SolutionCenter API Reference

REST API for managing a FrameworX installation remotely: list, run and stop solutions, file transfers, license activation, machine settings and server information. JWT-authenticated and off by default. An administrator enables it on FrameworX 10.1.6 and later. It pairs with the per-solution Runtime API on TServer ports 3101+.

The SolutionCenter API is off by default. On FrameworX 10.1.6 and later, an administrator enables it in `SolutionCenterApi-Settings.json` in the installation folder. Callers on the network reach it only over TLS. See Enabling the API below. The API is not available on FrameworX 10.1.5.

Overview
Enabling the API
Authentication
Scopes
Endpoints
Errors
Brand-neutrality and OEM hooks
OpenAPI 3.1 document
What is not in v1 (reserved for v2)
Deployment scenarios
Quick-start

Overview

The SolutionCenter API is the machine-wide management plane of FrameworX, exposed over HTTPS by TWebServices on port **10108**. Fleet operators, OEM administration tools, central monitoring solutions, and IT/DevOps automation use it to manage the solutions installed on a machine without touching the Designer or per-solution runtime ports.

Each FrameworX installation hosts the API once, regardless of how many solutions are installed or running on it. v1 is single-installation: each machine's API manages that machine's solutions. Cross-machine fleet aggregation is a v2 capability — orchestration today happens in the caller's tooling (Ansible, custom dashboards, PowerShell scripts) by fanning out to each machine's API.

The SolutionCenter API is one of two HTTP APIs FrameworX exposes:

API	Plane	Port	Server	Authentication
Runtime API	Data — tags, alarms, historian, datasets	3101 / 3201 / 3301 / 3401 (per profile)	TServer (per solution)	Bearer-GUID via `Security.LogOnAsync`
SolutionCenter API (this page)	Management — solutions, files, license, machine settings	10108	TWebServices (one per machine)	JWT (RFC 9068, `at+jwt`)

The two APIs are independent — different processes, different ports, different authentication. Integrating with one does not require the other.

Enabling the API

The API ships off. While it is off, every `/api/v1/...` call except `/api/v1/openapi` returns HTTP 503 with a `service-unavailable` Problem Details body. The OpenAPI document stays reachable for tooling.

Settings file

An administrator creates `SolutionCenterApi-Settings.json` in the installation folder, beside the TWebServices program the service runs:

- Windows service (.NET Framework 4.8 host): the installation folder, beside `TWebServices.exe`.
- .NET 10 host: the `net10.0` folder of the installation.
- Linux and macOS: the folder that holds TWebServices.

Each host reads only the file beside it. A file in `net10.0` does not enable the Windows service host, and the reverse. The product never creates or ships this file, so a product update keeps it. Editing the installation folder needs an elevated account. Restart the TWebServices service after every change to the file.

The members sit at the top level of the file, with no wrapper object:

