GIRA

Gira IoT REST API — Documentation

Date: 2026-08-31
Version: v3

Contents

This document describes **version 3** of the Gira IoT REST API. It supersedes the v2 documentation (dated 01.07.2020).

What's new in v3

Compared with v2:

- API version `3` is the current default; the availability response adds **`uidLength`**.
- UI configuration adds **`iconID`** (functions, locations, trades) and per-function **`scenes`**.
- Register-callbacks adds the **`batchEvents`** option; value events can be delivered batched.
- Value callbacks are **filtered per user** (a client receives events only for functions its user may see) and are **rate-limited**.
- The **Reporting** endpoint (`POST /api/messages`) is now documented (it was already available in v2).
- The error model documents `invalidToken` as **HTTP 401** and adds **`429`** (`tooManyRequests`), returned by the request rate limiter; `403`/`405` are not emitted.

---

1 About Gira IoT REST API

Using the Gira IoT REST API, you can programmatically read, write and watch data points of supported devices.

Supported devices:

Device type	Firmware version	Gira IoT REST API version
Gira X1	4.0	v3
Gira One Server	4.0	v3
Gira HomeServer / Gira FacilityServer	4.10	v2 only

The Gira HomeServer / FacilityServer support API version v2 only; there are currently no plans to support v3 on these devices. Clients targeting those devices should use the v2 documentation.

Functions are the representations of actual channels and data points in terms of the user interface configuration within the Gira Project Assistant (GPA). The functions are part of the UI configuration which also describes the locations (building structure) and the trades.

To make use of the API a client application has to register with an application identifier using a device's username and password for authorization. The registration for the API is using HTTPS with basic access authentication only.

To use callbacks, a separate HTTPS server has to be used, where the REST API can send the callback events to.

Important: The HTTPS connection is not fully trusted, because it is not technically possible that the server provides a trusted TLS certificate. Because of this, the client that is used to access the Gira IoT REST API needs to skip the certification check. Please consult the documentation of the client software or library on how to skip the certification check.

Target group. This documentation is aimed at persons who already have the following knowledge:

- Basic understanding of HTTP communication
- Basic understanding of JSON

2 Basic Usage

All Gira IoT REST API access is via HTTPS. All data is sent and received as JSON.

Request Method	Description
`GET`	Retrieve resources. Returns `200 OK` on success.
`PUT`	Update a resource. Returns `200 OK` on success.
`POST`	Create a new resource. Returns `201 Created` on success.
`DELETE`	Delete a resource. Returns `204 No Content` on success.

Response codes

Code	Description
`200 OK`	Request was successful, the requested resource is returned.
`201 Created`	Request was successful, the created resource is returned.
`204 No Content`	Request was successful, no content to return.
`400 Bad Request`	Invalid request.
`401 Unauthorized`	Missing or invalid authorization.
`404 Not Found`	Resource not found.
`422 Unprocessable Entity`	Unable to process the request.
`423 Locked`	Device is currently locked.
`429 Too Many Requests`	Too many requests sent in a given amount of time.
`500 Internal Server Error`	Internal server error.

> Note: unknown resources return `404 Not Found` (`resourceNotFound`). A request using an unsupported HTTP method is not answered by the API, and the fronting reverse proxy (nginx) returns `502 Bad Gateway`. The API does not emit `403` or `405`; it returns `429` (`tooManyRequests`) when a source IP exceeds the failed-authentication rate limit. Responses include the CORS header `Access-Control-Allow-Origin: *`.

In case of an error, additional information is returned in an `error` object:

```
{
  "error": {
    "code": "<errorCode>",
    "message": "<errorMessage>"
  }
}
```

The `code` field contains one of several valid error codes, the `message` field contains a more detailed error message.

Possible codes:

Code	HTTP	Message
`generic`	500	…
`unprocessable`	422	…
`invalidAuth`	401	Missing or invalid authentication.
`invalidToken`	401	Missing or invalid token.
`locked`	423	Device is currently locked.
`missingContent`	400	Missing content / Missing or invalid value in content.
`resourceNotFound`	404	Resource … not found.
`uidNotFound`	404	UID … not found.
`clientNotFound`	404	Client … not found.
`operationNotSupported`	400	Operation not supported for '…'.
`callbackTestFailed`	400	Callback test failed for '…'.
`refreshLicensesFailed`	500	Failed to refresh licenses: '…'.
`tooManyRequests`	429	Too many requests, try again later.
`unknown`	500	Unknown error.

`invalidAuth` and `invalidToken` responses also include a `WWW-Authenticate` header (`Basic realm="REST API"` and `Token realm="REST API"` respectively).

Failed authentication attempts are rate-limited per source IP address: five attempts, replenished over five seconds. A global limit of twenty attempts, replenished over one second, applies across all addresses. Exceeding either returns `429 Too Many Requests` (`tooManyRequests`) until the allowance recovers; a successful authentication is not counted.

2.1 Unique Identifier

The API relies on a concept of unique identifiers (UID) which are short (four characters), persistent and without instant recurrence.

The four-character-long UID consists of small letters and digits and starts with a letter. The UID length is also reported by the availability check as `uidLength` (see section 3).

2.2 Versioning via URL path

The Gira IoT REST API supports access to different versions of the API via the URL path in requests. The available versions are `v1`, `v2` and `v3`.

```
GET /api/v3/resource
```

By omitting the version part in the URL, the latest available version (currently **v3**) is used. Requesting a version outside the supported range returns `404 Not Found`.

2.3 Authorization

All API requests apart from the API availability check require a registration of a client with authorization in the form of username and password.

