GIRA

Gira IoT REST API Dokumentation

Stand: 01.07.2020

Version: v2

Inhaltsverzeichnis

1 Über die Gira IoT REST API

Mit der Gira IoT REST API können Sie Datenpunkte von unterstützten Geräten programmgesteuert lesen, schreiben und beobachten.

Zum Zeitpunkt Juli 2020 werden folgende Geräte unterstützt:

Gerätetyp	Firmware Version	Gira IoT REST API Version
Gira X1	2.4	v2
Gira HomeServer/ Gira FacilityServer	4.10	v2

Funktionen sind die Darstellungen der tatsächlichen Kanäle und Datenpunkte, die innerhalb des Gira Projekt Assistenten (GPA) zugewiesen werden. Die Funktionen sind Teil der UI-Konfiguration, die auch die Standorte (Gebäudestruktur) und die Gewerke beschreibt.

Um die API nutzen zu können, muss sich eine Client-Anwendung mit einer Anwendungs-ID registrieren und einen Benutzernamen und ein Kennwort für die Autorisierung des Geräts verwenden. Bei der Registrierung für die API muss HTTPS mit Basic Authentication verwendet werden.

Um Callbacks zu verwenden, muss ein separater HTTPS-Server verwendet werden, an den die REST-API die Callback Events senden kann.

Wichtig: Die HTTPS-Verbindung ist nicht vollständig vertrauenswürdig, da es technisch nicht möglich ist, dass der Server ein vertrauenswürdiges TLS-Zertifikat bereitstellt. Aus diesem Grund muss der Client, der für den Zugriff auf die Gira IoT REST API verwendet wird, die Zertifizierungsprüfung überspringen. Informationen zum Überspringen der Zertifizierungsprüfung finden Sie in der Dokumentation der Client-Software oder -Bibliothek.

Zielgruppe

Diese Dokumentation richtet sich an Personen, die bereits über folgende Kenntnisse verfügen:

- Grundlegendes Verständnis der HTTP-Kommunikation
- Grundlegendes Verständnis von JSON

2 Grundlegende Hinweise

Der gesamte Zugriff auf die Gira IoT REST API erfolgt über HTTPS. Alle Daten werden im JSON-Format gesendet und empfangen.

Anfragemethode	Beschreibung
`GET`	Ressource abrufen. Sendet `200 OK` bei Erfolg.
`PUT`	Ressource aktualisieren. Sendet `200 OK` bei Erfolg.
`POST`	Neue Ressource erstellen. Sendet `201 Created` bei Erfolg.
`DELETE`	Ressource löschen. Sendet `204 No Content` bei Erfolg.

Antwort Statuscodes:

Code	Description
`200 OK`	Anfrage war erfolgreich, die angefragte Ressource wird zurückgesendet.
`201 Created`	Anfrage war erfolgreich, die erstellte Ressource wird zurückgesendet.
`204 No Content`	Anfrage war erfolgreich. Kein Inhalt wird zurückgesendet.
`400 Bad Request`	Ungültige Anfrage.
`401 Unauthorized`	Fehlende oder ungültige Berechtigung.
`403 Forbidden`	Anfrage ist für den Nutzer unzulässig.
`404 Not Found`	Ressource nicht gefunden.
`405 Method not allowed`	Methode für diese Ressource unzulässig.
`422 Unprocessable Entity`	Anfrage kann nicht verarbeitet werden.
`423 Locked`	Gerät ist momentan gesperrt.
`429 Too Many Requests`	Zu viele Anfragen in einer bestimmten Zeit versendet.
`500 Internal Server Error`	Interner Serverfehler.

Im Fehlerfall werden zusätzliche Informationen in dem Objekt `error` zurückgegeben:

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

Das Feld `code` enthält einen von mehreren gültigen Fehlercodes, das Feld `message` enthält eine detailliertere Fehlermeldung.

Mögliche Fehlercodes:

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

Die API basiert auf einem Konzept eindeutiger Kennungen (Unique Identifiers, UID), welche kurz (vier Zeichen) und dauerhaft sind und sich nicht unmittelbar wiederholen.

Die vier Zeichen lange UID beginnt immer mit einem Buchstaben und besteht aus kleinen Buchstaben und Ziffern.

2.2 Versionierung über URL Pfad

Die Gira IoT REST API unterstützt Zugriff zu verschiedenen Versionen der API über den Pfad in der URL eines Aufrufs.

Wird der Versions-Teil der URL weggelassen, so wird automatisch die letzte verfügbare Version genommen.

