# Fabric Services System

Release 23.8.1

## Fabric Services System API Integration Guide

3HE 19800 AAAA TQZZA
Edition: 1
October 2023

Nokia is committed to diversity and inclusion. We are continuously reviewing our customer documentation and consulting with standards bodies to ensure that terminology is inclusive and aligned with the industry. Our future customer documentation will be updated accordingly.

---

# Table of contents

# 1 About this document

This document describes how use the Fabric Services System HTTP REST API.

This document is intended for network technicians, administrators, operators, service providers, and others who need to understand how to use the Fabric Services System HTTP REST API.

**Note:** This document covers the current release and may also contain some content that will be released in later maintenance loads. See the *Fabric Services System Release Notes* for information about features supported in each load.

## 1.1 What's new

This section lists the changes that were made in this release.

*Table 1: What's new in Release 23.8.1*

| Description | Location |
|---|---|
| Support for endpoint for gathering full device inventory | Displaying inventory |

## 1.2 Precautionary and information messages

The following are information symbols used in the documentation.

**DANGER:** Danger warns that the described activity or situation may result in serious personal injury or death. An electric shock hazard could exist. Before you begin work on this equipment, be aware of hazards involving electrical circuitry, be familiar with networking environments, and implement accident prevention procedures.

**WARNING:** Warning indicates that the described activity or situation may, or will, cause equipment damage, serious performance problems, or loss of data.

**Caution:** Caution indicates that the described activity or situation may reduce your component or system performance.

**Note:** Note provides additional operational information.

**Tip:** Tip provides suggestions for use or best practices.

## 1.3 Conventions

Commands use the following conventions:

- **Bold** type indicates a command that the user must enter.
- Input and output examples are displayed in `Courier` text.
- An open right angle bracket indicates a progression of menu choices or a simple command sequence, often selected from a user interface. For example: **start** > **connect to**
- Angle brackets (< >) indicate an item that is not used verbatim. For example, for the command **show ethernet <name>**, **name** should be replaced with the name of the interface.
- A vertical bar (|) indicates a mutually exclusive argument.
- Square brackets ([ ]) indicate optional elements.
- Braces ({ }) indicate a required choice. When braces are contained within square brackets, they indicate a required choice within an optional element.
- *Italic* type indicates a variable.

Examples use generic IP addresses. Replace these with the appropriate IP addresses used in your system.

# 2 Overview

The Fabric Services System includes an HTTP REST API to support software integration. By using the REST API, you can write your own software that can configure any feature of the Fabric Services System.

Detailed information about all of the individual API calls is available in the OpenAPI (v2) format, also referred to as Swagger documentation. You can download Swagger YAML files from your Fabric Services System deployment, from the same location as the Reference API documentation.

You can then use these files within your own tools that can work with the standard OpenAPI specifications.

## 2.1 Reference documentation

Within the Fabric Services System GUI you can find detailed API documentation for each of the services that exposes an API.

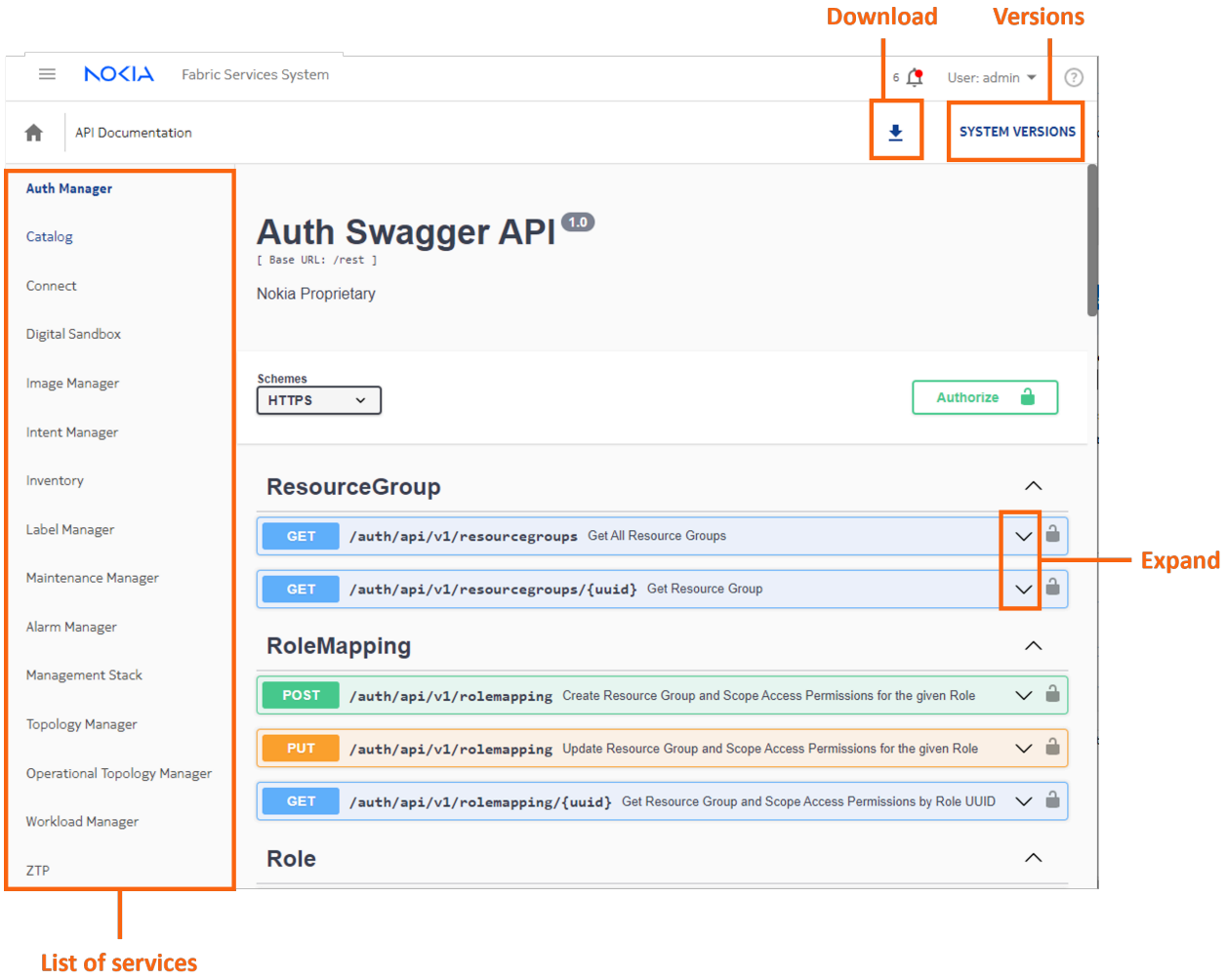
When the Fabric Services System is installed, this API GUI is available at the following URL:

```
https://fss.domain.tld/apidocs
```

where `fss.domain.tld` is your local installation FQDN

When you access the API GUI, the **API Documentation** page displays:

*Figure 1: Swagger documentation in the Fabric Services System*




The API GUI contains a list of the services on the left. Click on any of those services to display the Swagger (or OpenAPI) documentation.

Click the expand icon for an individual API call to see details for that call, including usage, parameter details, and a tool for issuing a sample call.

The download button allows you to download the Swagger YAML file for the selected service. You can use this file with any tools that are compatible with the OpenAPI standard. The Fabric Services System uses the OpenAPI v2 specifications format.

In addition to the reference API documentation, the same page in the Fabric Services System UI also provides a list of all the services deployed in the Fabric Services System application and the versions of each of these services. You can access this information by clicking **SYSTEM VERSIONS**.

**Related topics**

*Working with the Swagger UI*

## 2.2 Downloading the full OpenAPI definitions

You can download the OpenAPI definitions for all the services in a single file. This file can be downloaded in either YAML or JSON format using one of the following URLs:

- YAML: `https://fss.domain.tld/rest/about/swagger.yaml`
- JSON: `https://fss.domain.tld/rest/about/swagger.json`

## 2.3 URL endpoints

The Fabric Services System REST API URL path has the following format:

```
https://fss.domain.tld/rest/<service>/api/<version>
```

where `<version>` stands for the major release number of the API (such as 1)

**Example:**

```
POST https://fss.domain.tld/rest/intentmgr/api/v1/queueItems/
384258893435371520/deploy
```

**Related topics**

*Naming conventions and best practices*

## 2.4 Supported REST API versions

You can determine which versions of an API call are available for a specific deployment of the Fabric Services System by performing an unauthenticated HTTP GET request to the `https://fss.domain.tld/rest/about/version` URI:

```
# curl -s https://fss.domain.tld/rest/about/version | jq
{
  "fss": {
    "gitCommit": "v22.4.1-1462"
  },
  "fss-alarmmgr": {
    "gitCommit": "v22.4.1-1462-gca271333",
    "builtDate": "2022-03-23T05:24:14+00:00",
    "gitBranch": "22_4_1",
    "appName": "fss-alarmmgr",
    "apiVersions": [
      {
        "version": "v1",
        "status": "CURRENT",
        "baseURL": "fss.domain.tld/rest",
        "swaggerDoc": "fss.domain.tld/rest/alarmmgr/v1/docs/swagger.yaml"
      }
    ]
  },
  "fss-auth": {
    "gitCommit": "v22.4.1-1462-gca271333",
    "builtDate": "2022-03-23T05:15:55+00:00",
    "gitBranch": "22_4_1",
```

```
    "appName": "fss-auth",
    "apiVersions": [
      {
        "version": "v1",
        "status": "CURRENT",
        "baseURL": "fss.domain.tld/rest",
        "swaggerDoc": "fss.domain.tld/rest/auth/v1/docs/swagger.yaml"
      }
    ]
  },
...
```

All services and components of the Fabric Services System display in the resulting list. Only those that have an `apiVersions` attribute actually provide an API service.