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "Enabled": true,
  "LoopbackOnly": false,
  "AllowedPaths": [ "Solutions/", "Backup/" ],
  "OidcRedirectOrigins": [ "https://dashboard.example.com" ]
}
```

Member	Purpose
`Enabled`	Turns the API on. Default `false`.
`LoopbackOnly`	Accepts only connections made over a loopback address. Default `false`. See Transport below.
`AllowedPaths`	File-operation path allowlist. Default `Solutions/` and `Backup/`. See Files.
`MachineSettingsAllowlist`	MachineSettings files the machine-settings routes serve. Default `TWebServices.json`, `TWebServicesRun Programs.json`, `Logging.json`.
`MaxUploadBytes`	Byte cap per upload. Default 2 GiB.
`OidcRedirectOrigins`	Extra https origins allowed as single sign-on return addresses. Default empty. Entries that use http are ignored, and the start-up log names them.
`MaxSessionLifetimeSeconds`	Longest session from sign-in. Refresh is refused after it. Default 86400 (24 hours).
`AccessTokenLifetimeSeconds`	Lifetime of an issued token. Never longer than `MaxSessionLifetimeSeconds`.
`RefreshWindowSeconds`	How long after expiry a token is still refreshable. Default 24 hours.
`SigningKeys`	Keys that sign the tokens the API issues. Without them the API generates its own key. See Protecting the settings files.
`Issuer`, `Audience`, `OpenApiTitle`, `ErrorTypeBase`, `PublicBaseUrls`	OEM branding. See Brand-neutrality and OEM hooks.
`AcceptRiskOnOldBuild`	No effect on release builds. See Removed override below.

- A list in the file replaces the default list, so `AllowedPaths` and `MachineSettingsAllowlist` are narrowed by the file. An empty or missing list keeps the defaults.
- When the file cannot be read, is not valid JSON, holds a value of the wrong type, or wraps the members in a `SolutionCenterApi` or `appSettings` object, the API stays off. The log names the file, the error type and the member path, never a value.
- API settings left in `TWebServices.json` or in the shared MachineSettings folder are ignored. The start-up log names those files, never their content.
- At start-up, TWebServices logs one line, `SolutionCenter API: ENABLED` or `SolutionCenter API: DISABLED` with the reason, and names the settings file it read.

Transport: TLS or local-only

With `Enabled` set to `true`, the API also needs one of these:

- **A TLS certificate on the TWebServices listener.** The API checks the running listener, not the settings. A certificate thumbprint that matches no certificate in the machine store leaves the listener on plain HTTP, and the API stays off.
- **`"LoopbackOnly": true`.** The API accepts only connections whose peer is a loopback address. Every other caller gets 503 before authentication.

With neither, every call returns 503 and the start-up log reports that the listener has no TLS certificate.

In local-only mode:

- The listener accepts IPv4 only. Connect to `127.0.0.1`. A connection to `[::1]` is refused.
- A call from the same host through its host name or LAN address gets 503, because the peer is not a loopback address.
- A reverse proxy on the same host makes every caller it forwards look local.
- `/api/v1/openapi` stays reachable from the network.
- A remote client pointed at a local-only API sends its token before it receives the 503.

Protecting the settings files

On Windows, the installation folder gives write access to administrators and read access to all local users. `SolutionCenterApi-Settings.json`, `SolutionCenterApi-Keys.json` and `SolutionCenterApi-OidcProviders.json` hold signing keys and client secrets. Restrict read access on these three files to Administrators and SYSTEM. Prefer no inline `SigningKeys`: without them, the API signs with a key held in memory only.

Availability by version

- **FrameworX 10.1.6 and later:** `Enabled` plus a TLS certificate or `LoopbackOnly` turns the API on.