2.3 Autorisierung

Alle API-Anfragen bis auf die Verfügbarkeitsprüfung erfordern eine Registrierung eines Clients mit Autorisierung durch Benutzername und Passwort.

Um einen Client zu registrieren, müssen die Anmeldeinformationen als Basic Authentication im Header angegeben werden. Dies erzeugt einen neuen Client auf dem Server und liefert einen eindeutigen Token zurück. Nachfolgende API-Aufrufe benötigen diesen Token als zusätzliche Information für die Autorisierung.

Dieser Token muss als Query-Parameter in der URL angegeben werden:

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

3 API-Verfügbarkeitsprüfung

Die Verfügbarkeit der Gira IoT REST API kann durch Zugriff auf die Basis-URL überprüft werden.

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

Antwort

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

Field	Description
info	Immer der Text `GDS-REST-API`.
version	Die Versionsnummer der API.
deviceName	Der benutzerdefinierte Anzeigename des Gerätes.
deviceType	Der Gerätetyp, zum Beispiel `GIGSRVKX02` für den Gira X1.
deviceVersion	Firmware-Version des Gerätes im Format `n.n.n.n`, zum Beispiel `2.4.415.0`.

4 Registrierung

4.1 Client Identifier

Jeder Client muss eine eindeutige Client-ID verwenden. Um die Eindeutigkeit zu gewährleisten, müssen Client-IDs URNs innerhalb der Organisation des Clients sein (z. B. de.gira.gdsrestapi.clients.my_well_known_service).

4.2 Client registrieren

Die Client-seitige Registrierung in der API ist für die Verwaltung der Clientidentität und optionaler Callback-URLs für Eventing erforderlich. Wenn der Client über Konfigurationsänderungen oder Wertänderungen informiert werden soll, können entsprechende Callback-URLs nach der Registrierung angegeben werden. Es werden nur HTTPS-basierte Callback-URLs unterstützt.

Die API selbst sendet ein Clientzugriffs-Token bei einer erfolgreichen Registrierung zurück, der für die Authentifizierung in allen nachfolgenden API-Aufrufen verwendet werden muss.

```
POST /api/clients
```

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

Antwort

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

Antwort-Codes

HTTP Code	Fehlercode	Beschreibung
201 Created		Client wurde erfolgreich registriert.
400 Bad Request	missingContent	Body ist leer/ungültig.
401 Unauthorized	invalidAuth	Fehlende oder ungültige Berechtigung.
423 Locked	locked	Das Gerät ist momentan gesperrt.

Der Token ist eine zufällig erstellte Zeichenfolge mit 32 alphanumerischen Groß- / Kleinbuchstaben (`regex [0-9A-Za-z]{32}`).
Jedes Mal, wenn eine neue Kombination aus Benutzername aus dem HTTP Basic Authorization-Header und des `<client-identifier>` verwendet wird, wird ein neuer Token generiert. Wenn Sie sich erneut mit demselben Benutzernamen und `<client-identifier>` registrieren, ohne die Registrierung des Tokens aufzuheben, erhalten Sie denselben Token.

Es können mehrere Clients gleichzeitig registriert werden und die Gültigkeit eines Tokens ist zeitlich nicht begrenzt.

Der Token bleibt so lange gültig, bis der dazugehörige Client de-registriert, der Benutzer gelöscht oder der Benutzername geändert wird. Der Token bleibt gültig, wenn das Passwort des Benutzers geändert wird. Ein Token, der durch das Wegfallen eines Benutzernamens ungültig geworden ist, wird wieder gültig, wenn der Benutzername wiederverwendet wird.

4.3 Client abmelden

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

Antwort-Codes

HTTP Code	Fehlercode	Beschreibung
204 No Content		Client wurde erfolgreich abgemeldet.
401 Unauthorized	invalidToken	Der Token kann nicht gefunden werden.
423 Locked	locked	Das Gerät ist momentan gesperrt.
500 Internal Server Error	Generic	Client entfernen fehlgeschlagen.

4.4 Callbacks registieren

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

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

Antwort-Codes

HTTP Code	Fehlercode	Beschreibung
200 OK		Callback wurde erfolgreich registriert.
400 Bad Request	missingContent	Body ist leer/ungültig.
400 Bad Request	callbackTestFailed	Callback-Test hat nicht mit 200 OK geantwortet.
401 Unauthorized	invalidToken	Der Token konnte nicht gefunden werden.
422 Unprocessable	Unprocessable	Callbacks nutzen nicht HTTPS.
423 Locked	locked	Das Gerät ist momentan gesperrt.

