REST API User Guide

Privitar Data Security Platform, version 2.0.0

Publication date October 12, 2023

Privitar Data Security Platform, version 2.0.0

Table of Contents

1. Welcome to the DSP Platform REST API User Guide

Welcome to the DSP Platform REST API User Guide. If you are reading this document, you are likely to be a skilled developer. You may simply want to know the REST 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 REST 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 Data Security 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. The Platform REST APIs

To see all endpoints and methods, use Swagger. Go to:

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

The address of each API endpoint is:

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

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

There are platform APIs for:

Business Information

- Attributes
- Data Classes
- Tags
- Terms

Data Exchange

- Assets
- Connections
- Data Planes
- Datasets
- Fields

Policies

- Access Control Policies
- Access Control Policy Rules
- Transformations

- Transformation Policies
- Transformation Policy Rules

Projects

- Projects

Tasks

- Tasks

Users and User Groups

- Users
- User Groups

2.1.1. Business Information APIs

> **Note**
>
> You may not remove system attribute types, but you may change their list of valid values.

Table 1. Attribute Endpoints

Endpoint	Description
`GET/api/v1/attributes/{id}`	Retrieve details of an attribute type (such as a purpose).
`PUT/api/v1/attributes/{id}`	Modify an attribute type and its list of valid values.
`DELETE/api/v1/attributes/{id}`	Delete an attribute type.
`GET/api/v1/attributes`	Retrieve a list of all attribute types.
`POST/api/v1/attributes`	Create a new attribute type.

Table 2. Data Class Endpoints

Endpoint	Description
`GET/api/v1/data-classes/{id}`	Retrieve details of a data class.
`PUT/api/v1/data-classes/{id}`	Modify a data class.
`DELETE/api/v1/data-classes/{id}`	Delete a data class.
`GET/api/v1/data-classes/{id}/associated-terms`	Retrieve a list of associated terms for a data class.
`PUT/api/v1/data-classes/{id}/associated-terms`	Update the list of associated terms for a data class.
`GET/api/v1/data-classes`	Retrieve a list of all data classes.
`POST/api/v1/data-classes`	Create a new data class.

Table 3. Tag Endpoints

Endpoint	Description
GET/api/v1/project-tasks/{id}	Retrieve details of a project approval task.
PUT/api/v1/project-tasks/{id}	Approve or reject project tasks.
GET/api/v1/policy-tasks/{id}	Retrieve details of a policy approval task.
PUT/api/v1/policy-tasks/{id}	Approve or reject policy tasks.
GET/api/v1/asset-tasks/{id}	Retrieve details of an asset approval task.
PUT/api/v1/asset-tasks/{id}	Approve or reject asset tasks.
GET/api/v1/project-tasks	Retrieve a list of all project tasks.
GET/api/v1/policy-tasks	Retrieve a list of all policy tasks.
GET/api/v1/asset-tasks	Retrieve a list of all asset tasks.

Table 4. Term Endpoints

Endpoint	Description
GET/api/v1/terms/{id}	Retrieve details of a term.
PUT/api/v1/terms/{id}	Modify a term.
DELETE/api/v1/terms/{id}	Delete a term.
GET/api/v1/terms/{id}/associated-terms	Retrieve a list of associated terms for a term.
PUT/api/v1/terms/{id}/associated-terms	Update the list of associated terms for a term.
GET/api/v1/terms/{id}/associated-data-classes	Retrieve a list of associated data classes for a term.
PUT/api/v1/terms/{id}/associated-data-classes	Update the list of associated classes for a term.
GET/api/v1/terms	Retrieve a list of all terms.
POST/api/v1/terms	Create a new term.

2.1.2. Data Exchange APIs

Table 5. Asset Endpoints

Endpoint	Description
GET/api/v1/datasets/{datasetId}/assets/{id}	Retrieve details of an asset.
PUT/api/v1/datasets/{datasetId}/assets/{id}	Modify an asset.
DELETE/api/v1/datasets/{datasetId}/assets/{id}	Delete a published asset or discard a draft asset.
PUT/api/v1/datasets/{datasetId}/assets/{id}/register	Submit the asset for review.
GET/api/v1/datasets/{datasetId}/assets	Retrieve a list of all assets within a dataset.