- **FrameworX 10.1.5:** the API cannot be enabled on any build. Installations that carry `SolutionCenterApi` entries in `MachineSettings/TWebServices.json` should remove them and restart the TWebServices service.

On a build that predates the required security fixes, `Enabled` has no effect. The start-up log reports the reason, the Windows service writes a CRITICAL event to the Windows event log, and every call returns 503.

Removed override: AcceptRiskOnOldBuild

Earlier revisions of this page described a flag, `AcceptRiskOnOldBuild`, that forced the API on for builds without the required fixes. It has no effect on any release build. On FrameworX 10.1.5 Update 5c and earlier it still takes effect. **Do not use it on any build.** Remove it where it is set and restart the TWebServices service.

Authentication

All API calls authenticate with a JWT (RFC 9068, header `typ: at+jwt`) presented as `Authorization: Bearer <jwt>`. The API reads the token from this header only, never from a URL. Two issuance paths are supported.

OIDC primary path (humans)

Fleet operators authenticate against a corporate OIDC identity provider, and the SolutionCenter API mints a FrameworX-issued JWT on top of the IdP's identity claims:

1. The operator's browser or client calls `GET /api/v1/auth/oidc/login?provider=<name>&redirect=<return address>`.
2. The API checks the return address (see Return addresses below). It looks up the named provider in `SolutionCenterApi-OidcProviders.json` in the installation folder, builds the IdP authorize URL with PKCE S256 and state, and returns a 302 to the IdP.
3. The operator authenticates at the IdP. The IdP redirects the browser back to `POST /api/v1/auth/oidc/callback` with the authorization code.
4. The API exchanges the code with the IdP, validates the resulting token and mints its own JWT with the operator's identity and scope set. It returns a 200 HTML page that posts the JWT to the return address as the form field `token` (`application/x-www-form-urlencoded`). Without a `redirect`, it returns 200 with a JSON body containing `accessToken`. Both responses carry `Cache-Control: no-store`. The token never appears in a URL.
5. Subsequent API calls send `Authorization: Bearer <jwt>`.

OIDC providers are configured in `SolutionCenterApi-OidcProviders.json` in the installation folder. Client secrets live in that file as plaintext, protected by file-system ACL (see Protecting the settings files). There is no separate secret store in v1. Client-credentials grant and asymmetric signing are deferred to v2.

Return addresses

The `redirect` value must be one of these:

- A path on this server, starting with a single `/`.
- An http or https address on a loopback host, such as `http://localhost:<port>/`, on any port. Native clients such as the PowerShell module use this.
- An https address whose origin (scheme, host and port) is listed in `OidcRedirectOrigins`.

Any other address, an address with control characters, or an over-long address returns 400 `bad-request` "redirect is not an allowed return address." before any IdP round trip. A browser dashboard on another origin needs a server-side receiver for the posted form.

- A sign-in must finish within 10 minutes of its start. A later callback returns 400 "Unknown or expired state".
- At most 1000 sign-ins wait for completion at one time. Beyond that, a new sign-in gets 503 with `Retry-After: 60`.

Service-account JWT (machine integrations)

For service-to-service callers (central monitoring solutions, OEM admin tools, scripted automation) there is no login round trip. The OEM administrator registers a signing key in `SolutionCenterApi-Keys.json` in the installation folder of the target machine and issues a long-lived JWT signed with that key. The caller presents the pre-issued JWT in every request:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6ImF0K2p3dCIsImtpZCI6Im9...
```

