# Runtime REST API Reference

Read and write a running solution's live data over HTTPS (tags, historian, alarms, and the Object Model) on the solution's TServer port.



---

The Runtime REST API publishes a FrameworX solution's **live runtime data** over HTTPS: current and historical tag values, alarms, Object Model browse, and health. It runs on the per-solution TServer port and is the modern, versioned replacement for the legacy WebAccess API. For machine-wide management (starting/stopping solutions, licensing, files) see the companion SolutionCenter API on port 10108. That API is off by default, and an administrator enables it.

## Base URL and ports

A TServer process hosts exactly one solution, so the host and port together already name the solution and its execution profile. There is no `/solutions/{name}` segment in the URL. The port is `3100 + profile id`:

| Execution profile | Port | Base URL |
|---|---|---|
| Production | 3101 | `https://host:3101/api/v1/` |
| Development | 3201 | `https://host:3201/api/v1/` |
| Validation | 3301 | `https://host:3301/api/v1/` |
| Custom | 3401 | `https://host:3401/api/v1/` |

## Before Your First Call

- **Port.** A solution can run on a port other than the defaults above. The Designer shows the addresses the running solution serves at **Runtime Startup**.
- **Password.** `/auth/login` needs a non-empty password and refuses an empty one with `400`. The Guest user is the one account whose password is not checked: any non-empty password signs it in, and the session has the Guest user's permissions. Set a password for every other user in **Security Users**, and give the Guest user only the permissions an anonymous caller may hold.
- **Visibility.** The `/api/v1/uns/…` family serves tags whose Visibility is Public. A tag with any other Visibility is not listed and answers `404`, the same as a path that does not exist.
- **Restart.** Restart the runtime after you change a password or a tag's Visibility. A running solution keeps the previous settings until it restarts.
- **Quality.** A `quality` of `192` means Good.
- **Shell.** In Windows PowerShell, `curl` is an alias for `Invoke-WebRequest`, so type `curl.exe`. The JSON body is quoted differently in cmd and in PowerShell. `<server>` is the scheme, host and port shown in Runtime Startup.

**Login from a Windows shell**

```
cmd:
curl.exe -s -X POST <server>/api/v1/auth/login -H "Content-Type: application/json" -d "{\"user\":\"<user>\",\"
password\":\"<password>\"}"

PowerShell:
curl.exe -s -X POST <server>/api/v1/auth/login -H "Content-Type: application/json" -d '{\"user\":\"<user>\",\"
password\":\"<password>\"}'
```

## Authentication

Every data endpoint requires a Bearer token in the `Authorization` header. Two token kinds share the same wire format:

### Session token: user login

Exchange a solution user's credentials for a short-lived session token:

**POST /api/v1/auth/login**

```
POST /api/v1/auth/login HTTP/1.1
Host: scada.example.com:3101
Content-Type: application/json

{ "user": "partner-acme", "password": "<plaintext>" }

--- 200 OK ---
{
  "token": "9c30730a4-e2ee-4595-9131-d64769c5afbe",
  "expiresAt": "2026-07-07T18:35:00Z",
  "user": "partner-acme"
}
```

Send the token on every subsequent request. Session lifetime follows the user's Security Policy (inactivity / duration; `0` = never expires). Call `POST /api/v1/auth/logout` to invalidate it.

`expiresAt` in the `/auth/login` and `/auth/me` responses gives the time the session ends if it receives no further requests. Each authenticated request moves this time forward.

**Authenticated request**

```
GET /api/v1/uns/Plant1/Tank1.Level HTTP/1.1
Host: scada.example.com:3101
Authorization: Bearer 9c30730a4-e2ee-4595-9131-d64769c5afbe
```

### Pre-issued Bearer Token: machine-to-machine

For webhooks, headless scripts, and service accounts, issue a long-lived **Bearer Token** in the Designer **Security  Security Secrets** page and send it directly as `Authorization: Bearer <token>`, with no login round-trip. The token inherits the permission group of its associated user, is stored encrypted, shown once at issuance, and works until you disable or delete it.

