

Describe and organize data products and resources in a data mesh

In a data mesh, data consumers discover and trust data products based on how those data products are organized and described. The more consistent the organization and descriptions are, the easier it is for data consumers to discover the right data product for their needs. This consistency is also important for ensuring the interoperability of data products, and that they are used appropriately. The approaches that we recommend an organization take to govern their data products and the underlying data resources are described in this document.

This document is part of a series which describes how to implement a data mesh on Google Cloud. It assumes that you have read and are familiar with the concepts described in [Architecture and functions in a data mesh](/architecture/data-mesh) (/architecture/data-mesh) and [Build a modern, distributed Data Mesh with Google Cloud](https://services.google.com/fh/files/misc/build-a-modern-distributed-datamesh-with-google-cloud-whitepaper.pdf)

(<https://services.google.com/fh/files/misc/build-a-modern-distributed-datamesh-with-google-cloud-whitepaper.pdf>)

.

The series has the following parts:

- [Architecture and functions in a data mesh](/architecture/data-mesh) (/architecture/data-mesh)
- [Design a self-service data platform for a data mesh](/architecture/design-self-service-data-platform-data-mesh) (/architecture/design-self-service-data-platform-data-mesh)
- Describe and organize data products and resources in a data mesh (this document)
- [Build data products in a data mesh](/architecture/build-data-products-data-mesh) (/architecture/build-data-products-data-mesh)
- [Discover and consume data products in a data mesh](/architecture/discover-consume-data-products-data-mesh) (/architecture/discover-consume-data-products-data-mesh)

In large, distributed organizations, data is typically organized according to the functions of individual teams. Financial data, for example, is organized in ways that make sense to individuals working in finance. Logistics data is organized in ways that make sense to people working in areas such as supply chains. The problem is that this approach results in data which is inconsistent across the entire organization.

To help overcome these challenges, the data governance team of a data mesh-driven organization must do the following:

- Define a common vocabulary and approach for creating tags on all the data products of an organization.
- Create a common library of tags to enable the search, discovery, and understanding of data products from diverse businesses.

Metadata as discussed in this document refers to a collection of attributes that describe properties of the data. Metadata is represented through tag templates and tags in Google Cloud Data Catalog. A tag template is a collection of related fields that represent your vocabulary for classifying data entities. A tag is created from the tag template when it is associated with a data entity, and its fields are filled with values that describe attributes of that entity. Tag templates and tags are the building blocks that organizations can use to capture consistent metadata across the business. To learn more about tag templates and tags, see [Understanding the fundamentals of tagging in Data Catalog](#) (/blog/products/data-analytics/cloud-metadata-management-tagging-tips).

This common vocabulary and library serves as a foundational layer for a data mesh, helping to scale understanding of data across an organization. It also enables the central teams to automate privacy requirements and apply policy enforcement consistently across the organization.

How technical metadata is created

The data products from diverse data domains are typically deployed into multiple Google Cloud projects. Their resources (for example, dataset, tables, or views) are represented in Data Catalog as [entries](#) (/data-catalog/docs/entries-and-entry-groups). Entries contain the technical metadata of a resource, such as its unique name, its storage location, or creation time.

Data Catalog automatically creates the entries and the technical metadata for BigQuery datasets, tables, views, and Pub/Sub topics. However, automatic entry creation is not available for resource types such as Cloud Storage files, Looker Block objects, and machine learning (ML) models. To create entries for those resource types, you must use the Data Catalog API or console.

In addition to technical metadata, entries can be annotated with tags by using tag templates.

Tag templates

We recommend that you use the following types of tag templates to annotate data products with their data governance attributes:

- The [data product template](#) (#the_data_product_template)
- The [data resource template](#) (#the_data_resource_template)
- The [data attribute templates](#) (#the_data_attribute_templates_data_sensitivity_and_data_standardization):
 - The data sensitivity template
 - The data standardization template

You can find the YAML file for each of these templates in the [GitHub repository](#) (<https://github.com/GoogleCloudPlatform/datacatalog-templates>) that is associated with this document. This repository also contains a [Python script](#) (https://github.com/GoogleCloudPlatform/datacatalog-templates/blob/master/create_template.py) for creating the templates in Data Catalog. Before using these templates, we recommend that the data governance team in your organization customizes the templates to include the specific terms that your organization uses to describe metadata. This approach enables data producers to create the tags on their own, without assistance from a data governance specialist.