To register a client, it is required to provide the credentials as HTTP Basic auth in the `Authorization` header. This creates a new client on the server and returns a unique token. Subsequent API calls require this token for authorization, supplied **either** as a bearer token in the `Authorization` header **or** as a URL query parameter:

```
GET /api/v3/resource
Authorization: Bearer <token>
```

```
GET /api/v3/resource?token=<token>
```

If both are present, the `Authorization` header takes precedence.

3 API availability check

The availability of the Gira IoT REST API can be checked by accessing the base URL.

```
GET /api/v3/
```

Response

```
{
  "info": "GDS-REST-API",
  "version": "3",
  "deviceName": "<display name>",
  "deviceType": "<device type identifier>",
  "deviceVersion": "<device version number>",
  "uidLength": 4
}
```

Field	Description
`info`	Always the string `GDS-REST-API`.
`version`	The version number of the API (as a string).
`deviceName`	User-defined display name of the device. Returned from v2.
`deviceType`	Device type identifier, for example `GIGSRVKX02` for Gira X1. Returned from v2.
`deviceVersion`	Firmware version of the device with format `n.n.n.n`, for example `4.0.348.0`. Returned from v2.
`uidLength`	Length of a UID string. Currently `4`. **New in v3.**

This endpoint requires no authorization and is not blocked while the device is locked.

4 Registration

4.1 Client Identifier

Every client has to use a unique client identifier. To ensure uniqueness, client identifiers have to be URNs within the organization of the client (e.g. `de.gira.gdsrestapi.clients.my_well_known_service`).

4.2 Register client

Client-side registration at the API is necessary for management of client identity and optional callback URLs for eventing. In case the client wants to be notified of configuration changes or value changes, corresponding callback URLs can be specified after registration. HTTPS-based callback URLs are supported only.

The API returns a client access token on successful registration which has to be used for authentication in all subsequent API calls.

```
POST /api/clients
```

```
{ "client": "<client identifier>" }
```

Response

```
{ "token": "<client access token>" }
```

Response codes

HTTP Code	Error code	Description
201 Created		Client was successfully registered.
400 Bad Request	`missingContent`	Body is empty/invalid.
401 Unauthorized	`invalidAuth`	Missing or invalid authentication.
423 Locked	`locked`	The device is currently locked.
429 Too Many Requests	`tooManyRequests`	The source IP is temporarily blocked after too many failed authentication attempts.

The token is a randomly created string containing 32 alphanumeric upper/lower case characters (regex `[0-9A-Za-z]{32}`).

Every time a new combination of username from the HTTP Basic Authorization header and the `<client-identifier>` is used, a new token will be generated. Registering with the same username and the same `<client-identifier>` again without unregistering the token will yield the same token. Multiple clients can be registered at the same time and the lifetime of a token is not limited.

The token stays valid until the client is unregistered, the user is deleted or the username is changed. The token stays valid when the user's password is changed. A token that was invalidated due to a removed username will become valid again if the username is used again.

4.3 Unregister client

```
DELETE /api/clients/<access token>
```

Response codes

HTTP Code	Error code	Description
204 No Content		Client was successfully unregistered.
401 Unauthorized	`invalidToken`	The token cannot be found.
423 Locked	`locked`	The device is currently locked.
429 Too Many Requests	`tooManyRequests`	The source IP is temporarily blocked after too many failed authentication attempts.
500 Internal Server Error	`generic`	Failed to remove client.

4.4 Register callbacks

```
POST /api/clients/<access token>/callbacks
```

```
{
  "serviceCallback": "<callback URL of service events, optional>",
  "valueCallback": "<callback URL of value events, optional>",
  "testCallbacks": "<boolean value, optional>",
  "batchEvents": "<boolean value, optional>"
}
```

Field	Description
`serviceCallback`	HTTPS URL that receives service events. Optional.
`valueCallback`	HTTPS URL that receives value events. Optional.
`testCallbacks`	If `true`, each provided callback URL is probed during registration and must respond `200 OK`. Optional, default `false`.
`batchEvents`	**New in v3.** If `true`, value events for this client are delivered batched (multiple events per callback request) instead of one event per request. Optional, default `false`. See section 4.7.

At least one of `serviceCallback` / `valueCallback` must be provided. Only one set of callback URLs can be registered per `<access token>`; registering again replaces the previous set.

Callback URL requirements. Each callback URL must be an absolute `https://` URL of the form `https://<host>[:<port>]/<path>[?<query>]`. The host must be reachable from the device and must not be a loopback/localhost address. URLs that do not use HTTPS, contain control characters (including CR/LF), or are otherwise malformed are rejected with `422 unprocessable`.

Response codes

HTTP Code	Error code	Description
200 OK		Callback was successfully registered.
400 Bad Request	`missingContent`	Body is empty/invalid.
400 Bad Request	`callbackTestFailed`	Callback test did not respond with `200 OK`.
401 Unauthorized	`invalidToken`	The token cannot be found.
422 Unprocessable	`unprocessable`	The callback URL is invalid — not `https://`, contains control characters, targets local-host/loopback, or is malformed.
423 Locked	`locked`	The device is currently locked.
429 Too Many Requests	`tooManyRequests`	The source IP is temporarily blocked after too many failed authentication attempts.

4.5 Remove callbacks

```
DELETE /api/clients/<access token>/callbacks
```

Response codes

HTTP Code	Error code	Description
200 OK		Callback was successfully removed.
401 Unauthorized	`invalidToken`	The token cannot be found.
423 Locked	`locked`	The device is currently locked.
429 Too Many Requests	`tooManyRequests`	The source IP is temporarily blocked after too many failed authentication attempts.
500 Internal Server Error	`generic`	Failed to remove callback.

