOR OVERRIDE: the rule's checks failed, but the user is an administrator, so the app was relaunched elevated with the administrator's own real token.
5000	Error	Unexpected error. From 1.0.6.0 it can also mean that AEMSvc stopped itself after an internal safety failure; nothing is elevated until it runs again. If AEMSvc ends unexpectedly, Windows restarts it automatically after 5 seconds, twice; after a third failure within a day it stays stopped. Event 5000 is in the Application log (source `AEMS`); the restarts and the final stop appear as Service Control Manager events in the System log. If it stays stopped, start the AEMSvc service manually and contact support.
2014	Warning	Wildcard rules are inactive because the loaded driver does not support them (protocol minor below 2); logged once at start. They are never elevated, and launches they match first are refused (2013 `[WildcardInactive]`). Install the

ID	Level	Meaning
		current AEMS MSI; its upgrade normally unloads the old driver and starts the new one. If 2014 is still logged after that, the old driver is still loaded: reboot. If it stays after the reboot, the installed package does not contain driver 1.0.3.0 or later: install the current AEMS MSI (1.0.8.0) from Logic Unknown ApS.

The driver connection guard key after an uninstall

On a Windows build newer than the validated ones AEMSvc keeps the key `HKLM\SOFTWARE\Logic Unknown\AEMS-DriverBindGuard` (see Supported Windows versions). From 1.0.8.0 the uninstall removes it when it is empty (a leftover `ClearBindAttempt` is removed with it). It is kept, with its content, when it holds the value `BindAttemptBuild`. That value is written before every connection attempt and removed after it; it is left behind when the device stopped during the attempt, and also when AEMSvc itself was ended during it (for example by a forced stop at that moment), without any crash. A reinstall on the same Windows build then does not connect to the driver (error **3212**). If you are sure the device did not stop in the driver connection, set the REG_DWORD value `ClearBindAttempt` = 1 under that key and restart AEMSvc: AEMS tries once more and removes the value. Values that AEMS does not write also keep the key. The MSI log does not record this step; to see the result, check after the uninstall whether the key is still there (`reg query "HKLM\SOFTWARE\Logic Unknown\AEMS-DriverBindGuard"`): gone means it was removed, a remaining key holds `BindAttemptBuild` or a value AEMS does not write. Uninstalling 1.0.7.0 or earlier leaves the key in every case.

Event reference

Every event AEMSvc writes (Application log, source `AEMS`). Licensing events are described in Licensing, the upgrade and migration events in Upgrading from 1.0.5.0 or earlier, the wildcard events in Wildcard rule events, and the children-in-place events in A child program does not start in place.

ID	Level	Meaning	What to do
1000	Information	Service starting.	-
1001	Information	Service running (names the version).	-
1002	Information	Configuration loaded: number of allow-list entries, whether the policy allow-list is used, `LogLevel`, `AdminPassthrough`.	Check the entry count after a change.
1003 / 1004	Information	Service stopping / stopped.	-
1005 / 1006	Information	Service installed / uninstalled (by the MSI).	-
1007	Information	Driver connected (device opened, handshake done).	-
1008	Information	Allow-list sent to the driver.	-
1009	Information	A caught program was relaunched elevated (names image, session and rule).	-
1010	Information	The caught original exited before the decision; relaunched anyway. From 1.0.8.0 this happens only when the start was not held (see 2235 / 2236).	If it repeats with 1.0.8.0, look for 2235 or 2236 before it.
1011 / 1012	Information	Legacy settings copied / legacy key removed (upgrade from 1.0.5.0 or earlier).	-
1013 / 1014 / 1015	Information	Licensing: valid license / license applied / upgrade grace started.	See Licensing events.
2000	Warning	Configuration key not found or not readable; running with defaults.	Check `HKLM\SOFTWARE\Lo`

ID	Level	Meaning	What to do
			`gic Unknown\AEMS`.
2001	Warning	The allow-list is empty; nothing is elevated.	Add entries, then restart AEMSvc.
2002	Warning	A caught launch was not from an interactive session; skipped.	-
2003	Warning	The driver caught a launch but AEMSvc found no matching entry, or (administrator) the file does not match a `HashSHA`-only pin; details in Wildcard rule events.	See there.
2004	Warning	A caught original was left running unelevated; the text names why. Follows 2003, 2013 and 3009, and is also logged on its own when the launch is refused for another reason the text names (for example not the console user and not an administrator, or the process could not be opened). An unlicensed refusal is reported by 2011 alone, without 2004. From 1.0.8.0 a held original is resumed first and then left running; "could not be opened (error 87)" for a program that finishes very quickly then occurs only when the start was not held (2235 / 2236). At a service stop one 2004 names how many caught programs were not decided yet.	Read the event before it, or the reason in the text.
2005 / 2006 / 2007	Warning	Legacy settings key present or not removable / legacy policy key holds settings / loaded driver is older than 1.0.3.0 (`TamperProtection` not enforced).	See Upgrading from 1.0.5.0 or earlier.

ID	Level	Meaning	What to do
2008 / 2009 / 2010 / 2011 / 2015	Warning	Licensing: expiring soon / grace / upgrade grace / launch not elevated because unlicensed / license read retried.	See Licensing events.
2012 / 2013 / 2014 / 2016	Warning	Wildcard rules and pins.	See Wildcard rule events.
1200	Information	A child started in place, elevated (names the program, the pids and the session).	-
1201	Information	The console host of an elevated console program was allowed.	-
1202	Information	Further "started in place" lines of one elevated program in the last minute, summarised (count).	-
1221	Information	The service is connected to the AEMS driver for children in place (with the number of child starts refused while it was not connected).	-
1222	Information	Children in place is active again (after 2220).	-
1223	Information	Children in place, summary at service stop.	-
2200-2216, 2218, 2219	Warning	`Start of a child refused [reason]`: the child was not started, no copy. 2200 RequestInvalid, 2201 ServiceStopping, 2202 WorkersBusy, 2203 CreatorNotHeld, 2204 Unlicensed (the license stopped being valid while the application ran), 2205 DepthLimit, 2206 ChildOpenFailed, 2207 ChildIdentity, 2208	See A child program does not start in place.

ID	Level	Meaning	What to do
		UserMismatch, 2209 ChildFileBusy, 2210 FileOpenFailed, 2211 FileIdentity (not NTFS), 2212 ImageMismatch, 2213 PathGuard, 2214 InternalError, 2215 HashedEntryNoPrecompute, 2216 ArgumentsLocked, 2218 ConsoleHost, 2219 VerdictDeadline.	
2220	Warning	At least one entry has `ChildInPlace`, but children in place are not active (old driver loaded, no valid license, or the service could not connect to the driver); names the entries and the reason. Logged at service start and when the state changes.	See A child program does not start in place.
2221	Warning	Console hosts of elevated programs started while the AEMS service was stopped, starting or restarting, without the machine's drive letters (count).	Nothing, unless it repeats often; see Children in place.
2223	Warning	A blocked child of a program with children in place was not started again as a copy.	-
2224	Warning	The AEMS driver refused child starts `[reason]` (for example `PRE_DEPTH`, `PRE_TOKEN`, `LIMIT_CREATOR`, `LIMIT_SESSION`, `LIMIT_TOTAL`, `TIMEOUT`, `BREAKER_OPEN`); no process and no copy.	See A child program does not start in place.
2225	Warning	For one launch: `ChildInPlace` is configured but not active, so its children are blocked and started again elevated (see 2220).	See 2220.

ID	Level	Meaning	What to do
2226 / 2227 / 2228 / 2229	Warning	A child allowed in place did not start after all (the driver did not accept the allow, its answer was unclear, the driver was being unloaded), or a launch could not be recorded for children in place.	Start the application again; contact support if it repeats.
1224	Information	When the user's un-elevated original of an allow-listed program was ended, the un-elevated programs it had already started were ended too, so they do not run twice (count).	-
2232	Warning	Not every program that the un-elevated original had already started could be checked or ended; such a program may run twice.	Check the program; see Children in place.
1225	Information	At service start: `EndOriginalDescendants` = 0 is set, so programs that a user's un-elevated original start had already started keep running un-elevated next to the elevated start (as in 1.0.5.0).	Remove the value (or set it to 1) and restart AEMSvc if that was not intended.
1226	Information	1.0.7.0+: the elevated token for this program is built from the user's full token, because the user's logon has a split token (User Account Control). Text: `'<program>': the elevated token takes its identity from the user's full (linked) token, because the user's logon has a split token (UAC).` Written after the elevated start.	Nothing; see Users with a split token.
2234	Warning	1.0.7.0+: the program was not started elevated (a child: not started), because the user's logon	Log the user off and on again, then start the

ID	Level	Meaning	What to do
		has a split token and the user's full token could not be read. AEMS never uses the filtered token instead. Text: `'<program>' is not started elevated: the identity for the elevated token could not be taken from the user's token (<step>, <error>). When the user's logon has a split token (UAC), AEMS uses the user's full (linked) token and never the filtered one. Log the user off and on again; if this persists, contact support with this event.` (for a child: `The child '<program>' is not started: ...`)	program again. If it repeats, contact support with the event.
2235	Warning	1.0.8.0+: a caught program could not be held before it started, so it may have run without administrator rights for a moment; the decision then runs as in 1.0.7.0 (a program that finishes very quickly can run twice, 1010, or not elevated, 2004). At most once per program per hour. Text: `'<program>' (pid <n>) could not be held before it started (<reason>); it may have run without administrator rights for a moment. The decision runs as in AEMS 1.0.7.0 for this launch.` Reasons: the driver had already ended the hold (it timed out, or the hold breaker is open after repeated timeouts, or it was caught before the hold was armed at service start); the process could not be opened or suspended; no thread was found in the new process after the suspend; the decision queue is full; it was no longer pending in the driver; the decision worker is busy (an earlier caught program has	Occasional: nothing. Often: the service is overloaded or was suspended; check the device load. "no thread was found" on a Windows build that AEMS 1.0.8.0 was not validated on: contact support with the event.