Wenn `testCallbacks` auf `"true"` gesetzt wird, muss der Callback-Server mit 200 OK antworten. Es kann nur ein Satz an Callback-URLs pro `<access token>` registriert werden.

4.5 Callbacks entfernen

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

Antwort-Codes

HTTP Code	Fehlercode	Beschreibung
200 OK		Callback wurde erfolgreich entfernt.
401 Unauthorized	invalidToken	Der Token konnte nicht gefunden werden.
423 Locked	locked	Das Gerät ist momentan gesperrt.
500 Internal Server Error	generic	Entfernen des Callbacks fehlgeschlagen.

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>
}
```

Antwort-Codes

- 200 OK
- 404 Not Found
    - Wenn der Client auf den Callback mit 404 Not Found antwortet, wird der Client von der API implizit abgemeldet.

Event Typen

Event	Beschreibung
test	Das Callback-Test-Event wird nur im Szenario "Registrieren und Testen" verwendet.
startup	Das Gerät ist hochgefahren und betriebsbereit.
restart	Das Gerät wird neu gestartet.
projectConfigChanged	Die Projektkonfiguration hat sich geändert (z. B. GPA Download). Möglicherweise hat sich dabei die UI-Konfiguration nicht geändert. Eine direkte Reaktion auf dieses Event nicht sinnvoll, da nach dem Auslösen dieses Events der REST-Server noch einige Sekunden gesperrt ist.
uiConfigChanged	UI Konfiguration wurde geändert.

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>
}
```

Antwort-Codes

- 200 OK
- 404 Not Found
    - Wenn der Client auf den Callback mit 404 Not Found antwortet, wird der Client von der API implizit abgemeldet.

5 UI Konfiguration

5.1 UI Konfigurations-ID

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

Antwort

Eindeutiger Identifier der aktuellen Konfiguration. Dieser Identifier ändert sich bei jeder Änderung der Konfiguration (z. B. GPA-Projekt-Download, Konfigurationsänderungen mit der Gira Smart Home App).

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

5.2 Konfiguration abrufen

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

Antwort

Die vollständige UI-Konfiguration.

- uid – Der eindeutige UI-Konfigurations Identifier.
- functionType – Eindeutiger Ressourcenname für Funktionsarten.
  Siehe 9.1 Funktionsdefinitionen für weitere Informationen.
- channelType – Eindeutiger Ressourcenname für Kanalarten.
  Siehe 9.2 Kanaldefinitionen für weitere Informationen.
- displayName – UTF-8 basierter Anzeigename.
- functions – Eine Liste aller Funktionen.
- dataPoints – Eine Liste aller verfügbarer Datenpunkte in der Funktion.
- uid – Der eindeutige Identifier des Datenpunkts.
- name – Der Logikname des Datenpunkts, basierend auf der Kanaldefinition
- canRead – Ob der Datenpunkt gelesen werden kann.
  Wird nur zurückgesendet, wenn *dataPointFlags* im *expand* Parameter vorhanden sind.
- canWrite – Ob der Datenpunkt geschrieben werden kann.
  Wird nur zurückgesendet, wenn *dataPointFlags* im *expand* Parameter vorhanden sind.
- canEvent – Ob der Datenpunkt ein Event auslösen kann.
  Wird nur zurückgesendet, wenn *dataPointFlags* im *expand* Parameter vorhanden sind.
- parameters – Eine Liste von Funktionsparametern.
  Wird nur zurückgesendet, wenn es über den *expand* Parameters angefordert wird.
- locations – Eine verschachtelte Liste aller Gebäudeelemente und der darin enthaltenen eindeutigen Funktions-Identifier.
  Wird nur zurückgesendet, wenn es über den *expand* Parameters angefordert wird.
- trades – Eine Liste aller Gewerke und der darin enthaltenen eindeutigen Funktions-Identifier.
  Wird nur zurückgesendet, wenn es über den *expand* Parameters angefordert wird.

```
{
  "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 Werte

6.1 Wert(e) abrufen

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

Die UID kann sich auf Folgendes beziehen:

- Einen Datenpunkt. In diesem Fall wird nur der Wert dieses Datenpunkts zurückgesendet.
- Eine Funktion. In diesem Fall werden alle Datenpunktwerte der Funktion zurückgesendet.

Antwort

Der aktuelle Wert der referenzierten Datenpunkte.

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

6.2 Wert(e) setzen