This mirrors the existing FrameworX AccessKey deployment shape that has been in production for embedded and IPC scenarios for years — the credential is a file on the box, rotated by replacing the file. JWT lifetime is OEM-controlled; days or weeks is typical. Service-account tokens cannot be refreshed: rotate them instead.

Every signing key, in `SolutionCenterApi-Keys.json` or in `SigningKeys`, must be standard base64 of at least 32 bytes and have a non-empty `kid`. Base64url and unpadded JWK `k` values are rejected. A rejected key is dropped, and the log names its `kid`, never the key.

JWT shape

Tokens are signed HS256 (symmetric, kid-scoped). The `kid` header lets OEMs rotate signing keys without an algorithm change. `iss` and `aud` are OEM-configurable (see Brand-neutrality and OEM hooks below). Tokens the API issues also carry `auth_time`, the time of the original sign-in.

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "header": {
    "alg": "HS256",
    "typ": "at+jwt",
    "kid": "fleet-2026-q2"
  },
  "claims": {
    "iss":       "https://api.example.com",
    "aud":       "solutioncenter",
    "sub":       "alice@example.com",
    "iat":       1714324800,
    "exp":       1714328400,
    "auth_time": 1714324800,
    "scope":     "installation:read installation:control",
    "groups":    ["FleetOps"]
  }
}
```

Refresh and session lifetime

`POST /api/v1/auth/refresh` with an expiring or recently expired JWT returns a new JWT when all of these hold:

- The token is inside the refresh window (default 24 hours after expiry).
- The session started less than `MaxSessionLifetimeSeconds` ago (default 24 hours). The API reads the start from `auth_time`, or from `iat` when `auth_time` is absent. The last refreshed token expires at that limit.
- The token is not a service-account token.

Past the session limit, refresh returns 401 `session-expired` and the user signs in again. A service-account token returns 401 `authentication-required`. The refresh response carries `Cache-Control: no-store`.

Scopes

Authorization is JWT scope membership. Each endpoint declares the scope it requires; the validator middleware checks it before dispatching to the handler. Scopes do **not** subsume each other — holding `installation:admin` does not implicitly grant `installation:control`; callers that need both must hold both literally.

Scope	Grants
`installation:read`	Read-only queries: list solutions, solution info, server info, license info, installed protocols, installed product versions.
`installation:control`	Start and stop solutions; read and set the auto-start flag.
`installation:files`	File operations within the configured path allowlist: chunked upload/download, single-shot upload, existence checks, allowlist inspection.
`installation:license`	License operations: site code retrieval, license-key set, activation/deactivation by code, license info.
`installation:machine`	Read named MachineSettings files within the configured filename allowlist; list the allowlist. Reading `TWebServices.json` or any file that is not JSON also needs `installation:admin`.
`installation:admin`	Privileged operations: write MachineSettings files, read `TWebServices.json` and settings files that are not JSON, manage the file search paths, list active runtime connections, drop a connection.

Insufficient scope returns HTTP 404 (not 403). This is intentional — it prevents the API from leaking the existence of resources the caller is not authorized to see (BOLA defence).

Endpoints

All endpoints live under `/api/v1/`. Times are ISO 8601 UTC with millisecond precision and a `Z` suffix on output; inputs must carry an explicit `Z` or numeric offset (local-naive timestamps are rejected). List responses use a truncation envelope; cursor pagination is deferred to v2.

Solution lifecycle

Method	URL	Scope	Purpose
GET	`/api/v1/solutions/`	`installation:read`	List solutions installed on this machine, one row per (solution, profile) pair.
GET	`/api/v1/solutions/{name}/info`	`installation:read`	Full solution metadata snapshot including the running-state predicate.
GET	`/api/v1/solutions/{name}/isrunning`	`installation:read`	Cheap boolean predicate: is this solution running, yes or no.
POST	`/api/v1/solutions/{name}/run`	`installation:control`	Start TServer for this solution under the requested profile.
POST	`/api/v1/solutions/{name}/stop`	`installation:control`	Stop TServer for this solution.
GET	`/api/v1/solutions/{name}/autostart`	`installation:read`	Read the auto-start flag for this solution.

PUT	`/api/v1/solutions/{name}/autostart`	`installation:control`	Set the auto-start flag for this solution.
GET	`/api/v1/solutions/{name}/routing`	`installation:read`	Multi-version routing resolution — which installed product version this solution will start under.

Example — `GET /api/v1/solutions/`

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "solutions": [
    { "name": "PlantSCADA", "buildVersion": "build-127", "profile": "Production",  "status": "running", "port":
3101 },
    { "name": "PlantSCADA", "buildVersion": "build-127", "profile": "Development", "status": "stopped", "port":
null }
  ]
}
```

Example — `GET /api/v1/solutions/PlantSCADA/info`

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "name": "PlantSCADA",
  "description": "Plant 1 SCADA",
  "productVersion": "fx-2020.1.5.2000",
  "productFamily": "Enterprise",
  "productModel": "Unlimited",
  "targetFramework": ".net 10.0",
  "buildVersion": "build-127",
  "dateModified": "2026-04-28T11:14:02.115Z",
  "dateLastOpen": "2026-04-28T08:01:30.044Z",
  "path": "C:\\Solutions\\PlantSCADA.dbsln",
  "autoStart": false,
  "isRunning": true,
  "runningProfiles": ["Production"]
}
```

Example — `GET /api/v1/solutions/PlantSCADA/isrunning`

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{ "name": "PlantSCADA", "isRunning": true }
```

Example — `POST /api/v1/solutions/PlantSCADA/run`

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
Request body:
{ "profile": "Production" }

