API User Guide

Privitar Modern Data Provisioning Platform, version 1.1.0

Publication date December 30, 2022

Table of Contents

1. Welcome to the MDP Platform API User Guide 3
2. Introduction to the Platform REST APIs 4
    2.1. REST API Functions 4
    2.2. Submitting REST API Requests 4
    2.3. Platform REST API Essentials 5
3. Tools for REST APIs 6
    3.1. Use Swagger UI to Explore and Learn 6
    3.2. About cURL 8
4. Authorization 9
5. The Platform REST APIs 10
    5.1. Access Control Policy API Description 10
    5.2. Access Control Policy Rule API Description 12
    5.3. Asset API Description 13
    5.4. Attribute API Description 15
    5.5. Connection API Description 16
    5.6. Data Class API Description 18
    5.7. Data Plane API Description 20
    5.8. Dataset API Description 21
    5.9. Field API Description 22
    5.10. Project API Description 24
    5.11. Tag API Description 26
    5.12. Task API Description 27
    5.13. Term API Description 28
    5.14. Transformation API Description 31
    5.15. Transformation Policy API Description 32
    5.16. Transformation Policy Rule API Description 34
6. HTTP Status Codes and Error Handling 36
7. Glossary of Modern Data Provisioning Terminology 37

1. Welcome to the MDP Platform API User Guide

Welcome to the MDP Platform API User Guide. If you are reading this document, you are likely to be a skilled developer. You may simply want to know the API host URL in order to use Swagger, or how to create a token for authorization. That information is here.

If you are not an API developer, but you are interested in exploring the platform APIs, you may first want to read more about how Swagger UI works. This tutorial is written from the perspective of those documenting APIs, but it is also a description that offers a better understanding of Swagger UI for any casual user.

2. Introduction to the Platform REST APIs

The Privitar Modern Data Provisioning Platform uses Representational State Transfer (REST) APIs to manage assets.

The APIs use Hypertext Transfer Protocol Secure (HTTPS) to perform Create, Retrieve, Update, and Delete (CRUD) operations through the following methods:

- `POST` (Create)
- `GET` (Retrieve)
- `PUT` (Update)
- `DELETE`

2.1. REST API Functions

The REST APIs encrypt requests and data and provide responses using Transport Layer Security (TLS).

The combination of HTTPS method and API URL is referred to as an "endpoint." Each REST API endpoint is documented in Swagger, and is characterized as follows:

- It is "stateless"—that is to say, no context is stored, and each client needs to provide all the necessary information to service each request.
- It is "cache-able"—that is to say, the client's intermediary can store the responses for subsequent retrieval.
- It uses a uniform interface—all responses are provided in JavaScript Object Notation (JSON) format.

Note

JSON is a convenient format for representing REST resources because it is human-readable, easily compressed, and all modern programming languages support it. It is also easy to understand because it only has a few data types (`String`, `Number`, `Boolean`, `Null`, `Object`, and `Array`).

2.2. Submitting REST API Requests

To try out REST API calls from the Privitar Modern Data Provisioning Platform REST API documentation, go to:

```
https://<your_MDP_environment_url>/api/v1/api-docs/swagger-ui.html
```

To call a REST API on the platform, you send a request to the server, incorporating all the information required to tell the server what you want it to do. This must include the following:

- the relevant REST API method

- the URL
- the request parameters, including:
  - the authorization details
  - the object (term, data class, and so on) ID
  - the exchange ID

  To obtain the exchange ID:

  1. Log in to the platform.
  2. Click **Data Exchange** in the left navigation.
  3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**.
  4. **Data Exchange ID—**Select and copy the data exchange ID.

When the server finishes the call, it sends a response back to you, letting you know whether or not the operation was successful. In the event of a successful call, the response includes the data that you requested in the body section. The header section contains metadata about the response.

2.3. Platform REST API Essentials

Each platform REST API supports standard CRUD methods for managing data. In addition, Terms API and Data Classes API endpoints support methods on associated terms.

For an on-premises instance of the platform, each REST API endpoint is:

https://<your_MDP_environment_url>/api/v1/<ENDPOINT>

For example, for the Terms endpoint is:

https://<your_MDP_environment_url>/api/v1/terms

3. Tools for REST APIs

When you explore and test API endpoints, you will use one of the many graphical user interface (GUI) REST clients available to make API requests. You will also use a command-line interface, like cURL for some tasks too.

Swagger UI is used to explore and learn about APIs using the OpenAPI specification. Many developers also use Postman for testing and development, but there are plenty of choices out there that will work with the platform REST APIs.

3.1. Use Swagger UI to Explore and Learn

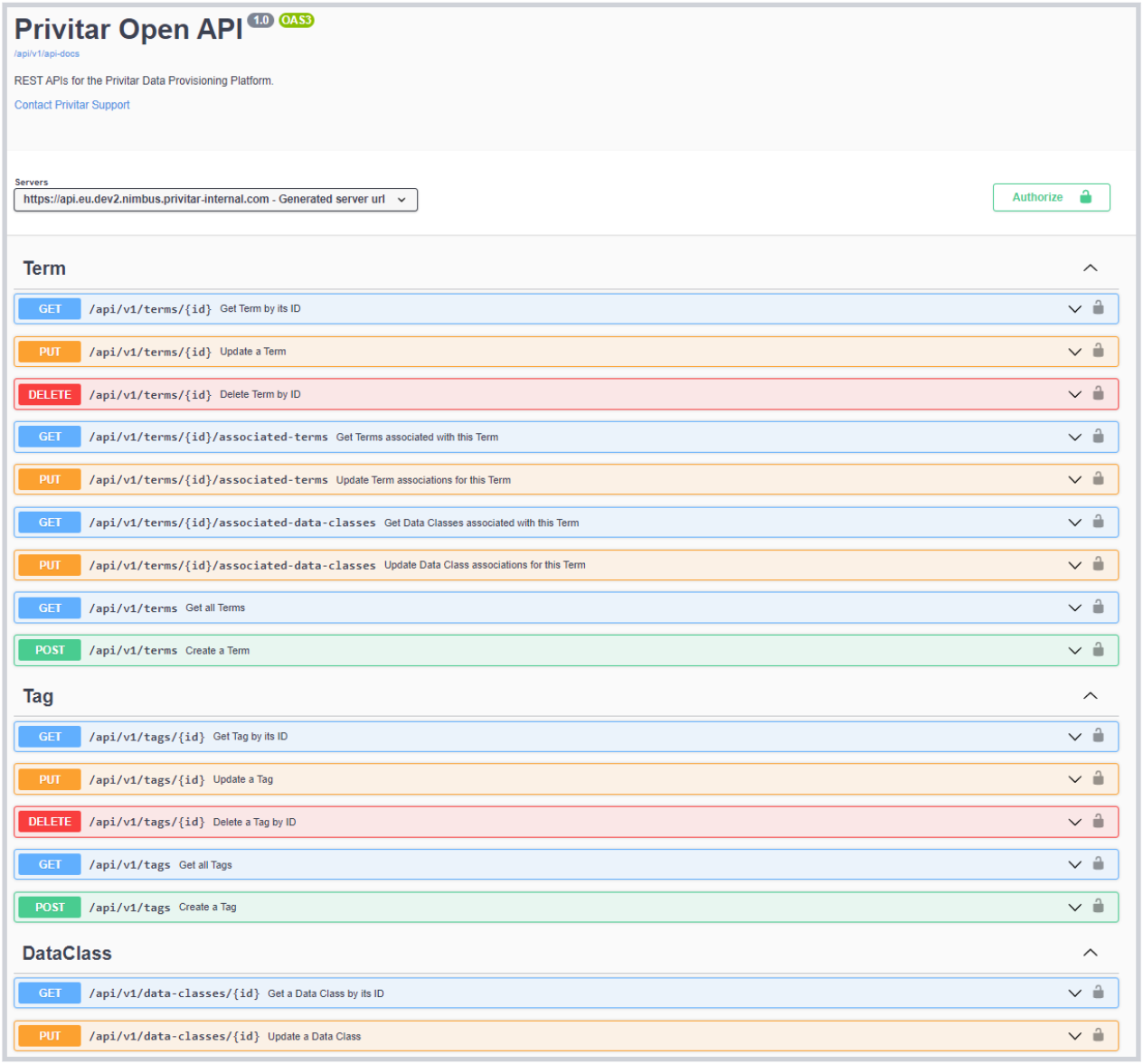
Swagger UI provides a display framework that reads an OpenAPI specification document and uses it to generate an interactive API console. This console allows you to quickly learn about the associated API and experiment with it by sending requests and viewing the responses generated by the requests.

To explore the endpoints of the APIs in Swagger UI, open a browser and enter the following URL, replacing YOUR_HOSTNAME with your host server name:

```
https://<YOUR_HOSTNAME>/api/v1/api-docs/swagger-ui.html
```

Press **Enter**.

The Privitar Open API landing page appears in Swagger UI. It displays a list of all available REST API endpoints.



Swagger UI effectively renders OpenAPI specs as interactive API documentation. All endpoints display here in one consolidated view. What you see here is everything that is currently available.

To interact with an endpoint method, you must first authorize access to the API with a bearer token. The platform APIs use JSON Web Tokens (JWT) for authorization. Click **Authorize**, and enter the JWT. The token is not used until a REST action is performed on an API.

Click an endpoint to expand the details and try it out. For each endpoint, you'll see the possible errors in the Responses section.

At the top of the page, click the `/api/v1/api -docs` link to open the schema in a new browser tab.

The schema is a document that's generated from the Open API specification. It gives a useful shape to the API, and is an additional quick reference guide where you can see all the endpoints and their parameters. It shows how requests and responses are formatted and the types of error or success responses to expect.

3.2. About cURL

Client Uniform Resource Locator (cURL) is a command-line tool that developers use to transfer data to and from a server. At its most fundamental, cURL allows you to talk to a server by specifying the location (in the form of a URL) and the data that you want to send. cURL supports several different protocols, including HTTP and HTTPS and runs on almost every platform. This makes cURL ideal for testing REST API calls from almost any device (provided that the device has a command line and network connectivity).

To run a cURL command, specify `curl` followed by the URL from which you wish to retrieve data, for example:

```
curl https://example.com
```

4. Authorization

Authorize REST actions on the API by passing a bearer authentication token into the header. The host server must first generate and return this JSON Web Token (JWT) to you by the host server in the following way.

Issue a `POST` request to the `/graph/enterprise-management/v1/registration/login` endpoint. To do this locally, open a terminal and use this cURL command:

```
curl --data '{"username": "YOUR EMAIL","password": "YOUR PASSWORD"}' \
--header "accept:*/*" \
--header "content-type:application/json" -k \
--url "https://<your_MDP_environment_url>/graph/enterprise-management/v1/
registration/login"
```

Replace `YOUR EMAIL` and `YOUR PASSWORD` with your platform email address and password. Replace `<your_MDP_environment_URL>` with the URL of your host server.

Press **Enter**.

If successful, the response text is your JWT. Note that if you execute the command from a `zsh` shell, your terminal will append a % symbol at the end (to let you know that there was no new line). When you copy the JWT, don't include that symbol.

You can now use the bearer token in Swagger, Postman, and other requests to the platform APIs.

5. The Platform REST APIs

The address of each API endpoint is:

```
https://<your_PDPP_environment_url>/api/v1/<ENDPOINT>
```

For example, the Term API endpoint is, `https://<your_PDPP_environment_url>/api/v1/terms`

There are platform APIs for:

- Access Control Policies
- Access Control Policy Rules
- Assets
- Attributes
- Connections
- Data Classes
- Data Planes
- Datasets
- Fields
- Projects
- Tags
- Tasks
- Terms
- Transformations
- Transformation Policies
- Transformation Policy Rules

The following sections are a brief description of the APIs, featuring a selected endpoint method in each case. To see all endpoints and methods, use Swagger. You can examine each API endpoint and see details, such as:

- the available parameters, such as the desired object (`term`, `tag`, and so on) to be acted on and the exchange ID in which it resides
- an example request body for create (`POST`) and update (`PUT`)
- an example response code and message description

5.1. Access Control Policy API Description

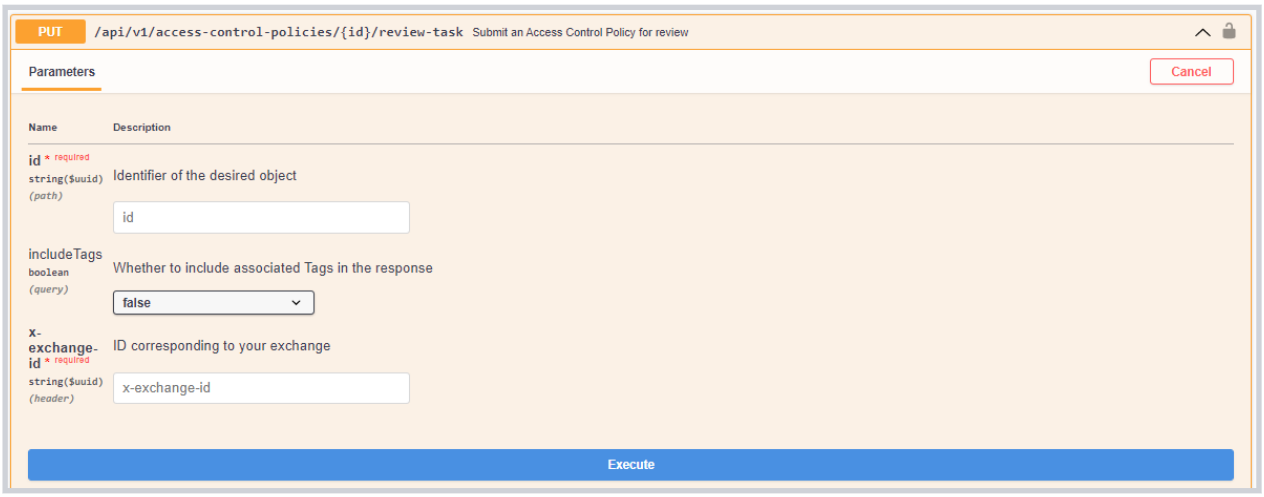
The Access Control Policy API has endpoints to create, read, update, and delete an access control policy. You can also:

- view details of an access control policy's rejection message
- submit an access control policy for review
- acknowledge that an access control policy has been rejected during a review by returning it to either the `Draft` or `Draft Update` state

When you POST (create) a new access control policy it is in draft status.

