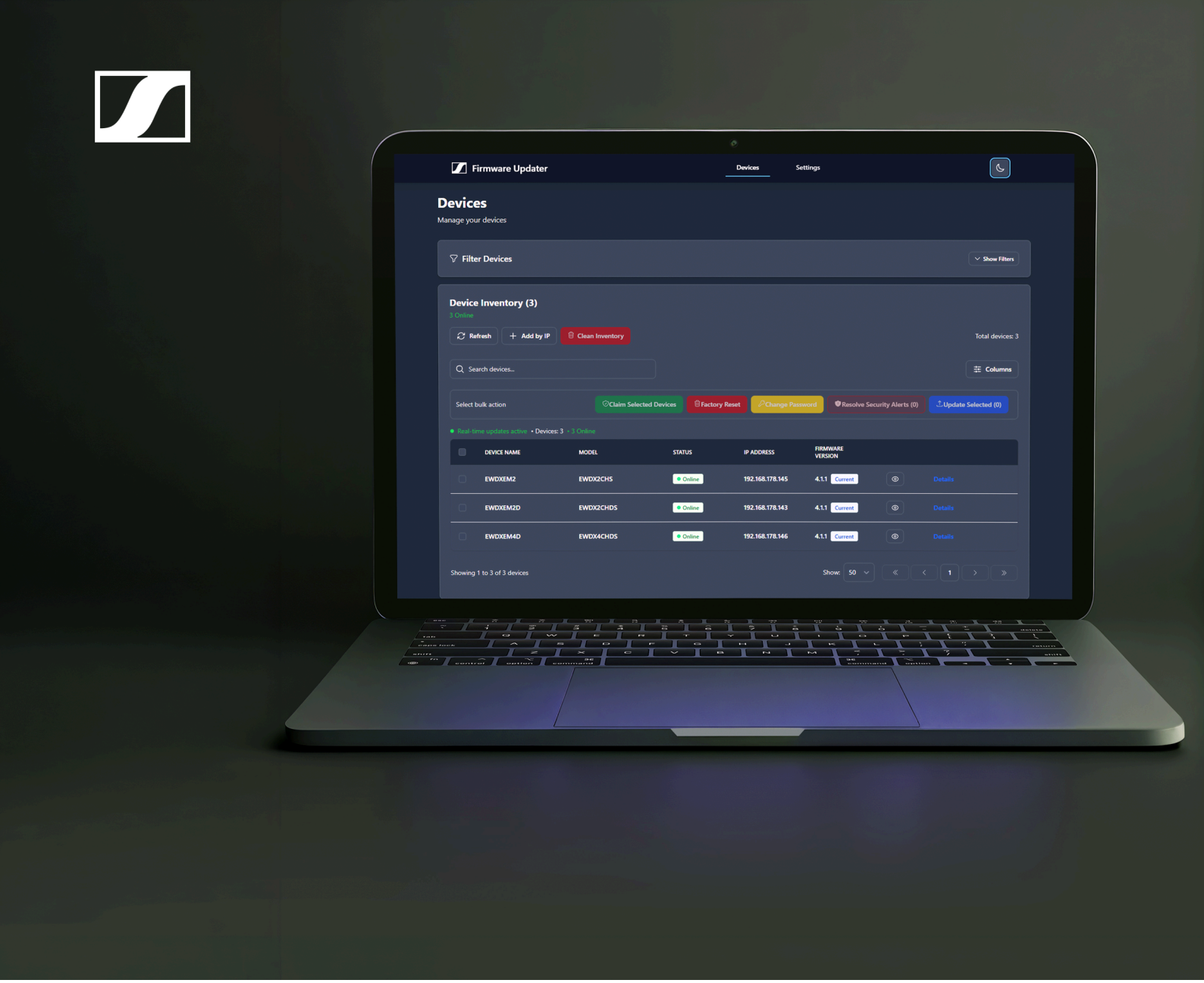



# Firmware Updater

## Guide for IT administrators

PDF export of the original HTML manual

# Contents

# 1. Overview

**The Sennheiser Firmware Updater discovers and updates EW-DX devices on a local network.**

The Sennheiser Firmware Updater (SFU) is a desktop application for discovering, managing, and updating the firmware of Sennheiser EW-DX wireless receivers on a local network.

