GIRA

Gira IoT REST API Documentation

Stand: 01.07.2020

Version: v2

Table of contents

1 About Gira IoT REST API 3
2 Basic Usage 3
2.1 Unique Identifier 5
2.2 Versioning via URL path 5
2.3 Authorization 5
3 API availability check 6
4 Registration 7
4.1 Client Identifier 7
4.2 Register client 7
4.3 Unregister client 8
4.4 Register callbacks 8
4.5 Remove callbacks 9
4.6 Service callback 9
4.7 Value callback 10
5 UI configuration 11
5.1 UI configuration identifier 11
5.2 Get configuration 12
6 Values 14
6.1 Get value(s) 14
6.2 Set value(s) 15
6.3 Set single value 15
7 Licenses 16
7.1 Get licenses 16
8 Examples 17
8.1 Registration of client 17
8.2 Getting the UI config 17
8.3 Setting values 19
8.4 Getting values 20
8.5 Getting licenses 21
8.6 Callbacks 22
9 Supported Functions 25
9.1 Function definitions 25
9.2 Channel definitions 26

1 About Gira IoT REST API

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

As of July 2020, supported devices are:

Device type	Firmware version	Gira IoT REST API version
Gira X1	2.4	v2
Gira HomeServer/ Gira FacilityServer	4.10	v2

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 devices 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.
`403 Forbidden`	Request is not allowed for the user.
`404 Not Found`	Resource not found.
`405 Method not allowed`	Method not allowed for this resource.
`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

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`	422	Missing or invalid token as URL parameter.
`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 '…'.
`unknown`	500	Unknown error.

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 consist of small letters and digits starting with a letter always.

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.

By omitting the version part in the URL, the latest available version will be used.

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 header basic auth. This creates a new client on the server and returns a unique token. Subsequent API calls require this token as additional information for authorization.

This token must be provided as a query parameter in the URL:

```
GET /api/v2/resource?token=<token>
```

3 API availability check

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

```
GET /api/v2/
```

Response

```
{
  "info": "GDS-REST-API",
  "version": "2",
  "deviceName": "<display name>",
  "deviceType": "<device type identifier>",
  "deviceVersion": "<device version number>"
}
```

Field	Description
info	Always the string `GDS-REST-API`.
version	The version number of the API.
deviceName	User defined display name of the device.
deviceType	Device type identifier, for example `GIGSRVKX02` for Gira X1.
deviceVersion	Firmware version of the device with format `n.n.n.n`, for example `2.4.415.0`.

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 itself does return a client access token within a 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.

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 will stay valid until the client is unregistered, the user is deleted or the username is changed. The token will stay 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.
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>
}
```

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	Callbacks are not using HTTPS.
423 Locked	locked	The device is currently locked.

If `testCallbacks` is set to true, the callback server must respond with 200 OK.
Only one set of callback-URLs can be registered per `<access token>`.

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.
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 ‘register and test’ scenario only.
startup	The device is up and running.
restart	The device is going to restart.
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>
}
```

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.

5 UI configuration

5.1 UI configuration identifier

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

Response

Unique identifier of 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>" }
```

5.2 Get configuration

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

Response

Get complete UI configuration.

- uid - The unique UI configuration identifier.
- functionType - Function types unique resource name.
  See 9.1 Function definitions for further information.
- channelType - Channel types unique resource name.
  See 9.2 Channel definitions for further information.
- displayName - UTF-8 based display name.
- functions - A list of all functions.
- 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 - Whether the data point can be read.
  Will be returned if *dataPointFlags* present within *expand* parameter only.
- canWrite - Whether the data point can be written.
  Will be returned if *dataPointFlags* present within *expand* parameter only.
- canEvent - Whether the data point can event.
  Will be returned if *dataPointFlags* present within *expand* parameter only.
- parameters - A list of function parameters.
  Will be returned if present within *expand* parameter only.
- locations - A nested list of all locations and the contained unique function identifiers.
  Will be returned if present within *expand* parameter only.
- trades - A list of all trades and the contained unique function identifiers.
  Will be returned if present within *expand* parameter only.

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

6 Values

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>"
    },
    ...
  ]
}
```