Example Access Control Policy API Endpoint

In Swagger, in the **Access Control Policy API** endpoints, click the arrow symbol on the **Submit an Access Control Policy for review** endpoint.



For each request in Swagger, you must supply a set of parameters. This endpoint has the following parameters:

Table 1. API Endpoint Parameters (Submit an Access Control Policy for review)

Parameter Name	Parameter Description	Notes
id	Identifier of the access control policy.	To find the ID, either: • look at the access control policy on the platform. It's in the URL. • or, it will be in the responses of `Get an access control policy by its ID`.
include Tags	Whether to include associated tags in the response.	Boolean (True/False)
x-exchange-id	The ID corresponding to your exchange on the platform.	The ID of the exchange in which the object exists. You can find the ID by looking at the exchange details on the platform: 1. Log in to the platform. 2. Click **Data Exchange** in the left navigation. 3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**. 4. **Data Exchange ID—**Select and copy the data exchange ID.

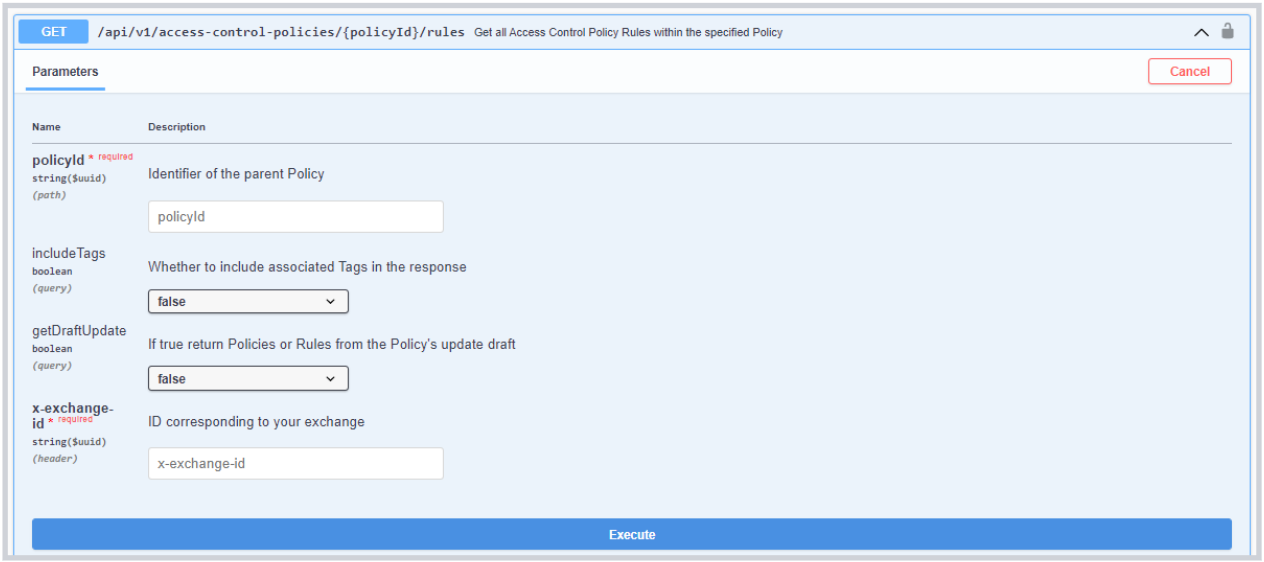
If the review submission is successful, a 200 status code returns the message, `The Task that must be completed to review the Access Control Policy`, and an example of the schema appears. If the PUT submission is unsuccessful, returns a 400 error with the message, `The Policy's current state does not allow it to be submitted for review` or, `The specified resource could not be found.`

5.2. Access Control Policy Rule API Description

The Access Control Policy Rule API has endpoints to create, read, update, and delete an access control policy rule. You can view the details of every access control policy rule in a specified policy, or view a single access control policy rule by its ID.

Example Access Control Policy Rule API Endpoint

In Swagger, in the **Access Control Policy Rule API** endpoints, click the arrow symbol on the **Get all Access Control Policy Rules within the specified Policy** endpoint.



For each request in Swagger, you must supply a set of parameters. This endpoint has the following parameters:

Table 2. API Endpoint Parameters (Get all Access Control Policy Rules within the specified Policy)

Parameter Name	Parameter Description	Notes
policyId	Identifier of the access control policy.	In this case, the desired object is an existing access control policy. You can find the ID by looking at the access control policy details on the platform, or by viewing the policy details through the Access Control Policy API.
include Tags	Whether to include associated tags in the response.	Boolean (True/False)

Parameter Name	Parameter Description	Notes
getDraftUpdate	Whether to return policies or rules from the current published version of a policy or a more recent draft update to the policy.	Boolean (True/False)
x-exchange-id	The ID corresponding to your exchange on the platform.	The ID of the exchange in which the object exists. You can find the ID by looking at the exchange details on the platform: 1. Log in to the platform. 2. Click **Data Exchange** in the left navigation. 3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**. 4. **Data Exchange ID—**Select and copy the data exchange ID.

If the GET is successful, the platform returns a list of rules in the specified policy. If no rules can be found, a 404 message displays, `The specified resource could not be found`.

If you searched for rules in a draft update, but the draft update does not exist, a 400 error displays:

```
The entity does not have a Draft Update, but the getDraftUpdate query
parameter was specified in the request
```

5.3. Asset API Description

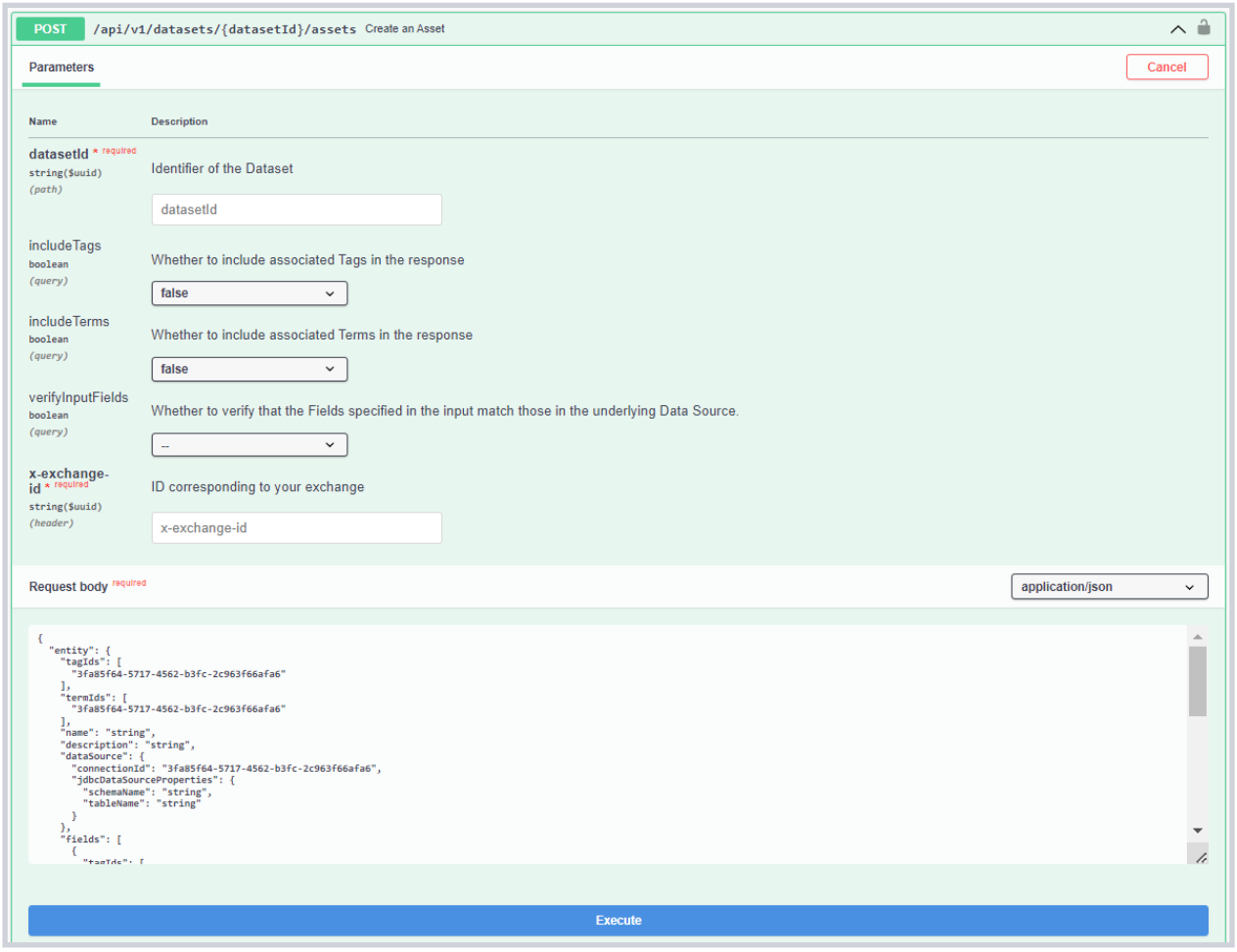
The Asset API has endpoints to create, read, update, and delete an asset. You can also:

- view details of an asset, by asset ID
- submit an asset for review

When you POST (create) a new asset you are also creating new fields and associated attributes within the asset.

Example Asset API Endpoint

In Swagger, in the **Asset API** endpoints, click the arrow symbol on the **Create an Asset** endpoint.



For each request in Swagger, you must supply a set of parameters. This endpoint has the following parameters:

Table 3. API Endpoint Parameters (Create an Asset)

Parameter Name	Parameter Description	Notes
datasetid	Identifier of the dataset.	In this case, the desired object is an existing dataset. You can find the ID by looking at the dataset details on the platform.
include Tags	Whether to include associated tags in the response.	Boolean (True/False)
include Terms	Whether to include associated terms in the response.	Boolean (True/False)
verifyInputFields	Whether to verify that the Fields specified in the input match those in the underlying data source.	Boolean (True/False) This will use the connection details to connect to the source, and validate the input. The API PUT will fail, if the connection fails. If you want to create an asset without testing if the connection is valid, set this to false.

Parameter Name	Parameter Description	Notes
x-exchange-id	The ID corresponding to your exchange on the platform.	The ID of the exchange in which the object exists. You can find the ID by looking at the exchange details on the platform: 1. Log in to the platform. 2. Click **Data Exchange** in the left navigation. 3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**. 4. **Data Exchange ID—**Select and copy the data exchange ID.

1. Click **Try it out**.
2. Enter the value of the asset in the `name` field of the request body.
3. Click **Execute**.

If the asset creation is successful, a 201 status code returns the message, `The Created Asset`, and an example of the schema appears. If the `POST` request is unsuccessful, it may return an error such as a 400 status, with the message:

```
Bad request due to an invalid format or because the requested name is
already in use.
```

5.4. Attribute API Description

The Attribute API has endpoints to read all location attribute types, or read all purpose attribute types.

Use this API to search for attributes required in other APIs:

- When you create or update a project through the Project API, you need to specify the purpose in the request body. You can obtain the set of allowable purposes by querying `GET /attributes/purpose` in the Attribute API.
- When you create or update a connection through the Connection API, you need to specify a geographical source location in the request body. You can obtain the set of allowable locations by querying `GET /attributes/location` in the Attribute API.