The application uses two processes:

- An Electron UI that runs in the user session.
- A bundled .NET backend service (SASS, Sennheiser Application Service Suite) that the UI starts on startup. SASS handles all device communication and firmware operations. It binds only to `localhost:8181` and is not accessible from the network.

The current release supports the following device families. Future versions are intended to expand coverage to nearly all Sennheiser networked devices.

Table 1. Supported devices

| Model | Description |
|---|---|
| EWDX2CH | EW-DX 2-Channel Receiver |
| EWDX2CHD | EW-DX 2-Channel Dante Receiver |
| EWDX4CHD | EW-DX 4-Channel Dante Receiver |
| EWDX2CHS | EW-DX 2-Channel Secure Receiver |
| EWDX2CHDS | EW-DX 2-Channel Dante Secure Receiver |
| EWDX4CHDS | EW-DX 4-Channel Dante Secure Receiver |
| L6000 | L 6000 Battery Charger |

# 2. System Requirements & Prerequisites

**The app requires ASP.NET Core Runtime 8.x on Windows and macOS, with platform-specific installation and verification steps.**

The bundled SASS backend is a framework-dependent .NET 8 application. It is not self-contained on either platform. The target machine must have the ASP.NET Core Runtime 8.x shared framework, Microsoft.AspNetCore.App 8.x.

The recommended way to satisfy this requirement on both platforms is to install the .NET 8 SDK. On macOS, Microsoft does not provide a standalone ASP.NET Core Runtime installer. It provides only a .tar.gz archive that must be unpacked and placed on PATH manually. The .NET 8 SDK is available as a standard .pkg installer, so it gives the same installation flow on Windows and macOS. A pre-provisioned ASP.NET Core Runtime 8.x is also valid.

**The dependency is handled differently by platform:**

- On Windows, the setup wizard detects a missing shared framework and can download and install the .NET 8 SDK automatically.
- On macOS, nothing is bundled and no auto-installer is available. Install the .NET 8 SDK on the machine before you start the application.

The application checks for an ASP.NET Core 8.x runtime every time it starts. It performs this verification independently of the installer. If no runtime is found, it shows a blocking error dialog with a download link and does not start the backend. This also covers cases where the runtime was declined during installation or removed later.

### Windows

Table 2. Windows requirements

| Requirement | Details |
|---|---|
| Operating system | Windows 10 or Windows 11, 64-bit only. |
| .NET | ASP.NET Core Runtime 8.x shared framework required. The setup wizard checks for %ProgramFiles%\dotnet\shared\Microsoft.AspNetCore.App\8.* and, if missing, offers to download and install the .NET 8 SDK automatically. The app re-verifies at launch by probing dotnet on PATH, %ProgramFiles%\dotnet\dotnet.exe, %ProgramFiles(x86)%\dotnet\dotnet.exe, and %LOCALAPPDATA%\Microsoft\dotnet\dotnet.exe. |
| Disk space | Approximately 300 MB, excluding .NET. |
| Network | Local network access to Sennheiser EW-DX devices. |

Table 2. Windows requirements (continued)

| Requirement | Details |
|---|---|
| Administrator rights | Not required for the app itself. It installs and runs per user. Administrator rights are required only for the optional .NET auto-install and for adding firewall rules. |