The templates are organized by the resource hierarchy of a *data product*, a *data resource*, and a *data attribute*. These concepts are defined in [Architecture and functions in a data mesh](#) (/architecture/architecture-functions-data-mesh). The templates are described in more detail in the following sections.

The data product template

The data product template, created by the [Python script](#) (https://github.com/GoogleCloudPlatform/datacatalog-templates/blob/master/create_template.py) from the [data_product.yaml](#) (https://github.com/GoogleCloudPlatform/datacatalog-templates/blob/master/data_product.yaml) file, operates at the dataset or folder level. This template is intended to record the intent of a data product. It captures the data domains of the product, the name and description of the product, and its ownership information. The template also captures the business criticality, the expected data latency, and the data retention period of the data product. The data producer should provide most of the field values for each tag in this template, except for the `number_of_data_resources` and `storage_location` fields. The information for these two fields can be inferred from the contents of the data product.

The following image shows what the data product tag template defined by the `data_product.yaml` file looks like in the Data Catalog console:

Template Details

Display Name	Data Product Template
Template ID	data_product
Project ID	data-mesh-344315
Location	us-central1
Visibility	Private 

Fields

ID	Display Name	Type		Description
data_domain	Data domain	ENUM	VALUES ▾	REQUIRED The broad category for the data
data_subdomain	Data subdomain	ENUM	VALUES ▾	The subcategory for the data
data_product_name	Data product name	STRING		The name of the data product
data_product_description	Data product description	STRING		Short description of the data product
data_confidentiality	Data confidentiality	ENUM	VALUES ▾	REQUIRED The confidentiality of the data product
business_criticality	Business criticality	ENUM	VALUES ▾	Business criticality of the data in the data product
business_owner	Business owner	STRING		REQUIRED Name of the business person who owns the data product
technical_owner	Technical owner	STRING		REQUIRED Name of the technical person who owns the data product
number_data_resources	Number of data resources	DOUBLE		Number of data resources in the data product
storage_location	Storage location	ENUM	VALUES ▾	REQUIRED The storage location for the data product
data_retention_period	Data retention period	ENUM	VALUES ▾	REQUIRED The retention period of the data in the data product
data_latency_slo	Latency SLO	ENUM	VALUES ▾	REQUIRED The guaranteed latency of the data in the data product
documentation_link	Documentation Link	STRING		Link to helpful documentation about the data product
access_request_link	Access request link	STRING		How to request access the data product
data_product_status	Data product status	ENUM	VALUES ▾	Status of the data product
last_modified_date	Last modified date	DATETIME		The last time this annotation was modified

Ideally, we recommend that you only tag a data product once from the data product template. The following image is an example of such a tag created from the template:

Data Product Template		
Display name	Value	
Data domain	Finance	
Data subdomain	Financial_Insights	
Data product name	Finwire Archive	
Data product description	This dataset contains financial information about companies and securities obtained from a financial newswire service that has been archived over an extended period of time.	
Data confidentiality	Public	
Business criticality	Medium	
Business owner	John Williams	
Technical owner	Emily Doe	
Number of data resources	409	
Storage location	us-central1	
Data retention period	7_years	
Latency SLO	quarterly	
Documentation Link	https://www.tpc.org/tpc_documents_current_versions/pdf/tpc-di_v1.1.0.pdf	
Access request link	access.corp.company.com/finwire-archive	
Data product status	RELEASED	
Last modified date	August 30, 2022 at 7:00:00 PM GMT-5	

If the data product is stored in BigQuery, you must attach the tag to the BigQuery dataset that contains the resources. If the data product is stored in Cloud Storage, you must attach

the tag to the top-level folder (or bucket) that holds the data resources for the product. In certain cases, some storage systems, such as Pub/Sub, don't provide a construct for grouping related data resources. When you're using the data product template and there's no logical container available, you must tag each individual data resource, for example, every Pub/Sub topic. It's better to have redundant data product tags on a subset of resource types than to be missing those tags.