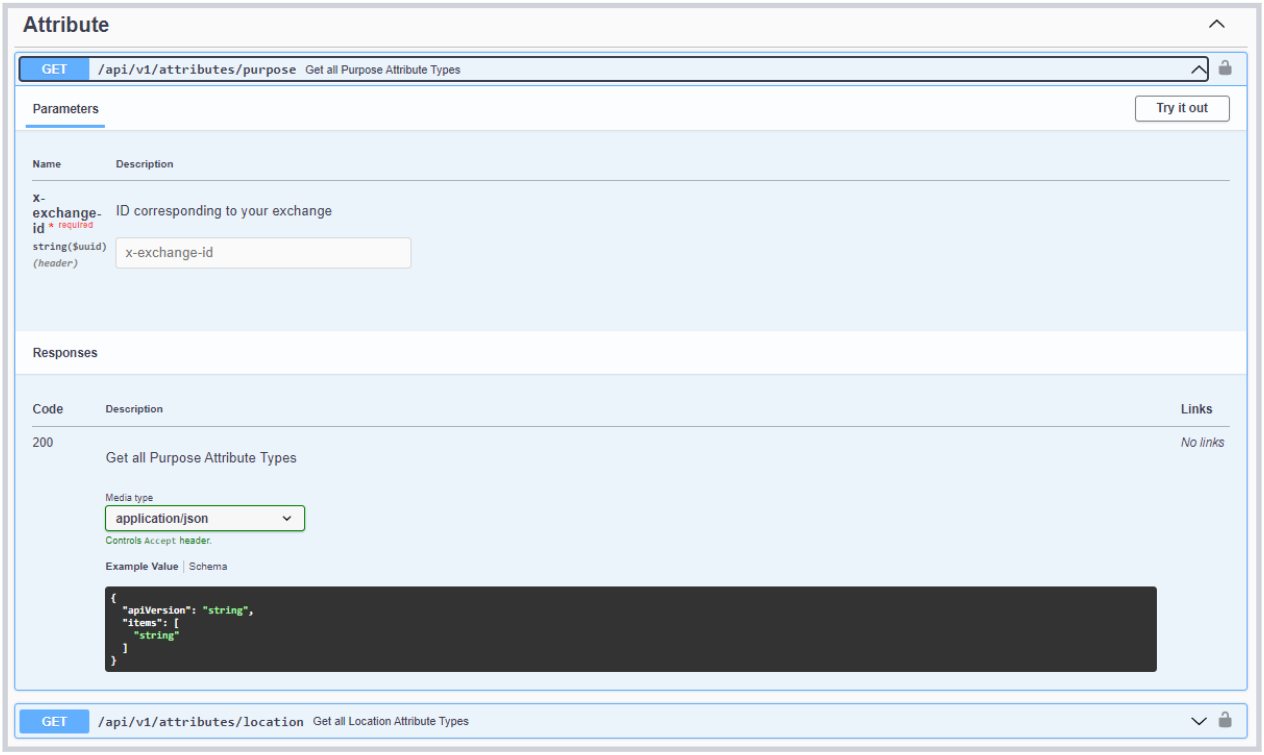
Example Attribute API Endpoint

In Swagger, in the **Attribute API** endpoints, click the arrow symbol on the **Get all Purpose Attribute Types** endpoint. For each request in Swagger, you must enter one or more parameters. This endpoint only needs the ID of the exchange on the platform where the object exists. To obtain the exchange ID:

1. Log in to the platform.
2. Click **Data Exchange** in the left navigation.
3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**.

4. **Data Exchange ID—**Select and copy the data exchange ID.

This image shows how the parameter looks in Swagger. Note the option to **Try it out**. You will need to be authorized to interact with an API endpoint.



5.5. Connection API Description

The Connection API has endpoints to create, read, update, and delete a connection. You can find all connections or search for a specific connection by connection ID. You select connections for deletion by their ID.

When creating (`POST`) or updating (`PUT`) a connection you need to specify a geographical source **location** in the request body. You can obtain the set of allowable locations by querying `GET /attributes/location` in the Attribute API, or look in the platform's user interface.

Example Connection API Endpoint

In Swagger, in the **Connection API** endpoints, click the arrow symbol on the **Create a Connection** endpoint. For each request in Swagger, you must supply a set of parameters. This endpoint has the following parameters:

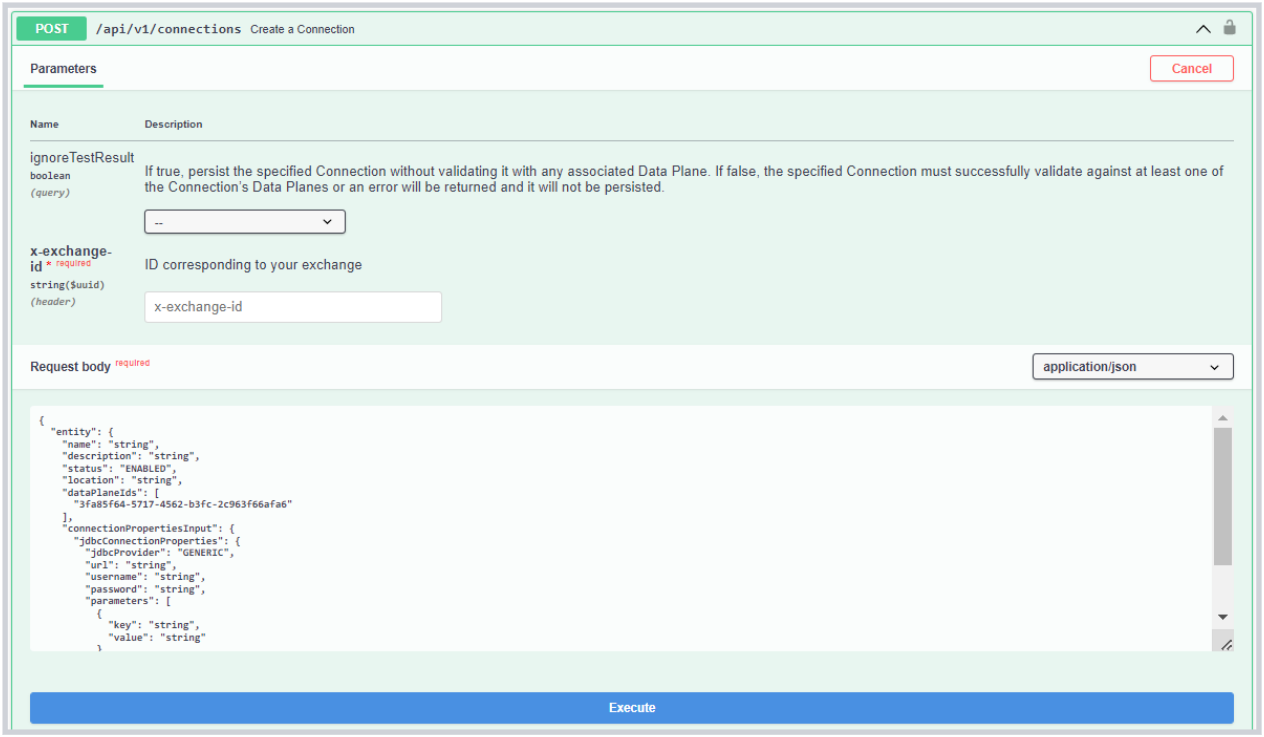
Table 4. Connection API Endpoint Parameters (Create a Connection)

Parameter Name	Parameter Description	Notes
ignoreTestResult	Whether to fail the creation request when connection tests for all of the data planes fail.	Boolean (True, False).

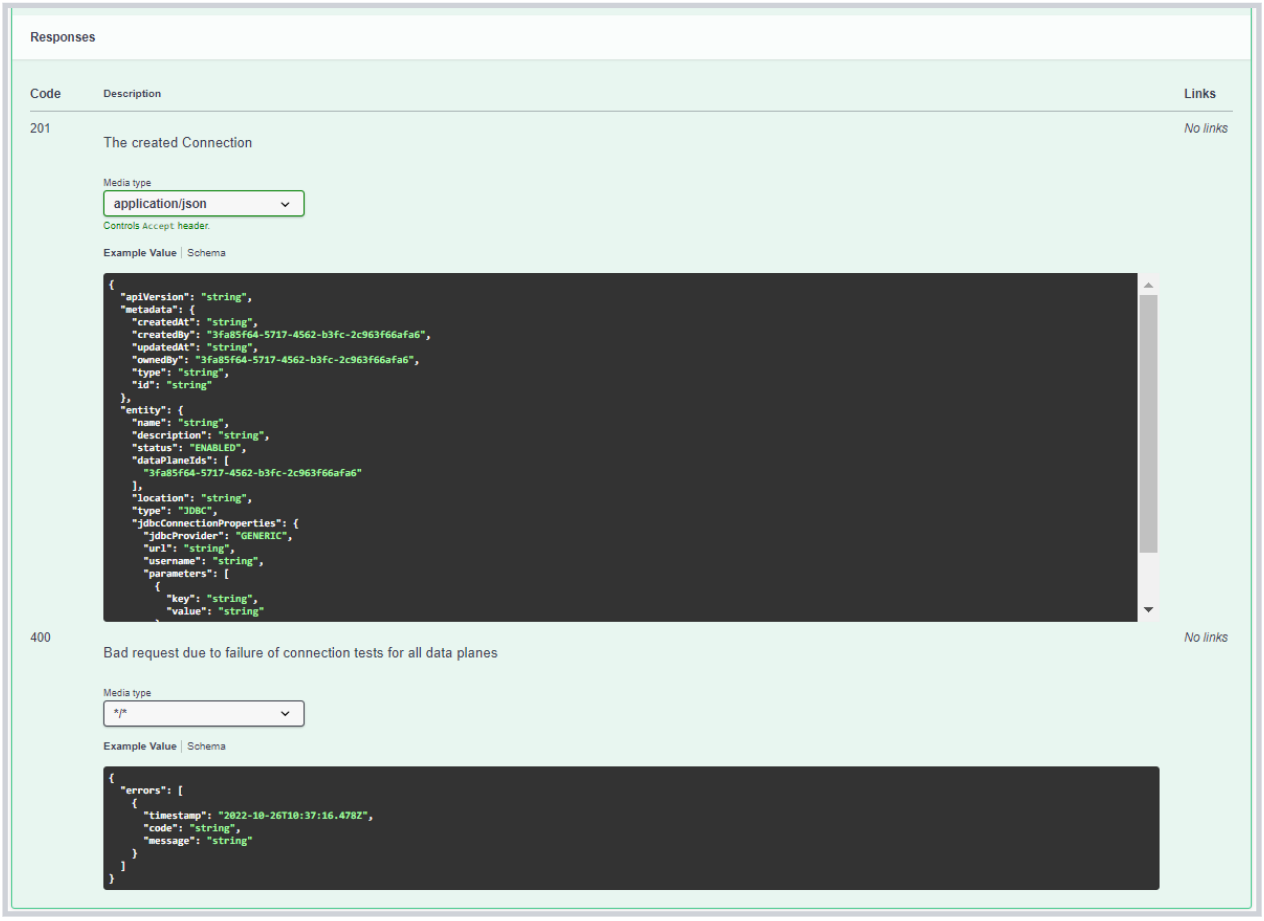
Parameter Name	Parameter Description	Notes
x-exchange-id	The ID corresponding to your exchange on the platform.	The ID of the exchange in which the object exists. You can find the ID by looking at the exchange details on the platform: 1. Log in to the platform. 2. Click **Data Exchange** in the left navigation. 3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**. 4. **Data Exchange ID—**Select and copy the data exchange ID.

Click **Try it out**.

In the **Request body**, you must enter the name, description, and location for the new connection. You must also enter connection properties, including the key-value pair.



The example **Responses** section shows both the status codes and the description for those codes on the platform API. There are also values, which which the platform extracts from the OpenAPI schema.



5.6. Data Class API Description

The Data Class API has endpoints to create, read, update, and delete a data class. You can also:

- view and update terms associated with the data class
- view a data class by its ID

Example Data Class API Endpoint

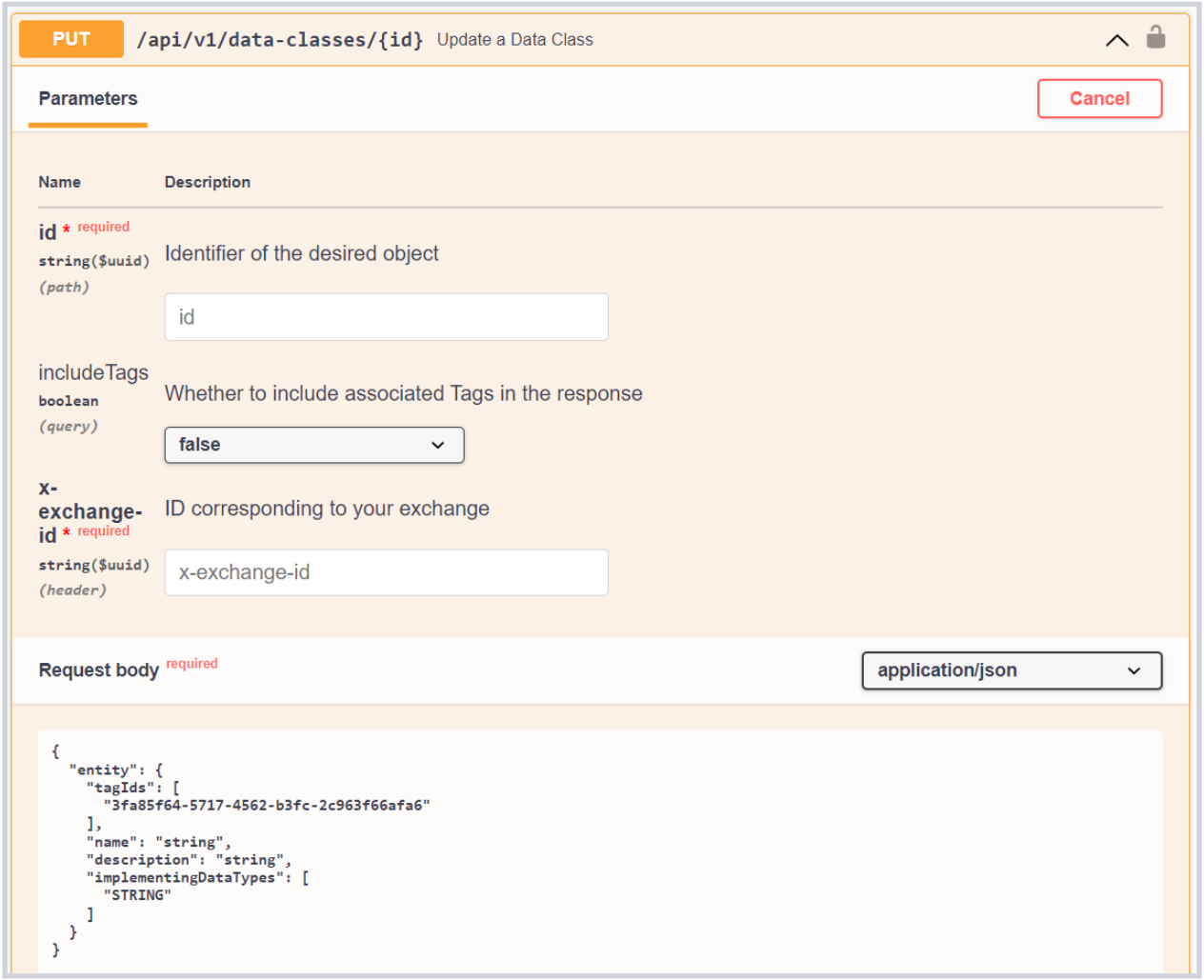
In Swagger, in the **DataClass** API endpoints, click the arrow symbol on the **Update a Data Class** endpoint. For each request in Swagger, you must supply a set of parameters. This endpoint has the following parameters:

Table 5. API Endpoint Parameters (Update a Data Class).