Response (HTTP 202):
{ "status": "starting", "statusUrl": "/api/v1/solutions/PlantSCADA/isrunning" }
```

Profile values map: `Production` = 0, `Development` = 1, `Validation` = 2, `Custom` = 3. Numeric values are accepted for forward compatibility.

Files

File operations work under the documents root: the parent of the Solutions folder of the account TWebServices runs as. Every path is resolved under that root and must sit inside one entry of `AllowedPaths` (default `Solutions/` and `Backup/`). An entry names a folder, so `Solutions/` admits `Solutions/a.dbsln` but not `SolutionsX/a.dbsln`. These are rejected with HTTP 400 `path-not-allowed`:

- Traversal (`..`) and paths outside the allowlist.
- Drive letters and any other `:`, control characters, and characters that are invalid in a path.
- A path through a junction, symbolic link, mount point or any other reparse point at or below the documents root. The check runs at start and again before every chunk and finish. It also refuses OneDrive placeholders and deduplicated or compressed files. A documents root or Solutions folder relocated with a junction loses every file route.

`Backup/` means the `Backup` folder under the documents root. Uploads there also need the absolute path of that folder added with `POST /api/v1/files/allowed-paths`.

Method	URL	Scope	Purpose
GET	`/api/v1/files/allowed-paths`	`installation:files`	Returns the active path allowlist.
POST	`/api/v1/files/allowed-paths`	`installation:admin`	Add a search path to the persistent allowlist. Body: `{ "path": "..." }`. Idempotent — re-adding an existing path is a no-op. Returns 200 with `{ "path": "...", "status": "added" }`.

DELETE	`/api/v1/files/allowed-paths`	`installation:admin`	Remove a search path from the persistent allowlist. Body: `{ "path": "..." }` (path is in the body, not the URL, to mirror the POST shape and avoid URL-encoding hell with Windows paths). Idempotent — removing a path not in the list still returns 200 with `{ "path": "...", "status": "removed" }`.
GET	`/api/v1/files/exists`	`installation:files`	Existence check for a file or directory under the allowlist.
POST	`/api/v1/files/upload/start`	`installation:files`	Begin a chunked upload; returns a transfer ID.
POST	`/api/v1/files/upload/chunk`	`installation:files`	Upload one chunk of an in-progress transfer.
POST	`/api/v1/files/upload/finish`	`installation:files`	Finalize a chunked upload.
POST	`/api/v1/files/upload/cancel`	`installation:files`	Cancel an in-progress chunked upload.
POST	`/api/v1/files/upload/small`	`installation:files`	Single-shot upload up to 4 MB, or up to `MaxUploadBytes` when that is lower.
POST	`/api/v1/files/download/start`	`installation:files`	Begin a chunked download; returns a transfer ID.
POST	`/api/v1/files/download/chunk`	`installation:files`	Read one chunk of an in-progress download.
POST	`/api/v1/files/download/finish`	`installation:files`	Finalize a chunked download.
POST	`/api/v1/files/download/cancel`	`installation:files`	Cancel an in-progress chunked download.
GET	`/api/v1/files/download/size`	`installation:files`	Pre-flight size check for a downloadable file.
POST	`/api/v1/files/delete-solution`	`installation:admin`	Not implemented in this version. Returns HTTP 501 `not-implemented` and changes nothing.

The chunked upload/download dance is the standard four-call sequence: `start` returns a transfer ID, then any number of `chunk` calls reference that ID, then a single `finish` call commits, or a `cancel` call discards. Chunks arriving out of order produce HTTP 400 `chunk-out-of-order`; an unknown transfer ID produces HTTP 404 `transfer-not-found`. Transfer limits:

- Only the caller that started a transfer continues, finishes or cancels it. Another caller gets 404.
- At most 8 open transfers per caller. The next one gets 429 `too-many-transfers`.
- At most 64 open transfers per server. The next one gets 503 with `Retry-After: 60`.
- A transfer idle for 10 minutes is cancelled within about a minute. A download chunk that returns no bytes does not count as activity.
- An upload that passes `MaxUploadBytes` (default 2 GiB) gets 413 `too-many-bytes`. The transfer is cancelled and its temporary file deleted.
- A chunk the server fails to write or read returns 500 and cancels the transfer.

License

Method	URL	Scope	Purpose
GET	`/api/v1/license/site-code`	`installation:license`	Retrieve the machine's site code (needed for offline activation).
PUT	`/api/v1/license/key`	`installation:license`	Set the license key.
POST	`/api/v1/license/activate`	`installation:license`	Activate the license by activation code.
POST	`/api/v1/license/deactivate`	`installation:license`	Deactivate the license by activation code.
GET	`/api/v1/license/info`	`installation:license`	License metadata: edition, dates, limits.

Example — `POST /api/v1/license/activate`

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
Request body:
{ "activationCode": "EXAMPLE-12345-67890-ABCDE-FGHIJ" }

Response (HTTP 200):
{ "status": "activated", "result": "..." }
```

A rejected key or activation code returns HTTP 400 with the `license-key-rejected` Problem Details type and the licensing service's reason in the `detail` field.

Machine settings

Reads and writes against files under `MachineSettings/` are confined to the filename allowlist (`MachineSettingsAllowlist`). Filenames outside the allowlist return HTTP 404 (not 403) so the API does not leak the existence of files the caller is not authorized to see. Reads need `installation:machine`. Reading `TWebServices.json` or any file that is not JSON also needs `installation:admin`, and writes need `installation:admin`. The `SolutionCenterApi-*.json` files are never served, whatever the allowlist says.