### macOS

Table 3. macOS requirements

| Requirement | Details |
|---|---|
| Operating system | macOS 12 Monterey or later. |
| Architecture | Apple Silicon, ARM64 only. |
| .NET | The .NET 8 SDK must be installed separately. Nothing is bundled and the app installs nothing itself. Install it via the official .pkg installer at https://dotnet.microsoft.com/en-us/download/dotnet/8.0 or with Homebrew by running brew install --cask dotnet-sdk@8. The SDK is recommended because the ASP.NET Core Runtime alone is only published as a .tar.gz for macOS. The app checks for the shared framework on PATH or at /usr/local/share/dotnet/dotnet, /usr/local/bin/dotnet, /opt/homebrew/opt/dotnet/bin/dotnet, /opt/homebrew/bin/dotnet, or ~/.dotnet/dotnet. A GUI-launched app does not inherit the login shell PATH, so the absolute locations are probed as well. |
| Disk space | Approximately 350 MB, excluding .NET. |
| Network | Local network access to Sennheiser EW-DX devices. |
| Security | Code-signed by Sennheiser electronic, XKNNU9WTL7, and notarized by Apple. No Gatekeeper warnings are shown. |

### Verify the runtime

Run `dotnet --list-runtimes` and confirm that an entry for `Microsoft.AspNetCore.App 8.x` is present.

# 3. Installing Firmware Updater

Install the Firmware Updater on Windows or macOS with the required system components.

**To install the Firmware Updater on Windows:**

- Run `Sennheiser.FirmwareUpdater.Setup.<version>.<buildId>.exe`.
  - The installer uses NSIS and configures a per-user installation with the following behavior:

    **Installation scope:** Per-user ( `%LOCALAPPDATA%\Programs\Sennheiser.FirmwareUpdater\` ). No elevation to Administrator is required.

    **Execution level:** asInvoker — the application always runs with standard user privileges.

    **Shortcuts:** Desktop and Start Menu shortcuts are created automatically.

    **Installer GUID:** `d8117db3-ae99-5b3e-8f83-67c056c83b7c`

    **Code signing:** Azure Code Signing — CN=Sennheiser electronic SE & Co. KG, O=Sennheiser electronic SE & Co. KG, L=Wedemark, S=Lower Saxony, C=DE

    **.NET 8 dependency:** The installer checks for the presence of `%ProgramFiles%\dotnet\shared\Microsoft.AspNetCore.App\8.*`. If not found, the user is prompted to download and silently install the .NET 8 SDK from https://aka.ms/dotnet/8.0/dotnet-sdk-win-x64.exe (~200 MB; it includes the ASP.NET Core and base .NET 8 runtimes). Installation runs /install /quiet /norestart, so no reboot is required. If the download fails, .NET 8 can be installed manually from https://dotnet.microsoft.com/en-us/download/dotnet/8.0 before the next launch. Declining the prompt aborts the installation; if .NET is removed later, the app reports it at launch (see System Requirements & Prerequisites).

    **Managed deployments:** To avoid the interactive ~200 MB download at install time, pre-provision .NET 8 on the target image (e.g. via your software distribution tool) before deploying the Firmware Updater. The ASP.NET Core Runtime 8.x alone is sufficient — it satisfies both the installer check and the launch-time check, and on Windows it is a ~11 MB install (https://aka.ms/dotnet/8.0/aspnetcore-runtime-win-x64.exe). The SDK is only what the interactive prompt downloads, so that the same instructions apply on macOS.

    **Firewall rules:** During installation the user is asked whether to add Windows Firewall rules for device discovery. See Firewall rules for details.

**To install the Firmware Updater in silent mode on Windows:**

- Run `Sennheiser.FirmwareUpdater.Setup.<version>.<buildId>.exe /S`.
  - The installer automatically answers No to the firewall-rules prompt.
- Configure firewall rules separately after installation.

**To install the Firmware Updater on macOS:**

- Open `Sennheiser.FirmwareUpdater.Setup.<version>.<buildId>.dmg` and drag the application to the Applications folder.
  - No additional steps are required. The application is code-signed and Apple-notarized, so Gatekeeper does not block it. Only Apple Silicon (ARM64) Macs are supported.