4.6 Service callback

```
POST <callback URL of service events>
```

Body

```
{
  "token": "<client access token>",
  "events": [ { "event": "<event type>" }, ... ],
  "failures": "<number of unsuccessful callback attempts since last successful one,
optional>"
}
```

Response codes

- `200 OK`
- `404 Not Found` — if the client responds to the callback event with `404 Not Found`, the API will unregister the client implicitly.

Event types

Event	Description
`test`	The callback test event type. Used in the 'register and test' scenario only.
`startup`	The device is up and running.
`restart`	The device is going to restart. Sent with a `Connection: close` header.
`projectConfigChanged`	Project configuration has changed (e.g. GPA download). In this case the UI configuration might not have changed as well. Reacting to this event immediately is not recommended, as the REST server will still be locked for a few seconds when this event is emitted.
`uiConfigChanged`	UI configuration has changed.

4.7 Value callback

```
POST <callback URL of value events>
```

Body

```
{
  "token": "<client access token>",
  "events": [ { "uid": "<unique identifier of data point>", "value": "<new data point
value>" }, ... ],
  "failures": "<number of unsuccessful callback attempts since last successful one,
optional>"
}
```

In case of a callback test, `events` is an empty list.

Response codes

- `200 OK`
- `404 Not Found` — if the client responds to the callback event with `404 Not Found`, the API will unregister the client implicitly.

Delivery, filtering and rate limiting

- **Per-user filtering:** a client receives value events only for data points belonging to functions its user is allowed to see. Events for other data points are not delivered.
- **Batching:** if the client registered with `batchEvents: true` (section 4.4), value events are collected and delivered as an array; otherwise each event is delivered in its own request.
- **Rate limiting:** to protect clients — and services that aggregate events from many devices — from event storms, value events are throttled. If a single data point exceeds 50 events within 5 seconds, further events for that data point are suppressed until the next project configuration change. If more than 500 value events occur within 5 seconds across all data points, a global safeguard trips and **all** value events are suppressed **until the device is restarted** — this is intentional and does not reset on a configuration change. Design clients so they do not generate such bursts. Value updates with a missing or error state are not delivered.

5 UI configuration

5.1 UI configuration identifier

```
GET /api/uiconfig/uid
```

Response

Unique identifier of the current configuration. This identifier changes each time the configuration is changed (e.g. GPA project download, configuration changes with the Gira Smart Home App).

```
{ "uid": "<unique ui configuration identifier>" }
```

Response codes

HTTP Code	Error code	Description
200 OK		The current configuration identifier is returned.
401 Unauthorized	`invalidToken`	Missing or invalid token.
423 Locked	`locked`	The device is currently locked.
429 Too Many Requests	`tooManyRequests`	The source IP is temporarily blocked after too many failed authentication attempts.
500 Internal Server Error	`unknown`	The device could not provide the configuration identifier.

5.2 Get configuration

```
GET /api/uiconfig[?expand=dataPointFlags,parameters,locations,trades]
```

Response

Get the complete UI configuration.

- `uid` — The unique UI configuration identifier.
- `functions` — A list of all functions.
  - `uid` — The unique identifier of the function.
  - `functionType` — Function type's unique resource name. See section 10.1, Function definitions.
  - `channelType` — Channel type's unique resource name. See section 10.2, Channel definitions.
  - `displayName` — UTF-8 based display name.
  - `iconID` — Numeric icon identifier of the function (`-1` if none). **New in v3.**
  - `dataPoints` — A list of all available data points in the function.
    - `uid` — The unique identifier of the data point.
    - `name` — The logical name of the data point based on the channel definition.
    - `canRead` / `canWrite` / `canEvent` — Data point capabilities. Returned only if `dataPointFlags` is present in the `expand` parameter.
  - `parameters` — A list of function parameters. Returned only if present in the `expand` parameter.

  - `scenes` — Scene definitions of the function, each an object `{ "name": <string>, "number": <int> }`. **New in v3.** Returned only when the function has scenes.
- `locations` — A nested list of all locations and the contained unique function identifiers. Returned only if present in the `expand` parameter. Each location has a `displayName`, `locationType`, `iconID` and `functions`, and may contain nested `locations`.
- `trades` — A list of all trades and the contained unique function identifiers. Returned only if present in the `expand` parameter. Each trade has a `displayName`, `tradeType`, `iconID` and `functions`.

```
{
  "uid": "<unique ui configuration identifier>",
  "functions": [
    {
      "uid": "<unique function id>",
      "functionType": "<function type urn>",
      "channelType": "<channel type urn>",
      "displayName": "<display name>",
      "iconID": -1,
      "dataPoints": [
        {
          "uid": "<unique id>",
          "name": "<name within channel definition>",
          "canRead": true,
          "canWrite": true,
          "canEvent": true
        }
      ],
      "parameters": [
        { "set": "<set name>", "key": "<key name>", "value": "<value>" }
      ],
      "scenes": [ { "name": "All on", "number": 2 } ]
    }
  ],
  "locations": [
    {
      "displayName": "<display name>",
      "locationType": "<predefined location type>",
      "iconID": -1,
      "functions": [ { "uid": "<unique function identifier>" } ],
      "locations": [ "..." ]
    }
  ],
  "trades": [
    {
      "displayName": "<display name>",
      "tradeType": "<predefined trade type>",
      "iconID": -1,
      "functions": [ { "uid": "<unique function identifier>" } ]
    }
  ]
}
```

Response codes

HTTP Code	Error code	Description
200 OK		The UI configuration is returned.
401 Unauthorized	`invalidToken`	Missing or invalid token.
401 Unauthorized	`invalidAuth`	The device refused the request for this user.
423 Locked	`locked`	The device is currently locked.
429 Too Many Requests	`tooManyRequests`	The source IP is temporarily blocked after too many failed authentication attempts.
500 Internal Server Error	`unknown`	The device could not provide the configuration.