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>"
    },
    ...
  ]
}
```

6.3 Set single value

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

Request

The value of the specified data point.

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

7 Licenses

7.1 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.

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.

8 Examples

Sample GPA project of X1 2.2 with 2 dimmer functions and one Sonos-Audio function.

IP address of the X1: 192.168.137.173

Device credentials: Username device, password device

8.1 Registration of client

Request

```
POST /api/v2/clients HTTP/1.1
Host: 192.168.137.173
Content-Type: application/json
Authorization: Basic dXNlcjpwYXNzd29yZA==
Cache-Control: no-cache

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

Response

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

8.2 Getting the UI config

Request without expand parameters

```
GET /api/v2/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",
      "uid": "a029"
    },
    {
      "channelType": "de.gira.schema.channels.KNX.Dimmer",
      "dataPoints": [
        {
          "name": "OnOff",
          "uid": "a02d"
        },
        {
          "name": "Brightness",
          "uid": "a02e"
        }
      ],
      "displayName": "Lampe Rechts",
      "functionType": "de.gira.schema.functions.KNX.Light",
      "uid": "a02c"
    },
    {
      "channelType": "de.gira.schema.channels.Sonos.Audio",
      "dataPoints": [
        {
          "name": "Play",
          "uid": "a02g"
        },
        {
          "name": "Volume",
          "uid": "a02h"
        },
        {
          "name": "Mute",
          "uid": "a02i"
        },
        ...
      ],
      "displayName": "Sonos-Audio",
      "functionType": "de.gira.schema.functions.Sonos.Audio",
      "uid": "a02f"
    }
  ],
  "uid": "a036"
}
```

8.3 Setting values

Setting brightness of left lamp to 70%

Request

```
PUT /api/v2/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%

Request

```
PUT /api/v2/values?token=wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD HTTP/1.1
Host: 192.168.137.173
Content-Type: application/json

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

8.4 Getting values

Getting brightness of the lamp

Request

```
GET /api/v2/values/a02b?token=wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD HTTP/1.1
Host: 192.168.137.173
```

Response

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

8.5 Getting licenses

Getting available licenses of the device

Request

```
GET /api/v2/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",
        "nl": "nieuwe licentie",
        "ru": "новая лицензия"
      },
      "uuid": "5cd92c49-8686-4c54-8b9a-b6eb62f35795",
      "vendor": "gira_de"
    },
    {
      "creationDate": "2019-07-05T11:31:48Z",
      "domain": "testing",
      "feature": "expired_license",
      "name": {
        "de": "abgelaufen",
        "en": "expired"
      },
      "uuid": "b56e0246-2b73-44e0-8547-24cc1ab9e01b",
      "validTo": "2010-01-01",
      "vendor": "tester"
    }
  ]
}
```

8.6 Callbacks

Callback server runs on local machine at 192.168.173.128.

Service callbacks shall be received at /service, value callbacks at /value.

Registering callbacks

Request

```
POST /api/v2/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
}
```

Callbacks received on callback server for callback test

Request

```
POST /service HTTP/1.1
Host: 192.168.137.128:5523
Content-Type: application/json
Content-Length: 72

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


POST /value HTTP/1.1
Host: 192.168.137.128:5523
Content-Type: application/json
Content-Length: 72

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

Callbacks when left lamp is set to 70%

Request

```
POST /value HTTP/1.1
Host: 192.168.137.128:5523
Content-Type: application/json
Content-Length: 96

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


POST /value HTTP/1.1
Host: 192.168.137.128:5523
Content-Type: application/json
Content-Length: 95

{
  "events": [
    {
      "uid": "a02a",
      "value": "1"
    }
  ],
  "failures": 0,
  "token": "wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD"
}


POST /value HTTP/1.1
Host: 192.168.137.128:5523
Content-Type: application/json
Content-Length: 103

{
  "events": [
    {
      "uid": "a02b",
      "value": "70.196078"
    }
  ],
  "failures": 0,
  "token": "wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD"
}
```

Callbacks of X1 restart with one previous missed event

Request

```
POST /service HTTP/1.1
Host: 192.168.137.128:5523
Connection: close
Content-Type: application/json
Content-Length: 88

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

...

POST /service HTTP/1.1
Host: 192.168.137.128:5523
Content-Type: application/json
Content-Length: 88

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

9 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.