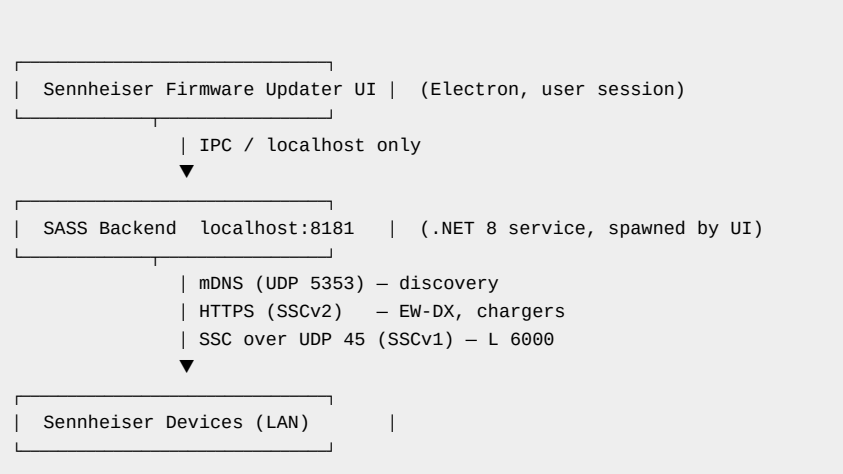
The Firmware Updater has been installed.

# 4. Network architecture

The application uses a two-process model with loopback-only communication between the Electron UI and the SASS backend.

The application uses a two-process model. All communication between the Electron UI and the SASS backend occurs exclusively over the loopback interface (127.0.0.1).

Figure 1. Network architecture




**Internal Loopback Endpoints** (no firewall exception required)

Table 4. Internal loopback endpoints

| Endpoint | Protocol | Purpose |
|---|---|---|
| http://localhost:8181/api | HTTP | REST API for device management and firmware operations |
| http://localhost:8181/api/health | HTTP GET | Backend health monitoring |
| http://localhost:8181/websocketHub | SignalR / WebSocket | Real-time device events |

These endpoints bind to localhost only and are never reachable from the network.

# 5. Firewall rules

**Configure firewall access so the SASS backend can discover Sennheiser devices and transfer firmware on the local network.**

The SASS backend must discover and communicate with Sennheiser devices on the local network. UI-to-backend traffic stays on the loopback interface and does not need a firewall exception.

### Required network access

The rules below cover device-facing traffic only.

| Purpose | Protocol / port | Devices |
|---|---|---|
| Discovery | mDNS over UDP 5353 | all |
| Control and firmware transfer | HTTPS (SSCv2) | EW-DX receivers, chargers |
| Control and firmware transfer | SSC over UDP 45 (SSCv1) | L 6000 battery charger |

The per-executable rules use protocol=any, so they already cover UDP 45 and HTTPS on a flat network. Consider UDP 45 explicitly only when traffic is filtered by port instead of by program, such as in Group Policy rule sets, VLAN or subnet ACLs, and third-party endpoint firewalls.

### What the script creates

The bundled firewall_rules.bat creates two categories of Windows Firewall rules. All rules use profile=any and interfacetype=any, so they also apply on the Public profile and on link-local interfaces.

### Per-executable allow rules

Each SASS executable receives one inbound and one outbound allow rule with protocol=any.

| Rule name | Direction | Protocol | Scope |
|---|---|---|---|
| Sennheiser Device API - Inbound Communication | Inbound | any | per executable |
| Sennheiser Device API - Outbound Communication | Outbound | any | per executable |

The rules apply to these five SASS executables in ...\resources\sass\win\:

- `Sennheiser.Application.Service.Suite.exe`
- `sennheiserservicebus\Sennheiser.ServiceBus.exe`
- `sennheiserapigateway\Sennheiser.ApplicationService.ApiGateway.Host.exe`
- `sennheiserdeviceapihost\Sennheiser.DeviceApi.Host.exe`
- `applicationservicedevicediscovery\Sennheiser.ApplicationService.DeviceDiscovery.Host.exe`

#### mDNS port rules

These rules are not bound to a program because the multicast reply may be received by any child process.

| Rule name | Direction | Protocol | Local port |
|---|---|---|---|
| Sennheiser Device API - mDNS Inbound | Inbound | UDP | 5353 |
| Sennheiser Device API - mDNS Outbound | Outbound | UDP | 5353 |

The script deletes existing rules before it adds new ones. This removes leftover block rules that would otherwise override the allow rules and block unsolicited inbound mDNS announcements after a reboot or firmware update. The deletions are limited to the five Sennheiser executables.