6 Values

Data-point values are exchanged as JSON. When reading values (section 6.1) the API returns each `value` as a **JSON string** (for example `"20"` or `"70.196078"`), regardless of the underlying data type. When writing values (section 6.2, section 6.3) the `value` may be supplied as a JSON string or a JSON primitive (for example `20` or `true`).

6.1 Get value(s)

```
GET /api/values/<uid>
```

The UID can refer to:

- A data point, in which case only this data point's value is returned.
- A function, in which case all the function's data point values are returned.

Response

The actual value of the referenced data point(s).

```
{
  "values": [
    { "uid": "<unique data point identifier>", "value": "<actual value>" }
  ]
}
```

Response codes

HTTP Code	Error code	Description
200 OK		Value(s) returned.
401 Unauthorized	`invalidToken`	Missing or invalid token.
401 Unauthorized	`invalidAuth`	The UID is not visible to the user — also returned for unknown or malformed UIDs, as visibility is checked first.
404 Not Found	`uidNotFound`	Edge case: the UID is visible but has no data-point mapping.
422 Unprocessable	`unprocessable`	The UID does not refer to a readable data point.
423 Locked	`locked`	The device is currently locked.
429 Too Many Requests	`tooManyRequests`	The source IP is temporarily blocked after too many failed authentication attempts.
500 Internal Server Error	`generic`	The device could not provide the value.

6.2 Set value(s)

```
PUT /api/values
```

Request

The value(s) of the specified data point(s).

```
{
  "values": [
    {
      "uid": "<unique data point identifier>",
      "value": "<actual value>",
      "hint": "<field bus specific additional information, optional>"
    }
  ]
}
```

Data points that are not visible to the user are silently skipped. If no settable data point remains, `422 unprocessable` is returned.

Response codes

HTTP Code	Error code	Description
200 OK		Values were set.
400 Bad Request	`missingContent`	`values` missing or not an array.
401 Unauthorized	`invalidToken`	Missing or invalid token.
401 Unauthorized	`invalidAuth`	The device refused the request for this user.
422 Unprocessable	`unprocessable`	An entry is invalid, no settable values remain, or the device failed to set one or more values.
423 Locked	`locked`	The device is currently locked.
429 Too Many Requests	`tooManyRequests`	The source IP is temporarily blocked after too many failed authentication attempts.

6.3 Set single value

```
PUT /api/values/<uid>
```

Request

The value of the specified data point.

```
{ "value": "<actual value>" }
```

Response codes

HTTP Code	Error code	Description
200 OK		Value was set.
400 Bad Request	`missingContent`	`value` missing or not a primitive.
400 Bad Request	`operationNotSupported`	The UID does not refer to a writable data point.
401 Unauthorized	`invalidToken`	Missing or invalid token.
401 Unauthorized	`invalidAuth`	The UID is not visible to the user — also returned for unknown or malformed UIDs.
404 Not Found	`uidNotFound`	Edge case: the UID is visible but has no data-point mapping.
423 Locked	`locked`	The device is currently locked.
429 Too Many Requests	`tooManyRequests`	The source IP is temporarily blocked after too many failed authentication attempts.
500 Internal Server Error	`generic`	The device could not set the value.

7 Reporting

A generic reporting interface that enables clients to report errors, warnings and general messages so they can be written to the device log.

7.1 Send a message to the device

```
POST /api/messages
```

Request

```
{
  "messageType": "<syslog compliant severity level>",
  "message": "<message text>",
  "timestamp": "<ISO 8601 timestamp>",
  "payload": "<payload object, optional>"
}
```

- `messageType` — a syslog compliant severity level.
- `message` — a message in English.
- `timestamp` — an ISO 8601 representation of date and time (e.g. `2012-04-23T18:25:43.511Z`).
- `payload` — any JSON object with further information about the message.

The body must be valid JSON. In the current implementation the structure above is not enforced — the body is written to the device log together with the user and source IP address.

Response codes

HTTP Code	Error code	Description
200 OK		Message accepted.
400 Bad Request	`missingContent`	Body is empty/invalid.
401 Unauthorized	`invalidToken`	Missing or invalid token.
423 Locked	`locked`	The device is currently locked.
429 Too Many Requests	`tooManyRequests`	The source IP is temporarily blocked after too many failed authentication attempts.

8 Licenses

8.1 Get licenses

```
GET /api/licenses[?refresh=true]
```

Requests all available licenses on the device. If the optional parameter `refresh=true` is specified, the device will first refresh the licenses from the license server. Requires API version v2 or higher.

Response

The list of all available licenses.

```
{
  "licenses": [
    {
      "vendor": "<vendor>",
      "domain": "<domain>",
      "feature": "<feature>",
      "name": { "<ISO 639-1 language code>": "<translated name>" },
      "uuid": "<uuid>",
      "creationDate": "<creation date>",
      "validTo": "<optional expiry date>",
      "target": "<optional target>",
      "configurationHint": "<optional configuration hint>"
    }
  ]
}
```

Field	Description
`vendor`	The vendor of the licensed feature.
`domain`	The domain of the licensed feature.
`feature`	The licensed feature.
`name`	The optional name of the license as a list of translated entries with an ISO 639-1 language code as key.
`uuid`	The UUID of the license.
`creationDate`	The creation date of the license in ISO 8601 format `YYYY-MM-DDThh:mm:ssZ`, for example `2019-05-22T11:16:46Z`.
`validTo`	The optional expiry date of the license in ISO 8601 format `YYYY-MM-DD`, for example `2025-12-31`.
`target`	The optional target for the license.
`configurationHint`	The optional specific configuration information for the feature.