9.1 Function definitions

Function name	Function type URN	Channel type URN	Description	X1	HS
Switched light	de.gira.schema.functions.Switch	de.gira.schema.channels.Switch	Toggle data point value between 0/1.	X	X
Dimmer	de.gira.schema.functions.KNX.Light	de.gira.schema.channels.KNX.Dimmer	Control a dimmer with optional shift and/or brightness.	X	X
Colored light	de.gira.schema.functions.ColoredLight	de.gira.schema.channels.DimmerRGBW	Set light to an RGB value with optional brightness and white values.	X	X
Tunable white	de.gira.schema.functions.TunableLight	de.gira.schema.channels.DimmerWhite	Set light to a color temperature with optional brightness value.	X	X
Shutter and blind	de.gira.schema.functions.Covering	de.gira.schema.channels.BlindWithPos	Move shutter/blind up/down, optionally set absolute position and/or slat position.	X	X
Trigger on/off	de.gira.schema.functions.Trigger	de.gira.schema.channels.Trigger	Set data point value to predefined value 0 or 1.	X	
Press and hold	de.gira.schema.functions.PressAndHold	de.gira.schema.channels.Trigger	Set data point value to 0 or 1 when pressing and 1 or 0 when releasing the button.	X	
Scene set	de.gira.schema.functions.Scene	de.gira.schema.channels.SceneSet	Execute/learn a specified scene.	X	
Scene extension	de.gira.schema.functions.Scene	de.gira.schema.channels.SceneControl	Execute a specified scene.	X	
Sauna temperature	de.gira.schema.functions.SaunaHeating	de.gira.schema.channels.RoomTemperatureSwitchable	Set the temperature of a sauna with optional on/off.	X	X
Heating and cooling	de.gira.schema.functions.KNX.HeatingCooling	de.gira.schema.channels.KNX.HeatingCoolingSwitchable	Control heating/cooling.	X	
KNX air conditioning / fan coil	de.gira.schema.functions.KNX.FanCoil	de.gira.schema.channels.KNX.FanCoil	Control air conditioning.	X	
Audio	de.gira.schema.functions.Audio	de.gira.schema.channels.AudioWithPlaylist	Control an audio player.	X	
Sonos-Audio	de.gira.schema.functions.Sonos.Audio	de.gira.schema.channels.Sonos.Audio	Control a sonos audio player.	X	
IP Camera	de.gira.schema.functions.Camera	de.gira.schema.channels.Camera	Show an IP camera. Data point value is set when the camera is being shown in UI.	X	
IP Link	de.gira.schema.functions.Link	de.gira.schema.channels.Link	Show a website. Data point value is set when website is being shown in UI.	X	
Binary status value	de.gira.schema.functions.BinaryStatus	de.gira.schema.channels.Binary	Show a binary value.	X	X
Unsigned status value	de.gira.schema.functions.NumericUnsignedStatus	de.gira.schema.channels.DWord	Show an unsigned value.	X	X
Signed status value	de.gira.schema.functions.NumericSignedStatus	de.gira.schema.channels.Integer	Show a signed value.	X	X
Decimal status value	de.gira.schema.functions.NumericFloatStatus	de.gira.schema.channels.Float	Show a decimal value.	X	X
Text status value	de.gira.schema.functions.TextStatus	de.gira.schema.channels.String	Show a text value.	X	X
Unsigned value transmitter (8 Bit)	de.gira.schema.functions.Unsigned8BitValue	de.gira.schema.channels.Byte	Set an unsigned 8-bit value.	X	X
Signed value transmitter (8 Bit)	de.gira.schema.functions.Signed8BitValue	de.gira.schema.channels.Integer	Set a signed 8-bit value.	X	X
Percent value transmitter	de.gira.schema.functions.PercentValue	de.gira.schema.channels.Percent	Set a percent value (0-100 or 0-255).	X	X
Temperature value transmitter	de.gira.schema.functions.TemperatureValue	de.gira.schema.channels.Temperature	Set a temperature value (decimal).	X	X
Unsigned value transmitter	de.gira.schema.functions.UnsignedValue	de.gira.schema.channels.DWord	Set an unsigned value.	X	X
Signed value transmitter	de.gira.schema.functions.SignedValue	de.gira.schema.channels.Integer	Set a signed value.	X	X
Decimal value transmitter	de.gira.schema.functions.DecimalValue	de.gira.schema.channels.Float	Set a decimal value.	X	X

9.2 Channel definitions

M/O: Whether the data point is Mandatory (always required) or Optional.

