IT SECURITY REFERENCE

DWSIM

Reference document describing how DWSIM, the DWSIM Patreon Extensions, and the dwsim-assistant interact with the network and the local filesystem — for use by corporate IT teams validating these components for installation on managed workstations.

Date	2026-04-27
Author	Daniel Wagner Oliveira de Medeiros — author and developer of DWSIM and of this report
Components	DWSIM (process simulator) · DWSIM Patreon Extensions (add-ons offered to active monthly Patreon supporters) · dwsim-assistant (AI helper)
Audience	Corporate IT and Security teams
Scope	Install footprint, network endpoints, inbound listeners, local secrets, code-execution surface, third-party components, data leaving the workstation, recommended firewall policy

---

A "deny by default" outbound rule plus the allowlist in §9 is sufficient to confine DWSIM to the use-cases your users need.

Contents

Purpose: describe how DWSIM, the DWSIM Patreon Extensions, and the dwsim-assistant interact with the network and the local filesystem so corporate IT can validate that they are safe to install and run on managed workstations. **Audience:** corporate IT / Security teams. **Author:** Daniel Wagner Oliveira de Medeiros — author and developer of DWSIM and of this report. **Date:** 2026-04-27.

> **Bottom line:**
>
> - **DWSIM core** is a desktop process simulator. Outbound network access is opt-in; the install footprint is limited to the user's profile and a single registry hive under `HKCU`. No background services, no auto-update channel.
> - **DWSIM Patreon Extensions** are add-on components offered to supporters who maintain an active monthly recurring donation on Patreon. They include extra unit operations and an optional convergence assistant; the convergence assistant uploads anonymized simulation parameters and can be turned off in user settings.
> - **dwsim-assistant** is an optional AI helper packaged as a single `.exe`. It binds only to localhost, is protected by a per-launch token, fetches its cloud credentials in memory at runtime (nothing on disk), keeps telemetry off by default, and refuses to run AI-suggested IronPython unless the user explicitly enables it.

---

1. Executive summary

Component	Outbound network	Inbound listeners	Local secrets	Code execution	Recommendation
DWSIM core	🟢 Opt-in (Simulate365 cloud, license check, manual chemistry lookups)	🟢 None by default	🟢 OAuth tokens stored under `HKCU\SOFTWARE\DWSIM` only after the user logs in	🟡 Optional in-process IronPython for user-attached flowsheet scripts	**Allow**
DWSIM Patreon Extensions	🟡 Anonymized convergence-assistant upload when enabled	🟢 None on client	🟢 No client-side secrets	🟢 None beyond core	**Allow**
dwsim-assistant	🟡 LLM provider, optional Supabase logging, optional OPC/PI read	🟢 `127.0.0.1` only, all endpoints token-gated	🟢 No on-disk secrets	🟢 IronPython execution off by default	**Allow**

Legend: 🟢 acceptable · 🟡 review and apply the recommended firewall policy · 🔴 blocking risk.

---

2. Install footprint

2.1 DWSIM core

- Windows desktop application (.NET Framework 4.6.2 / 4.8). Cross-platform Eto build for macOS/Linux is also available.
- Default install path on Windows: `%ProgramFiles%\DWSIM`. The installer requires admin only to write into Program Files; everything DWSIM does at runtime runs as a standard user.
- User data: `%AppData%\DWSIM` (settings, recent files, plugins).
- Registry: a single hive `HKCU\SOFTWARE\DWSIM`. Used to store Simulate365 OAuth tokens (`AccessTokenV2`, `RefreshTokenV2`) when the user logs in to the Simulate365 cloud feature; cleared on logout.
- No Windows service, no scheduled task, no auto-update mechanism, no driver, no kernel component.

2.2 DWSIM Patreon Extensions

- Add-on assemblies dropped into the existing DWSIM install folder. No separate installer, no separate registry footprint.
- Distributed only to supporters with an active monthly recurring Patreon donation. They are not part of the public DWSIM build.
- Includes the AI Convergence Assistant client and bundles the TensorFlow.NET runtime for local inference.

2.3 dwsim-assistant

- Single-file PyInstaller `.exe`. No separate installation step — it sits next to the DWSIM binary and is launched on demand by DWSIM when the user opens the AI Assistant panel.
- Per-user configuration file: `%AppData%\DWSIM\assistant.env` (optional; not created by the installer).
- Local data: a `logs/` folder and conversation `*.jsonl` files alongside the `.exe`. Optional Supabase mirror is off by default.
- No service, no auto-update.

---

3. Network behavior

All outbound calls are listed below. Anything not listed here does not happen.