Response codes

HTTP Code	Error code	Description
200 OK		Licenses returned (empty list if none).
401 Unauthorized	`invalidToken`	Missing or invalid token.
423 Locked	`locked`	The device is currently locked.
429 Too Many Requests	`tooManyRequests`	The source IP is temporarily blocked after too many failed authentication attempts.
500 Internal Server Error	`refreshLicensesFailed`	`refresh=true` was requested and the refresh failed or timed out.

9 Examples

Sample GPA project of an X1 with two dimmer functions and one Sonos-Audio function. IP address of the X1: `192.168.137.173`. Device credentials: username `device`, password `device`.

9.1 Registration of client

Request

```
POST /api/v3/clients HTTP/1.1
Host: 192.168.137.173
Content-Type: application/json
Authorization: Basic dXNlcjpwYXNzd29yZA==

{"client":"de.example.myapp"}
```

Response

```
{ "token": "wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD" }
```

9.2 API availability check

Request

```
GET /api/v3/ HTTP/1.1
Host: 192.168.137.173
```

Response

```
{
  "info": "GDS-REST-API",
  "version": "3",
  "deviceName": "My X1",
  "deviceType": "GIGSRVKX02",
  "deviceVersion": "4.0.348.0",
  "uidLength": 4
}
```

9.3 Getting the UI config

Request (without expand parameters)

```
GET /api/v3/uiconfig?token=wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD HTTP/1.1
Host: 192.168.137.173
```

Truncated response

```
{
  "functions": [
    {
      "channelType": "de.gira.schema.channels.KNX.Dimmer",
      "dataPoints": [
        { "name": "OnOff", "uid": "a02a" },
        { "name": "Brightness", "uid": "a02b" }
      ],
      "displayName": "Lampe Links",
```

```
      "functionType": "de.gira.schema.functions.KNX.Light",
      "iconID": -1,
      "uid": "a029"
    }
  ],
  "uid": "a036"
}
```

9.4 Setting values

Setting brightness of the left lamp to 70%

```
PUT /api/v3/values/a02b?token=wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD HTTP/1.1
Host: 192.168.137.173
Content-Type: application/json

{"value":70}
```

Setting brightness of both lamps to 20%/30%

```
PUT /api/v3/values?token=wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD HTTP/1.1
Host: 192.168.137.173
Content-Type: application/json

{ "values": [ { "uid": "a02b", "value": 20 }, { "uid": "a02e", "value": 30 } ] }
```

9.5 Getting values

Request

```
GET /api/v3/values/a02b?token=wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD HTTP/1.1
Host: 192.168.137.173
```

Response

```
{ "values": [ { "uid": "a02b", "value": "20" } ] }
```

9.6 Getting licenses

Request

```
GET /api/v3/licenses HTTP/1.1
Host: 192.168.137.173
Authorization: Bearer wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD
```

Response

```
{
  "licenses": [
    {
      "creationDate": "2019-11-04T11:48:47Z",
      "domain": "test_licenses",
      "feature": "new_test_license",
      "name": { "de": "neue lizenz", "en": "new license" },
```

```
      "uuid": "5cd92c49-8686-4c54-8b9a-b6eb62f35795",
      "vendor": "gira_de"
    }
  ]
}
```

9.7 Callbacks

Callback server at `192.168.137.128:5523`, receiving service callbacks at `/service` and value callbacks at `/value`.

Registering callbacks

```
POST /api/v3/clients/wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD/callbacks HTTP/1.1
Host: 192.168.137.173
Content-Type: application/json

{
  "serviceCallback": "https://192.168.137.128:5523/service",
  "valueCallback": "https://192.168.137.128:5523/value",
  "testCallbacks": true
}
```

Callback test received on the callback server

```
POST /service HTTP/1.1
Host: 192.168.137.128:5523
Content-Type: application/json

{ "events": [ { "event": "test" } ], "token": "wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD" }
```

```
POST /value HTTP/1.1
Host: 192.168.137.128:5523
Content-Type: application/json

{ "events": [], "token": "wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD" }
```

Value callback when the left lamp is set to 70%

```
POST /value HTTP/1.1
Host: 192.168.137.128:5523
Content-Type: application/json

{ "events": [ { "uid": "a02b", "value": "70" } ], "failures": 0, "token":
"wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD" }
```

Service callbacks on restart (with one previously missed event)

```
POST /service HTTP/1.1
Host: 192.168.137.128:5523
Connection: close
Content-Type: application/json
```

```
{ "events": [ { "event": "restart" } ], "failures": 1, "token":
"wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD" }
```

```
POST /service HTTP/1.1
Host: 192.168.137.128:5523
Content-Type: application/json

{ "events": [ { "event": "startup" } ], "failures": 0, "token":
"wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD" }
```

10 Supported Functions

All functions available in the GPA are also supported via the Gira IoT REST API. Functions that are added in newer firmware versions of the Gira server device will automatically be available in the Gira IoT REST API, even when the API's version does not change.

The tables below are generated from the function, channel and data-point definitions shipped in the device firmware (**Gira X1 4.0** and **Gira One Server 4.0**) and reflect what each device actually exposes. HomeServer / FacilityServer support API v2 only — see the v2 documentation for their function set.

The channel types `Switch2`, `Blind2`, `DimmerRGBW2` and `Trigger2`, and the function type `de.gira.schema.functions.Camera2`, were introduced with Gira X1 4.0 and Gira One Server 4.0. A device running earlier firmware does not expose them; a client that needs them should check the device version reported by the availability check (see section 3).