Endpoint	Description
`POST/api/v1/datasets/{datasetId}/assets`	Create a new draft asset.

Table 6. Connection Endpoints

Endpoint	Description
`GET/api/v1/connections/{id}`	Retrieve details of a connection.
`PUT/api/v1/connections/{id}`	Modify a connection.
`DELETE/api/v1/connections/{id}`	Delete a connection.
`GET/api/v1/connections`	Retrieve a list of all connections.
`POST/api/v1/connections`	Create a new connection.

Table 7. Data Plane Endpoints

Endpoint	Description
`GET/api/v1/data-planes`	Retrieve a list of data planes.
`GET/api/v1/data-planes/{id}`	Retrieve details of a data plane.

Table 8. Dataset Endpoints

Endpoint	Description
`GET/api/v1/datasets/{id}`	Retrieve details of a dataset.
`PUT/api/v1/datasets/{id}`	Modify a dataset.
`DELETE/api/v1/datasets/{id}`	Delete a dataset.
`GET/api/v1/datasets`	Retrieve a list of all datasets.
`POST/api/v1/datasets`	Create a new dataset.

Table 9. Field Endpoints

Endpoint	Description
`GET/api/v1/datasets/{datasetId}/assets/{assetId}/fields/{id}`	Retrieve details of a field from within an asset.
`PUT/api/v1/datasets/{datasetId}/assets/{assetId}/fields/{id}`	Modify a field.
`GET/api/v1/datasets/{datasetId}/assets/{assetId}/fields`	Retrieve a list of all fields from within an asset.

2.1.3. Policy APIs

Table 10. Access Control Policy Endpoints

Endpoint	Description
`GET/api/v1/access-control-policies/{id}`	Retrieve the details of an access control policy.

Endpoint	Description
`PUT/api/v1/access-control-policies/{id}`	Modify an access control policy.
`DELETE/api/v1/access-control-policies/{id}`	Delete an access control policy.
`PUT/api/v1/access-control-policies/{id}/review-task`	Submit an access control policy for review.
`GET/api/v1/access-control-policies`	Retrieve a list of all access control policies.
`POST/api/v1/access-control-policies`	Create a new draft of an access control policy.
`GET/api/v1/access-control-policies/{id}/review-response`	Retrieve the rejection message for a rejected access control policy.
`DELETE/api/v1/access-control-policies/{id}/review-response`	Discard a rejected draft of an access control policy.

Table 11. Access Control Policy Rule Endpoints

Endpoint	Description
`GET/api/v1/access-control-policies/{policyId}/rules/{id}`	Retrieve the details of an access control rule.
`PUT/api/v1/access-control-policies/{policyId}/rules/{id}`	Modify an access control rule.
`DELETE/api/v1/access-control-policies/{policyId}/rules/{id}`	Delete an access control rule.
`GET/api/v1/access-control-policies/{policyId}/rules`	Retrieve a list of all access control rules from within an access control policy.
`POST/api/v1/access-control-policies/{policyId}/rules`	Create a new draft of an access control rule.

Table 12. Transformation Endpoints

Endpoint	Description
`GET/api/v1/transformations/{id}`	Retrieve the details of a transformation.
`PUT/api/v1/transformations/{id}`	Modify a transformation.
`DELETE/api/v1/transformations/{id}`	Delete a transformation.
`GET/api/v1/transformations`	Retrieve a list of all transformations.
`POST/api/v1/transformations`	Create a new transformation.

Table 13. Transformation Policy Endpoints

Endpoint	Description
`GET/api/v1/transformation-policies/{id}`	Retrieve the details of a transformation policy.
`PUT/api/v1/transformation-policies/{id}`	Modify a transformation policy.
`DELETE/api/v1/transformation-policies/{id}`	Delete a transformation policy.