Method	URL	Scope	Purpose
GET	`/api/v1/machine/settings`	`installation:machine`	List the active filename allowlist.
GET	`/api/v1/machine/settings/{filename}`	`installation:machine`	Read the named MachineSettings file (returned as `application/json` for `.json` files, otherwise `text/plain`). Secret values are hidden.
PUT	`/api/v1/machine/settings/{filename}`	`installation:admin`	Write the named MachineSettings file. Body is the file contents.

Hidden secrets on read. A JSON file is returned with every secret value replaced by `"<redacted>"`. Values are hidden by member name (passwords, keys, tokens, secrets, credentials, connection strings) and by content (a password switch in a command line, a credential in a connection string, user information in a URL).

Write-back. When a PUT sends `"<redacted>"` back unchanged, the server puts the stored secret back at the same place, whatever JSON escaping the client used.

- A `.json` body must be valid JSON, and UTF-8 when it carries redacted values. Otherwise the PUT returns 400.
- When a redacted value no longer matches a stored secret at the same place, or the stored file cannot be read, the PUT returns 409 `conflict` and writes nothing. Read the file again, or send the real value.
- A list whose items hold redacted values must come back as it was read. Removing, adding or reordering its items, or changing an item that holds a secret, returns 409 `conflict` and writes nothing. The detail names the list. Send that list with the real values in place of every `"<redacted>"`, or leave the list as it was read.
- A redacted value in a member the stored file does not have, outside any list, is dropped.
- Renaming a member whose value is redacted does not carry the value. Send the real value under the new name. Renaming it onto the name of another member that held a secret gives it that member's stored value.
- A PUT that restores redacted values rewrites the file and drops JSON comments. A body without redacted values is written exactly as sent.

Server info and connections

Method	URL	Scope	Purpose
GET	`/api/v1/server/info`	`installation:read`	Hostname, OS, running processes and installation summary.
GET	`/api/v1/server/connections`	`installation:admin`	List active runtime client connections across all solutions on this machine.
DELETE	`/api/v1/server/connections/{guid}`	`installation:admin`	Force a runtime connection off. Logged for audit.
GET	`/api/v1/server/protocols`	`installation:read`	List installed communication-protocol drivers.
GET	`/api/v1/server/versions`	`installation:read`	List installed product versions on this machine (multi-version routing source).

Example — `GET /api/v1/server/info`

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "hostname": "plant1-srv07",
  "os": "Microsoft Windows NT 10.0.20348.0",
  "processCount": 12,
  "processInfo": "...",
  "currentTimeUtc": "2026-04-28T11:42:11.003Z",
  "serverInfoRows": [
    {
      "ProductVersion": "...",
      "ServerOS": "...",
      "InstalledServicesCount": "...",
      "ProcessRunningCount": "..."
    }
  ]
}
```

`serverInfoRows` returns only these installation-summary columns: `InstalledDate`, `InstalledVersions`, `ProductVersion`, `AvailableNETVersions`, `ServerOS`, `Processor`, `ValidRuntimeIdentifiers`, `InstalledServices`, `InstalledServicesCount`, `ProcessRunning`, `ProcessRunningCount`, `DirectorySeparatorChar`, `CanUpgradeProduct`, `AvailableDesignerWeb`, `AvailableRuntime`, `AvailableTSmartClient`, `AvailableDBServer`.

The host's IP addresses, installation paths and auto-start program list are intentionally **not** exposed. The caller already knows the address it connected to, and these values would give network topology and credentials to anyone holding `installation:read`. `processInfo` and `ProcessRunning` name a running solution by its full path, the same value the solution info endpoint returns.

Authentication endpoints

Method	URL	Scope	Purpose
GET	`/api/v1/auth/oidc/login`	*(none — pre-auth)*	Start single sign-on for a configured provider. Checks the return address, then returns 302 to the IdP's authorize URL.
POST	`/api/v1/auth/oidc/callback`	*(none — pre-auth)*	IdP callback target. Exchanges the authorization code for a FrameworX-issued JWT and posts it to the return address.
POST	`/api/v1/auth/refresh`	*(uses the supplied JWT)*	Refresh an expiring or recently expired JWT within the refresh window and the session lifetime. Service-account tokens are refused.

OpenAPI document

Method	URL	Scope	Purpose
GET	`/api/v1/openapi`	*(none — reachable even while the API is off)*	OpenAPI 3.1 document describing every endpoint, scope, and response shape. Suitable for code generation.

Errors

All error responses use the RFC 7807 Problem Details envelope with `Content-Type: application/problem+json`:

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "type":     "https://errors.example.com/solutioncenter-api/permission-denied",
  "title":    "Permission denied",
  "status":   403,
  "detail":   "JWT scope insufficient for this resource.",
  "instance": "/api/v1/server/connections"
}
```