### API version strategy and backward compatibility

The Fabric Services System is composed of multiple individual applications or services. Each of these services has its own API with its own API version. While most of the services use the same latest API version, it is important to keep track of API version changes.

For each service, at least two versions are always available unless a service is using version 1 for its API. The older version is available for customers upgrading from an older version to ensure that their scripts, tools, and integrations can continue functioning on the old API after the upgrade and before they are updated to the new API.

As soon as a new API version is introduced for that service, all new functionality for the service is available only on the new API version. The older API version can only be used for the older functionality.

The platform also guarantees backward compatibility within the same version of the API. This strategy allows for the creation of scripts, tools and integrations using the same API version across multiple releases, without having to adapt them every time the platform is upgraded.

**Note:** Between versions of the API, backward compatibility can sometimes break when there is a need for significant changes. When upgrading the scripts, tools, and integration to use the new API, investigate and understand the impact of these changes and act accordingly.

## 2.5 Using the UI to inspect API calls

The Fabric Services System UI is a graphical layer that uses the API directly. You can use it to better understand the workflow and API calls that are getting executed when taking specific actions in the UI.

You can use the Web Developer tools, specifically the Network tab of the tools, to see each individual API call that is executed. You can inspect the request and response data, including the headers for each API call.

# 3 API conventions

This section describes the conventions and guidelines for best practices used in the Fabric Services System REST API.

## 3.1 HTTP headers

The Fabric Services System REST API uses HTTP headers for various purposes such as authentication. This section describes the required and supported HTTP headers.

### Request headers

The following headers are required when making HTTP requests:

- ```
  Content-Type: application/json
  ```

  The server response is formatted in JSON. `Content-Type` is a required header field for HTTP POST and PUT requests is and strongly advised for any other HTTP request.

- ```
  Authorization: $AUTHORIZATION_STRING
  ```

  Each API call must be authenticated, so the `Authorization` header must be sent with each request unless a call is documented as not authenticated. The `$AUTHORIZATION_STRING` is defined in the API Authentication section.

### Response headers

The system returns the following headers in response to HTTP requests:

- ```
  Content-Type: application/json
  ```

  The server response is formatted in JSON.

- Security Headers: the server also returns multiple security headers, including:
  - `X-Content-Type-Options: nosniff`
  - `X-Frame-Options: SAMEORIGIN`
  - `X-XSS-Protection: 1; mode=block`

## 3.2 Server response codes

The Fabric Services System server replies using standard HTTP response codes as described in RFC9110, section 15.
  ```

### 2xx codes

2xx codes indicate that the request was processed, as listed in the following table:

*Table 2: 2xx code descriptions*

| Code | Description |
|---|---|
| 200 | Everything is okay, content is in the body |
| 201 | Everything is okay, the object is created, and the location header points to the object |
| 204 | Everything is okay, but there is no content in the body<br>This is the standard reply for an update or a deletion |

### 3xx codes

3xx codes indicate that further action is needed by the client, as listed in the following table:

*Table 3: 3xx code descriptions*

| Code | Description |
|---|---|
| 300 | The request requires a choice to be made for it to succeed. Typically used when a user needs to accept or acknowledge a warning; for instance, when deleting an entity. |

### 4xx codes

4xx codes indicate that a mistake was made by the client, as listed in the following table. The body of the response indicates what went wrong with more detail.

*Table 4: 4xx code descriptions*

| Code | Description |
|---|---|
| 400 | Bad request: something is wrong in the request |
| 401 | Not authenticated: wrong password or wrong API key |
| 403 | Not authorized: request for something to which you do not have rights |
| 404 | Not found: request for something that does not exist |
| 409 | Conflict: the requested change conflicts with existing configuration |
| 412 | Precondition failed: more information is required before this request can succeed |

### 5xx codes

5xx codes indicate a problem caused by the server

*Table 5: 5xx code descriptions*

| Code | Description |
|---|---|
| 500 | Internal server error: error is described in the body |
| 501 | Not implemented: the request you made is not supported |

## 3.3 API relationships

Resources have relationships to other objects. Objects in the Fabric Services System API can have the following relationships:

- 1:*n*–one-to-many
- *n*:*m*–many-to-many
- 1:1–one-to-one

### 1:*n* relationships

A parent-to-child relationship is an example of a 1:*n* relationship. 1:*n* relationships can expressed in the URI as sub-collections, for example:

- collection: `/subnets/{id}/sub-interfaces`
- singleton: `/subnets/{id}/sub-interfaces/{id}`

### *n*:*m* relationships

In an *n*:*m* relationship, two objects that have a link between each other, but are not in a parent-to-child relationship. A membership in a group exemplifies a many-to-many relationship: a member can belong to many groups; a group has many members, such as the relationship between a user and a user group.

### 1:1 relationships

In a 1:1 relationship, a resource is related only to one other resource.

## 3.4 Naming conventions and best practices

### URL conventions

The URL path is expressed in lowercase.
`https://fss.domain.tld/rest/workloadmgr/api/v2/aclprofiles`

instead of:

`https://fss.domain.tld/rest/workloadmgr/api/v2/aclProfiles`

### Attribute naming conventions

- Attributes are expressed as nouns.
- Acronyms are all uppercase.

**Related topics**
*URL endpoints*

## 3.5 Attributes

Anything that can be managed in the API is an object. Fabric Services System REST API objects have common attributes. Some objects also have an external attribute provided by orchestrators.

### `externalID` attribute

The `externalID` attribute specifies the ID used by an orchestrator to track an object in the Fabric Services System API. Orchestrators can use the `externalID` attribute as follows:

- to correlate external IDs in the systems they orchestrate
- in audit mechanisms between orchestrators and Fabric Services System, orchestrators can identify the objects that they need to audit or track and which ones to not inspect (that is, those without an `externalID`)

The following table describes the `externalID` attribute:

*Table 6: `externalID` attribute characteristics*

| Attribute | Type | Required | Description |
|---|---|---|---|
| `externalID` | String of up to 255 characters | No | The external identifier for the object.<br>• You can filter using the `externalID` value<br>• You cannot edit the value.<br>• You cannot update with the value with an empty string. |

The following objects have the `externalID` attribute:

- region
- deployment queue item
- fabric intent
- LAG
- workload intent
- workload intent subnet
- workload intent match label (sub-interface based on label)
- workload intent sub-interface (sub-interface based on port and interface)
- label
- QoS profile
- ACL profile

## 3.6 API error format

Fabric Services System REST API error handling is implemented according to RFC 7807 *Problem Details for HTTP APIs*. Errors are reported in the following JSON format:

```
{
```

```
    "type": URI
    "title": Translated string
    "detail": String
    "status": Int
    "object_ref" optional: URI
    "errors" optional: [ErrorData]
    "additional_info" optional: String
}
```

The following table defines the fields in the error message:

*Table 7: Field definitions*

| Name | Type | Description |
|---|---|---|
| `type` | URI | Uniform resource identifier for the relative name-spaced reference to the error |
| `title` | String | Fixed string |
| `detail` | URI | Detailed string with IDs and names |
| `status` | Integer | HTTP status code |
| `object_ref` | URI | Request URL |
| `errors` | Array | List of errors (if returning with multiple errors in single request) |

The Fabric Services System REST API returns all errors in a request, not just the first one that it encounters. For example, if a POST request fails because an attribute conflicts with another object and also because a mandatory attribute is not provided, both errors are returned. If there are multiple errors, all errors are wrapped in a single predefined error and all real errors are listed in the `errors` field.

### Example: single error

```
{
    "type": "/workloadmgr/api/v1/errors?key=missing-parameter",
    "title": "MISSING_PARAMETER",
    "detail": "[lsIntentFabricType] should be set",
    "status": 400,
    "object_ref": "/workloadmgr/api/v1/intents"
}
```

### Example: multiple errors