### Windows installation and manual setup

#### Automated configuration during installation

On a fresh install, the setup wizard prompts the user to add the rules. Yes runs firewall_rules.bat add with UAC elevation. No skips rule creation. Silent installations use No by default. The script logs to `%TEMP%\sennheiser-firewall-rules.log`.

#### Manual configuration after installation

The script remains available at `%LOCALAPPDATA%\Programs\Sennheiser.FirmwareUpdater\resources\firewall_rules.bat`.

- Run `firewall_rules.bat` add `%LOCALAPPDATA%\Programs\Sennheiser.FirmwareUpdater\resources\sass\win` from an elevated command prompt to add rules. Run `firewall_rules.bat` del to remove them. The script self-elevates through UAC if needed. The uninstaller removes the rules automatically.

#### Group Policy and managed environments

If Windows Firewall is centrally managed, local rules from firewall_rules.bat may be ignored when AllowLocalPolicyMerge or AllowLocalFirewallRules is disabled. Deploy equivalent rules through Group Policy instead.

- Allow inbound and outbound UDP 5353 for mDNS on all profiles, plus inbound and outbound UDP 45 for SSCv1 if L 6000 chargers are in use.
- Allow inbound and outbound access for the five SASS executables listed above. This is the preferred option because it covers mDNS, HTTPS, and UDP 45 in one rule set.

Use Get-FirewallDiagnostics.ps1 on a target machine to confirm whether Group Policy overrides local rules and to produce evidence for the domain administrator.

## Link-local networks and the Public profile

When a PC is connected directly to a device or to a small unmanaged switch with no DHCP server, both ends self-assign addresses in the 169.254.x.x range. Windows classifies the segment as a Public network and applies the stricter Public firewall profile.

Public-profile firewall rules or Group Policy block rules that target the Public profile can drop inbound mDNS. After a reboot or firmware update, the device may never re-announce itself and can appear offline.

This is why the bundled rules use profile=any and interfacetype=any.

### Mitigations

- Ensure the firewall rules are present through the installer, firewall_rules.bat add, or Group Policy so UDP 5353 is allowed on the Public profile.
- Use Add device by IP to reach the device directly when multicast discovery does not work.

On slow or link-local links, firmware transfer can take a long time. The application uses extended or disabled network timeouts for uploads, so no configuration is required.

## macOS

When the SASS backend starts for the first time, the macOS Application Firewall may ask whether to allow incoming connections. Select Allow. The application holds network.server and network.client entitlements in its code-signing profile, so no further configuration is needed.

## Corporate and managed network considerations

### VLAN segmentation

If workstations and Sennheiser devices are on separate VLANs, permit UDP 5353 for mDNS discovery, TCP 443 for EW-DX receivers and chargers, and UDP 45 for L 6000 battery chargers between the host running SFU and the device subnet.

An ACL that permits only 5353 and 443 often causes an L 6000 to stay discoverable but never controllable.

### Manual IP fallback

The application supports adding devices by IP address directly. This bypasses mDNS discovery. The control protocol must still be reachable.

### Loopback protection

Endpoint security software must not block `localhost:8181`. The SASS backend binds only to the loopback interface and is not reachable externally, but local traffic filtering can block communication with the UI.

### Internet access

Internet access is not required for device discovery or firmware updates. All firmware data is served by the local SASS backend. Internet access is needed only to download the latest device firmware images.

# 6. Firewall diagnostics

**Use the bundled `Get-FirewallDiagnostics.ps1` script to diagnose blocked device discovery on Windows.**

When device discovery fails on a Windows machine, run the bundled Get-FirewallDiagnostics.ps1 script before you change any firewall settings. The script is read-only. It reads firewall state, Group Policy, and network configuration only. It does not create, change, or delete firewall rules or policy.