ID	Level	Meaning	What to do
		waited suspended for its decision for more than 30 s; logged once for that period, new launches are not suspended until it is decided).	
2236	Warning	1.0.8.0+, once per service start: the loaded driver does not hold caught programs before they start (it speaks an older protocol, for example driver 1.0.6.0), so programs that finish very quickly can still run twice or only without elevation, as in 1.0.7.0. Not expected after a completed upgrade: the upgrade loads driver 1.0.8.0 at once, and when the old driver cannot be unloaded the upgrade fails and asks for a restart. The event text still says that this is expected right after an upgrade until the device restarts; with 1.0.8.0 treat 2236 as a reason to restart. Also logged when AEMSvc could not set up the hold (the text says why).	Restart the device to load driver 1.0.8.0. If 2236 stays after the restart, the installed package does not contain driver 1.0.8.0: install the AEMS 1.0.8.0 MSI.
2237	Warning	1.0.8.0+: AEMSvc could not resume a suspended caught program (count and status, at most once per hour). Such a program stays suspended until it is ended.	End the program in Task Manager and start it again; contact support with the event if it repeats.
2233	Warning	Once per service start: lists the exact-path allow-list entries on local fixed disks whose program or folder standard users can change. The entries keep working, but any user can replace the program.	Harden the folder (The folder rule) or move the program under `C:\Program Files`.
2230	Warning	At service start: an entry's `ChildInPlace` is invalid or set	Fix the entry, then restart AEMSvc.

ID	Level	Meaning	What to do
		without `AllowChildElevation=1`, so it counts as off; names the entry.	
2231	Warning	At service start, after a successful connection to the driver: the Windows build is newer than the builds this driver version was validated on (names the build); the driver's start-up self-check passed on this build.	Check for an updated AEMS version that is validated on this build.
3000	Error	`AEMSvc.exe --install` or `--uninstall` was run without administrator rights.	Run it elevated (the MSI does this).
3001 / 3002	Error	Service install / uninstall failed, or the service recovery settings could not be set (3001; the install still succeeds).	Repair or reinstall with the MSI.
3004	Error	Internal service error (thread or event creation failed).	Restart AEMSvc; contact support if it repeats.
3005	Error	The driver device could not be opened; the service runs idle.	Check `sc query AEMSDriver` and Driver Diagnostics.
3006	Error	Driver handshake failed.	Reinstall with the MSI; reboot.
3007	Error	The allow-list could not be sent to the driver; the entries are not enforced.	Restart AEMSvc; reboot if it repeats.
3008	Error	The privileges the driver grants could not be enabled; elevation is disabled.	Reboot; contact support if it repeats.

ID	Level	Meaning	What to do
3009	Error	Elevation failed; the text names the step. Includes a standard user's launch of a file that no longer matches the `HashSHA` of a hash-only entry ("On-disk hash ... no longer matches the configured HashSHA; refusing to relaunch", followed by 2004), a token, session or relaunch failure, a governed child whose command line is longer than the driver forwards, a governed child whose creator runs below High integrity ("Child ... is not relaunched: its creator runs below High integrity (0x...)", the child is not started in either child mode), an elevated start whose environment could not be built ("The environment for the elevated start of ... could not be built (machine environment, user profile folder or account name unavailable, ...); not elevated (fail closed)", followed by 2004 for a top-level launch: the app keeps running unelevated; a child is not started), and a token whose owner and default permissions could not be prepared ("Building the owner (Administrators) and the default DACL of the elevated token failed (error ...)", followed by "Building the elevated token ... failed (NTSTATUS 0xC000005A)").	For a hash mismatch: update `HashSHA` (after a vendor update) or investigate the file. For the environment: check that the user has a profile folder (`HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList`) and that the machine environment (`HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment`) is readable; see Environment of elevated programs. For the owner: restart AEMSvc; contact support if it repeats. Otherwise read the step named.
3010	Error	The driver refused to grant privileges because its start-up self-check does not recognise the Windows token layout on this build; elevation is disabled. Most likely after a Windows feature update to a build newer than the validated ones.	Check the supported Windows versions; install a current AEMS release.

ID	Level	Meaning	What to do
3011	Error	The driver protocol does not match the service (old driver still loaded, or driver and service from different packages); elevation is disabled.	Restart the device (an old driver can still be loaded after the upgrade); if 3011 stays, install driver and service from the same AEMS MSI.
3012	Error	Legacy settings migration failed; nothing was changed.	See Upgrading from 1.0.5.0 or earlier.
3013 / 3014	Error	Licensing: not licensed / license refused or not written.	See Licensing events.
3200 / 3201	Warning	`Start of a child refused [PathAlias]` (alias drive letter, symbolic link, junction) / `[VolumeNotFixed]` (network, UNC, removable, ISO). Logged at Warning level.	See A child program does not start in place.
3212	Error	On a Windows build newer than the validated ones, the last attempt to connect to the driver did not complete (the device probably stopped during it, or AEMSvc was ended during it), so AEMSvc did not connect again: elevation is off, the service keeps running.	Install an updated AEMS version. Only after you checked the cause: set REG_DWORD `ClearBindAttempt` = 1 under `HKLM\SOFTWARE\Logic Unknown\AEMS-DriverBindGuard` and restart AEMSvc to try once more. See Supported Windows versions and the

ID	Level	Meaning	What to do
			guard key after an uninstall.
3214	Error	The connection guard value `BindAttemptBuild` under `HKLM\SOFTWARE\Logic Unknown\AEMS-DriverBindGuard` could not be written (AEMS does not connect to its driver, elevation is off, retried) or could not be removed after the connection (the next start will not connect, event 3212).	Check the permissions of that registry key.
3213	Error	On a Windows build newer than the validated ones, the policy value `DisableDriverBindOnUnvalidatedBuild` = 1 forbids connecting to the driver: elevation is off.	Install an updated AEMS version, or remove the policy value.
3210 / 3211	Error	The service's connection to the AEMS driver for children in place failed (retried every second) / its children-in-place component could not start.	Restart AEMSvc; contact support if it repeats.
3240	Error	The machine's drive letters (global device map) could not be applied to an elevated program, a child started in place or its console host; that process was not started (fail closed). The text names the step and the status.	Restart AEMSvc; contact support if it repeats.
5000	Error	Unexpected error, or AEMSvc stopped itself after an internal safety failure (see Wildcard rule events). Event 5000 is in the Application log; restarts appear as Service Control Manager events in the System log.	If AEMSvc stays stopped, start it and contact support.

AEMS release notes

> **Applies to AEMS 1.0.8.0**
>
> This documentation describes AEMS 1.0.8.0 (service 0.1.9.0, driver 1.0.8.0, Allow-list Tool 1.3.0.0), released 2026-11-20. Install 1.0.8.0. Text marked "from 1.0.8.0" does not apply to earlier versions. Where the text says "from 1.0.6.0", it applies to 1.0.7.0 and 1.0.8.0 as well.

The AEMS documentation describes the current version only. Where behaviour depends on the version, the text says "from 1.0.x.0".

AEMS 1.0.8.0 (released 2026-11-20)

Components: AEMSvc **0.1.9.0**, AEMS driver **1.0.8.0** (a new driver, signed by Microsoft, protocol `0x00020004`; there is no driver 1.0.7.0: AEMS 1.0.7.0 shipped driver 1.0.6.0), Allow-list Tool (`AEMSAllowlist.exe`) 1.3.0.0. AEMS 1.0.8.0 is the version to install. AEMS stays fully offline: no new network or online dependency.