```
{
    "type": "/workloadmgr/api/v1/errors?key=multi-error",
    "title": "MULTI_ERROR",
    "detail": "See errors",
    "status": 207,
    "object_ref": "/workloadmgr/api/v1/intents",
    "errors": [
        {
            "type": "/workloadmgr/api/v1/errors?key=missing-parameter",
            "title": "MISSING_PARAMETER",
            "detail": "[lsIntentFabricType] should be set",
            "status": 400,
            "object_ref": "/workloadmgr/api/v1/intents"
        },
        {
```

```
            "type": "/workloadmgr/api/v1/errors?key=invalid-parameter",
            "title": "INVALID_PARAMETER",
            "detail": "Invalid parameter value [intentType]",
            "status": 400,
            "object_ref": "/workloadmgr/api/v1/intents"
        },
        {
            "type": "/workloadmgr/api/v1/errors?key=missing-parameter",
            "title": "MISSING_PARAMETER",
            "detail": "[templateName] should be set",
            "status": 400,
            "object_ref": "/workloadmgr/api/v1/intents"

        }
    ]
}
```


**Related topics**

*Server response codes*

# 4 Authentication

The Fabric Services System REST API supports authentication through a bearer API token. The user provides the username and password in an initial request and the REST API returns a token that is valid for a limited period and can be used to make subsequent requests.

By using Keycloak as the authentication and authorization backend of the API, the Fabric Services System REST API follows the OAuth2 (OpenID) authentication flow, which provides the client with the required OAuth2 tokens needed to authenticate subsequent API calls.

## 4.1 Getting the API token

To obtain an API authentication token, send a `POST /rest/auth/login` request with the required request POST data. This request returns the authentication information, including the access token. You can then use this access token for the actual API requests.

The POST data has the following format:

```
{
    "username": "<username>",
    "password": "<password>"
}
```

The authentication POST request response has the following format:

```
{
    "access_token": "<access token>",
    "id_token": "<id token>",
    "session_state: "",
    "scope": "",
    "refresh_token": "<refresh token>",
    "token_type": "Bearer",
    "expires_in": 300,
    "refresh_expires_in": 1800,
    "not-before-policy": 0
}
```

The following table describes the fields in the authentication POST request.

*Table 8: Field definitions*

| Field | Format | Description |
|---|---|---|
| `access_token` | String | This field specifies the access token to use for API calls toward the REST API. |
| `expires_in` | Integer | This field specifies how long the access token is valid, in seconds. Default: 300 |

| Field | Format | Description |
|---|---|---|
| id_token | JWT | This field specifies a JSON web token, which is not necessary for the actual API requests; it contains information from the Keycloak environment. |
| not-before-policy | Integer | This field specifies a custom value that indicates a time during which all tokens are invalid (unused by the Fabric Services System REST API). |
| refresh_expires_in | Integer | This field specifies how long the refresh token is valid, in seconds. Default: 1800 |
| refresh_token | String | This field specifies contains the refresh token that you can use to refresh the existing authentication (access token) after it expires and before the refresh token expires. |
| scope | Space-separated strings | This field specifies the OAuth2/OIDC scope of the token. For new authentications, this field is expected to be empty. |
| session_state | String | This field specifies is an optional internal session identifier. For new authentications, this field is expected to be empty. |
| token_type | String | This field specifies is always set to Bearer. |

### Example: Successful authentication request and response

Authentication request:

```
$ curl -s -v -XPOST -d '{"username":"admin","password":"NokiaFss1!"}' -H "Content-Type:
 application/json; charset=utf-8" "https://fss.domain.tld/rest/auth/login"
```

Authentication response:

```
...
> POST /rest/auth/login HTTP/2
> Host: fss.domain.tld
> User-Agent: curl/7.64.1
> Accept: */*
> Content-Type: application/json; charset=utf-8
> Content-Length: 44
>
...
< HTTP/2 200
< content-type: application/json; charset=utf-8
< date: Tue, 19 Jul 2022 23:20:34 GMT
< x-content-type-options: nosniff
< x-frame-options: SAMEORIGIN
< x-xss-protection: 1; mode=block
<
{
  "access_token": "eyJhbGciOiJSUzI...PQZUPN4BVu7tCT6qy7Q",
  "id_token": "eyJhbGciOiJSUzI...JMXlQi8y0KknyCVg9FQ",
  "session_state": "",
  "scope": "",
  "refresh_token": "eyJhbGciOiJIUzI1...V4_P0sNqZRjOBQ7ego",
  "token_type": "Bearer",
  "expires_in": 300,
```

```
  "refresh_expires_in": 1800,
  "not-before-policy": 0
}
```

### Example: Unsuccessful authentication response request

Authentication request:

```
$ curl -s -v -XPOST -d '{"username":"admin","password":"wrongpassword"}' -H "Content-Type:
 application/json; charset=utf-8" "https://fss.domain.tld/rest/auth/login"
```

Authentication response:

```
...
> POST /rest/auth/login HTTP/2
> Host: fss.domain.tld
> User-Agent: curl/7.64.1
> Accept: */*
> Content-Type: application/json; charset=utf-8
> Content-Length: 47
>
...
< HTTP/2 400
< content-type: application/json; charset=utf-8
< date: Tue, 19 Jul 2022 23:25:10 GMT
< x-content-type-options: nosniff
< x-frame-options: SAMEORIGIN
< x-xss-protection: 1; mode=block
<
{
  "type": "/auth/errors?key=token-error",
  "title": "TOKEN_ERROR",
  "detail": "Failed to get access token. failed to get token error : invalid_grant",
  "object_ref": "/auth/login",
  "status": 400
}
```

## 4.2 Refreshing the authentication API token

By default, the authentication token expires every 300 seconds. Refreshing the authentication API token does not require the use of the username or password; instead, use the `refresh_token` field. When you use the `access_token` and `refresh_token` fields, a new `access_token` and a new `refresh_token` is generated, which can be used again for 5 minutes. You refresh the authentication token by creating a `POST /rest/auth/refresh` request with the required POST data and the correct `Authorization: Bearer <expired access token>` header. This request returns the authentication information, including the access token that is used for the actual API requests.

The authentication refresh POST request data has the following format:

```
{
  "refresh_token":"eyJhbGciOiJIUzI1...V4_P0sNqZRjOBQ7ego"
}
```

### Example: Successful REST API authentication refresh request and response

Authentication refresh request:

```
$ curl -s -v -XPOST -d '{"refresh_token":"eyJhbGciOiJIUzI1...V4_P0sNqZRjOBQ7ego"}'
 -H "Authorization: Bearer eyJhbGciOiJSUzI...PQZUPN4BVu7tCT6qy7Q" -H "Content-Type:
 application/json; charset=utf-8" "https://fss.domain.tld/rest/auth/refresh"
> POST /rest/auth/refresh HTTP/2
```

Authentication refresh response:

```
> Host: fss.domain.tld
> User-Agent: curl/7.64.1
> Accept: */*
> Authorization: Bearer eyJhbGciOiJSUzI...PQZUPN4BVu7tCT6qy7Q
> Content-Type: application/json; charset=utf-8
> Content-Length: 3325
>
...
< HTTP/2 200
< content-type: application/json; charset=utf-8
< date: Tue, 19 Jul 2022 23:37:22 GMT
< x-content-type-options: nosniff
< x-frame-options: SAMEORIGIN
< x-xss-protection: 1; mode=block
<
{
  "access_token": "eyJhbGciOiJSUzI...Z0NmcOQthgEM2PfgA_g",
  "id_token": "eyJhbGciOiJSUzI...ZkNi8yReIyQM0FVt0rw",
  "session_state": "f7d9f6bc-7ab7-448a-8529-53d110e7ed8d",
  "scope": "openid",
  "refresh_token": "eyJhbGciOiJIUzI...eFXnTrYDEZmi8sH9Iwo",
  "token_type": "Bearer",
  "expires_in": 300,
  "refresh_expires_in": 1800,
  "not-before-policy": 0
}
```

### Example: Unsuccessful REST API Authentication refresh request and response

Authentication refresh request:

```
$ curl -s -v -XPOST -d '{"refresh_token":"eyJhbGciOiJIUzI1...V4_P0sNqZRjOBQ7_WRONG"}'
 -H "Authorization: Bearer eyJhbGciOiJSUzI...PQZUPN4BVu7tCT6qy7Q" -H "Content-Type:
 application/json; charset=utf-8" "https://fss.domain.tld/rest/auth/refresh"
```

Authentication refresh response:

```
> POST /rest/auth/refresh HTTP/2
> Host: fss.domain.tld
> User-Agent: curl/7.64.1
> Accept: */*
> Authorization: Bearer eyJhbGciOiJSUzI...PQZUPN4BVu7tCT6qy7Q
> Content-Type: application/json; charset=utf-8
> Content-Length: 3325
>
...
< HTTP/2 400
< content-type: application/json; charset=utf-8
< date: Tue, 19 Jul 2022 23:50:58 GMT
< x-content-type-options: nosniff
```

```
< x-frame-options: SAMEORIGIN
< x-xss-protection: 1; mode=block
<
{
  "type": "/auth/errors?key=token-error",
  "title": "TOKEN_ERROR",
  "detail": "Failed to get access token. failed to get token error : invalid_grant",
  "object_ref": "/auth/refresh",
  "status": 400
}
```

## 4.3 Using the Swagger UI to authenticate

### About this task

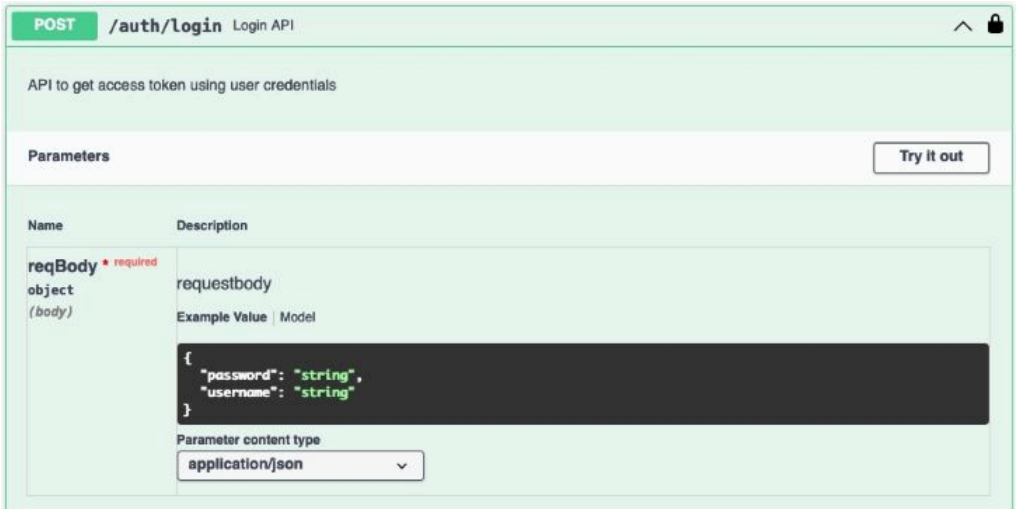
You can use the **Auth Manager** service from the Swagger UI to authenticate using the Auth service.

### Procedure

**Step 1.** From **Auth Manager**, under the **Auth**, click the **/auth/login** API request, then click **Try it out**.

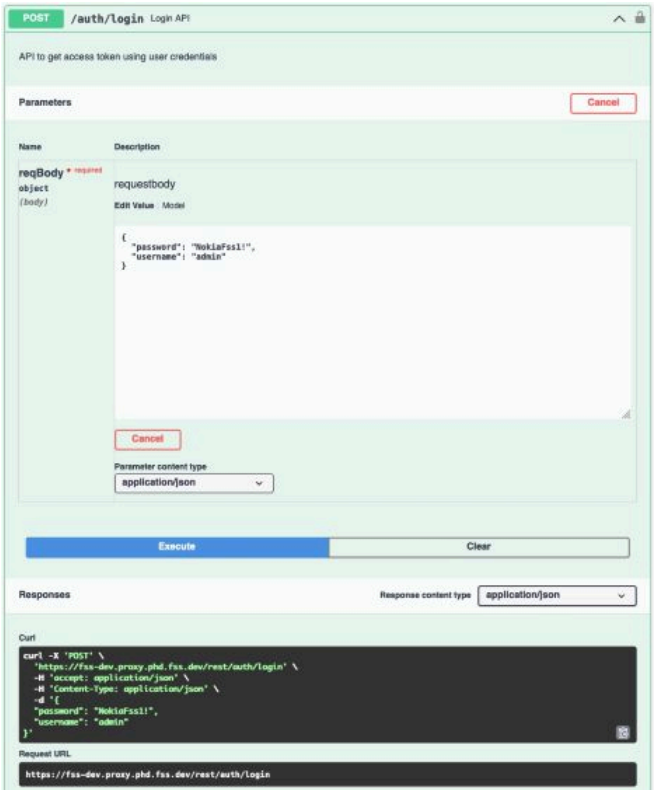
**Expected outcome**

The **password** and **username** fields become editable in the request body.




**Step 2.** Fill in the correct username and password in the fields, then click **Execute**.

**Example**



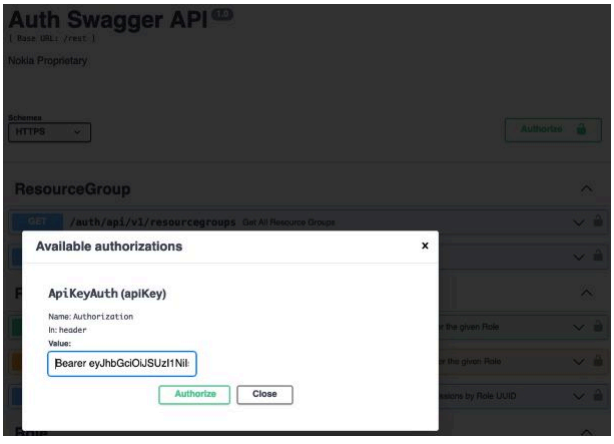

**Expected outcome**

The **Responses** section displays the access token. This token is valid for 5 minutes.

**Step 3.** Copy the access token value.

**Step 4.** Click the **Authorize** button and, in the **Available authorizations** form that displays, paste the token in the **Value** field.

**Example**



# 5 API management

This section provides an overview of the requests (methods) used in the Fabric Services System REST API and provides basic examples.

### Methods

The method defines the operation with a resource. The following methods are used in the Fabric Services System API:

| | |
|---|---|
| **GET** | Retrieves a resource or set of resources |
| **POST** | Creates a resource or set of resources |
| **PUT** | Updates or replaces an existing resource or set of resources |
| **PATCH** | Updates one or more specific properties of an existing resource or set of resources |
| **DELETE** | Removes a resource or a set of resources |

**Related topics**

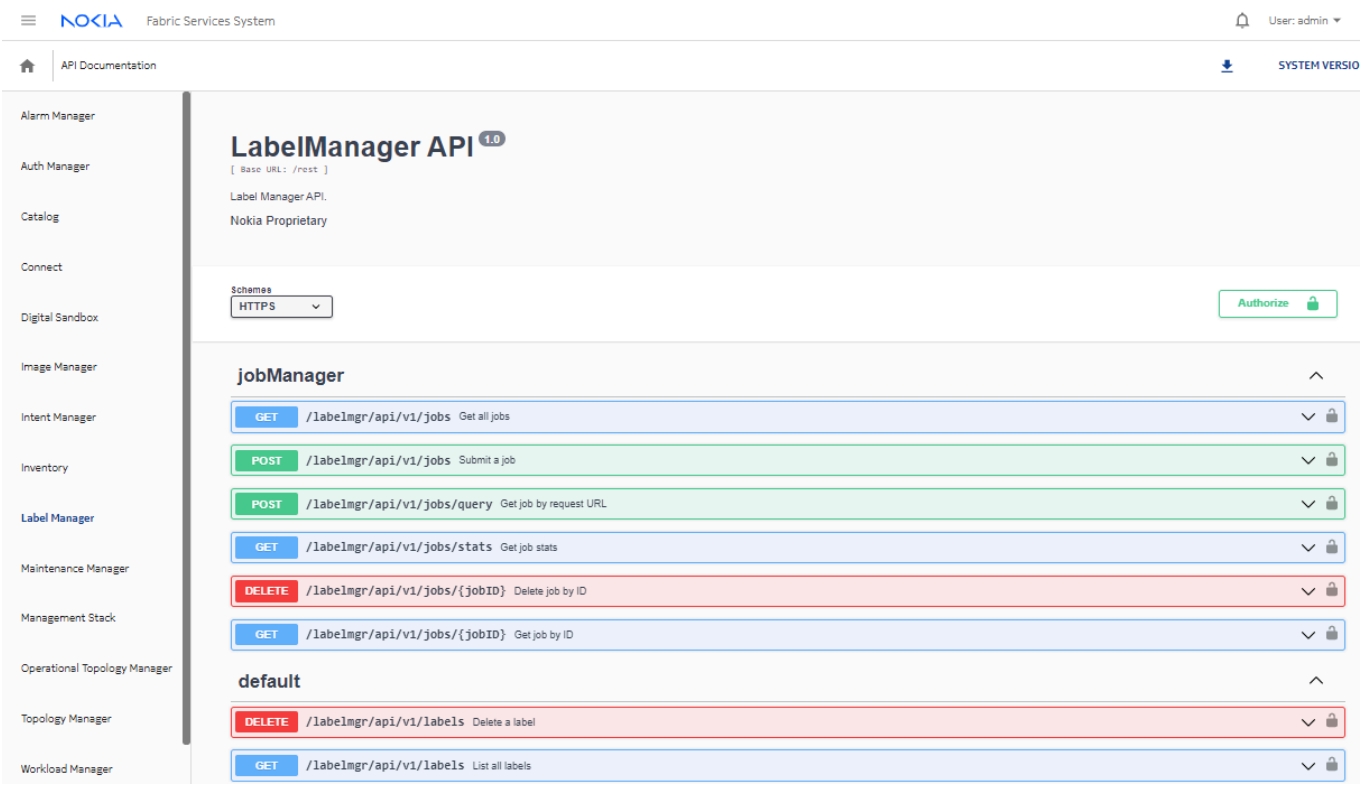
*Reference documentation*

## 5.1 Working with the Swagger UI

The Swagger UI describes each service (resource) in the API.

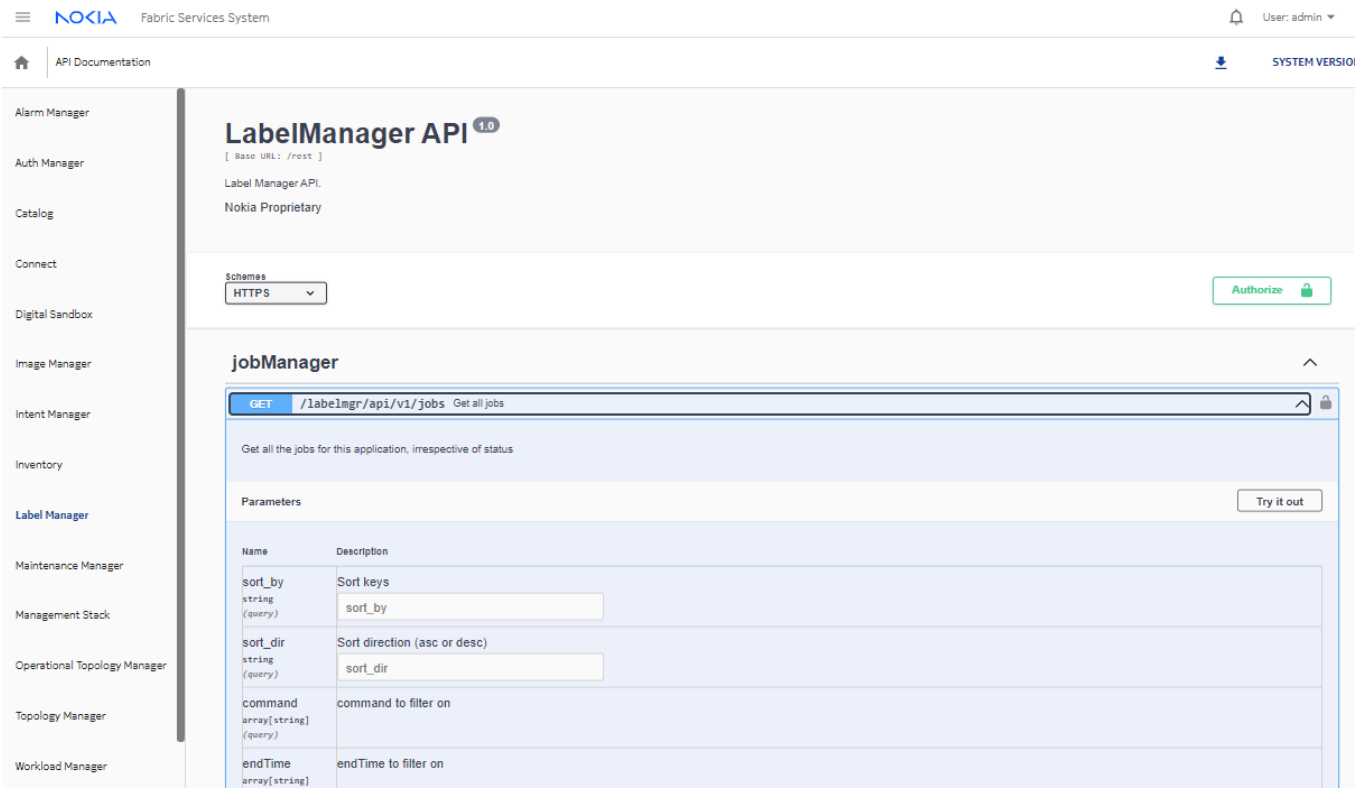
You can select a service and view the endpoints per service and the supported call methods that you can use for each endpoint.

*Figure 2: Endpoints for the LabelManager API*




When you click the expand icon for an endpoint, you can view the relevant input parameters for a call.

*Figure 3: Example of an expanded view*

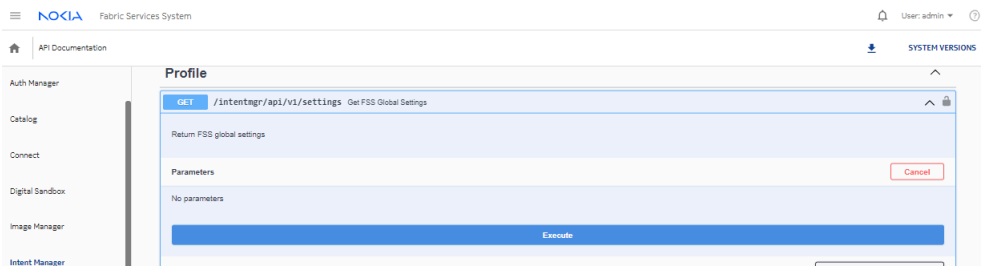



You can try and run API calls on the interactive Swagger UI.

**Note:** Before you execute a call through the Swagger UI, ensure that you have obtained an API access token, as described in Using the Swagger UI to authenticate. You need this access token when using the **Authorize** function on the Swagger UI.

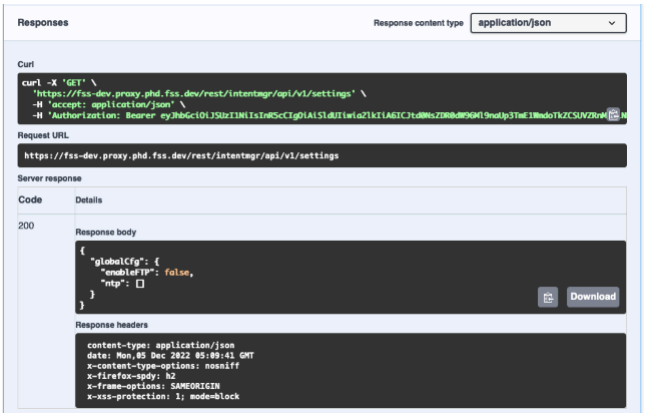
From the expanded view of an endpoint, click **Try it out**. In the request body, the field become editable so you can enter values.

*Figure 4: Executing a call on the Swagger UI*




Click **Execute** to send the request. After you click **Execute**, the **Responses** section shows the curl command that was submitted and the Request URL. The **Server response** section shows the code returned by the server.

*Figure 5: **Responses** section*




**Related topics**

*Reference documentation*

## 5.2 Retrieving a resource

Use a GET request to retrieve an object or group of objects. When you use the UUID, the request returns a single object in JSON format; if you do not use a UUID, the request returns a JSON list of objects.

### Example: Retrieve a list of fabric objects