The `type` URL base is OEM-configurable. Standard error type slugs:

HTTP status	Error type slug	When
400	`bad-request`	Malformed request, missing required parameter, wrong HTTP verb for the URL, a sign-in return address that is not allowed.
400	`path-not-allowed`	File operation references a path outside the allowlist, contains traversal segments, or passes through a link.
400	`chunk-out-of-order`	Upload chunks arrived out of sequence.
400	`license-key-rejected`	Licensing service refused the supplied key or activation code.
401	`authentication-required`	Token missing, malformed, or signature invalid. Refresh of a service-account token.
401	`session-expired`	JWT `exp` claim is in the past, the refresh window has been exceeded, or the session passed `MaxSessionLifetimeSeconds`.
404	`not-found`	Resource does not exist, OR caller's scope set is insufficient (BOLA defence — intentional collapse to 404 instead of 403).
404	`solution-not-found`	Named solution is not installed on this machine.
404	`file-not-found`	Requested file does not exist within the allowlist.
404	`filename-not-allowed`	MachineSettings filename is outside the configured allowlist.
404	`transfer-not-found`	Chunked-transfer ID is unknown or expired.
409	`conflict`	A machine-settings write-back whose redacted values cannot be restored. Nothing was written.

413	`too-many-bytes`	Chunked transfer exceeded the configured per-transfer byte cap.
429	`too-many-transfers`	The caller already has the maximum number of open file transfers.
500	`internal-error`	Unhandled server-side condition; details in the response `detail` field and TraceLog.
501	`not-implemented`	The operation does not exist in this version (`files/delete-solution`).
503	`service-unavailable`	The API is off: not enabled, no TLS certificate on the listener, or a caller that is not local in local-only mode. This response has no `Retry-After`: do not retry. The same status with `Retry-After` means too many pending sign-ins or open file transfers: retry later.

Brand-neutrality and OEM hooks

The SolutionCenter API is designed to be re-branded by OEMs without source edits. With default configuration, no response body contains the literal strings `FrameworX` or `Tatsoft` — this is verified by an automated test in the build. The following knobs are configurable per installation. Members named in the Override column sit in `SolutionCenterApi-Settings.json`.

Surface	Default	Override
OpenAPI document title	`SolutionCenter API`	`OpenApiTitle`
Problem Details `type` URL base	`https://errors.example.com/solutioncenter-api/`	`ErrorTypeBase`
JWT `iss` claim	`https://api.example.com`	`Issuer`
JWT `aud` claim	`solutioncenter`	`Audience`
Service-account JWT signing keys	(none — ephemeral key generated on first run)	`SolutionCenterApi-Keys.json` in the installation folder
OIDC providers + client secrets	(none — OIDC unavailable until at least one provider is enrolled)	`SolutionCenterApi-OidcProviders.json` in the installation folder
File-operation path allowlist	Solutions and backup directories	`AllowedPaths[]`
MachineSettings filename allowlist	`TWebServices.json`, `TWebServicesRunPrograms.json`, `Logging.json`	`MachineSettingsAllowlist[]`
API on or off	OFF	`Enabled` (see Enabling the API)
OpenAPI `servers[]` list	empty	`PublicBaseUrls[]`

The following are **not** configurable (they are part of the API contract):

- The `/api/v1/...` URL prefix.
- The JWT model (RFC 9068 `at+jwt`, HS256 in v1).
- The Problem Details envelope structure (RFC 7807).
- The set of scopes (additions are possible in v2; existing scopes will not lose meaning).
- The activation gate mechanism itself.