```
PUT /api/values
```

Anfrage

Die Werte der spezifizierten Datenpunkte.

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

6.3 Einzelnen Wert setzen

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

Anfrage

Der Wert des spezifischen Datenpunkts.

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

7 Lizenzen

7.1 Lizenzen abrufen

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

Fordert alle verfügbaren Lizenzen auf Gerät an. Wenn der optionale Parameter `refresh=true` gesetzt ist, aktualisiert das Gerät vorher die Lizenzen vom Lizenzserver.

Antwort

Die Liste aller verfügbaren Lizenzen.

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

Feld	Beschreibung
vendor	Der Hersteller des lizensierten Features.
domain	Die Domäne des lizensierten Features.
feature	Das lizensierte Feature.
name	Der optionale Name der Lizenz als eine Liste von übersetzten Einträgen mit einem ISO 639-1 Sprachcode als Schlüssel.
uuid	Die UUID der Lizenz.
creationDate	Das Erstellungsdatum der Lizenz im ISO 8601 Format `YYYY-MM-DDThh:mm:ssZ`, zum Beispiel `2019-05-22T11:16:46Z`.
validTo	Das optionale Ablaufdatum der Lizenz im ISO 8601 Format `YYYY-MM-DD`, zum Beispiel `2025-12-31`.
target	Das optionale Ziel der Lizenz.
configurationHint	Die optionale spezifische Konfiguration-Information vom Feature.

8 Beispiele

Beispiel-GPA-Projekt eines X1 2.2 mit zwei Dimmerfunktionen und einer Sonos-Audio-Funktion.

IP-Adresse des X1: 192.168.137.173

Geräte-Anmeldeinformationen: Benutzername device, Passwort device

8.1 Registrierung des Clients

Anfrage

```
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"}
```

Antwort

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

8.2 Abrufen der UI-Konfiguration

Anfrage ohne expand Parameter

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

Gekürzte Anwort


```
{
  "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 Werte setzen

Helligkeit der linken Lampe auf 70% einstellen

Anfrage

```
PUT /api/v2/values/a02b?token=wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD HTTP/1.1
Host: 192.168.137.173
Content-Type: application/json

{"value":70}
```

Helligkeit beider Lampen auf 20%/30% einstellen

Anfrage

```
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 Werte abrufen

Helligkeit der Lampe abrufen

Anfrage

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

Antwort

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

8.5 Lizenzen abrufen

Verfügbare Lizenzen vom Gerät abrufen

Anfrage

```
GET /api/v2/licenses HTTP/1.1
Host: 192.168.137.173
Authorization: Bearer wlkTNYIsYLJLsrw68Z3goYdQf6uiO4jD
```

Antwort


```
{
  "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

Ein Callback Server läuft auf einem lokalen Computer 192.168.173.128.

Der Callback Server empfängt Service Callbacks unter /service und Value Callbacks unter /value.

Callbacks registrieren

Anfrage

```
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
}
```

Auf dem Callback Server empfangene Callbacks für den Callback-Test

Anfrage

```
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 wenn die linke Lampe auf 70% eingestellt wird

Anfragen

```
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 bei einem Neustart des X1 und einem zuvor verpassten Event