```
curl --fail -s -X GET -H "Authorization: Bearer <ACCESS TOKEN>" -H "Content-Type:
 application/json; charset=utf-8" "https://fss.domain.tld/rest/intentmgr/api/v1/intents"

[
  {
    ...
    "fabric": [
      {
        "deployInfo": {
          "isFailed": false,
          "reason": "Delete Workload workload 1 Deployment Done.",
          "timestamp": "2022-11-22T19:19:12Z",
          "transactionId": "435561924323704832"
        },
        "isDigitalSandbox": false,
        "name": "edge1_realnet_fabric",
        "state": "DeploymentDone",
        "uuid": "434325057192460288",
        "version": "1.0"
      }
    ],
    ...
    "uuid": "434325057192329216",
    "version": "1.0",
    "wlCount": 0
  },
  {
    ...
    "fabric": [
      {
```

```
        "deployInfo": {
          "isFailed": false,
          "reason": "Workload APP01 Deployment Done",
          "timestamp": "2022-11-22T22:58:46Z",
          "transactionId": "435584022265987072"
        },
        "isDigitalSandbox": true,
        "name": "DSPOD01_gatenet_fabric",
"state": "DeploymentDone",
        "uuid": "435562509898940416",
        "version": "1.0"
      }
    ],
    ...
    "uuid": "435562509865320448",
    "version": "1.0",
    "wlCount": 0
  }
]
```

**Example: Retrieve a single fabric with its UUID**