Fixed

- **Programs that finish very quickly no longer run twice or only without elevation.** Up to 1.0.7.0 the user's own start of an allow-listed program ran without administrator rights for a fraction of a second while AEMS decided. A program that did its whole work in that time ran twice (once without administrator rights, then elevated, event 1010), or, when the program that started it had already let go of it, it was not elevated at all (event 2004, "could not be opened (error 87)"). From 1.0.8.0 the driver holds the start of such a program for a moment until AEMS has suspended the new process, before its first instruction runs. The process is then ended and started again elevated, or resumed unchanged when it is not to be elevated. The program that started it (Explorer, `cmd.exe`, PowerShell, a script) sees a normal, successful start: no "Access is denied", no error dialog. See Programs that finish very quickly.
- **The uninstall removes the empty key `HKLM\SOFTWARE\Logic Unknown\AEMS-DriverBindGuard`.** On a device that ran on a Windows build newer than the validated ones, the uninstall of 1.0.7.0 and earlier left this empty key (and with it the `Logic Unknown` vendor key). The uninstall of 1.0.8.0 removes it, unless it holds the value `BindAttemptBuild`: that value means a connection attempt to the driver did not complete, and it is kept on purpose (see Driver connection guard). Uninstalling 1.0.7.0 or earlier still leaves the key; it goes with the first uninstall of 1.0.8.0 or later.

Changed

- **An allow-listed program that is to be elevated waits, suspended, until AEMS has decided.** AEMS decides one launch at a time. When several allow-listed programs start at once (for example a logon script), the later ones wait their turn suspended instead of running without administrator rights first. See Programs that finish very quickly.
- **The upgrade loads the new driver at once.** The installer stops the old driver and starts driver 1.0.8.0, so the hold works right after the upgrade, without a restart (measured on the upgrade from 1.0.7.0). When the old driver cannot be unloaded, the upgrade fails and asks for a restart. If the old driver 1.0.6.0 is nevertheless still loaded after a completed upgrade (not expected), AEMS behaves exactly as 1.0.7.0 for these programs and logs warning **2236** once at every service start; restart the device. The text of 2236 still says that it is expected right after an upgrade until the restart; with 1.0.8.0 treat it as a reason to restart.
- **New events** 2235, 2236 and 2237 (Warning). See the event reference.

Validation of the new driver behaviour. Measured on Windows 11 25H2 (build 26200) and on build 26300. Windows 11 24H2 (build 26100) has the same kernel code base as build 26200 and is covered by that measurement. On build 26300 AEMS 1.0.8.0 still logs warning 2231: the driver's validated build limit stays 26200. **Not validated:** Windows 11 23H2 (build 22631). Windows 10 22H2 (build 19045) was not in the scope of this validation. On every build AEMS checks at every launch that the suspend took effect; if it did not, that launch is handled as in 1.0.7.0 and warning 2235 is logged.

Upgrade: install the AEMS 1.0.8.0 MSI over 1.0.7.0 or 1.0.6.0 (`msiexec /i AEMS.msi /qn` , one run); settings, allow-list and license are kept, and the new driver is loaded by the upgrade. A device that runs 1.0.5.0 or earlier is upgraded directly to 1.0.8.0 in one run, without installing 1.0.6.0 or 1.0.7.0 first; restart the device once afterwards. The upgrade moves the settings as described for 1.0.6.0 (Upgrading from 1.0.5.0 or earlier). Details: Upgrade and repair.

AEMS 1.0.7.0 (released 2026-11-20)

Components: AEMSvc **0.1.8.0**, AEMS driver **1.0.6.0** (unchanged: the same driver file as in AEMS 1.0.6.0, protocol `0x00020003`), Allow-list Tool (`AEMSAllowlist.exe`) 1.3.0.0. AEMS 1.0.7.0 is a service-only fix on top of 1.0.6.0; everything in the 1.0.6.0 notes below applies to it.

Fixed

- **Children in place did not work for standard users whose Windows logon has a split token.** Windows gives a standard user a split token when the user is a member of a group that User Account Control filters (for example Network Configuration Operators, Power Users, Backup Operators or Print Operators) or holds a privilege that User Account Control filters. For such a user every program that an elevated allow-listed program started was refused with "Access is denied" (warning 2224 with `[PRE_TOKEN]`): in 1.0.6.0 children in place did not work for them at all. AEMS now builds the elevated token from the user's full token, and children in place work for these users too (measured: the elevated program, its children

and a chain of them started in place, with no 2224). Event 1226 shows that this path was taken. See Users with a split token.

- **Failed installs no longer remove the driver service.** When an installation failed, the installer's rollback could remove the driver service of an existing AEMS installation. The rollback no longer does this.
- Users without a split token are not affected: they behave exactly as in 1.0.6.0 (measured: same events and results as with 1.0.6.0).

Changed

- For a user with a split token the elevated program now runs in the user's full logon session, as a program elevated by User Account Control does; groups that are "deny only" in the user's normal session are enabled in the elevated program. The same applies to such users who do not use children in place. Details and consequences: Users with a split token.
- **New events** 1226 (Information) and 2234 (Warning). See the event reference.

Upgrade: install the AEMS 1.0.7.0 MSI over 1.0.6.0 (`msiexec /i AEMS.msi /qn`); the driver file is unchanged. A device that runs 1.0.5.0 or earlier is upgraded directly to 1.0.7.0 as described for 1.0.6.0 (Upgrading from 1.0.5.0 or earlier).

AEMS 1.0.6.0 (released 2026-11-19)

Components: AEMSvc 0.1.7.0, AEMS driver 1.0.6.0 (protocol `0x00020003`), Allow-list Tool (`AEMSAllowlist.exe`) 1.3.0.0. The components have their own version numbers: "driver 1.0.6.0" is the driver inside the AEMS 1.0.6.0 MSI; "driver 1.0.3.0" elsewhere in this documentation means the first driver with protocol `0x00020002` (wildcard matching), not the AEMS 1.0.3.0 MSI listed under earlier versions.

Highlights

- **Children in place.** An approved application can start other programs elevated as real children, with their output, exit code and handle, instead of having them blocked and relaunched as a detached copy. Per entry, `ChildInPlace` (with `AllowChildElevation=1`): `1` direct children, `2` the whole tree up to 8 levels. For wrapper programs that call other programs use the whole tree. The programs must lie in a folder that standard users cannot change; a folder created directly under `C:\` does not qualify until its permissions are hardened. See Children in place.
- **Machine drive letters for elevated programs.** Elevated programs no longer follow the user's own drive letters. See Drive letters of elevated programs.
- **No upper Windows build limit.** The driver loads on Windows 10 22H2 (build 19045) and every later build. This release, including children in place, was measured on Windows 11 builds 26200 and 26300. The driver is validated up to 26200; on 26300 AEMSvc logs

warning 2231 and elevation worked. Children in place were not validated on Windows 10. See Supported Windows versions.

Security

- **Elevated programs no longer create objects owned by the standard user.** The token AEMS builds for a standard user is now owned by Administrators, with the default permissions of an administrator elevated through UAC. Files and keys an elevated program creates in protected places (for example an installer under `C:\Program Files`) can no longer be changed by the user afterwards. Objects created under AEMS 1.0.5.0 or earlier keep their owner; reinstall the affected applications. See The elevated token.
- **Elevated programs no longer receive the user's own environment variables.** They get the machine environment plus profile variables computed by AEMS, never the user's `HKCU\Environment`, so a standard user can no longer make an allow-listed program load their code through variables such as `COR_PROFILER`, `DOTNET_STARTUP_HOOKS` or `PATH`. If that environment cannot be built, the program is not elevated (error 3009). The administrator passthrough keeps the administrator's own environment. See Environment of elevated programs.
- **Elevated programs use the machine's drive letters.** Every program AEMS elevates for a standard user, every child started in place and the console host of each elevated console program resolve drive letters through the machine's drive letters, not through the user's own. A standard user can no longer redirect a drive letter (also `C:`) for an elevated program. Consequence: elevated programs do not see the user's own mapped network drives or `subst` letters. A thread that impersonates the user still sees the user's drive letters (known limit). If the machine drive letters cannot be applied, the program is not started (error 3240).
- **Children in place are checked against the file that runs.** A child started in place must lie in a folder that standard users cannot change, on a local fixed drive, and must run as the same user with the same token. Everything else is refused, with no copy. See Children in place.
- **The driver** closes a gap where an elevated application could start a program elevated through the system console host (`conhost.exe`), bypassing default-deny child governance. Elevated console programs such as `cmd.exe` work as before. A parent with `AllowChildElevation=1` that starts a tool through `conhost.exe --headless <tool>` now gets the tool non-elevated.
- **Known limit: programs that finish very quickly run twice** (fixed in 1.0.8.0, see above). The user's original start of an allow-listed program runs without administrator rights for a fraction of a second until AEMS has started the elevated program. A wrapper that does its whole work in that time runs twice: once without administrator rights (its work typically fails or is incomplete) and once elevated (measured in 10 of 10 quick starts, with `EndOriginalDescendants` on and off). This is how allow-listed programs have always been caught, also in 1.0.5.0; no AEMS setting changes it. Programs whose first action must not run without administrator rights are affected; programs that take longer than about half a

second to reach their real work, or whose first attempt simply fails without side effects, are not affected in practice. See Children in place.