> ⚠ Permissions are enforced server-side by the runtime's tag security on every call. A token can only read or write what its associated user is allowed to. A request for an object the caller cannot see returns `404`, not `403`.

To confirm a token and see the resolved identity, call `GET /api/v1/auth/me`.

The probes `/ping`, `/health`, `/ready`, and `/api/v1/openapi.json` are anonymous and require no token.

## Cross-origin requests (CORS)

A browser page on another origin calls the API with a Bearer token in the `Authorization` header. The CORS preflight response names `Authorization` as an allowed header, so the browser sends the token.

By default, the runtime answers every origin with `Access-Control-Allow-Origin: *` and no credentials. To limit cross-origin access to known origins, list them in `MachineSettings\TServer\Cors.settings.json` under the FrameworX public documents folder (by default `C:\Users\Public\Documents\FrameworX\`), then restart the runtime:

**Cors.settings.json**

```
{ "AllowedOrigins": [ "https://ops.plant.example", "http://localhost:5173" ] }
```

With a list in place, only a listed origin receives CORS headers, and a listed origin also receives `Access-Control-Allow-Credentials: true`. A request from any other origin receives no CORS headers, and the browser blocks it.

## Endpoints

### UNS: tags and UserTypes

The `/api/v1/uns/…` family is UNS-strict (the `Tag.*` namespace, Pillar 1). Tag paths are written directly (`Plant1/Tank1.Level`); the `Tag.` prefix is added automatically.

| Method & path | Purpose |
|---|---|
| `GET /api/v1/uns/{path}` | Read current value, quality, timestamp of one tag. `?include=ontology` adds DisplayText / Labels / SourceIri / attributes. |
| `GET /api/v1/uns?query=&type=&limit=` | Search the UNS by path substring and optional type filter. |
| `GET /api/v1/uns/{path}/history?from=&to=` | Raw historian samples over a time range. |
| `GET /api/v1/uns/{path}/aggregated?from=&to=&aggregation=&interval=` | Time-bucketed aggregates (Average, Minimum, Maximum, and other kinds; ISO 8601 `interval`, e.g. `PT5M`). |
| `GET /api/v1/uns/{path}/describe` | Full metadata: inheritance, derived types, ontology, current value, alarm state. |
| `GET /api/v1/uns/by-iri?iri=` | Resolve an external ontology IRI back to the FrameworX object. |
| `POST /api/v1/uns/group` | Bulk read up to 1000 tags in one request. |
| `PUT /api/v1/uns/group` | Write one or N tag values (idempotent; governed by tag security). |
| `GET /api/v1/uns/usertypes` | List UserTypes with member and instance counts. |
| `GET /api/v1/uns/usertypes/{name}` | Describe one UserType (members, inheritance, derived types, ontology). |
| `GET /api/v1/uns/usertypes/{name}/instances` | List the tag instances of a UserType. |

### Runtime Object: the Object Model roots

The `/api/v1/runtime-object/…` family spans the whole Object Model (`Tag`, `Server`, `Client`, `Info`, `Alarm`, `Device`, `Historian`, `Dataset`, `Script`, `Display`, `Layout`, `Report`, `Security`, `Enumeration`, `TagProvider`). Paths here are dot-rooted and must be fully qualified (`Tag.Plant1/Tank1.Level`, `Server.Now`).

| Method & path | Purpose |
|---|---|
| `GET /api/v1/runtime-object/{path}` | Describe a single node (properties, methods, current values) across any of the Object Model roots. |
| `GET /api/v1/runtime-object/browse?root=&path=&depth=` | Walk the Object Model tree from a starting node (depth 1 to 5). |
| `POST /api/v1/runtime-object/group` | Bulk read up to 1000 cross-namespace paths. |

### Alarms and runtime info

| Method & path | Purpose |
|---|---|
| `GET /api/v1/alarms/active` | Currently active alarms. |
| `GET /api/v1/alarms/history?from=&to=&filter=` | Alarm history rows (Activated / Normalized / Acked) with optional filter. |
| `GET /api/v1/info` | Running solution metadata: build, profile, target framework, uptime, license edition. |
| `GET /health`, `/ready`, `/ping` | Liveness / readiness probes (anonymous). Kubernetes-ready. |

The API has no endpoint for Script Class methods.