Anfrage

```
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 Unterstützte Funktionen

Alle im GPA verfügbaren Funktionen werden auch über die Gira IoT REST API unterstützt. Funktionen, die in neueren Firmware-Versionen des Gira-Server-Geräts hinzugefügt wurden, stehen in der Gira IoT REST API automatisch zur Verfügung, auch wenn sich die API-Version nicht ändert.

9.1 Funktionsdefinitionen

Funktionsname	Funktionstyp URN	Kanaltyp URN	Beschreibung	X1	HS
Switched light	de.gira.schema.functions.Switch	de.gira.schema.channels.Switch	Datenpunktwert zwischen 0 und 1 schalten.	X	X
Dimmer	de.gira.schema.functions.KNX.Light	de.gira.schema.channels.KNX.Dimmer	Dimmer steuern mit optionaler relativer und/oder absoluter Helligkeit.	X	X
Colored light	de.gira.schema.functions.ColoredLight	de.gira.schema.channels.DimmerRGBW	Lampe auf einen RGB-Wert mit optionalen Helligkeits- und Weißwerten einstellen.	X	X
Tunable white	de.gira.schema.functions.TunableLight	de.gira.schema.channels.DimmerWhite	Lampe auf eine Farbtemperatur mit optionalem Helligkeitswert setzen.	X	X
Shutter and blind	de.gira.schema.functions.Covering	de.gira.schema.channels.BlindWithPos	Jalousie hoch-/runterfahren, optional Absolutposition und/oder Lamellenposition einstellen.	X	X
Trigger on/off	de.gira.schema.functions.Trigger	de.gira.schema.channels.Trigger	Datenpunktwert auf den vordefinierten Wert 0 oder 1 setzen.	X	
Press and hold	de.gira.schema.functions.PressAndHold	de.gira.schema.channels.Trigger	Datenpunktwert auf 0 oder 1 beim Drücken und 1 oder 0 beim Loslassen eines Schalters setzen.	X	
Scene set	de.gira.schema.functions.Scene	de.gira.schema.channels.SceneSet	Eine bestimmte Szene ausführen/einlernen.	X	
Scene extension	de.gira.schema.functions.Scene	de.gira.schema.channels.SceneControl	Eine bestimmte Szene ausführen.	X	
Sauna temperature	de.gira.schema.functions.SaunaHeating	de.gira.schema.channels.RoomTemperatureSwitchable	Temperatur einer Sauna einstellen, optional mit an/aus.	X	X
Heating and cooling	de.gira.schema.functions.KNX.HeatingCooling	de.gira.schema.channels.KNX.HeatingCoolingSwitchable	Heizen/Kühlen regeln.	X	
KNX air conditioning / fan coil	de.gira.schema.functions.KNX.FanCoil	de.gira.schema.channels.KNX.FanCoil	Klimatisierung regeln.	X	
Audio	de.gira.schema.functions.Audio	de.gira.schema.channels.AudioWithPlaylist	Audio-Player regeln.	X	
Sonos-Audio	de.gira.schema.functions.Sonos.Audio	de.gira.schema.channels.Sonos.Audio	Sonos Speaker regeln.	X	
IP Camera	de.gira.schema.functions.Camera	de.gira.schema.channels.Camera	Bild einer IP-Kamera anzeigen. Datenpunktwert wird gesetzt, wenn die Kamera im UI angezeigt wird.	X	
IP Link	de.gira.schema.functions.Link	de.gira.schema.channels.Link	Webseite anzeigen. Datenpunktwert wird gesetzt, wenn die Webseite im UI angezeigt wird.	X	
Binary status value	de.gira.schema.functions.BinaryStatus	de.gira.schema.channels.Binary	Binärwert anzeigen.	X	X
Unsigned status value	de.gira.schema.functions.NumericUnsignedStatus	de.gira.schema.channels.DWord	Vorzeichenlosen Zahlenwert anzeigen.	X	X
Signed status value	de.gira.schema.functions.NumericSignedStatus	de.gira.schema.channels.Integer	Vorzeichenbehafteten Zahlenwert anzeigen.	X	X
Decimal status value	de.gira.schema.functions.NumericFloatStatus	de.gira.schema.channels.Float	Dezimalwert anzeigen.	X	X
Text status value	de.gira.schema.functions.TextStatus	de.gira.schema.channels.String	Text anzeigen.	X	X
Unsigned value transmitter (8 Bit)	de.gira.schema.functions.Unsigned8BitValue	de.gira.schema.channels.Byte	8-Bit Wert ohne Vorzeichen setzen.	X	X
Signed value transmitter (8 Bit)	de.gira.schema.functions.Signed8BitValue	de.gira.schema.channels.Integer	8-Bit Wert mit Vorzeichen setzen.	X	X
Percent value transmitter	de.gira.schema.functions.PercentValue	de.gira.schema.channels.Percent	Prozentwert setzen (0-100 oder 0-255).	X	X
Temperature value transmitter	de.gira.schema.functions.TemperatureValue	de.gira.schema.channels.Temperature	Temperaturwert setzen (dezimal).	X	X
Unsigned value transmitter	de.gira.schema.functions.UnsignedValue	de.gira.schema.channels.DWord	Zahlenwert ohne Vorzeichen setzen.	X	X
Signed value transmitter	de.gira.schema.functions.SignedValue	de.gira.schema.channels.Integer	Zahlenwert mit Vorzeichen setzen.	X	X
Decimal value transmitter	de.gira.schema.functions.DecimalValue	de.gira.schema.channels.Float	Dezimalwert setzen.	X	X

9.2 Kanaldefinitionen

M/O: Gibt an, ob der Datenpunkt zwingend (Mandatory/immer erforderlich) oder optional ist.

R/W/E: Gibt an, ob der Datenpunkt Lesen/Schreiben/Eventing unterstützt (Reading/Writing/Eventing). Wenn ein Datenpunkt “Eventing” unterstützt, werden Werteänderungen von Datenpunkten sofort an alle registrierte „value“-Callbacks in Echtzeit gesendet (siehe Kapitel 4.7 Value Callback).

Kanaltyp URN	Datenpunktname	Typ	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

Kanaltyp URN	Datenpunktname	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