## Script location and requirements

After installation, the script is available at `%LOCALAPPDATA%\Programs\Sennheiser.FirmwareUpdater\resources\Get-FirewallDiagnostics.ps1`.

Use Windows PowerShell 5.1 or later. Run the script from an elevated Administrator PowerShell or command prompt whenever possible. Elevated runs return complete computer-scoped Group Policy data.

## Run the script

Use the following command:

```
powershell -ExecutionPolicy Bypass -File .\Get-FirewallDiagnostics.ps1
```

Optional: use `-OutputDir <path>` to set the report folder. If you omit it, the script writes to the current user's Desktop ( `%USERPROFILE%\Desktop` ).

## Output files

The script writes two timestamped files to the output directory. Collect both files from the user's machine for analysis.

| File | Contents |
|---|---|
| `Sennheiser-Firewall-Diagnostics-<yyyyMMdd-HHmmss>.txt` | The full human-readable diagnostic report |
| `gpo-report-<yyyyMMdd-HHmmss>.html` | The gpresult /h Group Policy report in HTML format |

## What the script checks

The report is organized into sections that each target a likely cause of dropped mDNS discovery.

- **System:**
  host name, user, OS build, and whether the script runs elevated.
- **Firewall profiles (ActiveStore):**

DefaultInboundAction for each profile and whether AllowLocalFirewallRules is False, which means Group Policy ignores locally created rules.

- **Group Policy firewall registry:**
  `HKLM\SOFTWARE\Policies\Microsoft\WindowsFirewall`
  and AllowLocalPolicyMerge = 0, which means local rules are ignored.
- **Network connection profiles and IPv4 addresses:**
  each interface's NetworkCategory and connectivity, including link-local 169.254.* adapters. Windows classifies such networks as Public.
- **Inbound BLOCK rules:**
  enabled inbound block rules that match the Sennheiser executables or UDP 5353, and whether they come from Group Policy or local settings.
- **All rules that touch UDP 5353:**
  every rule that references the discovery port.
- **Installer rules present:**
  checks whether the four named Sennheiser rules exist.
- **GPO-delivered rules (RSOP) and gpresult:**
  the effective policy on the machine.
- **Summary / verdict:**
  a plain-language conclusion with flags and a recommended action.

## Scope

The script diagnoses discovery only. It inspects rules that reference the Sennheiser executables and UDP 5353, but it does not examine UDP 45. If a device is discovered and listed but never becomes controllable, check UDP 45 separately for L 6000 chargers, and TCP 443 for EW-DX receivers.

## Interpreting the verdict

Use the verdict flag to determine the next action.

| Verdict flag | Meaning | Recommended action |
|---|---|---|
| GpoManaged / LocalIgnored | Group Policy manages the firewall and local rules are ignored. | The domain administrator must allow inbound and outbound UDP 5353 and/or the five SASS executables by Group Policy, or re-enable local rule merge. Running `firewall_rules.bat add` locally does not help. |
| GpoBlock | A Group Policy block rule matches UDP 5353 or the Sennheiser executables. | Only a domain administrator can remove or override the blocking GPO rule. |
| LocalBlock | A local block rule, typically from a declined firewall prompt, drops the traffic. | Run firewall_rules.bat add from an elevated prompt. It deletes the stale block rules and creates the correct allow rules. |

| Verdict flag | Meaning | Recommended action |
|---|---|---|
| (none) | No firewall obstruction is detected. | Look elsewhere: link-local or Public profile, VLAN or multicast routing, or the device itself. |

# 7. Application Data & Log Files

This topic lists the application data and log file locations for Windows and macOS.