Parameter Name	Parameter Description	Notes
id	Identifier of the desired object.	In this case, the desired object is an existing data class. You can find the ID by looking at the data class details on the platform.
include Tags	Whether to include associated tags in the response.	Boolean (True/False)

Parameter Name	Parameter Description	Notes
x-exchange-id	The ID corresponding to your exchange on the platform.	The ID of the exchange in which the object exists. You can find the ID by looking at the exchange details on the platform: 1. Log in to the platform. 2. Click **Data Exchange** in the left navigation. 3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**. 4. **Data Exchange ID—**Select and copy the data exchange ID.

Enter the name and description of the data class in the **Request body**.



If the data class is updated, a 200 status code returns the message, `The Updated Data Class`, and an example of the schema appears. If the PUT request is unsuccessful, it may return an error such as a 400 status, with the message:

```
Bad request due to an invalid format or because the requested name is
already in use.
```

5.7. Data Plane API Description

The Data Plane API has endpoints to view all data planes or to view data planes within a single exchange.

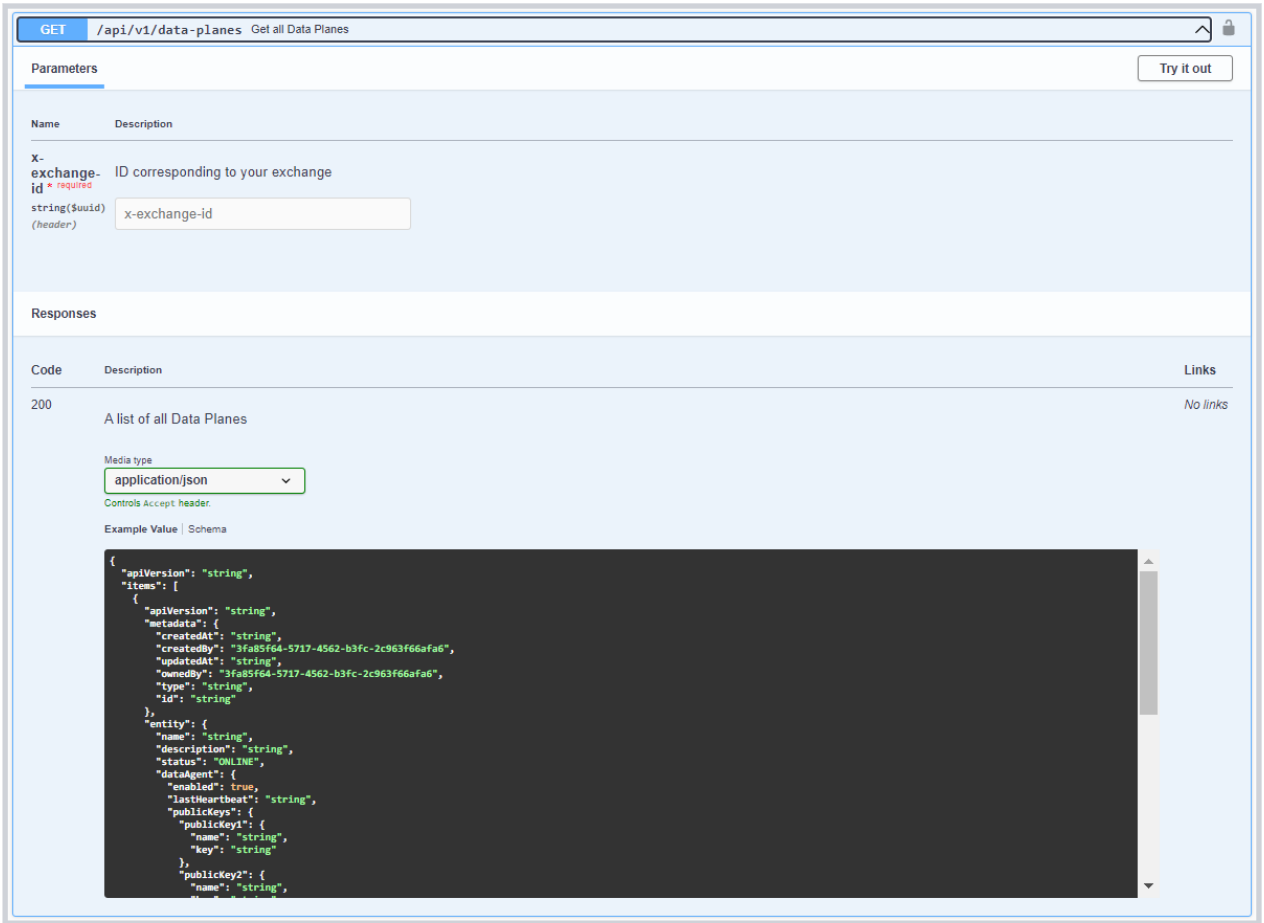
There is no POST DataPlane endpoint. Administrators create data planes within the platform's user interface as part of the exchange.

Example Data Plane API Endpoint

In Swagger, in the Data Plane API endpoints, click the arrow symbol on the **Get all Data Planes** endpoint. For each request in Swagger, you must supply one or more parameters. This endpoint requires the ID of the exchange in which the data plane exists. You can find the ID by looking at the exchange details on the platform:

1. Log in to the platform.
2. Click **Data Exchange** in the left navigation.
3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**.
4. **Data Exchange ID—**Select and copy the data exchange ID.

This image shows how the parameters look in Swagger. Note the option to **Try it out**. You will need to be authorized to interact with an API endpoint.



5.8. Dataset API Description

The Dataset API has endpoints to create, read, update, and delete a dataset. You can find all datasets or filter them by dataset ID. You select datasets for deletion by their ID.

The `datasetid` parameter is required when you create or get an Asset by its ID.

Example Dataset API Endpoint

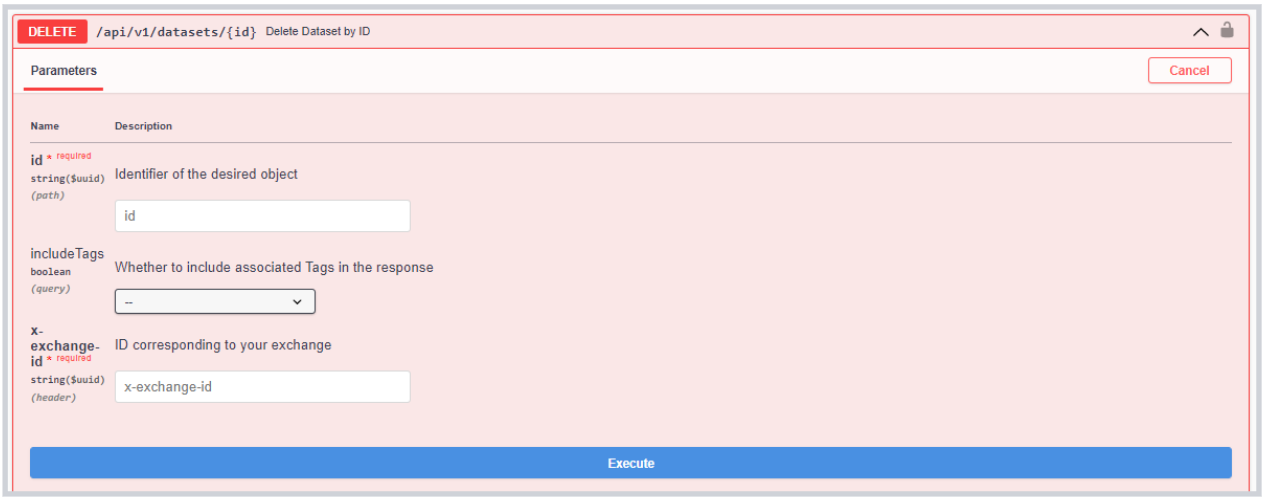
In Swagger, in the **Dataset API** endpoints, click the arrow symbol on the **Delete Dataset by ID** endpoint. For each request in Swagger, you must supply a set of parameters. This endpoint has the following parameters:

Table 6. API Endpoint Parameters (Delete Dataset by ID)

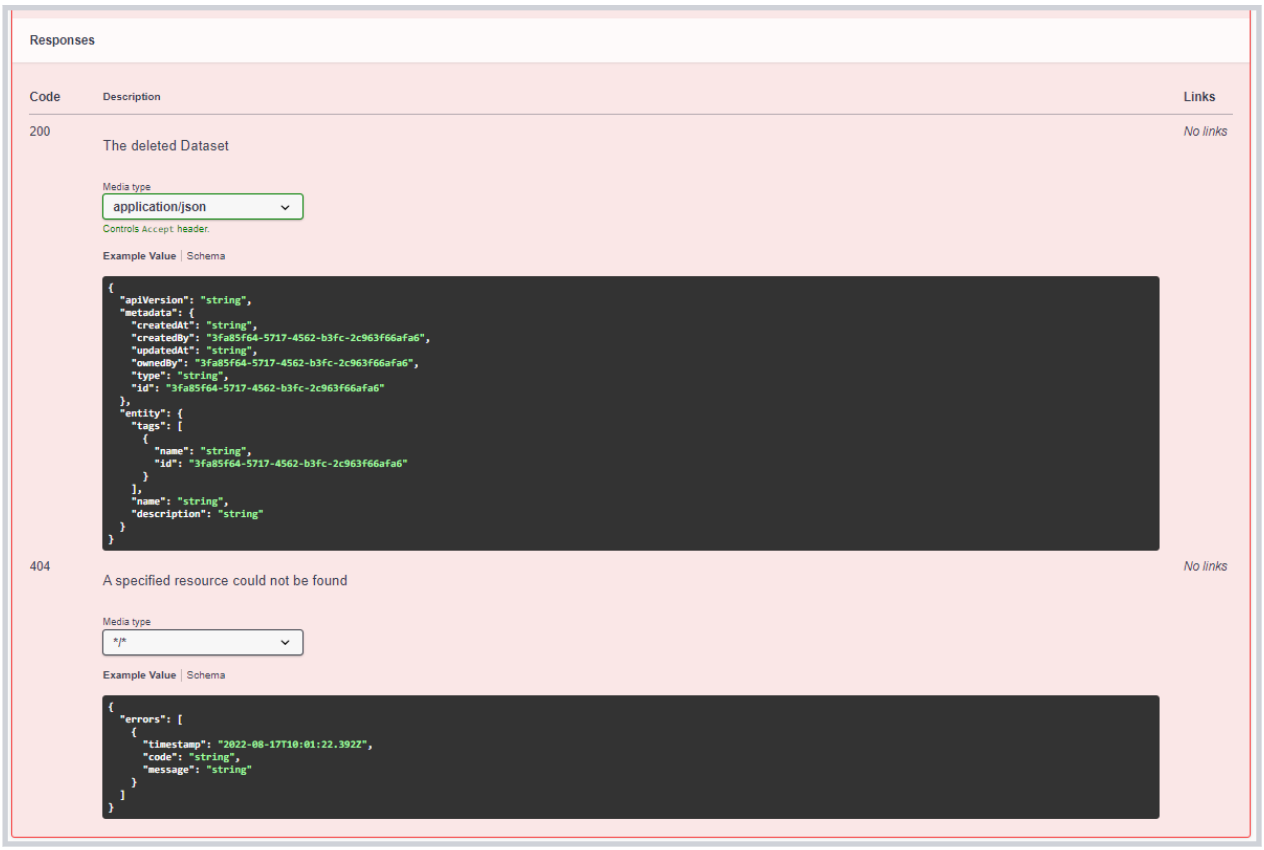
Parameter Name	Parameter Description	Notes
id	Identifier of the desired object.	In this case, the desired object is an existing dataset. You can find the ID by looking at the data class details on the platform.
include Tags	Whether to include associated tags in the response.	Boolean (True/False)
x-exchange-id	The ID corresponding to your exchange on the platform.	The ID of the exchange in which the object exists. You can find the ID by looking at the exchange details on the platform: 1. Log in to the platform. 2. Click **Data Exchange** in the left navigation. 3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**. 4. **Data Exchange ID—**Select and copy the data exchange ID.

Click **Try it out**.

Enter the name of the dataset that you wish to delete in **id**.



This **Responses** section shows both the status codes and the description for those codes in the platform API. There are also example values, which which the platform extracts from the OpenAPI schema.



5.9. Field API Description

The Field API has endpoints to search all fields or to find a field within a selected asset within a dataset. You can also modify field metadata within a selected asset.

Their is no POST Field endpoint. Data owners create fields when they create a new asset.