- **Known limits of children in place** (see What cannot be closed):
- helpers that an installer unpacks into the user's `%TEMP%` and starts are refused, with no copy (documented limit of 1.0.6.0; leave "Children in place" off for such installers);
- an MSI that an in-place program installs runs its impersonated custom actions in the Windows Installer custom-action server, outside the elevated tree and outside AEMS control; the installation completes, but those actions are not governed;
- scheduled tasks, COM servers and WMI processes that an elevated program starts are outside the tree;
- children started through an 8.3 short name, an alias drive letter, a symbolic link or a junction, or from a network drive, a UNC path, removable media or a mounted ISO are refused;
- sandboxed helpers of Chromium, Edge WebView2 and Electron programs are refused; more than 6 child starts waiting at once per program and trees deeper than 8 levels are refused;
- children of programs started before an AEMS service restart are refused until the program is restarted.
- Not changed by this release: `%TEMP%` and the profile stay writable by the user, and a file an installer stages in `%TEMP%` and then moves into place keeps the user's write access (see Limits).

Compatibility

- Per-user `PATH` entries and all other per-user environment variables (for example proxy variables, `JAVA_HOME`, vendor variables) no longer reach elevated programs; set them machine-wide. `OneDrive*`, `HOMESHARE` and `CLIENTNAME` are not set; `HOMEDRIVE` / `HOMEPATH` and `APPDATA` always point to the local profile (an Active Directory home drive or redirected AppData is not reflected). Full list: Compatibility effects.
- Files an elevated program creates in folders where users have write access only as creator (for example under `C:\ProgramData`) are no longer writable by the user's non-elevated programs.

New

- **Children in place** (`ChildInPlace`, see Highlights). Entries without it handle child processes exactly as in 1.0.5.0.
- **License key.** AEMS elevates only with a valid license key from Logic Unknown ApS; a new installation without a key does not elevate (event 3013). The key is checked offline on the device; AEMS has no portal and no online dependency. A trial license runs 30 days and has no grace after it expires. Keys can be applied with the MSI property `LICENSEKEY`, with `AEMSvc.exe --set-license`, as a policy value or bundled into a customer MSI. Devices that ran 1.0.5.0 or earlier get a one-time 30-day upgrade grace. See Licensing.

- **Wildcard path rules.** An allow-list entry can name an executable by a path pattern (`*`, `?`), pinned by publisher or hash and elevated only from a folder that only administrators can write. See Wildcard path rules.
- **Publisher pins.** New values `Publisher` and `PublisherThumbprint` check the file's embedded Authenticode signature, also on exact-path entries. An entry with a malformed `HashSHA` (not 64 hex digits) or `PublisherThumbprint` (not 40 hex digits) is kept as a refusing entry (warning 2012; every launch it matches is refused with 2013 `[MalformedRule]`). An entry with a missing or empty `DataPath` is dropped (warning 2012).
- **Administrator override.** When a rule's checks fail for an administrator, AEMS relaunches the app with the administrator's own real token and logs warning 2016.
- **Fail closed on unmatched catches.** When the driver catches a launch but AEMS finds no matching entry, a standard user's program is not elevated (warning 2003, then 2004).
- **Inactive wildcard rules refuse.** While the driver of the previous version is still loaded, wildcard rules never elevate, and a launch a wildcard rule matches first is refused (warning 2014 at start, 2013 `[WildcardInactive]`).
- **Service recovery.** If AEMSvc ends unexpectedly, Windows restarts it after 5 seconds, twice; after a third failure within a day it stays stopped (event 5000).
- **New events** 1011, 1012, 1013, 1014, 1015, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 3012, 3013 and 3014, and for children in place 1200-1202, 1221-1225, 2200-2216, 2218-2221, 2223-2233, 3200, 3201, 3210-3214 and 3240; event 5000 can also mean that AEMSvc stopped itself after an internal safety failure. See the event reference.

Changed

- **Registry path.** The configuration moves to `HKLM\SOFTWARE\Logic Unknown\AEMS` and the policy override to `HKLM\SOFTWARE\Policies\Logic Unknown\AEMS`. The MSI upgrade migrates the settings key and verifies the copy; a policy that writes the legacy policy key must be re-targeted. Value names and meanings are unchanged. See Upgrading from 1.0.5.0 or earlier.
- **TamperProtection.** The driver protects exactly `HKLM\SOFTWARE\Logic Unknown\AEMS` and its subkeys, and allows 64-bit registry reads with an explicit registry view (for example Intune detection rules) while `TamperProtection` is on.
- **Upgrade.** The MSI removes the old version first and refuses to start while the driver of the previous installation is still loaded and marked for deletion ("Restart the computer, then run this installation again").
- **Repair.** With `TamperProtection` on, a repair skips its registry write instead of failing with Error 1406.
- **Uninstall** also removes a leftover legacy settings key from 1.0.5.0 or earlier.
- **Downgrade** over 1.0.6.0 is blocked by the MSI.

- **Allow-list Tool** 1.3.0.0, the tool version for AEMS 1.0.6.0: writes the new registry path, adds wildcard path rules, reads `Publisher` and `PublisherThumbprint` from a signed file, and has the "Children in place" choice (Off, Direct children, Whole tree). It always writes `Arguments`, `AllowChildElevation`, `AllowUserArguments` and `ChildInPlace`, so an import replaces a stale value.
- **Fewer double starts.** When the user's un-elevated original of an allow-listed program is ended, the un-elevated programs it had already started are ended too (event 1224, warning 2232 when not all could be ended). A program that already did its work in that fraction of a second is not undone. The new value `EndOriginalDescendants` = 0 keeps the 1.0.5.0 behaviour (event 1225).
- **Warning for open folders.** Warning 2233 at every service start lists the exact-path entries on local fixed disks whose program or folder standard users can change.
- **Driver connection guard on new Windows builds.** On a build newer than the validated ones, AEMSvc does not connect to its driver again after a connection attempt that did not complete (error 3212; `ClearBindAttempt` under `HKLM\SOFTWARE\Logic Unknown\AEMS-DriverBindGuard` to try once more), and the value `DisableDriverBindOnUnvalidatedBuild` = 1 forbids the connection on such builds (error 3213); error 3214 when the guard value cannot be written or removed. See Supported Windows versions.
- **Windows builds.** The driver no longer has an upper Windows build limit. It loads on build 19045 and later; on a build newer than the validated ones elevation depends on the driver's start-up self-check (warning 2231 when it passed, error 3010 when it failed).
- **Customer MSI bundler** (`Build-MsiWithSettings.cmd`, provided by Logic Unknown ApS). Optional license field; legacy key lines in older `.reg` files are rewritten to the new path.
- **License policy value.** An empty or blank policy value `License` is ignored, so clearing the policy field never hides a valid base license.

Action required: run the MSI once and restart the device after the upgrade (again if AEMSvc logs 3011, 2014 or 2220 after the restart: the old driver is then still loaded); harden the folders of programs that should start in place, and check the folder of every exact-path entry, not only of in-place entries; apply a license key within the 30-day upgrade grace, and move policies and scripts to the new registry path after the devices run 1.0.6.0. Set environment variables that elevated programs need machine-wide, and reinstall applications that were installed elevated under 1.0.5.0 or earlier.

Earlier versions

AEMS MSI version	Change stated in the documentation
1.0.5.0	Last version that uses the legacy registry key; upgrade path to 1.0.6.0 documented in Upgrading from 1.0.5.0 or earlier.
1.0.3.0	Driver protocol 2: the driver forwards the full command line and the identity of each caught launch. New value `AllowUserArguments`. The decision follows the token of the launched process ("Run as different user" with an administrator account, other standard accounts).
1.0.2.0	The allow-list is kept when AEMS is upgraded.
1.0.1.0	Upgrading from 1.0.1.0 removes the base allow-list; export it first.

AEMS security model

> **Applies to AEMS 1.0.8.0**
>
> This documentation describes AEMS 1.0.8.0 (service 0.1.9.0, driver 1.0.8.0, Allow-list Tool 1.3.0.0), released 2026-11-20. Install 1.0.8.0. Text marked "from 1.0.8.0" does not apply to earlier versions. Where the text says "from 1.0.6.0", it applies to 1.0.7.0 and 1.0.8.0 as well.

This page describes what AEMS trusts, how it decides whether a launch is elevated, what protects its configuration and license, and where the design stops. Read it before a production rollout.