## Examples

**Read one tag value**

```
GET /api/v1/uns/Plant1/Tank1.Level
Authorization: Bearer <token>

--- 200 OK ---
{ "objectName": "Plant1/Tank1.Level", "value": 47.3, "quality": 192,
  "utcTimestamp": "2026-07-07T15:30:42.123Z" }
```

**Bulk read**

```
POST /api/v1/uns/group
Authorization: Bearer <token>
Content-Type: application/json

{ "paths": ["Plant1/Tank1.Level", "Plant1/Tank2.Level", "Plant1/Motor1.Speed"] }
```

**Write tag values**

```
PUT /api/v1/uns/group
Authorization: Bearer <token>
Content-Type: application/json

{ "writes": [
    { "path": "Plant1/Tank1.Setpoint", "value": 50.0 },
    { "path": "Plant1/Tank2.Setpoint", "value": 60.0 }
] }
```

**Aggregated history**

```
GET /api/v1/uns/Plant1/Tank1.Level/aggregated
        ?from=2026-07-07T00:00:00Z&to=2026-07-07T01:00:00Z
        &aggregation=Average&interval=PT5M
Authorization: Bearer <token>
```

## Errors

All errors use the RFC 7807 Problem Details envelope: `type`, `title`, `status`, `detail`, `instance`.

| HTTP | When |
|---|---|
| 400 | Malformed request or body, out-of-range parameter, over-limit batch (`too-many-objects`), invalid time format, or a non-Tag path on a UNS-strict route (`uns-namespace-mismatch`). |
| 401 | Missing / invalid token, or expired session (`authentication-required`). |
| 404 | Object or resource not found, *or* it exists but is not visible to the caller's permission set (BOLA defence). |
| 405 | HTTP method not allowed on that route (e.g. a non-GET on a read-only endpoint). |
| 422 | Invalid argument, or an unknown aggregation kind (`invalid-aggregation`). |
| 500 | Unexpected error on the server (`internal-error`). |
| 501 | The route could not reach the runtime module that serves it (`method-not-yet-implemented`). This is an internal fault, not a problem with the request. |

**Per-item results in bulk calls.** Bulk read and write return HTTP `200` with a `results` array and an `errors` array. A single failing item (a write rejected by tag security, a path that doesn't resolve, a value that doesn't match the tag's type) appears in `errors` while the remaining items still succeed in `results`. HTTP-level 4xx codes are reserved for request-level problems (bad body, over-limit, auth).

## Time format

All timestamps are ISO 8601 strict UTC with a mandatory `Z` suffix (`2026-07-07T15:30:42.123Z`). Durations use ISO 8601 duration syntax (`PT5M` = 5 minutes, `PT1H` = 1 hour, `P1D` = 1 day). A local-naive timestamp is rejected with `400`.

## OpenAPI specification

The runtime serves its **OpenAPI 3.0.3** document at `GET /api/v1/openapi.json`. The endpoint is anonymous, and the alias `GET /api/v1/openapi` redirects to it. Load the document into Swagger UI, Postman, or a client generator to work from the current contract. The document lists only the routes the runtime serves.

## Migrating from the legacy WebAccess API

Existing `/rtServices/api/…` integrations continue to work, but new integrations should use the endpoints above. Direct equivalents:

| Legacy WebAccess (`/rtServices/api/`) | Runtime REST API (`/api/v1/`) |
|---|---|
| `GET SetServer` (mint bearer) | `POST /api/v1/auth/login` |
| `GET SyncObjects/ReadObject?ObjectName=Tag1` | `GET /api/v1/uns/Tag1` |
| `GET SyncObjects/ReadObjects` | `POST /api/v1/uns/group` |
| `POST SyncObjects/WriteObject` | `PUT /api/v1/uns/group` (single-element) |
| `POST SyncObjects/WriteObjects` | `PUT /api/v1/uns/group` |

For convenience during migration, `GET /api/v1/tags/…` and `GET /api/v1/historian/…` also resolve: they issue a `308` redirect to the canonical `/api/v1/uns/…` path.

---

**In this section...**

---