Example Field API Endpoint

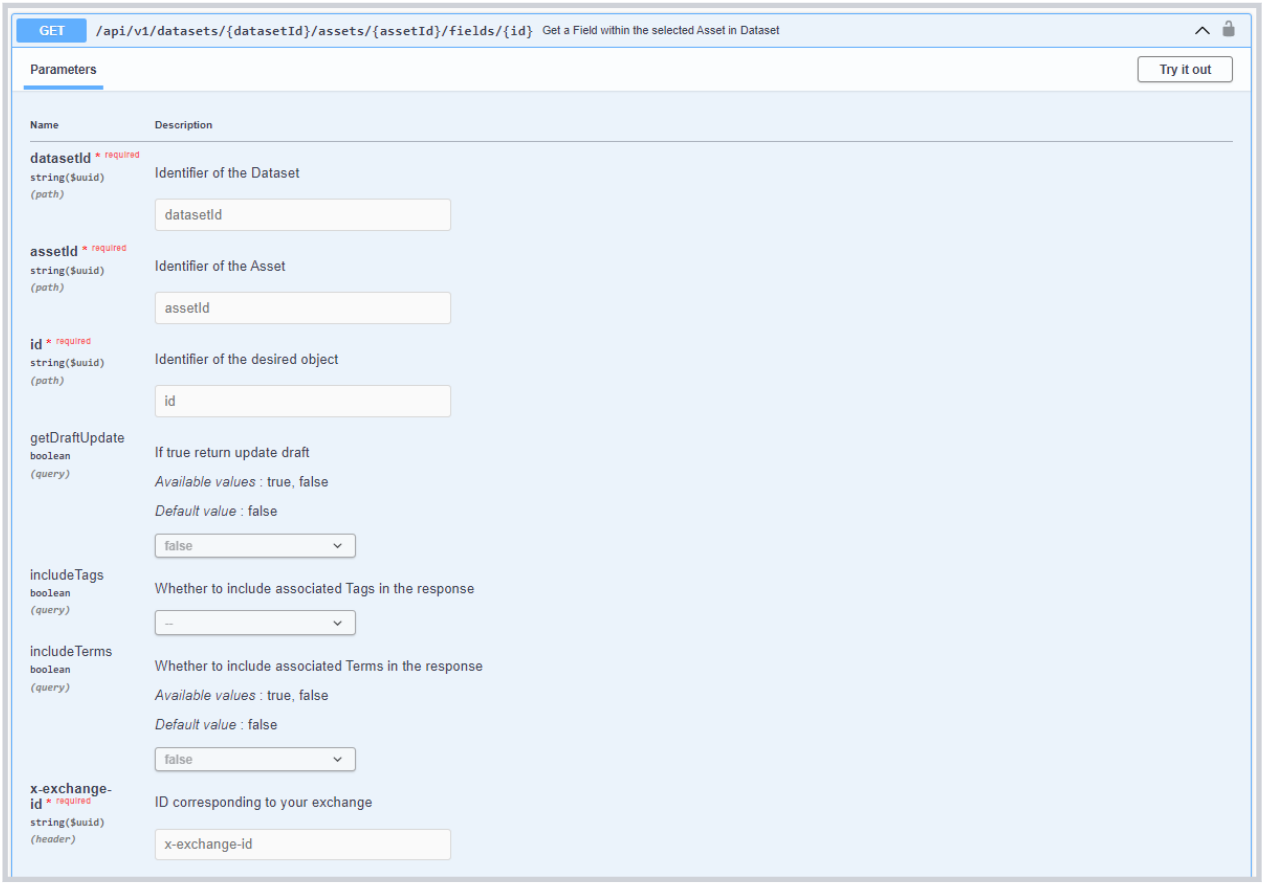
In Swagger, in the **Field API** endpoints, click the arrow symbol on the **Get a Field within the selected Asset in a Dataset** endpoint. For each request in Swagger, you must supply a set of parameters. This endpoint has the following parameters:

Table 7. API Endpoint Parameters (Get a Field within the selected Asset in a Dataset)

Parameter Name	Parameter Description	Notes
datasetid	Identifier of the dataset.	In this case, the desired object is an existing dataset. You can find the ID by looking at the dataset details on the platform.

Parameter Name	Parameter Description	Notes
assetid	Identifier of the asset.	In this case, the desired object is an existing asset. You can find the ID by looking at the asset details on the platform.
id	Set this value to the id of the Field you are interested in.	In this case, the desired object is an existing field. You can find the ID by looking at the field details on the platform.
getDraftUpdate	If true return update draft.	Boolean (True/False)
include Tags	Whether to include associated tags in the response.	Boolean (True/False)
include Terms	Whether to include associated terms in the response.	Boolean (True/False)
x-exchange-id	The ID corresponding to your exchange on the platform.	The ID of the exchange in which the object exists. You can find the ID by looking at the exchange details on the platform: 1. Log in to the platform. 2. Click **Data Exchange** in the left navigation. 3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**. 4. **Data Exchange ID—**Select and copy the data exchange ID.

This image shows how the parameters look in Swagger. Note the option to **Try it out**. You will need to be authorized to interact with an API endpoint.



5.10. Project API Description

The Project API has endpoints to create, read, update, and delete a project. You can also:

- view (GET) all projects
- view all the proxy URLs of a project
- submit a project for review
- update the list of assets associated with a project

Update the general characteristics of a project (name, description, purpose, collaboration mode, and collaboration groups), with the **Update a Project** endpoint, and add assets to the project with the **Set assets on a project** endpoint.

When creating (POST) or updating (PUT) a project you need to specify the **purpose** in the request body. You can obtain the set of allowable purposes by querying `GET /attributes/purpose` in the Attribute API rather than having to look in the platform's user interface.

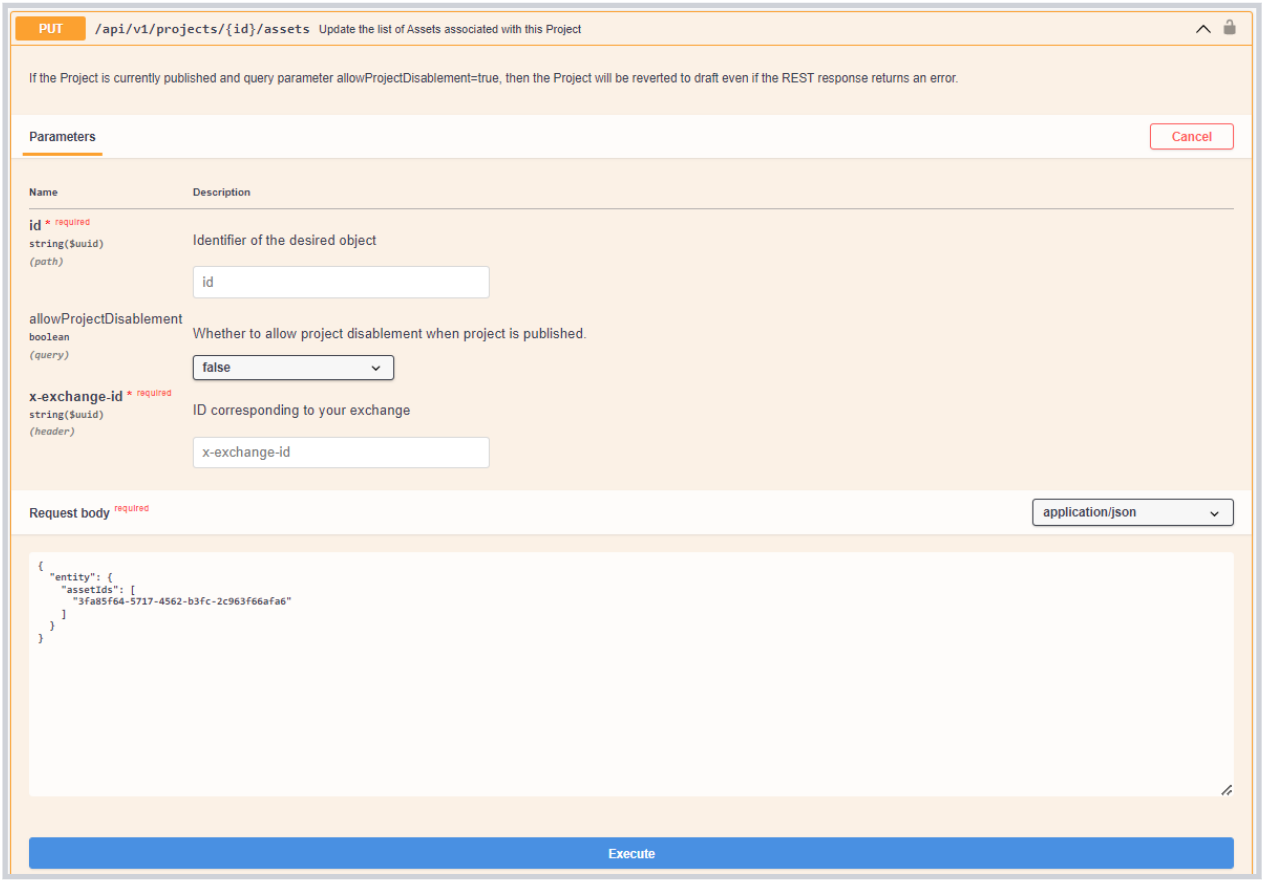
Example Project API Endpoint

In Swagger, in the Project API endpoints, click the arrow symbol on the **Update the list of Assets associated with this project** endpoint. For each request in Swagger, you must supply a set of parameters. This endpoint has the following parameters:

Table 8. Project API Endpoint Parameters (Update the list of Assets associated with this project)

Parameter Name	Parameter Description	Notes
Id	Set this value to the ID of the Term whose associated data classes you are interested in.	In this case, the desired object is the term whose associated data classes you want to retrieve. You can find the ID by looking at the term details n the platform.
allowProjectDisablement	Whether to allow project disablement when project is published.	Boolean: true, false Default value: false
x-exchange-id	The ID corresponding to your exchange on the platform.	The ID of the exchange in which the object exists. You can find the ID by looking at the exchange details on the platform: 1. Log in to the platform. 2. Click **Data Exchange** in the left navigation. 3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**. 4. **Data Exchange ID—**Select and copy the data exchange ID.

This image shows how the parameters look in Swagger. Note the option to **Try it out**. You will need to be authorized to interact with an API endpoint.



As well as entering the parameters outlined in the table, you must enter the ID of the asset that you wish to add to the project.

> **Warning**
>
> Take note of the warning that **if the Project is currently published and query parameter allowProjectDisablement=true, then the Project will be reverted to draft even if the REST response returns an error**.

5.11. Tag API Description

The Tag API has endpoints to create, read, update, and delete a tag. You can view all tags or for view an individual tag by ID.

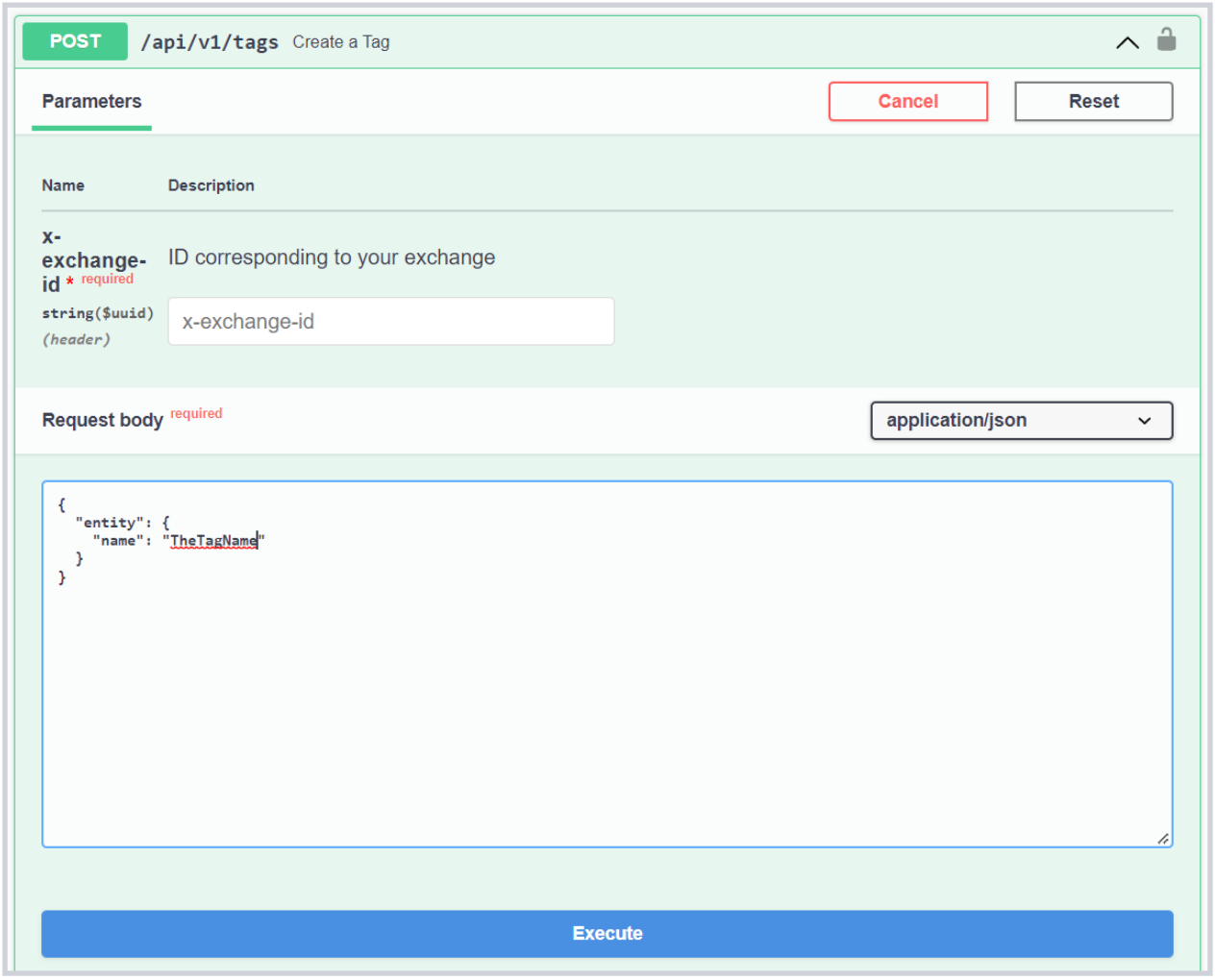
Example Tag API Endpoint

In Swagger, in the Tag API endpoints, click the arrow symbol on the **Create a Tag** endpoint. For each request in Swagger, you must supply parameters. In this endpoint you supply only the exchange ID:

1. Log in to the platform.
2. Click **Data Exchange** in the left navigation.
3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**.