Trust anchors

Part	Protection
AEMS kernel driver	Signed by Microsoft. Its device `\\.\AEMS` can be opened by SYSTEM only. In addition the driver accepts a caller only when its image path ends in `\Program Files\AEMS\AEMSvc.exe`; this is a path check, not a signature check.
`AEMSvc` service	Runs as LocalSystem. Signed by Logic Unknown ApS (the code-signing certificate subject reads "Logic Unknown Aps").
Configuration	Registry values under `HKLM\SOFTWARE\Logic Unknown\AEMS`, or under the policy hive `HKLM\SOFTWARE\Policies\Logic Unknown\AEMS`, which wins when present.
Audit	Windows Application event log, source `AEMS`. No network or cloud dependency.
Decisions	Made on the device only, from the allow-list in the registry. AEMS has no portal, no cloud service and no online dependency: it never contacts a server, and license keys are checked offline on the device. Signature checks use only the revocation information already cached on the device; AEMS does not fetch it from the network.

How a launch is decided

The driver catches a launch of an allow-listed path; AEMSvc decides from the token of the process that was actually launched:

How the allow-listed app was started	What AEMS does
"Run as administrator" (any admin account), by an administrator with UAC off, or by SYSTEM	Nothing. The process already runs elevated; the driver does not catch it.
"Run as different user" / `runas` with a local administrator account	Relaunched with that account's real full token, child governance off.
By the console user (standard user)	AEMS-built elevated token, default-deny child governance, `HashSHA` enforced. From 1.0.7.0, for a user whose logon has a split token, the token is built from the user's full token (see Users with a split token).
"Run as different user" with another standard account, or from a non-interactive session	Left running unelevated with its own token (event 2004). AEMS never elevates an identity it cannot vouch for.

From 1.0.8.0 the driver holds a caught launch until AEMS has suspended it, so the original does not run while AEMS decides: it stays suspended, with the launching user's own token, and is then either ended and replaced by the elevated relaunch or resumed unchanged (see Programs that finish very quickly for the cases in which it is not held and runs as in 1.0.7.0). If AEMSvc is stopped, nothing is held and top-level launches run unelevated. If AEMSvc ends unexpectedly while a launch is suspended, the driver ends that process (within about 10 seconds); start it again. Child processes of a governed (elevated) application do not start while the service is stopped. With `AdminPassthrough=0`, administrators are treated exactly like standard users.

When the original is ended, the un-elevated programs it had already started in that short time are ended too (one snapshot; same session and user, in its process chain, created after their parent and the original, up to 8 levels and 64 processes; event 1224, or warning 2232 when not all could be ended). A program that already did its work in that time, was started after the snapshot, or broke away from the original's process chain, is not undone; when the original had already exited before the decision (event 1010), nothing is ended. `EndOriginalDescendants` = 0 turns this off (event 1225): such programs then run twice, as in 1.0.5.0.

Every exact-path entry on a local fixed disk whose program or folder standard users can change is listed once per service start in warning 2233: such an entry is elevated, but any user can replace its program.

The elevated token

For a standard user AEMS builds the elevated token itself (top-level relaunch and elevated child relaunch). From AEMS 1.0.6.0 this token has:

- **Owner** `BUILTIN\Administrators`, as the token of an administrator elevated through UAC. Up to 1.0.5.0 the owner was the standard user.
- **Default permissions** (used for an object that is created without explicit permissions and inherits none from its parent): Administrators full control, SYSTEM full control, the user's logon session read and execute. Up to 1.0.5.0 they were the user's own (the user full control).

What follows from it:

- Files, folders and registry keys that an elevated program creates in protected places (`C:\Program Files`, `HKLM\SOFTWARE`) are owned by Administrators. The standard user has no owner rights over them and only the access the parent folder or key grants (in `C:\Program Files`: read and execute), so after an elevated installation the user cannot change or replace the installed files. Up to 1.0.5.0 the user owned them and could change their permissions and contents.
- Where a folder gives users write access only through `CREATOR OWNER` (for example subfolders of `C:\ProgramData`, or folders created directly under `C:\`), files that an elevated program creates there are **no longer writable by the user**, also not by the user's non-elevated programs, including the non-elevated child processes of the same elevated application. If an application writes data there elevated and later changes it unelevated, grant the Users group the needed rights on that folder explicitly.
- Files that an elevated program saves in the user's profile (Documents, AppData, `%TEMP%`) are owned by Administrators, but the user keeps full access through the profile's inherited permissions.
- Objects that inherit no permissions, such as the elevated process itself: the user's non-elevated programs in the same logon session get read and execute access only; the same user in another logon session (a scheduled task with stored credentials, `runas`, a remote session) gets no access. This is the same as for an administrator elevated through UAC.
- **Objects created by earlier versions keep their owner.** Files, folders and keys that elevated programs created under AEMS 1.0.5.0 or earlier stay owned by the standard user; the update does not change them, and their contents may have been changed by the user. Reinstall the affected applications after the update (uninstall, delete the remaining folder, install again). To find such objects, list the owners, for example (read-only):

```
Get-ChildItem 'C:\Program Files','C:\Program Files (x86)','C:\ProgramData' -Recurse -
Force -ErrorAction SilentlyContinue |
  Where-Object { $o = (Get-Acl -LiteralPath $_.FullName -ErrorAction
SilentlyContinue).Owner
                 $o -and $o -notmatch 'SYSTEM$|\\Administrators$|\\TrustedInstaller$' }
|
  Select-Object FullName, @{n='Owner';e={(Get-Acl -LiteralPath $_.FullName).Owner}}
```

- **Administrator passthrough is unchanged.** An administrator's launch uses the administrator's real Windows token, as before.

Environment of elevated programs

From AEMS 1.0.6.0 a program that AEMS elevates for a standard user (top-level and elevated child relaunch) gets:

- the **machine environment** (`HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment` and the values Windows computes). A reference to another machine variable that Windows left unexpanded (for example `%ProgramFiles%` in `PSModulePath`) is expanded; `PATH` and `PSModulePath` keep only absolute folders (drive or network paths), any relative or unresolvable element is left out;
- **profile variables that AEMS computes** for the user: `USERPROFILE` (the profile folder Windows has registered), `HOMEDRIVE`, `HOMEPATH`, `APPDATA` (`<profile>\AppData\Roaming`), `LOCALAPPDATA` (`<profile>\AppData\Local`), `TEMP` and `TMP` (`<profile>\AppData\Local\Temp`), `USERNAME` and `USERDOMAIN` (the names of the user's logon session, for local, domain and Microsoft Entra ID accounts alike), `LOGONSERVER`, `USERDNSDOMAIN` and `SESSIONNAME`;
- **nothing** from the user's `HKCU\Environment` or `HKCU\Volatile Environment`.

Reason: a standard user can write `HKCU\Environment`. Variables such as `COR_PROFILER`, `DOTNET_STARTUP_HOOKS`, `JAVA_TOOL_OPTIONS`, `NODE_OPTIONS` or `PATH` would otherwise make an allow-listed .NET, Java, Electron or similar program load the user's code with administrator rights. If this environment cannot be built (no profile folder, the account name cannot be determined), the program is not elevated (error 3009; a top-level program keeps running unelevated, 2004).

Unchanged: a program that AEMS relaunches **non-elevated** (a child of a governed application) keeps the user's own environment. The **administrator passthrough** keeps the administrator's own environment, exactly as a UAC elevation of that administrator does. With **UAC turned off**, the session token of an administrator is itself elevated, so a child process that AEMS relaunches **non-elevated** for that administrator (a child of a governed application) gets the machine environment described above instead of the administrator's own (the safe choice).

Compatibility effects

- **Per-user `PATH` entries are not used.** Programs installed per user are not found by name by elevated programs: the `%LOCALAPPDATA%\Microsoft\WindowsApps` aliases (such as `winget` and

Store Python), the per-user setup of Visual Studio Code, per-user Python, npm global tools, .NET tools and user-scope Git.

- **Every other `HKCU\Environment` value is dropped**: proxy variables (`HTTP_PROXY`, `HTTPS_PROXY`, `NO_PROXY`, `ALL_PROXY`), `JAVA_HOME`, `GOPATH`, per-user `PSModulePath` additions, vendor and licensing variables. Set a variable that an elevated program needs machine-wide (System Properties > Environment Variables > System variables, or Group Policy / Intune), then restart the program.
- `OneDrive`, `OneDriveCommercial`, `OneDriveConsumer`, `HOMESHARE`, `CLIENTNAME` and `USERDOMAIN_ROAMINGPROFILE` are not set.
- `HOMEDRIVE` / `HOMEPATH` always point to the local profile: an Active Directory home drive (for example `H:`) is not reflected in the variables. The mapped drive itself is not visible to elevated programs either (see Drive letters of elevated programs).
- `APPDATA` / `LOCALAPPDATA` always use the local default layout: AppData that is redirected by folder redirection is not reflected.
- A `TEMP` / `TMP` that the user moved is ignored.
- A machine variable that refers to `%USERPROFILE%` or `%TEMP%` points to the Default profile or `C:\Windows\TEMP` (Windows expands such references without a user).