A trailing `2` is not a general marker of this. In data point names it is an instance index, not a version: the `GiraOne.WeatherStationVisu` channel has `Sun-Protection-1` through `Sun-Protection-4`.

10.1 Function definitions

Function name	Function type URN	Channel type(s)	Description	X1	Gira One
Switched light	de.gira.schema.functions.Switch	Switch, Switch2	Toggle a data point between 0 and 1.	X	X
Dimmer	de.gira.schema.functions.KNX.Light	KNX.Dimmer	Control a dimmer with optional shift and/or brightness.	X	X
Colored light	de.gira.schema.functions.ColoredLight	DimmerRGBW, DimmerRGBW2	Set an RGB(W) colour with optional brightness.	X	X
Tunable white	de.gira.schema.functions.TunableLight	DimmerWhite	Set a colour temperature with optional brightness.	X	X
Philips Hue light	de.gira.schema.functions.Hue.Light	Hue.Light	Control a Philips Hue light.	X	X
Shutter and blind	de.gira.schema.functions.Covering	BlindWithPos, Blind2	Move shutter/blind; optional absolute and/or slat position.	X	X
Heating and cooling	de.gira.schema.functions.KNX.HeatingCooling	KNX.HeatingCoolingSwitchable	Control heating/cooling.	X	X
Heating and cooling (relative)	de.gira.schema.functions.KNX.HeatingCoolingRelative	KNX.HeatingCoolingRelative	Control heating/cooling with a relative set-point.	X	

Function name	Function type URN	Channel type(s)	Description	X1	Gira One
KNX fan coil / air conditioning	de.gira.schema.functions.KNX.FanCoil	KNX.FanCoil	Control air conditioning / fan coil.	X	
Sauna temperature	de.gira.schema.functions.SaunaHeating	RoomTemperatureSwitchable	Set the sauna temperature, optionally with on/off.	X	
Weather station	de.gira.schema.functions.KNX.WeatherInformation	KNX.WeatherInformation	Show weather-station and climate values.	X	X
Threshold setting	de.gira.schema.functions.ThresholdSetting	ThresholdSetting	Configure a threshold / limit value.	X	X
Trigger on/off	de.gira.schema.functions.Trigger	Trigger, Trigger2	Set a data point to a predefined value 0 or 1.	X	X
Press and hold	de.gira.schema.functions.PressAndHold	Trigger, Trigger2	Set 0/1 on press and 1/0 on release.	X	
IoT trigger	de.gira.schema.functions.iot.Trigger	iot.Trigger	Trigger an IoT / automation action.	X	
Scene	de.gira.schema.functions.FunctionScene	FunctionScene	Execute/learn the scenes of a function.	X	X

Function name	Function type URN	Channel type(s)	Description	X1	Gira One
Scene (legacy)	de.gira.schema.functions.Scene	SceneSet, SceneControl	Legacy scene execute/learn (superseded by Function Scene).	X	
Audio	de.gira.schema.functions.Audio	AudioWithPlaylist, AudioWithCover	Control an audio player.	X	X
Sonos audio	de.gira.schema.functions.Sonos.Audio	Sonos.Audio	Control a Sonos audio player.	X	X
IP camera	de.gira.schema.functions.Camera2	Camera	Show an IP camera.	X	X
ONVIF camera	de.gira.schema.functions.Onvif.Camera	Onvif.Camera	Show an ONVIF-compliant IP camera.	X	X
IP camera (legacy)	de.gira.schema.functions.Camera	Camera	Legacy IP camera (superseded by IP camera).	X	
Link	de.gira.schema.functions.LinkOverlay	Link	Show a website / link overlay.	X	X
Link (legacy)	de.gira.schema.functions.Link	Link	Legacy website link (superseded by Link).	X	
Binary status value	de.gira.schema.functions.BinaryStatus	Binary	Show a binary value.	X	X

Function name	Function type URN	Channel type(s)	Description	X1	Gira One
Unsigned status value	de.gira.schema.functions.NumericUnsignedStatus	DWord	Show an unsigned value.	X	
Signed status value	de.gira.schema.functions.NumericSignedStatus	Integer	Show a signed value.	X	
Decimal status value	de.gira.schema.functions.NumericFloatStatus	Float	Show a decimal value.	X	X
Text status value	de.gira.schema.functions.TextStatus	String	Show a text value.	X	
Multi-status	de.gira.schema.functions.multi.Status	multi.Status	Show a combined multi-status value.	X	
Unsigned value transmitter (8-bit)	de.gira.schema.functions.Unsigned8BitValue	Byte	Set an unsigned 8-bit value.	X	
Signed value transmitter (8-bit)	de.gira.schema.functions.Signed8BitValue	Integer	Set a signed 8-bit value.	X	
Percent value transmitter	de.gira.schema.functions.PercentValue	Percent	Set a percent value.	X	
Temperature value transmitter	de.gira.schema.functions.TemperatureValue	Temperature	Set a temperature value.	X	

Function name	Function type URN	Channel type(s)	Description	X1	Gira One
Unsigned value transmitter	de.gira.schema.functions.UnsignedValue	DWord	Set an unsigned value.	X	
Signed value transmitter	de.gira.schema.functions.SignedValue	Integer	Set a signed value.	X	
Decimal value transmitter	de.gira.schema.functions.DecimalValue	Float	Set a decimal value.	X	
Remote access	de.gira.schema.functions.RA.RemoteAccess	RA.RemoteAccess	Gira One remote-access function.		X

On the X1 the classic `Scene`, `Camera` and `Link` function types remain available. On Gira One the equivalent capabilities are provided by **Function Scene**, **IP camera** (`Camera2`) / **ONVIF camera**, and **Link** (`LinkOverlay`).