The data resource template

The data resource template, created by the [Python script](#)

(https://github.com/GoogleCloudPlatform/datacatalog-templates/blob/master/create_template.py)

from the [data_resource.yaml](#)

(https://github.com/GoogleCloudPlatform/datacatalog-templates/blob/master/data_resource.yaml)

file, contains aggregate statistics about the data in the resource. You use this template to tag the data resources in the data product (for example, the tables, views, or files). Analysts can then use this information to determine whether the data within the resource matches their use case.

The data resource template contains information about the data sensitivity of the resource, the number of fields, the number of records, and the volume of the data. It also contains a listing of the global identifiers that are present in the resource. The identifiers are useful for understanding whether one data resource can be joined with another, potentially from a different domain.

The following image shows what the data resource tag template that's defined by the `data_resource.yaml` file looks like in the Data Catalog console:

Template Details

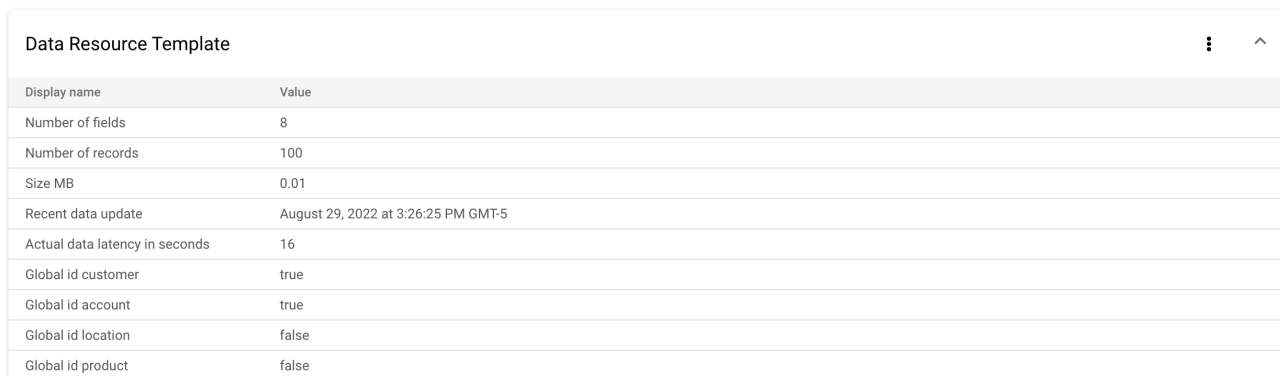
Display Name	Data Resource Template
Template ID	data_resource
Project ID	data-mesh-344315
Location	us-central1
Visibility	Private 

Fields

ID	Display Name	Type		Description
data_sensitivity	Data sensitivity	ENUM	VALUES ▾	Sensitivity of data in the data resource, inferred from the sensitivity of the data elements
num_fields	Number of fields	DOUBLE		Number of fields present in the data resource
num_records	Number of records	DOUBLE		Number of records present in the data resource
size	Size MB	DOUBLE		Size of the data resource in MB
recent_data_update	Recent data update	DATETIME		The most recent data update time
actual_data_latency	Actual data latency in seconds	DOUBLE		The average data latency in seconds over the past 180 days
global_id_customer	Global id customer	BOOL		Global customer identifier present in the data
global_id_account	Global id account	BOOL		Global account identifier present in the data
global_id_location	Global id location	BOOL		Global location identifier present in the data
global_id_product	Global id product	BOOL		Global product identifier present in the data

In the preceding image, the fields for the data resource tag are a mix of user-defined and auto-generated values. The `data_sensitivity` field is inferred from the data attribute tags, which are discussed in the data attribute section. The aggregate statistics fields (`num_fields`, `num_records`, and `size`) are computed by using basic data profiling techniques. When the data resource is the source-of-truth for the data, the global identifier field values are user-defined. When the data resource is derived, the global identifier field values can be auto-generated from data flows. The goal is to auto-generate as many values as possible without compromising on the accuracy of the metadata.