4. **Data Exchange ID—**Select and copy the data exchange ID.

Enter the value of the tag in the **name** field of the **Request body**. In this image a tag is created in the **Try it out** interaction.



If the tag creation is successful, a 201 status code returns the message, `The Created Tag`, and an example of the schema appears. If the POST request is unsuccessful, it may return an error such as a 400 status, with the message:

```
Bad request due to an invalid format or because the requested name is
already in use.
```

5.12. Task API Description

The Task API has endpoints to:

- view all project review request tasks, or to view a single project review request task by ID
- view all asset review request tasks, or to view a single asset review request task by ID
- view all policy review request tasks, or to view a single policy review request task by ID
- publish or reject a policy

- approve or disapprove a project task or asset task

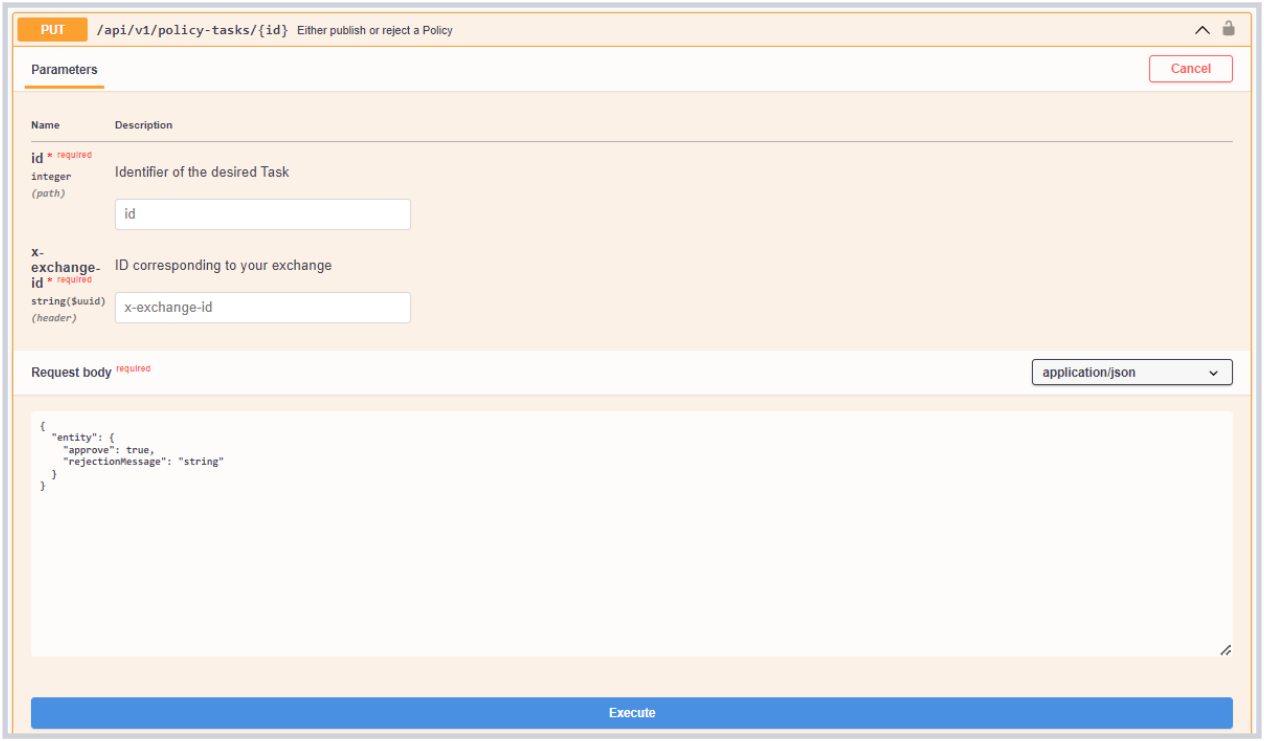
Example Task API Endpoint

In Swagger, in the **Task API** endpoints, click the arrow symbol on the **Either publish or reject a Policy** endpoint. For each request in Swagger, you must supply parameters. In this endpoint, supply the exchange ID and the ID of the Policy you want to publish or reject. To obtain the exchange ID:

1. Log in to the platform.
2. Click **Data Exchange** in the left navigation.
3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**.
4. **Data Exchange ID—**Select and copy the data exchange ID.

You will need to be authorized to interact with an API endpoint.

This image shows how the parameters looks in Swagger. In the approve field enter `true`, to publish, or `false` to reject the policy. In the `rejectionMessage` field enter an optional free text message about the policy rejection. A number of example responses appear below.



5.13. Term API Description

The Term API has endpoints to create, read, update, and delete a term. You can also read and update other terms associated with the term or data classes associated with the term. Delete terms selectively by ID.

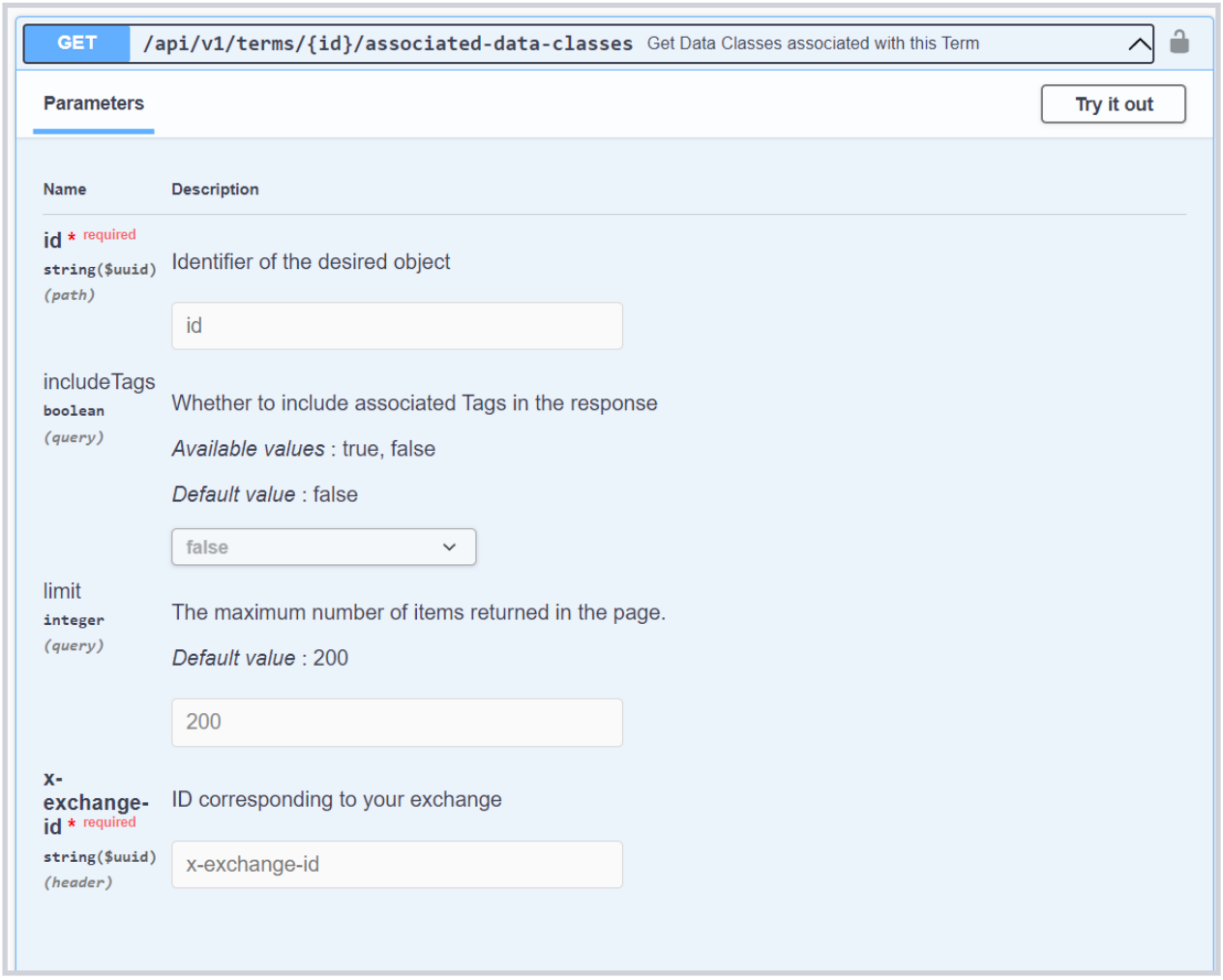
Example Term API Endpoint

In Swagger, in the Term API endpoints, click the arrow symbol on the **Get Data Classes associated with this Term** endpoint. For each request in Swagger, you must supply a set of parameters. This endpoint has the following parameters:

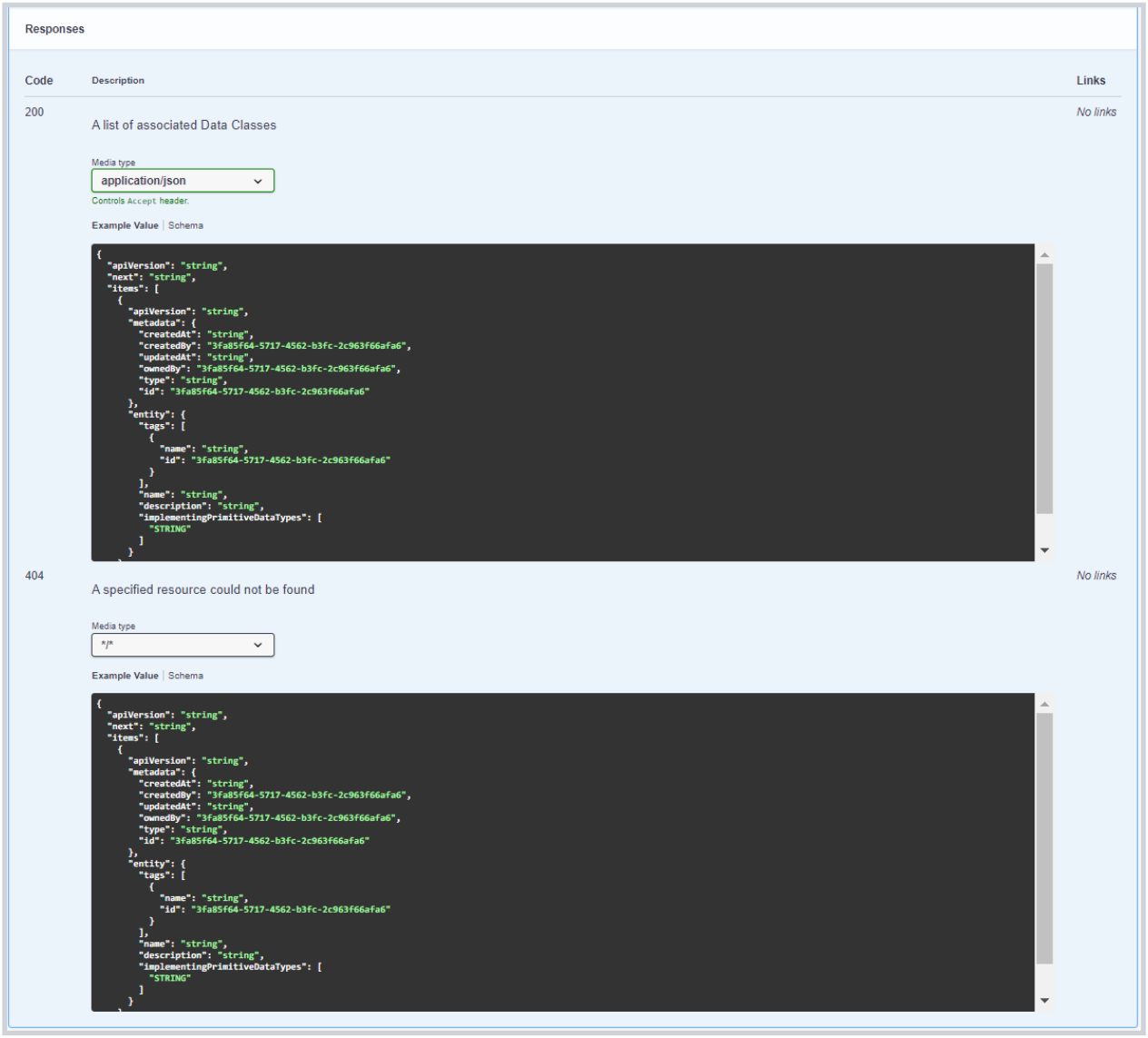
Table 9. Term API Endpoint Parameters (Get Data Classes associated with this Term)

Parameter Name	Parameter Description	Notes
Id	Set this value to the ID of the term whose associated data classes you are interested in.	In this case, the desired object is the term whose associated data classes you want to retrieve. You can find the ID by looking at the term details on the platform.
include Tags	Whether to include associated tags in the response.	Boolean (True/False).
limit	The maximum number of items returned in the page.	
x-exchange-id	The ID corresponding to your exchange on the platform.	The ID of the exchange in which the object exists. You can find the ID by looking at the exchange details on the platform: 1. Log in to the platform. 2. Click **Data Exchange** in the left navigation. 3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**. 4. **Data Exchange ID—**Select and copy the data exchange ID.