Limits

- **A program that finishes very quickly** runs once, elevated, from 1.0.8.0. While an old driver is still loaded after a completed upgrade (not expected; warning 2236) and after a hold timeout it can still run twice, once without administrator rights and once elevated (see What cannot be closed, item 11).
- **`%TEMP%` and the profile stay writable by the user.** The environment change removes values the user chose; it does not make the user's folders trustworthy. An installer or self-extractor that unpacks to `%TEMP%` and runs or loads files from there can still be given planted or replaced files, and programs still load content from the profile by themselves (for example PowerShell profile scripts and the user module folder `Documents\WindowsPowerShell\Modules`, which Windows PowerShell adds itself, and application settings under AppData).
- **A file staged in `%TEMP%` and then moved into place stays writable by the user.** A move within one drive keeps the file's permissions, so a file that an installer creates in the user's `%TEMP%` and then moves into `C:\Program Files` is owned by Administrators but keeps the user's full-control entry. Only files created in place are protected.
- **Machine `PATH` folders that standard users can write** are the search path of every elevated program. Make sure no folder in the machine `PATH` is writable by Users (installers sometimes add such folders, for example `C:\Python310\`).
- **The signed-in user can still end an elevated program.** An elevated program runs in the user's own logon session and on the user's desktop, so the signed-in user can still close or end an elevated program they started. This is by design.

- **Programs that sandbox their own helper processes are not supported as elevated applications** in AEMS 1.0.6.0: Chromium-based browsers, Electron applications and applications that host WebView2. Such a program starts its helper processes (renderer, GPU, utility) with a deliberately reduced, sandboxed token. AEMS governs these helpers like any other child process: each is blocked when it is created and started again by AEMS outside the sandbox the program asked for (non-elevated, or elevated under `AllowChildElevation=1`). Such a program may therefore not work when elevated, and a helper meant to handle untrusted content does not run in its sandbox. With "Children in place" the sandboxed helpers are refused instead: the program starts, its helpers do not (blank windows or an exit). Do not allow-list such programs.

Default-deny rules

- AEMS elevates **only the applications you allow-list**.
- **Prefix matching.** A `DataPath` without wildcards is matched as a prefix of the launched path. Always write the full path including the file name; a folder path approves every program below it.
- **Child processes are not elevated by default.** With `AllowChildElevation=1` an app's **direct** children are relaunched elevated; elevation continues down a chain only where every link is opted in. With `ChildInPlace` as well, children start elevated in place (see Children in place). Opt in only specific, trusted parent apps. Opting in a general shell or interpreter (`cmd.exe`, `powershell.exe`, `pwsh.exe`, `wscript.exe`, `python.exe`, ...) is allowed, but it makes the user an administrator while it runs: such a parent elevates every child it spawns, with no per-child allow-list or hash gate, and with "Whole tree" everything the user types in it. Without `ChildInPlace` the guards below decide only in part for such a child: a child whose file fails its own entry's hash, or whose entry is malformed or an inactive wildcard rule, is **not started at all**; a `Publisher`, thumbprint or trusted-location failure does **not** stop it - the parent's `AllowChildElevation=1` still elevates it.
- **Programs started by a console host are never elevated.** When an elevated application starts the console host process with a program to run (for example `conhost.exe <program>` or `conhost.exe --headless <tool>`), the program that the console host starts runs non-elevated, whatever the application's `AllowChildElevation` setting. Commands typed in the console window of an elevated, allow-listed console program such as `cmd.exe` are children of that program, not of the console host: they follow that entry's `AllowChildElevation` as before (elevated with `1`, non-elevated otherwise). Elevated console programs themselves work as before.
- **Arguments are locked by default.** A standard user's elevated relaunch is `"<DataPath>" <Arguments>`; `AllowUserArguments=1` forwards the user's own command line. Never set it on a host that executes what its arguments name (`cmd.exe`, `powershell.exe`, `mmc.exe`, `rundll32.exe`, `regedit.exe`, script hosts).

- **Pins.** `HashSHA` is enforced for standard users (for administrators it is informational). From 1.0.6.0 `Publisher` and `PublisherThumbprint` are enforced on any entry that sets them. An entry with a malformed `HashSHA` or `PublisherThumbprint` is kept in its place as a refusing entry (warning 2012), so a later, broader entry cannot elevate the same file without the pin.
- **Fail closed.** When the driver catches a launch but AEMS finds no matching entry, a standard user's program is not elevated (warning 2003).
- **Administrators.** A failed check does not block an administrator: AEMS relaunches the app with the administrator's own real token and logs warning 2016. With the default UAC setting an administrator gets nothing they could not already get by accepting a UAC prompt; a standard user cannot opt in.

Children in place

With `ChildInPlace` on an entry that also has `AllowChildElevation=1`, the programs that the elevated application starts are checked while Windows creates them and, when they pass, start elevated as real children (see Children in place on the Configure page). This section states what that changes for security.

What gets stronger

- **A child starts only from a folder that standard users cannot change**, on a local fixed drive, with the same user and the same elevated token as its parent. Without `ChildInPlace` a direct child of an `AllowChildElevation=1` parent is relaunched elevated from wherever it lies.
- **The decision is bound to the file that actually runs.** The driver reports the path and the volume of the file Windows is about to start, not the name the parent used, and AEMS judges exactly that file. A child started through an alias drive letter, a symbolic link or a junction, or from a network or removable drive, is refused, so the user cannot point the start at a different file.
- **No copies.** Inside an in-place tree every child is either started in place or refused, at every depth. Nothing is relaunched as a separate elevated copy.
- **Locked arguments stay locked.** A child whose own entry locks its arguments starts only with exactly those arguments.
- **Machine drive letters for the whole elevated tree.** The elevated application, every child started in place and the console host of each elevated console program use the machine's drive letters, not the user's own (see Drive letters of elevated programs).

What changes, and what you accept with it

- The command line, working directory, environment, inherited handles and console of a child started in place are chosen by its parent, not by AEMS.

- A `Publisher` or `PublisherThumbprint` pin on a child's own entry is not checked for an in-place start; it only ends the chain there. The folder rule is what protects the file.
- "Whole tree" spreads elevation to every program below the application, up to 8 levels. Without it, elevation reaches direct children plus chains in which every link is opted in.
- With "Direct children", a grandchild is refused instead of relaunched as a copy, also when it has a plain `AllowChildElevation=1` entry of its own.
- Each child waits up to 2 seconds while AEMS checks it. An application with more than 6 child starts waiting at the same moment gets "Access is denied" for the extra ones; they are refused, not queued.
- A user who holds the program file of a child open in a special way (an oplock) can make that child be refused. AEMS fails closed: the child does not start.
- A child started from a network share that does not answer can make the starting application wait until Windows gives up on the share; the child is then refused (by design, not measured).

What cannot be closed (read before you opt in)

1. **A program that can start other programs makes the user an administrator while it runs.** This is true for PowerShell, `cmd.exe`, a terminal, a script host, and anything with an Open dialog or plugins. An elevated shell with "Whole tree" elevates whatever the user types. The same was already true for `AllowChildElevation=1`.
2. A child with an Open or Save dialog can change any file as administrator without starting a process.
3. The parent chooses the arguments, working directory and environment of a child started in place.
4. A helper started from the user's `%TEMP%` or any other folder a standard user can change is refused and no copy starts (documented limit, from 1.0.6.0). Libraries (DLLs) and data files that a child started in place loads from the user's `%TEMP%` are not checked at all: loading a file does not start a process.
5. **Processes that Windows components start on the program's behalf are outside the elevated tree and outside all AEMS control**, some of them with the user's elevated token: the Windows Installer custom-action server that runs an MSI's impersonated custom actions, scheduled tasks, COM servers that the elevated program activates, processes created through WMI (`Win32_Process.Create`), and services that start a process for their caller (for example the Secondary Logon service). For an MSI that an in-place program installs, "works" means the installation completes, not that its custom actions are governed. They also keep the user's drive letters.
6. **A thread that impersonates the user still sees the user's own drive letters.** The machine drive letters protect only threads that do not impersonate. A thread of an elevated program, or of one of its in-place children, that impersonates a token of the user's logon session (some

installers and installer frameworks, RPC or COM servers that impersonate a client of the same session, programs that call `ImpersonateSelf`) resolves drive letters through the user's own mappings, which the user can change. Such a program can be redirected by the user (measured). AEMS's own checks never run impersonated, and the check of which file a child starts does not depend on drive letters.