Endpoint	Description
PUT/api/v1/transformation-policies/{id}/review-task	Submit a transformation policy for approval.
GET/api/v1/transformation-policies	Retrieve a list of all transformation policies.
POST/api/v1/transformation-policies	Create a new draft transformation policy.
GET/api/v1/transformation-policies/{id}/review-response	Retrieve the rejection message for a rejected transformation policy.
DELETE/api/v1/transformation-policies/{id}/review-response	Discard a rejected draft of a transformation policy.

Table 14. Transformation Policy Rule Endpoints

Endpoint	Description
GET/api/v1/transformation-policies/{policyId}/rules	Retrieve the details of a transformation policy rule.
PUT/api/v1/transformation-policies/{policyId}/rules	Modify the order of transformation policy rules.
POST/api/v1/transformation-policies/{policyId}/rules	Create a new draft of a transformation policy rule.
GET/api/v1/transformation-policies/{policyId}/rules/{id}	Retrieve a list of all transformation policy rules from within a transformation policy.
PUT/api/v1/transformation-policies/{policyId}/rules/{id}	Modify a transformation policy rule.
DELETE/api/v1/transformation-policies/{policyId}/rules/{id}	Delete a transformation policy rule.

2.1.4. Project APIs

> **Note**
>
> These endpoints only support consumption projects.

Table 15. Project Endpoints

Endpoint	Description
GET/api/v1/projects/{id}	Retrieve the details of a project.
PUT/api/v1/projects/{id}	Modify a project.
DELETE/api/v1/projects/{id}	Delete a project.
PUT/api/v1/projects/{id}/review-task	Submit the project for review.
PUT/api/v1/projects/{id}/assets	Update and replace the assets associated with a project.
GET/api/v1/projects	Retrieve a list of all projects.
POST/api/v1/projects	Create a new draft project.

Endpoint	Description
GET/api/v1/projects/{id}/proxyurls	Retrieve the proxy URLs of a project.

2.1.5. Task APIs

Table 16. Task Endpoints

Endpoint	Description
GET/api/v1/project-tasks/{id}	Retrieve the details of a project approval task.
PUT/api/v1/project-tasks/{id}	Approve or reject project tasks.
GET/api/v1/policy-tasks/{id}	Retrieve details of a policy approval task.
PUT/api/v1/policy-tasks/{id}	Approve or reject policy tasks.
GET/api/v1/asset-tasks/{id}	Retrieve details of an asset approval task.
PUT/api/v1/asset-tasks/{id}	Approve or reject asset tasks.
GET/api/v1/project-tasks	Retrieve a list of all project tasks.
GET/api/v1/policy-tasks	Retrieve a list of all policy tasks.
GET/api/v1/asset-tasks	Retrieve a list of all asset tasks.

2.1.6. User and User Group APIs

Table 17. User and User Group Endpoints

Endpoint	Description
GET/api/v1/users	Retrieve a list of all users.
GET/api/v1/users/{id}	Retrieve the details of a specific user.
GET/api/v1/users/{id}/user-groups	Retrieve a list of all user groups of which a specific user is a member.
GET/api/v1/user-groups	Retrieve a list of all user groups.
GET/api/v1/user-groups/{id}	Retrieve the details of a specific user group.

2.2. 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`).