```
curl --fail -s -X GET -H "Authorization: Bearer <ACCESS TOKEN>" -H "Content-Type:
 application/json; charset=utf-8" "https://fss.domain.tld/rest/intentmgr/api/v1/intents/
434325057192329216"

{
  ...
  "fabric": [
    {
      "deployInfo": {
        "isFailed": false,
        "reason": "Delete Workload workload 1 Deployment Done.",
        "timestamp": "2022-11-22T19:19:12Z",
        "transactionId": "435561924323704832"
      },
      "isDigitalSandbox": false,
      "name": "edge1_realnet_fabric",
      "state": "DeploymentDone",
      "uuid": "434325057192460288",
      "version": "1.0"
    }
  ],
  ...
  "uuid": "434325057192329216",
  "version": "1.0",
  "wlCount": 0
}
```

## 5.3 Create a resource

Use a POST request to create a new resource or set of resources. When you create a new object, you must provide a JSON object that contains all the attributes of the object as defined in the API specifications.

**Example: Create a label**

```
curl -X 'POST' \
  'https://fss.domain.tld/rest/labelmgr/api/v1/labels' \
  -H 'accept: application/json' \
```

```
  -H 'Authorization: Bearer <ACCESS TOKEN>' \
  -H 'Content-Type: application/json' \
  -d '{
  "comments": "",
  "defaultLabel": false,
  "externalID": "",
  "name": "Edge-Link",
  "value": "edgelabel"
}'

{
  "externalID": "",
  "name": "Edge-Link",
  "value": "edgelabel",
  "comments": "",
  "defaultLabel": false
}
```

## 5.4 Updating an object

Use a PUT request to update an existing object.

### Example: Update a label

The following PUT request updates the comments for a label:

```
curl -X 'PUT' \
  'https://fss.domain.tld/rest/labelmgr/api/v1/labels' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer <ACCESS TOKEN>' \
  -H 'Content-Type: application/json' \
  -d '{
  "externalID": "",
  "name": "Edge-Link",
  "value": "edgelabel",
  "comments": "Updated Comment",
  "defaultLabel": false
}'

{
  "externalID": "",
  "name": "Edge-Link",
  "value": "edgelabel",
  "comments": "Updated Comment",
  "defaultLabel": false
}
```

## 5.5 Deleting an object

Use a DELETE request to remove an object or a set of objects.

### Example: DELETE requests

The following DELETE request removes minor severity alarms:

```
curl -X 'DELETE' \
```

```
  'https://fss.domain.tld/rest/alarmmgr/api/v1/alarms?severity=minor' \
  -H 'accept: application/json' -H 'Authorization: Bearer <ACCESS TOKEN>'
```

The following DELETE request removes an alarm with a UUID 424197038574403584:

```
curl -X 'DELETE' \
  'https://fss.domain.tld/rest/alarmmgr/api/v1/alarms/424197038574403584' \
  -H 'accept: application/json' -H 'Authorization: Bearer <ACCESS TOKEN>'
```

## 5.6 Bulk API operations

A bulk API request is a single HTTPS request that contains multiple objects instead of a single object. For example, a regular POST requests contains the definition of a single object and creates that object, and then returns the created object. A bulk POST request contains a list of objects (of the same type and with the same hierarchy) where each object is handled individually and created; the return contains a list of created objects instead of a single object.

Bulk API requests use the same URIs as for single object requests. If single object is received, it handles it as regular API call. If a list of objects is received, it handles it as a bulk API request.

Bulk API requests are supported for POST, PUT and DELETE bulk API calls.

### Limitations

The maximum number of objects allowed for each bulk operation is as follows:

*Table 9: Number of objects per operation*

| Operation | Number of objects |
|---|---|
| POST | 200 |
| PUT | 200 |
| DELETE | 150 |

If request body is a JSON array, it is considered to be a bulk request. The following APIs already support arrays in the body and are not compatible with bulk API requests.

*Table 10: Unsupported endpoints*

| Method | Endpoint | Details |
|---|---|---|
| POST/ DELETE | `/auth/api/{version}/users/:uuid/roles` | add or remove role from user |
| POST/ DELETE | `/auth/api/{version}/users/:uuid/usergroups` | add or remove users from user-group |
| POST/ DELETE | `/auth/api/{version}/usergroups/:uuid/users` | associate or disassociate user from user-group |
| POST | `/intentmgr/api/{version}/intents/:uuid/ fabrics/:fabricid/accept` | accept, commit, or update diff |

### 5.6.1 POST and PUT requests

All POST and PUT endpoints that accept the JSON body (object) to create or update an object also expose bulk POST and bulk PUT endpoints.

#### Example: Create a single region

```
POST http://fss.domain.tld/rest/intentmgr/api/v1/regions
{ /* single region object*/ }
```

#### Example: Create multiple regions

```
POST http://fss.domain.tld/rest/intentmgr/api/v1/regions
[ { /* region1 object*/ }, { /* region2 object*/ }, { /* region3 object*/ } ]
```

### 5.6.2 Delete requests

DELETE endpoints that accept the resource ID in the URI or accept the JSON object as body (as identifier) also expose bulk DELETE endpoints.

#### Example: Delete devices identified by UUID

The following example deletes a single device by its UUID:

```
DELETE http://fss.domain.tld/rest/inventory/api/v1/devices/{uuid}
```

The following example deletes multiple devices identified by their UUIDs:

```
DELETE http://fss.domain.tld/rest/inventory/api/v1/devices?uuid=1234&uuid=5678&uuid=4534
```

#### Example: Delete objects by their labels

The following example deletes a single label identified by its name and value:

```
DELETE http://fss.domain.tld/rest/labelmgr/api/v1/labels
{ "name": "Tenant", "value": "tenant1" }
```

The following example deletes multiple labels:

```
DELETE http://fss.domain.tld/rest/labelmgr/api/v1/labels
[ { "name": "Tenant", "value": "tenant1" }, { "name": "tenant", "value": "tenant2" },
 { "name": "tenant", "value": "tenant3" } ]
```

### 5.6.3 Bulk API response

A bulk API request can include the following response codes:

*Table 11: Response codes*

| Code | Description |
|---|---|
| 200 | All actions were successful; information for each object is provided in the body |
| 207 | Not all actions were successful; check the body for each object to identify the errors |
| 4XX | Something is wrong with the entire bulk API request and it cannot be handled; this code is returned when incorrect formatting is used or when too many objects have been provided |

### Response structure

The response structure allows for the easy handling of the response for each separate object; each object has a unique response. You can use the index in the response to map the response back to each object in the original request. The response also includes the overall status that lists how many actions were successful and how many failed.

The response has the following structure:

```
{
    "response": [
        {
            "status": $STATUS_CODE_FIRST_OBJECT,
            "data": $RETURN_DATA_FIRST_OBJECT
        },
        {
            "status": $STATUS_CODE_SECOND_OBJECT,
            "data": $RETURN_DATA_SECOND_OBJECT
        }
    ],
    "responseMetadata": {
        "success": $NUMBER_OF_SUCCESSFUL_OBJECTS,
        "failure": $NUMBER_OF_FAILED_OBJECTS,
        "total": $TOTAL_NUMBER_OF_OBJECTS
    }
}
```

The following table describes the variables in the preceding response message.

*Table 12: Variable definitions*

| Variable | Description |
|---|---|
| $STATUS_CODE_*_OBJECT | This variable specifies the HTTP response code that would be returned if a single object API request had been made with the specified action. For possible values, see Table 11: Response codes. |
| $RETURN_DATA_*_OBJECT | This variable specifies the HTTP response data that would be returned if a single object API request had been made with the specified action. For instance, it contains the object's data if the creation succeeded or an error output if something went wrong for the object. |

| Variable | Description |
|---|---|
| $NUMBER_OF_SUCCESSFUL_OBJECTS | This variable specifies for how many objects the action was successful. |
| $NUMBER_OF_FAILED_OBJECTS | This variable specifies for how many objects the action failed. |
| $TOTAL_NUMBER_OF_OBJECTS | This variable specifies the total number of objects provided. This value is the sum of $NUMBER_OF_SUCCESSFUL_OBJECTS and $NUMBER_OF_FAILED_OBJECTS. |

## 5.7 Job manager framework

The Fabric Services System API job framework provides a way to handle REST API calls in a non-blocking manner so that the call returns to the invoking client immediately. The job manager manages the life cycle of a job.

At a high level, the job life cycle is as follows:

1. An API client sends a request.
2. The job manager creates a new job, validates it, and marks its status as `Submitted`.
3. The new job is persisted in the Fabric Services System database.
4. The job manager sends the client details about the accepted server operation in the form of a job with an ID, tracking URL, and other details.

The client can then use the tracking URL to poll for the job status.

### Example: Single auto-deploy command request body and response

The following examples show the request body that uses the **auto-deploy** command and the response from the system.