7. While the AEMS service is stopped, starting or restarting, elevated console programs keep working, but their console host does not get the machine drive letters (counted in warning 2221). If Windows refuses to apply the machine drive letters at all, no elevated program starts (error 3240, fail closed).
8. Children that ask for their own elevation (`requireAdministrator` in their manifest) are expected to fail with error 740, as they do with the relaunch (in place: not measured).
9. Sandboxed helper processes of Chromium, Edge WebView2 and Electron programs are refused, so such a program starts but its helpers do not. Programs on a volume that is not NTFS (ReFS, Dev Drive, FAT, exFAT) are refused in place. Trees deeper than 8 levels are refused.
10. A child started through an alias drive letter, a symbolic link or a junction, from a network drive, a UNC path, removable media or a mounted ISO is refused in place. For programs that must start children from such places, leave "Children in place" off: today's block-and-relaunch then applies, with its weaker checks.
11. **A program that finishes very quickly can run twice** in the cases below; otherwise, from 1.0.8.0, it runs once, elevated: the driver holds the start until AEMS has suspended the new process before its first instruction (at most 1 second; see Programs that finish very quickly). The hold only delays a start, it never refuses one, and a suspended process carries only the user's own token, so nothing in it grants more than the user has without AEMS. If AEMSvc ends unexpectedly while a process is suspended, the driver ends that process. The cases: if an old driver is still loaded after a completed upgrade (not expected; warning 2236, restart the device), after a hold timeout or while the hold breaker is open (warning 2235), and when the suspend could not be checked (warning 2235). Then, as in 1.0.7.0, the user's original start of an allow-listed program runs without administrator rights for a fraction of a second until AEMS has started the elevated program. A wrapper that does its whole work in that time has run once without administrator rights (its work typically fails or is incomplete) and then once elevated (measured in 10 of 10 quick starts; `EndOriginalDescendants` does not change it, because it stops only programs that are still running). This is how allow-listed programs have always been caught, also in 1.0.5.0, and no AEMS setting changes it. A program whose first action changes something before it needs administrator rights is affected; programs that take longer than about half a second to reach their real work, or whose first attempt simply fails without side effects, are not affected in practice.

Users with a split token

Windows gives a standard user a **split token** when the user is a member of a group that User Account Control filters (for example Network Configuration Operators, Power Users, Backup Operators, Print Operators, and some domain groups) or holds a privilege that User Account Control filters. The user's normal programs then run with a filtered token, in which such a group is "deny only". `whoami /groups` shows such a group as "Group used for deny only".

From AEMS 1.0.7.0 the elevated token for such a user is built from the user's **full** token (event 1226). What that means:

- The elevated program runs in the user's full logon session, as a program elevated by User Account Control does. Windows reports it as fully elevated.
- Groups that are "deny only" in the user's normal session are **enabled** in the elevated program, in addition to Administrators (as before). Locally this gives nothing that the Administrators group of the elevated program does not already give.
- The privileges are the fixed AEMS set, exactly as before; none are taken from the user's token.
- On the network nothing changes: an elevated program always acted with the user's own account, with every right that account has on the network (also the rights of a powerful domain group). Putting a program on the allow-list for such a user does not limit it to local administration.
- Network connections that the user made only in the normal session (for example `net use` with other credentials) are not shared with the elevated program (by design, not measured). Drive letters do not change: elevated programs already use the machine's drive letters (see Drive letters of elevated programs).
- If the user's full token cannot be read, the program is not elevated (warning 2234, fail closed); AEMS never falls back to the filtered token.
- An **administrator** whose logon has a split token, on a device where administrator passthrough is switched off (`AdminPassthrough=0`), is handled the same way: the elevated token is built from the administrator's full token, with the fixed AEMS privileges and child governance. With administrator passthrough on (the default) nothing changes for administrators.

In AEMS 1.0.6.0 the elevated token of such a user was built from the filtered token. Children in place were then always refused (warning 2224 `[PRE_TOKEN]`).

Drive letters of elevated programs

From AEMS 1.0.6.0 every program that AEMS elevates for a standard user (top-level relaunch, elevated child relaunch and every child started in place) resolves drive letters through the machine's drive letters (the global device map), not through the user's own. A standard user can create drive letters for their own logon session (`subst` , a mapped network drive, or a private

redirect of an existing letter such as `C:`). Up to 1.0.5.0 an elevated program followed such letters, so a user could make it open a different file than intended.

What follows:

- **An elevated program does not see the user's own mapped network drives or `subst` letters.** Paths on them are "not found". Drive letters defined for the whole machine stay visible. Measured: the user saw a `subst` letter; the elevated application, its child started in place and that child's child did not.
- This applies to every elevated launch, with or without `ChildInPlace`. The administrator passthrough is not affected (it runs with the administrator's own token).
- If the machine drive letters cannot be applied to a process, AEMS does not start it (error 3240).
- The limits are in What cannot be closed items 5 to 7: impersonating threads and processes outside the elevated tree keep the user's drive letters.

Protecting the configuration

- The policy hive wins over the base hive; a complete `Applications` subtree there replaces the base allow-list, so a local edit of the base hive cannot weaken a pushed list.
- `TamperProtection=1` (opt-in, off by default) makes the driver deny registry changes under `HKLM\SOFTWARE\Logic Unknown\AEMS` for every process except AEMSvc. It is tamper evidence against casual edits, **not a barrier against a determined local administrator**, and it does not cover the policy hive. See the TamperProtection notes.

License keys

- A license key is checked offline, on the device, against the signature of Logic Unknown ApS. AEMS never contacts a license server.
- A license key is not bound to a device: whoever has it can use it until it expires. Treat it as confidential.
- Every local user, also a standard user, can **read** the applied key in the registry (`HKLM\SOFTWARE\Logic Unknown\AEMS\License`, the policy value and a staged request use the default `HKLM\SOFTWARE` access rights). Only administrators (with `TamperProtection=1`, only AEMSvc) can change it.
- `LICENSEKEY=<key>` puts the key on the msiexec command line. The MSI log masks it, but process-creation auditing, Sysmon, EDR telemetry and management tools record command lines. For fleets prefer the policy value or a customer MSI; `--set-license` reads the key from standard input.
- The license string and the e-mail address in it are never written to the event log, and `--license-status` never prints the key.

- Licensing is enforced against standard users; it relies on administrators not tampering with the device (see Upgrade grace).

Residual risks

- The driver depends on undocumented Windows kernel structures. It checks them at every start and grants nothing when they do not match (error 3010, no elevation). On a Windows build newer than the validated ones it loads, and elevation depends on that self-check alone; a part of the driver then runs on a kernel that was not tested (see Supported Windows versions).
- Children in place: see What cannot be closed.
- The driver's caller check is a path check, not a signature check; the protection against other callers is that its device can be opened by SYSTEM only.
- An attacker who already has SYSTEM or local administrator rights is out of scope: administrators can stop the driver, change the configuration and uninstall AEMS.
- The trusted-location check of wildcard rules decides by storage bus, not by "internal" or "external": eSATA, Thunderbolt or USB4 (NVMe) enclosures, external SAS and Storage Spaces pools on USB disks are not detected as external (see Wildcard rule limitations). Point wildcard rules at the system drive and restrict external storage with device control.
- Paths without wildcards are prefix matches (see "Default-deny rules").
- `%TEMP%` and the profile of the user stay writable by the user, also for elevated programs; a file staged in `%TEMP%` and moved into place keeps the user's write access; user-writable folders in the machine `PATH` reach every elevated program (see Limits).
- Objects that elevated programs created under AEMS 1.0.5.0 or earlier stay owned by the standard user (see The elevated token).
- Programs that sandbox their own helper processes (Chromium-based browsers, Electron, WebView2 hosts) are not supported as elevated applications (see Limits).

Scope and limits

- AEMS targets a kiosk with **one interactive user** and elevates into the active console session. Multi-user Remote Desktop Services is out of scope.
- If AEMSvc stops, nothing is elevated until it runs again. If it ends unexpectedly, Windows restarts it after 5 seconds, twice; after a third failure within a day it stays stopped (event 5000) until it is started manually.
- The driver refuses to load on Windows builds below 19045 (Windows 10 22H2). There is no upper build limit; after a feature update to a newer build, elevation works only when the driver's start-up self-check passes (see Supported Windows versions).

- A "Run as different user" administrator window left open on the kiosk desktop lets whoever is at the keyboard start allow-listed apps elevated without a prompt. Sign such windows out after use.

Wildcard rule guards

The driver only **catches** a launch that matches the pattern; AEMSvc decides. Every configured check must pass. A check that fails or cannot be completed counts as failed (fail closed): a standard user's process is released and continues **unelevated**, and AEMSvc logs warning 2013 with the image, the entry name and the precise reason (for an administrator see 2016 in the outcomes table).