R/W/E: Whether the data point can support Reading/Writing/Eventing. When a data point supports “eventing”, all data point value changes are immediately posted to registered “value” callbacks in real-time. (See chapter 4.7 Value callbacks)

Channel type URN	Data point name	Type	M/O	R/W/E
de.gira.schema.channels.Switch	OnOff	Binary	M	RWE
de.gira.schema.channels.KNX.Dimmer	OnOff	Binary	M	RWE
	Shift	Percentage	O	-W-
	Brightness	Percent	O	RWE
de.gira.schema.channels.DimmerRGBW	OnOff	Binary	M	RWE
	Brightness	Percent	O	RWE
	Red	Percent	M	RWE
	Green	Percent	M	RWE
	Blue	Percent	M	RWE
	White	Percent	O	RWE
de.gira.schema.channels.DimmerWhite	OnOff	Binary	M	RWE
	Brightness	Percent	O	RWE
	Color-Temperature	Float	M	RWE
de.gira.schema.channels.BlindWithPos	Step-Up-Down	Binary	M	-W-
	Up-Down	Binary	M	-W-
	Movement	Binary	O	R-E
	Position	Percent	O	RWE
	Slat-Position	Percent	O	RWE
de.gira.schema.channels.Trigger	Trigger	Binary	M	-WE
de.gira.schema.channels.SceneSet	Execute	Integer	M	-WE
	Teach	Integer	O	-WE
de.gira.schema.channels.SceneControl	Scene	Integer	M	-W-
de.gira.schema.channels. RoomTemperatureSwitchable	Current	Float	M	R-E
	Set-Point	Float	M	RWE
	OnOff	Binary	O	RWE
de.gira.schema.channels.KNX. HeatingCoolingSwitchable	Current	Float	M	R-E
	Set-Point	Float	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	Binary	O	RWE
	OnOff	Binary	O	RWE
de.gira.schema.channels.KNX.FanCoil	Current	Float	M	R-E
	Set-Point	Float	M	RWE
	OnOff	Binary	M	RWE
	Mode	Byte	M	RWE

Channel type URN	Data point name	Type	M/O	R/W/E
	Fan-Speed	Byte	O	RWE
	Vanes-UpDown-Level	Byte	O	RWE
	Vanes-UpDown-StopMove	Binary	O	RWE
	Vanes-LeftRight-Level	Byte	O	RWE
	Vanes-LeftRight-StopMove	Binary	O	RWE
	Error	Binary	O	R-E
	Error-Text	String	O	R-E
de.gira.schema.channels.AudioWithPlaylist	Play	Binary	M	RWE
	Volume	Percent	M	RWE
	Mute	Binary	O	RWE
	Previous	Binary	O	-WE
	Next	Binary	O	-WE
	Title	String	O	R-E
	Album	String	O	R-E
	Artist	String	O	R-E
	Playlist	Byte	O	RWE
	PreviousPlaylist	Binary	O	-WE
	NextPlaylist	Binary	O	-WE
	PlaylistName	String	O	R-E
	Shuffle	Binary	O	RWE
	Repeat	Binary	O	RWE
de.gira.schema.channels.Sonos.Audio	(all of audio with playlist above)			
	Shift-Volume	Percentage	O	-W-
	Playlists	String	O	R-E
	Cover	String	O	R-E
	ValidPlayModes	String	O	R-E
	TransportActions	String	O	R-E
	ZoneName	String	O	R-E
de.gira.schema.channels.Camera	Camera	Binary	M	RWE
de.gira.schema.channels.Link	Link	Binary	M	RWE
de.gira.schema.channels.Binary	Binary	Binary	M	RWE
de.gira.schema.channels.DWord	DWord	DWord	M	RWE
de.gira.schema.channels.Integer	Integer	Integer	M	RWE
de.gira.schema.channels.Float	Float	Float	M	RWE
de.gira.schema.channels.String	String	String	M	RWE
de.gira.schema.channels.Byte	Byte	Byte	M	RWE
de.gira.schema.channels.Percent	Percent	Percent	M	RWE
de.gira.schema.channels.Temperature	Temperature	Float	M	RWE