3.1 DWSIM core

Endpoint	Purpose	When it fires	Data sent
`dashboard-service.simulate365.com`, `excel-runner-service.simulate365.com`, `sensitivity-study-service.simulate365.com`, `take-home-exams-service.simulate365.com`	Simulate365 cloud sync, Excel runner, exam delivery	Only after the user signs in to Simulate365 from inside DWSIM	OAuth tokens, flowsheet files the user explicitly uploads, basic profile
`login.microsoftonline.com`	Azure AD OAuth flow used by Simulate365	Same as above	Standard OAuth handshake
`dwsimlicensingcheck.azurewebsites.net`, `dsw-license-manager.azurewebsites.net`	License validation for Pro features	At startup if Pro features are licensed	Machine identifier, license key
`pubchem.ncbi.nlm.nih.gov`	Compound metadata lookup	Manual: when the user adds a compound through the Compound Creator	Compound name / CAS / SMILES
`api.crossref.org`	Citation metadata enrichment	Manual: when the user requests citations for a report	DOI

If Simulate365 is not used and Pro features are not licensed, DWSIM core does not contact any of these endpoints.

3.2 DWSIM Patreon Extensions

Endpoint	Purpose	When it fires	Data sent
`dwsim-convergence-assistant-akbsbbd7h3f5enby.brazilsouth-01.azurewebsites.net`	Crowd-sourced convergence training data	Every 60 s while a simulation is running, **only when** `UploadToServer = true` in user settings	Compressed JSON: component names, molar flows, flash/equilibrium results, and a per-machine GUID for deduplication. No flowsheet labels, no user identity, no document metadata

Disabling the upload is one toggle in the user-settings file (`UploadToServer = false`). IT can also block the domain at the firewall as defense-in-depth.

3.3 dwsim-assistant

Endpoint	Purpose	When it fires	Data sent
`api.openai.com`, `api.anthropic.com`, `*.cognitiveservices.azure.com`, `openrouter.ai`, custom endpoint	LLM completions for the chat panel	Every chat turn, only when the user is on the cloud backend	System prompt, conversation history, and the active flowsheet's structural XML (no images)
`dsw-license-manager.azurewebsites.net`	Per-launch credential fetch	At startup of the assistant	License key only
`*.supabase.co`	Conversation logging mirror	Only when the user has explicitly opted in to data sharing in Settings	Message content, tool arguments, model name, session UUID
MCP servers from the user's `mcp_servers.json`	External tool calls the user configured	User-driven	Whatever the user-configured tool requires
OPC UA server (port 4840)	Read-only browse of process tags	Only if the user configures an OPC server	Tag browse / read requests
OSIsoft / AVEVA PI Data Archive	Read-only access to historian	Only if the user configures PI	Tag search / read

The OPC UA and PI clients in this build are read-only — they browse and read tags but never write back to control systems or historians.

---

4. Inbound listeners

Component	Bind address	Port	Authentication
dwsim-assistant Python server	`127.0.0.1` (localhost only)	5834	Per-launch GUID token. Every HTTP, WebSocket and EventSource call must carry the matching `X-DWSIM-Token` header (or `?token=` query parameter for streams). Requests without it are rejected with HTTP 401
DWSIM bridge inside the DWSIM process	`localhost`	5002	Same per-launch GUID token. Used by the assistant to read the active flowsheet and to issue read/write commands to the simulation
DWSIM Automation TCP / Azure Server	varies	varies	Standalone tools — only started if the user explicitly runs them

The token is generated fresh every time DWSIM starts the assistant; it never leaves the local machine.

---

5. Local secrets and credentials

Where	What	Sensitivity	Notes
`HKCU\SOFTWARE\DWSIM` (registry)	Simulate365 OAuth refresh / access tokens	🟡	Only present after the user signs in to Simulate365. Cleared on logout
Azure AD tenant and client IDs embedded in the binary	OAuth public-client identifiers used for Simulate365 login	🟢	These are **not** secrets — they are by-design public identifiers for OAuth public clients
dwsim-assistant Azure / Supabase credentials	LLM provider + optional logging	🟢	Not stored on disk in any distributed build. Fetched at runtime from a license-validated, RSA-signed endpoint and held only in process memory

There are no embedded passwords, API keys, database connection strings, or hardcoded credentials in the distributed binaries.

---

6. Code-execution surface

What can run code inside the DWSIM process tree, and how IT can control it:

Surface	When it runs	Default	How to disable for a corporate deployment
Built-in unit operations and property packages	Whenever a flowsheet is solved	Always available	Cannot be disabled — these are the simulator's core function
IronPython runtime (in-process, used by user-supplied scripts attached to flowsheet objects via the Script Manager)	Only when the user opens a flowsheet that contains attached scripts, or attaches one manually	Available, but only fires on user-attached scripts	Remove `IronPython.dll`, `IronPython.Modules.dll`, `IronPython.SQLite.dll`, `IronPython.Wpf.dll` and `Microsoft.Scripting.dll` from the install folder
AI Assistant `generic_script` tool	Only when the assistant is configured to use it	**Disabled by default**. The user must enable it from the assistant Settings panel for the current session	Do not deploy `dwsim-assistant.exe`, or distribute the assistant with scripting hard-locked off (the binary will then refuse the toggle)
Plugins folder	Loaded at startup	Empty by default	Restrict NTFS write permission on `%AppData%\DWSIM\Plugins`

External Python (`python.exe` via Python.NET) and Octave subprocess execution are **not** part of this distribution. The legacy `DWSIM.Libraries.PythonLink.dll` and `DWSIM.Libraries.OctaveSharp.dll` have been removed from the build.