OpenAPI 3.1 document

`GET /api/v1/openapi` returns the OpenAPI 3.1 document describing every endpoint, scope, and response. All operation IDs are namespaced under `solutioncenter.*` to avoid collision with the Runtime API in shared tooling. The document is suitable as input to standard client generators (`openapi-generator-cli`, `NSwag`, `autorest`).

The OpenAPI endpoint stays reachable while the API is off, so tooling inspects the surface before an administrator enables the API.

The OpenAPI document advertises base URLs via its `servers[]` array. By default this array is empty — client generators and Swagger UI then fall back to the host that served the document, which is correct in most local-network deployments. If the OpenAPI document will be consumed away from the API host (developer portals, partner SDK distributions, or reverse-proxy fronts where the externally-reachable hostname differs from the listener), populate `PublicBaseUrls` in `SolutionCenterApi-Settings.json` with the canonical external URL(s) — one entry per URL — so generated clients and rendered docs target the right host.

What is not in v1 (reserved for v2)

Two URL families are intentionally reserved and return HTTP 404 in v1, so callers that depend on them cannot mistake an unimplemented capability for a network failure or a permission issue:

- `/api/v1/installations/...` — reserved for multi-installation aggregation (a centralized SolutionCenter API instance proxying queries across many remote machines).
- `/api/v1/fleet/...` — reserved for fleet-wide rollups, agent registry, and heartbeat ingestion.

v1 is single-installation by design: each machine's API manages that machine's solutions. Operators with multiple installations call each machine's API directly from their own orchestration layer (Ansible, custom dashboards, PowerShell scripts). Multi-installation aggregation is a v2 capability when a real fleet operator drives the requirements.

Other capabilities deferred to v2: asymmetric JWT signing (RS256 / ES256) with a JWKS endpoint, OIDC client-credentials grant for service accounts, cursor pagination, audit-log query API, WebSocket push for live state, bulk multi-installation rollout.

Deployment scenarios

The SolutionCenter API ships in **both** TWebServices binaries on every FrameworX installation, on the same port, with the same surface:

Binary	Path	Target framework	Use when
TWebServices (Windows .NET Framework)	`TWebServices.exe` in the installation folder	.NET Framework 4.8	Default Windows install. Runs on every supported Windows host without needing a separate runtime.
TWebServices (.NET 10)	`net10.0\TWebServices.exe` in the installation folder	.NET 10	Cross-platform deployments where the host already runs the .NET 10 runtime (Linux, container, or Windows hosts standardized on .NET 10).

Both binaries listen on port **10108** by default and expose identical `/api/v1/...` behaviour. An installation runs one or the other — not both at once on the same machine. Each binary reads `SolutionCenterApi-Settings.json` from its own folder.

Quick-start

The examples below assume the API is enabled, the TWebServices listener has a TLS certificate, and a service-account JWT is in place. Replace `plant1-srv07` with the target host and `$jwt` with the actual token.

PowerShell — list solutions

```
$jwt = Get-Content -Path "C:\secure\fleet-jwt.txt" -Raw

$headers = @{ Authorization = "Bearer $jwt" }

$resp = Invoke-RestMethod `
    -Uri    "https://plant1-srv07:10108/api/v1/solutions/" `
    -Method GET `
    -Headers $headers

$resp.solutions | Format-Table name, profile, status, port
```

curl — check whether a solution is running

```
curl --silent \
     --header "Authorization: Bearer $JWT" \
     "https://plant1-srv07:10108/api/v1/solutions/PlantSCADA/isrunning"

# {"name":"PlantSCADA","isRunning":true}
```

Python — start a solution

```
import os, requests

jwt  = os.environ["FLEET_JWT"]
host = "plant1-srv07"

resp = requests.post(
    f"https://{host}:10108/api/v1/solutions/PlantSCADA/run",
    headers={"Authorization": f"Bearer {jwt}"},
    json={"profile": "Production"},
    timeout=30,
)

resp.raise_for_status()
print(resp.json())
# {"status": "starting", "statusUrl": "/api/v1/solutions/PlantSCADA/isrunning"}
```

In this section…