10.2 Channel definitions

M/O: whether the data point is Mandatory (always present) or Optional. **R/W/E:** whether the data point supports Reading / Writing / Eventing (letter shown if supported, - otherwise). When a data point supports eventing, its changes are posted to registered value callbacks (see section 4.7). **Type** is the data-point value type from the definition. The **X1** / **Gira One** columns mark on which device the channel is available.

Channel type URN	Data point name	Type	M/O	R/W/E	X1	Gira One
AudioWithCover	Play	Bool	M	RWE	X	X
	Volume	Percent	M	RWE		
	Mute	Bool	O	RWE		
	Previous	Trigger	O	-WE		
	Next	Trigger	O	-WE		
	Title	String	O	R-E		
	Album	String	O	R-E		
	Artist	String	O	R-E		
	Playlist	Byte	O	RWE		
	PreviousPlaylist	Trigger	O	-WE		
	NextPlaylist	Trigger	O	-WE		
	PlaylistName	String	O	R-E		
	Shuffle	Bool	O	RWE		
	Repeat	Bool	O	RWE		
	Cover	Uri	O	R-E		
AudioWithPlaylist	Play	Bool	M	RWE	X	X
	Volume	Percent	M	RWE		
	Mute	Bool	O	RWE		
	Previous	Trigger	O	-WE		
	Next	Trigger	O	-WE		
	Title	String	O	R-E		
	Album	String	O	R-E		

Channel type URN	Data point name	Type	M/O	R/W/E	X1	Gira One
	Artist	String	O	R-E		
	Playlist	Byte	O	RWE		
	PreviousPlaylist	Trigger	O	-WE		
	NextPlaylist	Trigger	O	-WE		
	PlaylistName	String	O	R-E		
	Shuffle	Bool	O	RWE		
	Repeat	Bool	O	RWE		
Binary	Binary	Binary	M	RWE	X	X
Blind2	Step-Up-Down	UpDown	M	-W-	X	
	Up-Down	UpDown	M	-W-		
	Movement	Binary	O	R-E		
	Position	Percent	O	RWE		
	Slat-Position	Percent	O	RWE		
	Upper-Endstop	Bool	O	R-E		
	Lower-Endstop	Bool	O	R-E		
	Status	DWord	O	R-E		
	Lock	Switch	O	RWE		
BlindWithPos	Step-Up-Down	UpDown	M	-W-	X	X
	Up-Down	UpDown	M	-W-		
	Movement	Binary	O	R-E		
	Position	Percent	O	RWE		
	Slat-Position	Percent	O	RWE		

Channel type URN	Data point name	Type	M/O	R/W/E	X1	Gira One
Byte	Byte	Byte	M	RWE	X	
Camera	Camera	Binary	O	RWE	X	X
DWord	DWord	DWord	M	RWE	X	
DimmerRGBW	OnOff	Switch	M	RWE	X	
	Brightness	Percent	O	RWE		
	Red	Percent	M	RWE		
	Green	Percent	M	RWE		
	Blue	Percent	M	RWE		
	White	Percent	O	RWE		
DimmerRGBW2	OnOff	Switch	M	RWE	X	X
	Brightness	Percent	O	RWE		
	Red	Percent	M	RWE		
	Green	Percent	M	RWE		
	Blue	Percent	M	RWE		
	White	Percent	O	RWE		
	RGB	DWord	O	RWE		
	RGBW	QWord	O	RWE		
DimmerWhite	OnOff	Switch	M	RWE	X	X
	Brightness	Percent	O	RWE		
	Color-Temperature	Temperature	M	RWE		
Float	Float	Float	M	RWE	X	X
FunctionScene	Execute	Integer	M	-WE	X	X

Channel type URN	Data point name	Type	M/O	R/W/E	X1	Gira One
	Teach	Integer	O	-WE		
GiraOne.WeatherStationVisu	Wind-Speed	Float	O	R-E		X
	Brightness	Lux	O	R-E		
	Day-Night	Bool	O	R-E		
	Automatic-Day-Night-Operation	Bool	O	R-E		
	Outdoor-Temperature	Temperature	O	R-E		
	Rainfall	Bool	O	R-E		
	Sun-Protection-1	Bool	O	R-E		
	Sun-Protection-2	Bool	O	R-E		
	Sun-Protection-3	Bool	O	R-E		
	Sun-Protection-4	Bool	O	R-E		
	Wind-Alarm-1	Bool	O	R-E		
	Wind-Alarm-2	Bool	O	R-E		
	Rain-Alarm	Bool	O	R-E		
	Frost-Alarm	Bool	O	R-E		
	Fault	Bool	O	R-E		
Hue.Light	OnOff	Switch	M	RWE	X	X
	Shift	Percentage	O	-W-		
	Brightness	Percent	O	RWE		
	ColorTemperature	Temperature	O	RWE		
	ShiftColorTemperature	Percentage	O	-W-		
	RelativeBrightnessRelativeColorTemperature	DWord	O	-W-		

Channel type URN	Data point name	Type	M/O	R/W/E	X1	Gira One
	xyY	QWord	O	RWE		
	RGB	DWord	O	RWE		
Integer	Integer	Integer	M	RWE	X	
KNX.Dimmer	OnOff	Switch	M	RWE	X	X
	Shift	Percentage	O	-W-		
	Brightness	Percent	O	RWE		
KNX.FanCoil	Temperature	Temperature	M	RWE	X	
	Set-Point	Temperature	M	RWE		
	OnOff	Switch	M	RWE		
	Mode	Byte	M	RWE		
	Fan-Speed	Byte	O	RWE		
	Vanes-UpDown-Level	Byte	O	RWE		
	Vanes-UpDown-StopMove	Switch	O	RWE		
	Vanes-LeftRight-Level	Byte	O	RWE		
	Vanes-LeftRight-StopMove	Switch	O	RWE		
	Error	Bool	O	R-E		
	Error-Text	Iso88591	O	R-E		
KNX.HeatingCoolingRelative	Set-Point-Shift	Temperature	M	RWE	X	
	Set-Point-Status	Temperature	M	RWE		
	Current	Temperature	M	R-E		
	Mode	Byte	O	-WE		
	Status	Byte	O	R-E		