```
{
    "command": "AUTO_DEPLOY",
    "params": [
        {
            "key": "intentId",
            "value": 123
        },
        {
            "key": "intentVersion",
            "value": "1.0"
        }
    ]
}
```

### Example

```
{
    "id": "443896864224313344",
    "requestURI": "",
    "requestMethod": "",
    "command": "AUTO_DEPLOY",
    "params": [
        {
```

```
            "key": "intentId",
            "value": 123,
            "type": ""
        },
        {
            "key": "intentVersion",
            "value": "1.0",
            "type": ""
        }
    ],
    "enqueueTime": "2023-01-19T07:19:09.096617184Z",
    "endTime": null,
    "trackingURL": "/jobs/443896864224313344",
    "status": "Submitted",
    "failureReason": "",
    "result": null
}
```

### Example: Bulk API request and response

```
[
    {
        "command": "AUTO_DEPLOY",
        "params": [
            {
                "key": "intentId",
                "value": 123
            },
            {
                "key": "intentVersion",
                "value": "1.0"
            }
        ]
    },
    {
        "command": "AUTO_DEPLOY",
        "params": [
            {
                "key": "intentId",
                "value": 456
            },
            {
                "key": "intentVersion",
                "value": "2.0"
            }
        ]
    }
]
```

```
{
    "response": [
        {
            "data": {
                "id": "443896864224313344",
                "requestURI": "",
                "requestMethod": "",
                "command": "AUTO_DEPLOY",
                "params": [
                    {
                        "key": "intentId",
                        "value": 123,
                        "type": ""
```

```
                    },
                    {
                        "key": "intentVersion",
                        "value": "1.0",
                        "type": ""
                    }
                ],
                "enqueueTime": "2023-01-19T07:19:09.096617184Z",
                "endTime": null,
                "trackingURL": "/jobs/443896864224313344",
                "status": "InProgress",
                "failureReason": "",
                "result": null
            },
            "status": 202
        },
        {
            "data": {
                "id": "443896864291422208",
                "requestURI": "",
                "requestMethod": "",
                "command": "AUTO_DEPLOY",
                "params": [
                    {
                        "key": "intentId",
                        "value": 456,
                        "type": ""
                    },
                    {
                        "key": "intentVersion",
                        "value": "2.0",
                        "type": ""
                    }
                ],
                "enqueueTime": "2023-01-19T07:19:09.134867794Z",
                "endTime": null,
                "trackingURL": "/jobs/443896864291422208",
                "status": "Submitted",
                "failureReason": "",
                "result": null
            },
            "status": 202
        }
    ],
    "responseMetadata": {
        "success": 2,
        "failure": 0,
        "total": 2
    }
}
```


### Job retention

The system checks every hour for jobs that are older than 24 hours and purges them from the database to keep storage usage to a minimum.

## 5.8 Displaying inventory

### About this task

You can gather and build a detailed inventory of all hardware in a data center. The hardware inventory includes details about chassis, CPMs, line cards, fans, PSUs, media adapters, or any hardware part with a part number or serial number.
The **INVENTORY_UPDATE** job command is used to gather and update information about all the hardware in a node.

To display the hardware details after updating it using the `INVENTORY_UPDATE` job command, use the following API endpoints:

- to display hardware details for a single node: `http://fss.domain.tld/rest/inventory/api/v1/regions/{regionid}/pdevices/{node uuid}/hardwaredetails`
- to display hardware details for a fabric intent: `http://fss.domain.tld/rest/inventory/api/v1/inventory/api/v1/regions/{regionid}/intents/{fabric intent uuid}/pdeviceshardwaredetails`

### Prerequisites

- The fabric intent must be successfully deployed and the planned node has been associated with real-world hardware.
- The nodes must be in the Ready state.

### Procedure

**Step 1.** Create a job using **INVENTORY_UPDATE** command.

The endpoint is `http://fss.domain.tld/rest/inventory/api/v1/jobs` API.

**Example**

**INVENTORY_UPDATE** command request and response:

```
[
  {
    "command": "INVENTORY_UPDATE",
    "params": [
      {
        "key": "intentID",
        "value": "471251337657472994"   // the fabric intent uuid
      }
    ]
  }
]
```

```
{
    "response": [
        {
            "data": {
                "regionID": "471251302223992802",
                "id": "471251417886119925",
                "requestURI": "",
                "requestMethod": "",
                "command": "INVENTORY_UPDATE",
                "params": [
                    {
                        "key": "intentID",
                        "value": "471251337657472994",
```

```
                        "type": ""
                    }
                ],
                "enqueueTime": "2023-07-27T00:22:13.638365301Z",
                "endTime": null,
                "trackingURL": "/regions/471251302223992802/jobs/471251417886119925",
                "status": "Submitted",
                "failureReason": "",
                "result": null
            },
            "status": 202
        }
    ],
    "responseMetadata": {
        "success": 1,
        "failure": 0,
        "total": 1
    }
}
```

**Example**

Response when getting the latest state of the job, showing successful execution of the **INVENTORY_UPDATE** command:

```
{
    "regionID": "471251302223992802",
    "id": "471251971181926389",
    "requestURI": "",
    "requestMethod": "",
    "command": "INVENTORY_UPDATE",
    "params": [
        {
            "key": "intentID",
            "value": "471251337657472994",
            "type": ""
        }
    ],
    "enqueueTime": "2023-07-27T00:27:43.42Z",
    "endTime": "2023-07-27T00:27:43.66Z",
    "trackingURL": "/regions/471251302223992802/jobs/471251971181926389",
    "status": "Success",
    "failureReason": "",
    "result": {
        "devicesFailed": "0",
        "devicesProcessed": "2",
        "devicesSucceeded": "2",
        "errors": [],
        "successList": [
            "471251378744874970",
            "471251378845538266"
        ]
    }
}
```

**Example**

Response when getting the latest state of the job, showing partially successful execution of the **INVENTORY_UPDATE** command:

```
{
    "regionID": "471251302223992802",
```

```
    "id": "471252375965816821",
    "requestURI": "",
    "requestMethod": "",
    "command": "INVENTORY_UPDATE",
    "params": [
        {
            "key": "intentID",
            "value": "471251313649276898",
            "type": ""
        }
    ],
    "enqueueTime": "2023-07-27T00:31:44.69Z",
    "endTime": "2023-07-27T00:31:44.877Z",
    "trackingURL": "/regions/471251302223992802/jobs/471252375965816821",
    "status": "Success",
    "failureReason": "",
    "result": {
        "devicesFailed": "2",
        "devicesProcessed": "4",
        "devicesSucceeded": "2",
        "errors": [
            {
                "additional_info": null,
                "detail": "Device 471251331919730650 is not ready",
                "errors": null,
                "object_ref": "/inventory/api/v1/pdevices/471251331919730650/
hardwaredetails",
                "status": 412,
                "title": "DEVICE_NOT_READY",
                "type": "/inventory/errors?key=device-not-ready"
            },
            {
                "additional_info": null,
                "detail": "Device 471251329940019162 is not ready",
                "errors": null,
                "object_ref": "/inventory/api/v1/pdevices/471251329940019162/
hardwaredetails",
                "status": 412,
                "title": "DEVICE_NOT_READY",
                "type": "/inventory/errors?key=device-not-ready"
            }
        ],
```


**Example**

Response when getting the latest state of the job, showing failed execution of the **INVENTORY_UPDATE** command:


```
{
    "regionID": "471251302223992802",
    "id": "471251417886119925",
    "requestURI": "",
    "requestMethod": "",
    "command": "INVENTORY_UPDATE",
    "params": [
        {
            "key": "intentID",
            "value": "123",
            "type": ""
        }
    ],
    "enqueueTime": "2023-07-27T00:22:13.638Z",
    "endTime": "2023-07-27T00:22:13.736Z",
    "trackingURL": "/regions/471251302223992802/jobs/471251417886119925",
```

```
    "status": "Fail",
    "failureReason": "did not have intent for  123",
    "result": null
}
```

**Step 2.** Display hardware details.

**Example**

Device and component inventory by fabric intent - sample request and response

```
curl -X 'GET' \
  'https://fsp.nokia.com:8090/rest/inventory/api/v1/regions/471251302223992802/
intents/471251337657472994/pdeviceshardwaredetails' \
  -H 'accept: application/json' \
  -H 'Authorization: NokiaFss1!'
```

**Example**

Device and component inventory by planned node UUID - sample request and response