This image shows how the parameters look in Swagger. Note the option to **Try it out**. You will need to be Authorized to interact with an API endpoint.



This **Responses** section shows both the status code and the description for that code on the platform API. There is also an example value, which the platform extracts from the OpenAPI schema.



5.14. Transformation API Description

The Transformation API has endpoints to create, delete, update, and view transformations. You can view all transformation or view by transformation ID.

Example Transformation API Endpoint

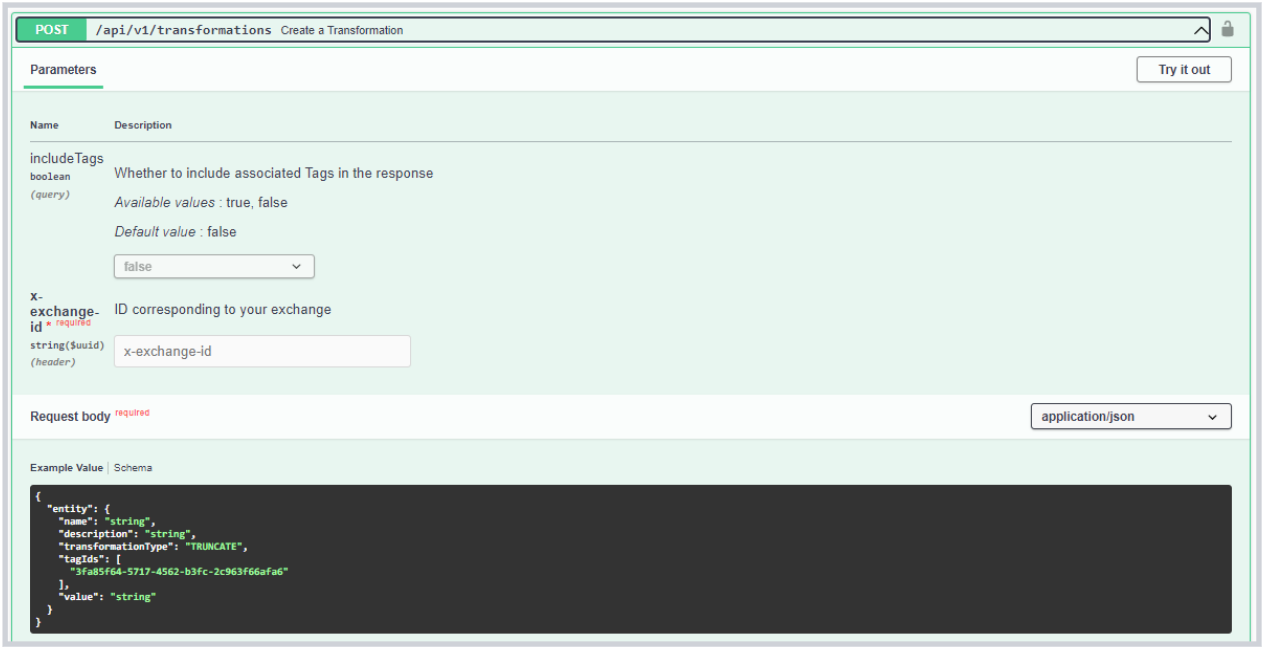
In Swagger, in the **Transformation API** endpoints, click the arrow symbol on the **Create a Transformation** endpoint. For each request in Swagger, you must supply a set of parameters. This endpoint has the following parameters:

Table 10. Transformation API Endpoint Parameters (Create a Transformation)

Parameter Name	Parameter Description	Notes
include Tags	Whether to include associated tags in the response.	Boolean (True/False)
x-exchange-id	The ID corresponding to your exchange on the platform.	The ID of the exchange in which the object exists. You can find the ID by looking at the exchange details on the platform: 1. Log in to the platform. 2. Click **Data Exchange** in the left navigation. 3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**. 4. **Data Exchange ID—**Select and copy the data exchange ID.

To add rules to the transformation policy, you will need to edit it on the platform user interface.

This image shows how the parameters look in Swagger. Note the option to **Try it out**. You will need to be authorized to interact with an API endpoint.



5.15. Transformation Policy API Description

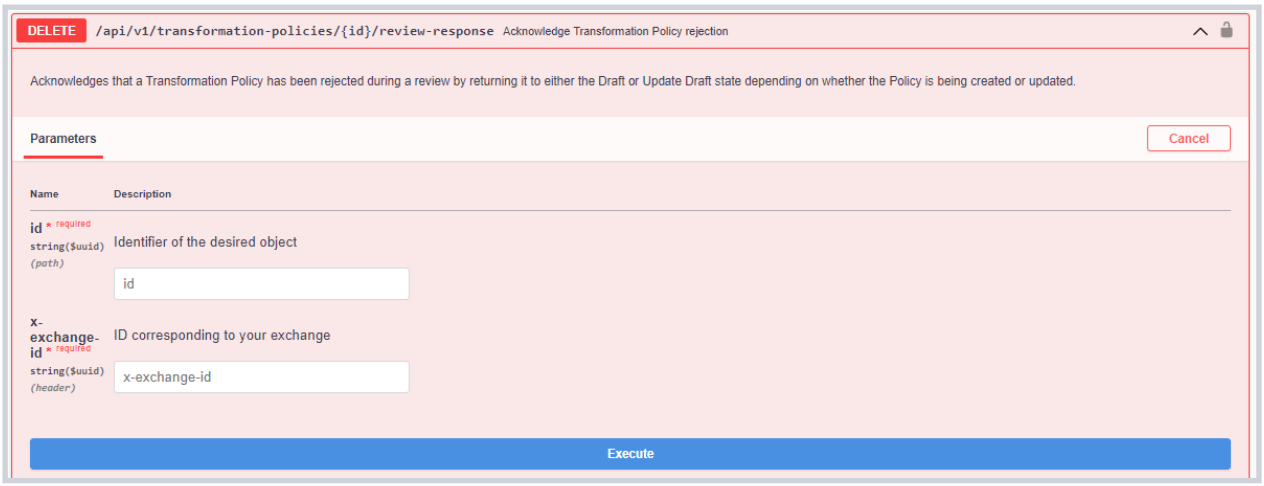
The Transformation Policy API has endpoints to create, read, update, and delete a transformation policy. You can also:

- view a transformation policy's rejection message.

- acknowledge that a transformation policy has been rejected during a review by returning it to either the Draft or Update Draft state, depending on whether the policy is being created or updated. This is a **Delete** endpoint.
- create policies that include conditional triggers.

Example Transformation Policy API Endpoint

In Swagger, in the **Asset API** endpoints, click the arrow symbol on the **Acknowledge Transformation Policy Rejection** endpoint.



For each request in Swagger, you must supply a set of parameters. This endpoint has the following parameters:

Table 11. API Endpoint Parameters (Acknowledge Transformation Policy Rejection)

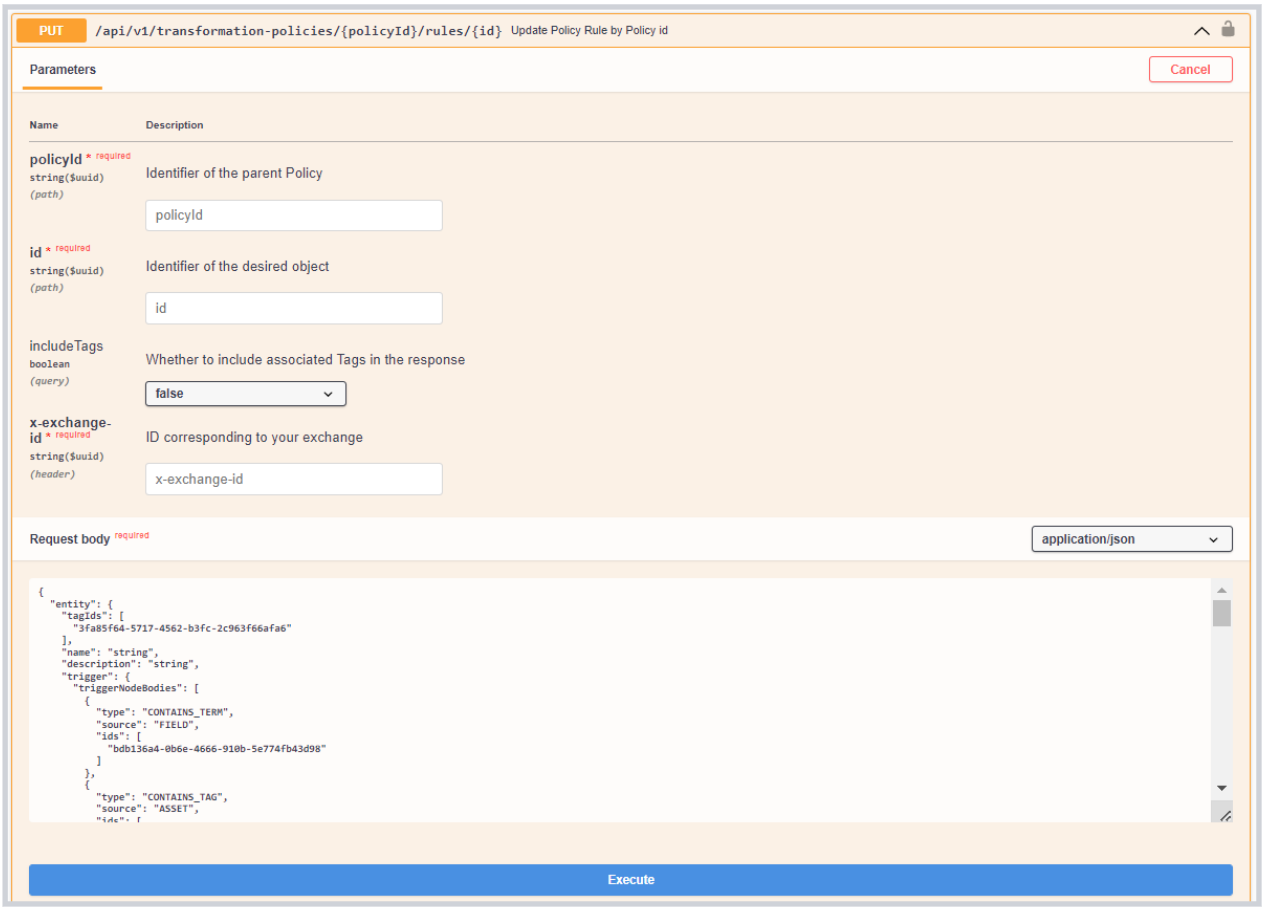
Parameter Name	Parameter Description	Notes
id	Identifier of the transformation policy.	In this case, the desired object is an existing transformation policy. You can find the ID by looking at the policy details on the platform or by searching with the **Get a Transformation Policy by its ID** endpoint.
x-exchange-id	The ID corresponding to your exchange on the platform.	The ID of the exchange in which the object exists. You can find the ID by looking at the exchange details on the platform: 1. Log in to the platform. 2. Click **Data Exchange** in the left navigation. 3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**. 4. **Data Exchange ID—**Select and copy the data exchange ID.

5.16. Transformation Policy Rule API Description

The Transformation Policy Rule API has endpoints to create, read, update, and delete a transformation policy rule. You can also reorder the transformation policy rules within a specified policy and specify a conditional trigger.

Example Transformation Policy Rule API Endpoint

In Swagger, in the **Transformation Policy Rule API** endpoints, click the arrow symbol on the **Update Policy Rule by Policy id** endpoint.



For each request in Swagger, you must supply a set of parameters. This endpoint has the following parameters:

Table 12. API Endpoint Parameters (Update Policy Rule by Policy id)

Parameter Name	Parameter Description	Notes
policyId	Identifier of the transformation policy.	You can find the ID by looking at the policy details on the platform or by searching with the **Get a Transformation Policy by its ID** endpoint.
id	ID of the transformation policy rule.	You can find the ID by looking at the transformation policy details on the platform or by searching with the **Get a Transformation Policy Rule by its ID** endpoint.

Parameter Name	Parameter Description	Notes
include Tags	Whether to include associated tags in the response.	Boolean (True/False)
x-exchange-id	The ID corresponding to your exchange on the platform.	The ID of the exchange in which the object exists. You can find the ID by looking at the exchange details on the platform: 1. Log in to the platform. 2. Click **Data Exchange** in the left navigation. 3. Click the avatar symbol in the top right corner of the page, and select **View Exchange**. 4. **Data Exchange ID—**Select and copy the data exchange ID.

1. Click **Try it out**.
2. Enter the name and description of the transformation policy rule in the **Request body**.
3. For the conditional trigger of the policy rule, enter the `type` and `source` information in the **Request body**.
4. Click **Execute**.

If the transformation policy rule update creation is successful, a 201 status code returns the message, `A single Transformation Policy Rule`, and an example of the schema appears. If the PUT request is unsuccessful, it may return an error such as a 400 status, with the message:

```
Policy in statuses IN_REVIEW and REJECTED should not be updated with a Rule.
```

6. HTTP Status Codes and Error Handling

The platform APIs use HTTP status codes to convey the results of a client request. There are the following standard status code categories, though not all are currently in use on the platform:

- `1xx: Informational`—Communicates transfer protocol-level information.
- `2xx: Success`—The client's request was accepted successfully. For example:
  - a successful `PUT` (update) request for a data class on the platform will return a 200 code, and the response will show the contents of the updated object.
  - a successful `POST` (create) will return a 201 code, and the response will show the contents of the created object.
- `3xx: Redirection`—The client must take additional action in order to complete the request.
- `4xx: Client Error`—This type of error status code indicates a client error. For example, an unsuccessful request on the platform may return a 400 status code: `"Bad Request due to an invalid format or because the requested name is already in use"`.
- `5xx: Server Error`—This type of error status code indicates a server error.