Channel type URN	Data point name	Type	M/O	R/W/E	X1	Gira One
	Presence	Binary	O	RWE		
	Heating	Binary	O	R-E		
	Cooling	Binary	O	R-E		
	Heat-Cool	Bool	O	RWE		
	OnOff	Bool	O	RWE		
KNX.HeatingCoolingSwitchable	Temperature	Temperature	M	RWE	X	X
	Set-Point	Temperature	M	RWE		
	Mode	Byte	O	-WE		
	Status	Byte	O	R-E		
	Presence	Binary	O	RWE		
	Heating	Binary	O	R-E		
	Cooling	Binary	O	R-E		
	Heat-Cool	Bool	O	RWE		
	OnOff	Bool	O	RWE		
KNX.WeatherInformation	Wind-Speed	Float	O	R-E	X	
	Wind-Direction	Float	O	R-E		
	Brightness	Lux	O	R-E		
	Day-Night	Bool	O	R-E		
	Automatic-Day-Night-Operation	Bool	O	R-E		
	Outdoor-Temperature	Temperature	O	R-E		
	Indoor-Temperature	Temperature	O	R-E		
	Outdoor-Humidity	Percent	O	R-E		

Channel type URN	Data point name	Type	M/O	R/W/E	X1	Gira One
	Indoor-Humidity	Percent	O	R-E		
	Air-Pressure	Float	O	R-E		
	Rainfall	Bool	O	R-E		
	Sun-Protection	Bool	O	R-E		
	Wind-Alarm	Bool	O	R-E		
	Rain-Alarm	Bool	O	R-E		
	Frost-Alarm	Bool	O	R-E		
	Fault	Bool	O	R-E		
Link	Link	Binary	O	RWE	X	X
Onvif.Camera	Json	Json	O	-WE	X	X
	Execute	Json	O	-WE		
	Status	Json	O	R-E		
	CameraInfo	Json	O	R-E		
Percent	Percent	Percent	M	RWE	X	
RA.RemoteAccess	Enabled	Bool	M	RWE		X
	User-Access	Bool	M	RWE		
	Installer-Access	Bool	M	RWE		
	Connection-State	Bool	O	R-E		
	Access-State	Bool	O	R-E		
	User-Access-State	Bool	O	R-E		
	Installer-Access-State	Bool	O	R-E		
	Error	Bool	O	R-E		

Channel type URN	Data point name	Type	M/O	R/W/E	X1	Gira One
	Connection-Info	String	O	R-E		
	Error-Info	String	O	R-E		
RestClient.BlindWithPos	Step-Up-Down	UpDown	M	-W-		X
	Up-Down	UpDown	M	-W-		
	Movement	Binary	O	R-E		
	Position	Percent	O	RWE		
	Slat-Position	Percent	O	RWE		
RestClient.KNX.Dimmer	OnOff	Switch	M	RWE		X
	Shift	Percentage	O	-W-		
	Brightness	Percent	O	RWE		
RestClient.Switch	OnOff	Switch	M	RWE		X
RoomTemperatureSwitchable	Temperature	Temperature	M	RWE	X	
	Set-Point	Temperature	M	RWE		
	OnOff	Switch	O	RWE		
SceneControl	Scene	Scene	M	-W-	X	
SceneSet	Execute	Integer	M	-WE	X	
	Teach	Integer	O	-WE		
Sonos.Audio	Play	Bool	M	RWE	X	X
	Volume	Percent	M	RWE		
	Mute	Bool	O	RWE		
	Previous	Trigger	O	-WE		
	Next	Trigger	O	-WE		

Channel type URN	Data point name	Type	M/O	R/W/E	X1	Gira One
	Title	String	O	R-E		
	Album	String	O	R-E		
	Artist	String	O	R-E		
	Playlist	Byte	O	RWE		
	PreviousPlaylist	Trigger	O	-WE		
	NextPlaylist	Trigger	O	-WE		
	PlaylistName	String	O	R-E		
	Shuffle	Bool	O	RWE		
	Repeat	Bool	O	RWE		
	Shift-Volume	Percentage	O	-W-		
	Playlists	Json	O	RWE		
	Cover	Uri	O	R-E		
	ValidPlayModes	String	O	R-E		
	TransportActions	String	O	R-E		
	ZoneName	String	O	R-E		
String	String	String	M	RWE	X	
Switch	OnOff	Switch	M	RWE	X	X
Switch2	OnOff	Switch	M	RWE	X	
	Status	DWord	O	R-E		
	Lock	Switch	O	RWE		
Temperature	Temperature	Temperature	M	RWE	X	
ThresholdSetting	Automatic	Bool	M	RWE	X	X

Channel type URN	Data point name	Type	M/O	R/W/E	X1	Gira One
	Threshold	Float	M	RWE		
	Sensor-Value	Float	O	R-E		
Trigger	Trigger	Trigger	M	-WE	X	X
Trigger2	Trigger	Trigger	M	-WE	X	
	Status	DWord	O	R-E		
	Lock	Switch	O	RWE		
iot.Trigger	Trigger	Trigger	M	-WE	X	
multi.Status	Visu-Data	Json	M	RWE	X	