2.3. Submitting REST API Requests

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

```
https://<your_DSP_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 bearer token that represents the authorization details
  - for PUT and DELETE endpoints, 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.

3. Use Swagger UI to Explore and Learn

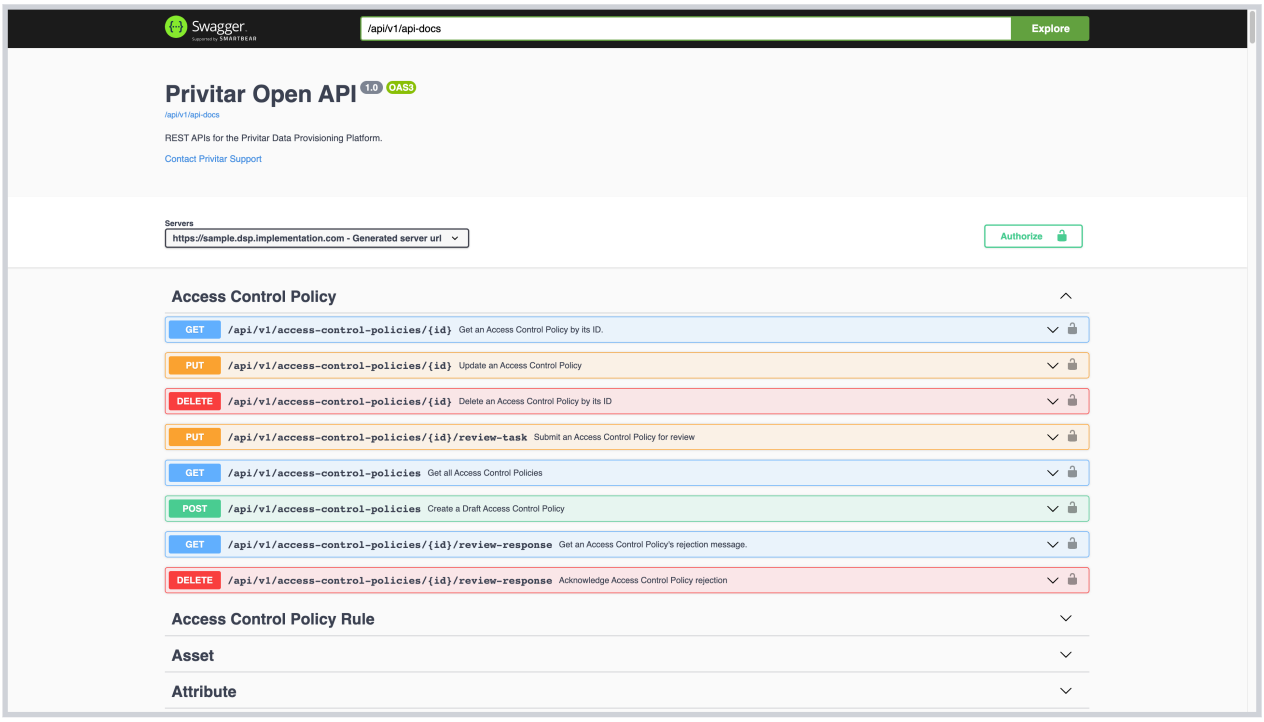
Swagger UI provides a display framework that reads an OpenAPI specification document and uses it to generate an interactive API console. This console allows you to quickly learn about the associated API and experiment with it by sending requests and viewing the responses generated by the requests.

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.

To open the schema definition for all supported API endpoints, click the `/api/v1/api-docs` link at the top of the page.

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.

4. 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.

4.1. 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://<YOUR_HOSTNAME>.com/api/v1/attributes
```

5. 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 USERNAME","password": "YOUR PASSWORD"}' \
--header "accept:*/*" \
--header "content-type:application/json" -k \
--url "https://<your_DSP_environment_url>/graph/enterprise-management/v1/
registration/login"
```

Replace `YOUR USERNAME` and `YOUR PASSWORD` with your platform username and password. Replace `<your_DSP_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.

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 Data Security Terminology

This glossary defines terms that relate to the Privitar Data Security 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, *consumption 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 Data Security 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 Data Security 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 Data Security 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 Data Security 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: 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 Data Security 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 a 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 Data Security 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.

M

migration project	A migration project is a collection of raw data assets that a data owner wishes to mask and move to cloud storage or cross jurisdiction. A data consumer with appropriate authorization may consume the masked assets from the new location where, according to policy, masked data may be reversed.

P

consumption project	A consumption 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 consumption projects, including linkability between assets. However, data consumers will not have access to the data within a consumption project until a data guardian approves their access.
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 Data Security 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 Data Security 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 request (request)	A project request is an inquiry made either by a data consumer to use the assets in a data consumption project or by a data owner to create a migration project. Data guardians approve or deny all 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 consumption 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 consumption 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 (RLAC)	Record-level access controls (RLACs) 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 (RLACs) 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 Data Security 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 Data Security 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 Data Security 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 their fields and give them context. When data consumers are browsing assets, terms allow them to understand the business meaning and semantics of the physical asset. 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 Amazon 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.