## Windows

The application stores user data and log files in the following locations on Windows.

| Data | Path |
|---|---|
| Application user data | `%APPDATA%\Sennheiser.FirmwareUpdater\` |
| Settings store | `%APPDATA%\Sennheiser.FirmwareUpdater\config.json` |
| Backend PID tracking file | `%APPDATA%\Sennheiser.FirmwareUpdater\dapi.pid` |
| Backend log configuration | `%APPDATA%\Sennheiser.FirmwareUpdater\nlog.json` |
| Application log file | `C:\Users\Public\Sennheiser\Sennheiser.FirmwareUpdater\log\console_main.log` |

## macOS

The application stores user data and log files in the following locations on macOS.

| Data | Path |
|---|---|
| Application user data | `~/Library/Application Support/Sennheiser.FirmwareUpdater/` |
| Settings store | `~/Library/Application Support/Sennheiser.FirmwareUpdater/config.json` |
| Backend PID tracking file | `~/Library/Application Support/Sennheiser.FirmwareUpdater/dapi.pid` |
| Backend log configuration | `~/Library/Application Support/Sennheiser.FirmwareUpdater/nlog.json` |
| Application log file | `/Users/Shared/Sennheiser/Sennheiser.FirmwareUpdater/log/console_main.log` |

## Log file details

The application log `console_main.log` is written to a shared, machine-wide location so that log files are accessible regardless of which user account runs the application.

This setup simplifies log collection in managed environments.

**Rotation:**

- The log file is capped at 5 MB. The application keeps up to five rotated backup files, `console_main.log.1` to `console_main.log.5`, and compresses them with gzip.

**Fallback:**

- If the shared directory cannot be created, for example because of permissions, the application uses a temporary fallback path:
    - Windows: `%TEMP%\Sennheiser.FirmwareUpdater\log\console_main.log`
    - macOS: `$TMPDIR/Sennheiser.FirmwareUpdater/log/console_main.log`

## Log levels

Configure the backend logging level in the application settings.

Available levels in ascending verbosity are `error`, `warn`, `info`, and `debug`.

# 8. Security notes

These security notes summarize signing, execution, network, isolation, and process-handling behavior.

## Overview

| Topic | Details |
|---|---|
| Code signing (Windows) | Signed via Azure Code Signing — CN=Sennheiser electronic SE & Co. KG, L=Wedemark, S=Lower Saxony, C=DE |
| Code signing (macOS) | Signed with Apple identity Sennheiser electronic (XKNNU9WTL7), hardened runtime enabled, Apple-notarized |
| Execution level | asInvoker — The application always runs with the privileges of the logged-in user. No automatic elevation occurs. |
| Network exposure | The SASS backend binds exclusively to `127.0.0.1:8181`. It is not accessible from any network interface. |
| Renderer isolation | The Electron renderer process has no direct access to Node.js or operating system APIs. All host access is mediated through a typed IPC bridge in the preload script. |
| Orphan process handling | On startup, SFU checks for a leftover SASS process from a previous session via a `PID` file and terminates it before starting a new one. |

# 9. Uninstalling Firmware Updater

Uninstall the application on Windows or macOS and remove leftover user data manually if needed.

Use the procedure for your operating system.

**To uninstall the Firmware Updater under Windows:**

- Open **Settings** > **Apps** > **Installed Apps**, or run the uninstaller directly from the installation directory.
  - ✓ The uninstaller removes all Windows Firewall rules that were added during installation.
- Delete `%APPDATA%\Sennheiser.FirmwareUpdater\` manually if you need a clean removal.
  - ✓ The uninstaller does not remove the user data stored in this directory.

**To uninstall the Firmware Updater under macOS:**

- Drag `Sennheiser.FirmwareUpdater` from the Applications folder to the Trash.
  - ✓ The application is removed from the system.
- Delete `~/Library/Application Support/Sennheiser.FirmwareUpdater/` manually if you need a clean removal.
  - ✓ The application support data is not removed by the uninstaller.

> **i** Delete the remaining user data manually when you need a complete uninstall.

✓ The Firmware Updater has been uninstalled.

Sennheiser electronic SE & Co. KG | Am Labor 1 | 30900 Wedemark | Germany