1. **Trusted location** (wildcard rules only):
2. the file can be opened without write or delete sharing; a file that is still being copied or that another process holds open for writing is refused (retry when the copy is done), and a launch can be delayed while another process holds the file open;
3. the launched path is the file's canonical long path: an 8.3 short path such as `C:\PROGRA~1\...` or an alternate data stream is refused (use the long path);
4. the drive is reported as a fixed drive, and the disk's storage bus is one AEMS accepts: SATA, SAS, SCSI, ATA/ATAPI, NVMe, RAID, Storage Spaces, SCM or UFS (this includes the virtual disks of Hyper-V, VMware and Azure virtual machines). Refused: removable media (whatever the bus), and disks on the USB, FireWire (1394), SD, MMC (eMMC), iSCSI, Fibre Channel, SSA, virtual and file-backed virtual (mounted VHD/VHDX or ISO) buses, an unknown bus or any other bus not in the accepted list, a volume that cannot be opened or queried, and network drives and RAM disks that are not reported as fixed drives (code `RemovableVolume` or `NotFixedVolume`). The check is by bus, not by where the disk sits: disks on an accepted bus that are attached externally (for example a Thunderbolt/USB4 NVMe or eSATA enclosure, external SAS, or a Storage Spaces pool built on USB disks) are **not** detected as external, so point wildcard rules at the system drive;
5. the drive letter is a global one: a drive letter created or redirected for one user (`subst`, a mapped drive, a per-user redirect of an existing letter) is refused (code `NotFixedVolume`). The reason says which view saw the different drive letter: the "original's logon session" (the user who started the program) or the "logon session of the elevated process"; in both cases use a real local path on a global drive letter;
6. no component of the path (the file or any folder above it) is a reparse point (junction, symbolic link, mount point);
7. the file and every folder up to and including the drive root are owned by SYSTEM, BUILTIN\Administrators or NT SERVICE\TrustedInstaller;
8. no allow ACE for any other SID grants a modifying right: on the file `FILE_WRITE_DATA`, `FILE_APPEND_DATA`, `FILE_WRITE_EA`, `FILE_WRITE_ATTRIBUTES`, `DELETE`, `WRITE_DAC`, `WRITE_OWNER` (or a generic right that maps to one of them); on the file's folder `FILE_ADD_FILE`,

`FILE_ADD_SUBDIRECTORY`, `FILE_DELETE_CHILD`, `DELETE`, `WRITE_DAC`, `WRITE_OWNER`; on higher folders `FILE_DELETE_CHILD`, `DELETE`, `WRITE_DAC`, `WRITE_OWNER`; on the volume root (`C:\`, `D:\`) `FILE_DELETE_CHILD`, `WRITE_DAC` and `WRITE_OWNER` (a `DELETE` right on a volume root is ignored, because a volume root cannot be deleted, so the default root of a data volume such as `D:\` does not block a rule). (The inherit-only right of Authenticated Users to create subfolders in `C:\` does not count for `C:\` itself.)

9. **A folder created directly in the drive root fails.** On a default Windows installation `C:\` passes Modify for Authenticated Users on to every new folder below it, so a folder such as `C:\Contoso` fails this check even when an administrator or an installer created it. Use a folder under `C:\Program Files`. If a folder in the drive root is required, remove the inherited entries and grant only administrators full control (see Wildcard path rules for the `icacls` commands).
10. A NULL DACL fails. Deny ACEs are ignored (conservative). Inherit-only ACEs do not count for the object itself; an effective `CREATOR OWNER` or `OWNER RIGHTS` ACE is evaluated like any other SID.
11. **Identity pin**: the file carries a valid **embedded** Authenticode signature whose leaf certificate CN equals `Publisher` (and whose SHA-1 thumbprint equals `PublisherThumbprint` when set), and/or its SHA-256 equals `HashSHA`. `Publisher` compares the certificate's common name (CN) only, not the issuing authority. Revocation is not checked online, only from the local cache (the launch never waits for the network), so a revoked certificate may still be accepted; `PublisherThumbprint` or `HashSHA` is the strict option. An expired certificate with a valid timestamp is valid, as in normal Authenticode.

AEMSvc opens the file once without write or delete sharing, runs the checks while it holds that handle and keeps it open until the elevated relaunch has started, so the file cannot be swapped between the check and the launch.

For a locked wildcard rule the launch line is `"<launched file>" <Arguments>`: the concrete file the user started, never the pattern.

`Publisher` / `PublisherThumbprint` on an **exact-path** entry are enforced too (a renewal-proof alternative to `HashSHA`); the trusted-location check applies to wildcard rules only. An exact entry that sets none of the new values behaves exactly as before.

Outcomes:

Situation	Result
Standard user, top-level launch, all checks pass	Elevated; information event 1009 names the rule.
Standard user, top-level launch, a check fails	Released unelevated; warning 2013 with the reason.
Child of a governed parent, a check fails	Warning 2013, two cases. (a) The file itself does not pass its own entry's identity (`HashMismatch`, `HashFailed`, `MalformedRule`) or the entry is an inactive wildcard rule (`WildcardInactive`): the child is NOT started at all, whatever the parent's policy. (b) A `Publisher`, thumbprint or trusted-location failure: the child is handled like a child that is not on the allow-list (its own entry is ignored: no configured `Arguments`, its own children are not elevated), and the parent decides: unelevated when the parent does not allow child elevation; still elevated, by design, when the parent has `AllowChildElevation=1` (the 2013 text says which).
Administrator (passthrough), a check fails	Warning 2016 "ADMINISTRATOR OVERRIDE": relaunched elevated with the administrator's own real token, as administrator passthrough always does. A standard user would have been refused (2013).
Standard user, the driver caught the launch but AEMSvc finds no matching entry	Warning 2003 "NOT elevated (fail closed)", followed by 2004: the original keeps running unelevated. Check that the driver and AEMSvc come from the same MSI and that `DataPath` matches the path.
Unlicensed	Unchanged: the license check wins before any of this.

The guards protect top-level launches and a child's own rule. A child whose file fails its own entry's hash (or whose entry is malformed or an inactive wildcard rule) is not started at all. A `Publisher`, thumbprint or location failure does **not** stop the child of a parent with `AllowChildElevation=1` and no `ChildInPlace`: such a parent elevates its direct children whatever rule they match, so the warning about `AllowChildElevation` applies unchanged. A child started in place always gets the folder check (see Children in place).

Wildcard rule limitations

- **Embedded signatures only.** A file that is only catalog-signed (many Windows binaries, such as `notepad.exe`) has no embedded signature and cannot satisfy `Publisher`; use `HashSHA` for such a file.
- **Storage check by bus.** Network shares, removable media, disks on the USB, FireWire (1394), SD, MMC (eMMC), iSCSI, Fibre Channel, SSA, virtual, file-backed virtual or unknown buses, a volume that cannot be queried, mounted VHD/VHDX files and drive letters created for one user (`subst`, mapped drives) always fail the trusted-location check. Disks on an accepted bus that are attached externally (Thunderbolt/USB4 NVMe or eSATA enclosures, external SAS, Storage Spaces pools on USB disks) are not detected as external; point wildcard rules at the system drive and restrict external storage with device control.
- **Long paths and closed files only.** A launch through an 8.3 short path, or of a file that is open for writing (for example still being copied), is refused.
- **No online revocation check.** Revocation is read from the local cache only, so a revoked signing certificate may still be accepted. `PublisherThumbprint` or `HashSHA` is the strict option.
- `Publisher` **is the common name only.** It does not pin the certificate authority; add `PublisherThumbprint` when that matters.
- **A pin identifies the executable, not its folder.** `Publisher`, `PublisherThumbprint` and `HashSHA` check the `.exe` only. Programs load libraries (DLLs) from their own folder, so no folder of **any** elevated entry, wildcard or exact, may be writable by standard users; otherwise a user can plant a library that runs elevated. Wildcard rules enforce this; for exact entries it is the administrator's responsibility.
- **Launch delay.** The launch can be delayed while another process holds the file open.
- **Trusted SIDs are fixed** to SYSTEM, BUILTIN\Administrators and NT SERVICE\TrustedInstaller. A folder that a deployment account or a group such as "Software installers" can write fails the check.
- **Elevation on launch, not on arrival.** AEMS elevates a matching file when a user launches it. Nothing is started when a file merely appears in the folder.
- **Driver and service versions.** Wildcard rules need driver 1.0.3.0 or later with protocol `0x00020002` or later (AEMS 1.0.6.0 ships driver 1.0.6.0, protocol `0x00020003`), which runs after the 1.0.6.0 upgrade and the restart that follows it (restart again if 2014 or 3011 shows the old driver is still loaded). With an older driver AEMSvc does not send wildcard rules and logs warning 2014 once at start; exact entries keep working. The wildcard rules are then never elevated, and a launch that a wildcard rule matches first is refused even when a later, broader entry would catch it (2013 `[WildcardInactive]`; administrator 2016; child not started). If 2014 is still logged after the reboot, the installed package does not contain driver 1.0.3.0 or later: install the current AEMS MSI (1.0.8.0) from Logic Unknown ApS. AEMSvc 0.1.4.0 and earlier (AEMS 1.0.5.0 and earlier) ignore `Publisher` and `PublisherThumbprint` entirely and treat a pattern as a literal path that never matches.

Folders of elevated programs

`Publisher`, `PublisherThumbprint` and `HashSHA` identify the executable only. Programs load libraries (DLLs) from their own folder, so the folder of **every** elevated entry, wildcard or exact, must not be writable by standard users; otherwise a user can place a library there that then runs elevated. Wildcard rules check this; for exact entries it is your responsibility (install allow-listed programs under `C:\Program Files`).

Report a security issue

Report a security issue to Logic Unknown ApS through the contact page (mark the message as a security report).