Nothing in the distribution downloads code from a remote location, performs auto-update, or loads DLLs from outside the install folder.

---

7. Third-party components

7.1 DWSIM core

UI/runtime: Eto.Forms, GTK# (Linux only), OpenTK, SkiaSharp, WebView2 (Edge/Chromium component), Newtonsoft.Json, SharpZipLib, NetOfficeFw (Excel interop). Cloud: Microsoft.Graph and

Microsoft.Identity.Client (MSAL) — used only by the Simulate365 feature.

7.2 DWSIM Patreon Extensions

Adds TensorFlow.NET and the matching native TensorFlow runtime for local inference in the convergence assistant. LiteDB ships only with the server-side component (not part of any client install).

7.3 dwsim-assistant

A pinned set of Python packages (see `requirements.txt` shipped with the source). Notable runtime dependencies: FastAPI + Uvicorn (web server), httpx (HTTP client), cryptography (RSA signature verification), pythonnet (Windows-only .NET bridge), MCP client, asyncua (OPC UA), PIconnect (OSIsoft PI). All dependencies are pinned to exact versions.

7.4 Code signing

The DWSIM `.exe`, the Patreon-extension DLLs and the `dwsim-assistant.exe` are not Authenticode-signed by the publisher. Corporate environments running AppLocker or WDAC therefore need either a hash-based exemption for these binaries or a corporate code-signing wrapper applied during deployment.

---

8. Data leaving the workstation

Channel	What	When
Simulate365	Whole flowsheet files	Only when the user explicitly uploads through Simulate365
Convergence-assistant upload (Patreon extensions)	Anonymized simulation parameters	Every 60 s while a simulation runs, only when `UploadToServer = true`
LLM provider (assistant)	The active flowsheet's structural XML and the chat conversation	Each chat turn while the cloud backend is selected
Supabase mirror (assistant)	Conversation history and tool arguments	Only when the user has opted in via the Settings UI
OPC UA / PI (assistant)	Tag-read requests to the user-configured server	User-driven; reads only — no writes

DWSIM does not collect or transmit biometric data, user keystrokes, screenshots of unrelated applications, file-system contents outside its own folders, or PII beyond what the user types into Simulate365 login or the chat panel.

---

9. Recommended corporate firewall and policy

A "deny by default" outbound rule for the DWSIM and dwsim-assistant processes, combined with the allowlist below, is sufficient to confine the install to the use-cases your users actually need.

```
# Always required if the user has a paid license
dwsimlicensingcheck.azurewebsites.net
dsw-license-manager.azurewebsites.net

# Required only if Simulate365 is used
*.simulate365.com
login.microsoftonline.com
graph.microsoft.com

# Optional — manual chemistry data lookups
pubchem.ncbi.nlm.nih.gov
api.crossref.org

# Required only if the AI Assistant is used (pick one provider)
api.openai.com                  # OR
api.anthropic.com               # OR
<resource>.cognitiveservices.azure.com

# Recommended DENY (no business reason on a corporate workstation)
dwsim-convergence-assistant-akbsbbd7h3f5enby.brazilsouth-01.azurewebsites.net
*.supabase.co
```

Additional policies an administrator may apply:

- Set `UploadToServer = false` in the DWSIM Patreon Extensions user-settings file under `%AppData%\DWSIM` and protect the file with read-only ACLs or a Group Policy preference.
- Monitor `HKCU\SOFTWARE\DWSIM` for OAuth token presence on workstations not approved for Simulate365.
- For environments where AI assistance is not desired, do not deploy `dwsim-assistant.exe` (DWSIM will function normally without it) or remove `DWSIM.Extensions.AI.Assistant.dll`.
- For environments where in-app scripting (IronPython attached to flowsheet objects) is not desired, remove `IronPython.dll`, `IronPython.Modules.dll`, `IronPython.SQLite.dll`, `IronPython.Wpf.dll` and `Microsoft.Scripting.dll` from the install folder. External Python and Octave subprocess execution are already not part of the distribution.
- Apply a corporate Authenticode signing wrapper (or hash-based AppLocker exemption) to the DWSIM and dwsim-assistant binaries.