If an API call is not successful, for example if the server is unable to generate a `200 OK` or `201 CREATED` response, then the server will return the appropriate standard error code in a set format.

In Swagger, click the arrow symbol on any endpoint to see example status codes in the **Responses** section.

You can also view every status code in use in the schema. To open the schema, click `/api/v1/api-docs` just below the title on the Swagger page.

7. Glossary of Modern Data Provisioning Terminology

This glossary defines terms that relate to the Privitar Modern Data Provisioning Platform.

A

access control policy	An access control policy is a reusable set of access control rules that serves a business context. An access control policy is a flexible construct that allows you to apply access control rules according to desired conditions. For example, you can write access control policies to define rules that examine and drop rows (records) according to the business condition and the actual data in those records.
access control rule	Access control rules act on the field level. Access control rules examine the actual data and discard each record being queried (requested) according to the rule's conditions.
access request	See project request.
asset	Assets are data structures; for example the tables in an Oracle® or PostgreSQL database.
asset registration request	An asset registration request is an inquiry made by a data owner to add a data asset (a database table, for example) to a dataset. A data guardian approves or denies asset registration requests.
attribute-based access control (ABAC)	Attribute-based access controls (ABACs) are conditional policies and rules that regulate how users' access fields or rows, based on specific attributes, such as location, terms, and tags. ABACs determine how the platform applies policies and rules. In contrast, field-level access controls and record-level access controls determine where (on which assets, rows, or fields) the platform applies the policies and rules.

B

business information	Business information provides definition, structure, and clarity to data assets, *projects*, *policies*, and *rules* by representing the context and semantics of an organization. Business information includes *data classes*, *tags*, *terms*, and *purpose*.

	Business information assists users to find and understand content on the platform and guides when to apply transformations based on attributes and conditions.

C

cell-level transformation	Cell-level transformations allow you to select a different transformation for each distinct record of a specified field (column), that is, a cell, based on varying (logical) conditions. For example, you can instruct the platform to apply different transformations to an identity number or postal code in a given record based on the value of country of residence in a specific cell.
connection	A connection is a configuration for connecting to and reading data from a data source, such as a JDBC connection string. The platform uses this connection information to read metadata attributes from a data asset, to read the data itself, and to write the processed data to the target location.
control plane	The control plane is a logical perimeter that does not have direct access to data but may host components that drive operations in the data plane. The control plane is where policies, rules, projects, and assets are created and managed. The architectural split between the control plane and the data plane allows for configuration, orchestration, and administration (control) without the need to access data, but the ability to process data close to the source within a given jurisdiction. The control plane allows for this by using metadata, data classes, and other representations of the data.

D

data agent	The data agent provides access to the data plane whenever required by the control plane, for example to retrieve the schema for a data asset. It makes a long-lived connection to the *data bridge* on startup.
data bridge	The data bridge is the component in the control plane that handles communication with the data plane. It acts as a Google Remote Procedure Call (gRPC) server. It is replicated, and it sits behind an ingress with a load-balancer.
data class (class)	A data class is a categorization that data owners apply to fields within data assets to indicate the category of data. Within the Privitar Modern Data Provisioning Platform, data

	owners can apply a data class to identify the data's category and ensure that that kind of data is classified consistently throughout your organization. For example, data classes can classify birth dates, national identifiers, and postal codes.
data consumer (consumer)	Data consumers are users on the Privitar Modern Data Provisioning Platform who request and consume data from the platform. Data consumers require direct access to data as part of their job responsibilities.
data exchange (exchange)	A data exchange is a secure online portal where data owners can classify sensitive datasets, and data consumers can access them, without compromising data safety. Each data exchange is separate and different from other data exchanges, being a discrete entity within an enterprise.
data guardian (guardian)	Data guardians are users on the Privitar Modern Data Provisioning Platform who develop and maintain company policies and rules that govern data usage, including how the organization adheres to regulatory and compliance guidelines and requirements. Data guardians are responsible for approving all data requests, including requests to register data on the platform and requests to access data outside the platform.
data owner (owner)	Data owners are users on the Privitar Modern Data Provisioning Platform who register and classify data on the platform. Data owners understand where the data comes from, its quality, its meaning, and for what purposes it can be used.
data plane	A data plane is a set of services used for the reading, writing, and processing of data. It contains a data agent and services capable of provisioning data, such as a data proxy or an integration using the Privitar SDK.
data proxy (proxy)	The data proxy is a Java Database Connectivity proxy (JDBC proxy) that allows data consumers to access sensitive data to which de-identification policies have been applied. It makes calls to the *data bridge* to fetch the information it needs, for example the details of how to connect to the sensitive data and the policies to be applied.
dataset	A dataset is a logical container of assets that is also known as a "data product." Its purpose is to group and facilitate an easier search experience. Data owners make datasets available to data consumers.
data type (type)	A data type is the data's categorization that is read from the source. Examples include: Boolean, integer, and string. The data type references how data is stored in a database,

	and each data type can have a different corresponding transformation. For example, you can store a person's age as an integer or a string.

E

encryption	Encryption is the act of using a cryptographic algorithm to derive a value that is applied to a value in a dataset in such a way that only authorized parties can access the original value. In an encryption scheme, the original value, referred to as plaintext, is encrypted using an encryption algorithm to generate ciphertext that authorized parties can only read if it is decrypted. Encryption can be used as a de-identification technique. It is good practice to encrypt data at rest and in transit. However, while encryption can help protect against unauthorized access, it does not protect the privacy of individuals' data when it's used by people who are authorized. This is known as an insider attack.
enterprise administrator (enterprise admin)	Enterprise administrators are users who perform operations within the Privitar Modern Data Provisioning Platform, such as creating a data exchange, creating a data plane, and configuring a data plane.
exchange	See data exchange.
exchange administrator (exchange admin)	Exchange administrators are users who perform tasks within an data exchange, such as creating and editing a data plane, managing users and groups, and performing everyday administration tasks.

F

field-level access control	Field-level access controls are conditional policies and rules that regulate users' ability to access individual fields of a data asset. Field-level access controls determine which fields of the original dataset the platform retrieves prior to applying data transformation rules. Field-level access controls are implemented through drop field transformation, conditioned on attributes (ABAC), data consumer roles (RBAC), or purpose (PBAC). Field-level access controls determine where (on which fields) the platform applies policies and rules.
field-level transformation	Field-level transformations apply the same transformation to the entire field (column). The platform determines whether to apply a field-level transformation based on the data class of the column.

H

HashiCorp® Vault Key Management System (HashiCorp® Vault KMS)	The HashiCorp® Vault KMS is a key management system (KMS) used to create and control encryption keys, which you use to encrypt data. A KMS is a system for the management (generation, distribution, storage, and more) of cryptographic keys and their metadata.

K

key format	The Privitar Modern Data Provisioning Platform uses "asymmetric" (or public key) encryption, which uses a pair of distinct, yet related keys. One key (the public key) is used for encryption, while the other in the pair (the private key) is used for decryption by an authenticated recipient (user).

L

linkability	"Linkability" is the probability of inferring the original value of transformed data by linking values from different datasets. Applying different tokens to the same value in different datasets reduces the ability to re-identify or de-anonymize data.

P

policy	A policy is a reusable set of rules that serves a business context. Users of the platform can utilize the following types of policies: • *access control policies* • *transformation policies*
privacy enhancing technology (PET)	A privacy enhancing technology is a transformation type used to modify raw data to remove sensitive data elements. The Privitar Modern Data Provisioning Platform offers many PETs. These are the transformation types that data guardians select when building policies.
Privitar NOVLT	Privitar NOVLT is a feature of the Privitar Modern Data Provisioning Platform that applies consistent tokenization without a token vault. NOVLT allows for data linkability across regions. NOVLT also offers faster throughput and less latency than most vaulted solutions.
Privitar Query Engine	The Privitar Query Engine retrieves relevant policies and applies them to assets. The Query Engine transforms SQL queries, and the data retrieved with them, in compliance with the applicable policies.

project	A project is a collection of data assets that a team of data consumers wishes to provision safely. While data owners manage the data assets themselves, data consumers manage projects, including linkability between assets. However, data consumers will not have access to the data within a project until a data guardian approves their access.
project request (request)	A project request is an inquiry made by a data consumer to use the assets in a data project. A data guardian approves or denies project requests.
provision	Provisioning is the act of making data available in a secure way to users and applications.
purpose	A purpose is the data consumer's intended use for the data in a project. Data guardians use purposes as attributes in rules. Examples might include, "to find sources of bad loans" or "to build customer 360 profiles."
purpose-based access control (PBAC)	Purpose-based access controls (PBACs) are conditional policies and rules that regulate how users' access fields, rows, or entire data assets, based on a project purpose selected by a data consumer. PBACs determine how the platform applies policies and rules. In contrast, field-level access controls, and RLACs determine where (on which fields, rows, or assets) the platform applies policies and rules.

R

record-level access control	Record-level access controls are conditional policies and rules that regulate users' ability to access individual records of an asset based on the values of selected fields of the same record. Record-level access controls determine which records of the original dataset the platform retrieves prior to applying transformation rules. Unlike data transformation rules, which are based solely on metadata, record-level access control rules are based on a combination of the data itself and metadata. Record-level access controls determine where (on which records) the platform applies policies and rules. Attribute-based access controls (ABACs), purpose-based access controls (PBACs), and role-based access controls (RBACs) determine how the platform applies those policies and rules.
region	1. In the Privitar Modern Data Provisioning Platform, a region is a name for the geographical location, such as the location of a data exchange or a data agent. This is closely tied to jurisdiction. Some regulations require that data must remain within certain jurisdictions.

2. In cloud computing a region, (aka “geography”), is a named set of cloud resources in the same geographical area. A region is comprised of availability zones.

regular expression (regex)

A regular expression is a series of characters that specifies a pattern to match text and numeric data formats. The Privitar Modern Data Provisioning Platform uses regular expressions to replace text strings and numbers with random characters.

For example, for an initial value of `abcdef`, you could use the following regular expression `[a-z]{6}` to produce an output such as `mvskyc`.

request

See project request and *asset registration request*.

role-based access control (RBAC)

Role-based access controls (RBACs) are conditional policies and rules that regulate how users access fields or rows, based on specific roles provided as user groups.

RBACs determine how the platform applies policies and rules. In contrast, field-level access controls, and record-level access controls determine where (on which fields, rows, or assets) the platform applies policies and rules.

rule

Rules are building blocks of policies. Rules are conditional based on attributes, such as user groups, terms, tags, locations, and so on. Rules also take actions specific to data classes and transformations.

Users of the platform can utilize the following types of rules:

- *access control rules*
- *transformation rules*

S

source connection

A source connection is from where a data owner reads data.

system administrator (SysAdmin)

System administrators are users who perform activities to install and set up the Privitar Modern Data Provisioning Platform. Most of these activities are external to the platform, such as deploying the platform, managing secrets required for installation, performing backup and restore activities, and performing updates to the platform.

T

tag

A tag is a keyword that you can define to describe objects, such as when you want to group objects together or add context to those objects. For example, you might want to define tags that correspond to geography, line of business,

	project names, or applications. Tags help facilitate search and filtering.
target connection	A target connection is to where a data consumer provisions data.
term	Terms are words used within your organization to describe business concepts in plain language. Adding them to the platform ensures consistent use of those words throughout your organization. Terms also lend meaning to physical assets and give them context. Examples of terms could be “account type,” “customer level,” or “credit risk rating.”
tokenization	Tokenization is a form of fine-grained data protection that replaces a clear value with a randomly generated synthetic value that stands in for the original as a "token." The pattern for the tokenized value is configurable and can retain the same format as the original, which means fewer down-stream application changes, enhanced data sharing, and more meaningful testing and development with the protected data.
token vault	A token vault is a secure database (for example, PostGreSQL or AWS DynamoDB) where you can store tokens generated during the de-identification of a dataset. Token vaults are only used for consistent tokenization (always returning the same token for the input value). Each token in a token vault is unique. That is, each token is only returned for one value. Token vaults allow for re-identification. That is, you are able to take a token from a de-identified dataset and look up the original input value.
transformation	A transformation defines a set of behaviors (privacy enhancing technologies) for the platform to execute on a field in a dataset to de-identify it, while still preserving data utility.
transformation policy	A transformation policy is a reusable set of transformation rules that serves a business context. A transformation policy is a flexible construct that allows you to apply transformation rules in the way that best meets your needs. For example, you can write a policy around a regulation (such as HIPAA or GDPR) or around a business context (such as provisioning data for marketing analytics). The order of transformation policies matters. The platform applies them in the order that they are arranged by the data guardian.
transformation rule	Transformation rules are conditional based on attributes, such as user group, terms, tags, location, and so on. Transformation rules apply pre-defined transformations to data classes.

W

watermark

A watermark is a unique digital pattern created by the Privitar platform that is added into the records of de-identified datasets for traceability reasons. The platform adds watermarks to the data during the process of de-identification. They are invisibly embedded and distributed throughout the data, and as a result are robust against tampering and operations, such as filtering or reorganizing of the data.

In the event of a leak or data breach, watermarks can be used to identify the data and plug potential security holes faster. Watermarks can also act as a deterrent to anyone handling the data, encouraging them to take the security of the dataset seriously when they know that the data can be traced.