```
'https://fsp.nokia.com:8090/rest/inventory/api/v1/regions/471251302223992802/pdevices/
471251337657472994/hardwaredetails' \
  -H 'accept: application/json' \
  -H 'Authorization: NokiaFss1!'
```

```
{
    "data": {
        "system": {
            "name": {
                "host-name": "fssi468972054851374514-020003ff0000-665b644676-dzxzk"
            },
            "information": {
                "description": "SRLinux-v22.11.1-184-g6eeaa254f7 7250 IXR-6 Copyright
 (c) 2000-2020 Nokia. Kernel 4.18.0-425.3.1.el8.x86_64 #1 SMP Wed Nov 9 20:13:27 UTC
 2022",
                "version": "v22.11.1-184-g6eeaa254f7"
            }
        },
        "platform": {
            "chassis": {
                "clei-code": "Sim CLEI",
                "hw-mac-address": "02:00:03:FF:00:00",
                "manufactured-date": "01012019",
                "part-number": "Sim Part No.",
                "serial-number": "Sim Serial No.",
                "type": ""
            },
            "control": {
                "A": {
                    "clei-code": "",
                    "manufactured-date": "",
                    "part-number": "",
                    "serial-number": "",
                    "slot": "A",
                    "software-version": "",
                    "type": "cpm2-ixr"
                },
                "B": {
                    "clei-code": "",
                    "manufactured-date": "",
                    "part-number": "",
                    "serial-number": "",
```

```
                    "slot": "B",
                    "software-version": "",
                    "type": ""
                }
            },
            "fabric": null,
            "fan-tray": {
                "1": {
                    "clei-code": "",
                    "manufactured-date": "",
                    "part-number": "",
                    "serial-number": "",
                    "type": "",
                    "id": 1
                },
                "2": {
                    "clei-code": "",
                    "manufactured-date": "",
                    "part-number": "",
                    "serial-number": "",
                    "type": "",
                    "id": 2
                },
                "3": {
                    "clei-code": "",
                    "manufactured-date": "",
                    "part-number": "",
                    "serial-number": "",
                    "type": "",
                    "id": 3
                }
            },
            "linecard": {
                "1": {
                    "clei-code": "",
                    "manufactured-date": "",
                    "part-number": "",
                    "serial-number": "",
                    "software-version": "",
                    "type": "",
                    "slot": 1
                },
                "2": {
                    "clei-code": "",
                    "manufactured-date": "",
                    "part-number": "",
                    "serial-number": "",
                    "software-version": "",
                    "type": "",
                    "slot": 2
                },
                "3": {
                    "clei-code": "",
                    "manufactured-date": "",
                    "part-number": "",
                    "serial-number": "",
                    "software-version": "",
                    "type": "",
                    "slot": 3
                },
                "4": {
                    "clei-code": "",
                    "manufactured-date": "",
                    "part-number": "",
```

```
                    "serial-number": "",
                    "software-version": "",
                    "type": "",
                    "slot": 4
                }
            },
            "power-supply": {
                "1": {
                    "clei-code": "",
                    "manufactured-date": "",
                    "part-number": "",
                    "serial-number": "",
                    "type": "",
                    "id": 1
                },
                "2": {
                    "clei-code": "",
                    "manufactured-date": "",
                    "part-number": "",
                    "serial-number": "",
                    "type": "",
                    "id": 2
                },
                "3": {
                    "clei-code": "",
                    "manufactured-date": "",
                    "part-number": "",
                    "serial-number": "",
                    "type": "",
                    "id": 3
                },
                "4": {
                    "clei-code": "",
                    "manufactured-date": "",
                    "part-number": "",
                    "serial-number": "",
                    "type": "",
                    "id": 4
                },
                "5": {
                    "clei-code": "",
                    "manufactured-date": "",
                    "part-number": "",
                    "serial-number": "",
                    "type": "",
                    "id": 5
                },
                "6": {
                    "clei-code": "",
                    "manufactured-date": "",
                    "part-number": "",
                    "serial-number": "",
                    "type": "",
                    "id": 6
                }
            }
        },
        "interface": null
    },
    "lastUpdated": "2023-07-11T16:22:24Z",
    "name": "ds2-spine-1",
    "mgmtAddress": "31.107.200.68",
    "serialNumber": "468972119493987729",
    "uuid": "468972119493987729",
```

```
    "softwareVersion": "22.11.1-184"
```

# 6 API filtering, sorting, and pagination

The Fabric Services System REST API supports the ability to filter, sort, and paginate results from REST API GET requests.

## 6.1 Filtering

To filter the results of an HTTP GET requests for lists of objects, specify an attribute name and the value of the attribute for which you want to filter.

Filtering can be combined with pagination and sorting. When you apply filtering, the HTTP HEAD request returns the total number of objects that would be returned with a GET request, and the filter is applied to this result. For example, if there are 100 objects and a filter is used that results in the GET API call returning 20 of them, the HEAD request also returns the number 20.

The following table describes the parameters used for filtering.

*Table 13: Filtering parameters*

| Name | Type | Required | Description |
|---|---|---|---|
| `attribute_name` | String | No | The name of an attribute |
| `value` | String | No | The text string to match |

### Using wildcards

You can use the wildcard character (*) at the beginning or the end of a specified value. The wildcard character * represents one or more characters.

As multiple fields in the API can contain an asterisk (*) as a valid character, to use it as a wildcard in the filtering, you must also include `wildcardsearch=true` in the expression.

**Example:**

The following expression returns `externalId` values that start with `connect-`:

```
?externalId=connect-*&wildcardsearch=true
```

The following expression returns `name` values that contain `example`:

```
?name=*example*&wildcardsearch=true
```

### Combining expressions

You can further refine the results returned by combining multiple query expressions. Use the ampersand character & to combine multiple filters.

When you search on multiple different attributes, these attributes are combined in an AND-based search. When using multiple search values for the same attribute, the search looks for any of the specified values for that attribute in an OR-based search.

### Example: Using the AND operator

The following example returns results where the `name` attribute contains `example` and the `externalId` starts with `connect-`:

```
?name=*example*&externalId=connect-*&wildcardsearch=true
```

### Example: Using the OR operator

The following expression returns results where the `name` attribute contains `example` or `sample`:

```
?name=*example*&name=*sample*&wildcardsearch=true
```

### Example: AND and OR operators

The following example returns results where the `name` attribute value contains `example` or `sample` and the `externalId` attribute starts with `connect-` or `con-`:

```
?name=*example*&name=*sample*&externalId=connect-*&externalId=con-
*&wildcardsearch=true
```

## 6.2 Sorting

You can sort the results of an HTTP GET request. You can sort on a single attribute or multiple attributes. You can combine sorting with pagination and filtering.

The following table describes the parameters used for sorting.

*Table 14: GET query parameters*

| Name | Type | Required | Description |
|---|---|---|---|
| `sort_by` | String | No | Use this field to sort the results. |
| `sort_dir` | String | No | This field specifies whether to sort in ascending (**asc**) or descending (**desc**) order. The value is case insensitive.<br>Default: **asc** |

### Example: Sort on a single attribute

```
sort_by=name&sort_dir=asc
```

### Example: Sort on multiple attributes

```
sort_by=name,externalId&sort_dir=asc,desc
```

## 6.3 Pagination

Pagination prevents the overloading of the API backend when it has to return hundreds or even thousands of objects in a single HTTP GET request. Pagination also prevents overloading the receiving side in the amount of data and the number of objects to process.

Provide information about pagination in the GET request parameters.

A link header is returned to highlight the next page or previous page URIs for easy reference; they are only present if they are relevant.

The table below describes the parameters used for the pagination request.

*Table 15: Request parameter definitions*

| Name | Type | Required | Description |
|---|---|---|---|
| `page` | Integer | No | The page number to display.<br>Default: 1 |
| `page_size` | Integer | No | The number of items to return in the response.<br>Default: 100 |

The table below describes the link header parameter.

*Table 16: Link header parameter definitions*

| Name | Type | Required | Description |
|---|---|---|---|
| `rel` | String | Yes | This parameter specifies link relation for the specific page results. The link relation can be:<br>• `next` - the next page of results<br>• `prev` - the previous page of results |

### Example: Pagination request for the second page with a page size of 100

```
GET https://fss.domain.tld/rest/...?page=2&page_size=100
```

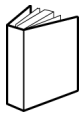
### Example: Link header that contains a link to the next page

```
<https://fss.domain.tld/rest/...?page=2&page_size=100>; rel=next
```

### Example: Link header that contains a link to the previous page

```
<https://fss.domain.tld/rest/...?page=1&page_size=100>; rel=prev
```

# Customer document and product support



## Customer documentation

Customer documentation welcome page

## Technical support

Product support portal

## Documentation feedback

Customer documentation feedback