The following image is an example of a tag that is generated by this template:



Data Resource Template	
Display name	Value
Number of fields	8
Number of records	100
Size MB	0.01
Recent data update	August 29, 2022 at 3:26:25 PM GMT-5
Actual data latency in seconds	16
Global id customer	true
Global id account	true
Global id location	false
Global id product	false

Before adopting this template, you must customize the specific global identifiers in the template to match the identifiers that your organization uses. The fields in this template are not intended to be relevant to every resource type. For example, the `num_records` field is not applicable to a Pub/Sub topic or ML model resource type. In those cases, you would omit the non-applicable field from the tag.

The data attribute templates: Data sensitivity and data standardization

The data attribute templates consist of the data sensitivity template and the data standardization template.

The data sensitivity template, created by the [Python script](https://github.com/GoogleCloudPlatform/datacatalog-templates/blob/master/create_template.py)

(https://github.com/GoogleCloudPlatform/datacatalog-templates/blob/master/create_template.py)

from the [`data_sensitivity.yaml`](#)

(https://github.com/GoogleCloudPlatform/datacatalog-templates/blob/master/data_sensitivity.yaml)

file, classifies the sensitivity of individual elements in a data resource. The `sensitive_type` field is a refinement of the `data_sensitivity` field from the data resource template and applies to a single data attribute. The following image shows what the data sensitivity tag template defined by the `data_sensitivity.yaml` file looks like in Data Catalog:

Template Details

Display Name	Data Sensitivity Template
Template ID	data_sensitivity
Project ID	data-mesh-344315
Location	us-central1
Visibility	Private ?

Fields

ID	Display Name	Type		Description
sensitive_field	Sensitive field	BOOL	REQUIRED	Field has sensitive information, such as PII
sensitive_type	Sensitive type	ENUM	VALUES ▾	Type or category of sensitive information

The data standardization template, created by the [Python script](#) (https://github.com/GoogleCloudPlatform/datacatalog-templates/blob/master/create_template.py) from the [data_standardization.yaml](#) (https://github.com/GoogleCloudPlatform/datacatalog-templates/blob/master/data_standardization.yaml)

file, indicates the degree to which the data in a field is standardized in relation to a standard set of terms. This data quality metric states by how much the data element conforms to a standard vocabulary. The following image shows what the data standardization tag template defined by the `data_standardization.yaml` file looks like in the Data Catalog console:

Template Details

Display Name	Data Standardization Template
Template ID	data_standardization
Project ID	data-mesh-344315
Location	us-central1
Visibility	Private ?

Fields

ID	Display Name	Type		Description
degree	Degree	DOUBLE	REQUIRED	Degree to which the data in the field conforms to a controlled vocabulary

You attach data attribute tags to the columns of a data resource. However, you don't need to tag every column. The data sensitivity tags should be created on all sensitive columns and the data standardization tags should be created only on the columns that contain standard terms (for example, a country or an industry).

The following image is an example of the tags that the data sensitivity template applies:

Data Sensitivity Template		VIEW TAG TEMPLATE	✎ EDIT TAG	🗑 REMOVE TAG	✕ CLOSE
Display name	Value				
Sensitive field	true				
Sensitive type	Personal_Identifiable_Information				

The following image is an example of the tags that the data standardization template applies:

Data Standardization Template		VIEW TAG TEMPLATE	EDIT TAG	REMOVE TAG	CLOSE
Display name	Value				
Degree	75				

You can auto-generate both types of data attribute tags from these templates. You can use Cloud Data Loss Prevention to automate the sensitivity tag. With this approach, the output from a [DLP inspection job](#) (/dlp/docs/inspecting-storage) is mapped to a classification for data sensitivity types (for example, personal identifiable information). Similarly, you can automate the standardization tags by joining the data element with a reference column that contains only the terms in use.

Policies

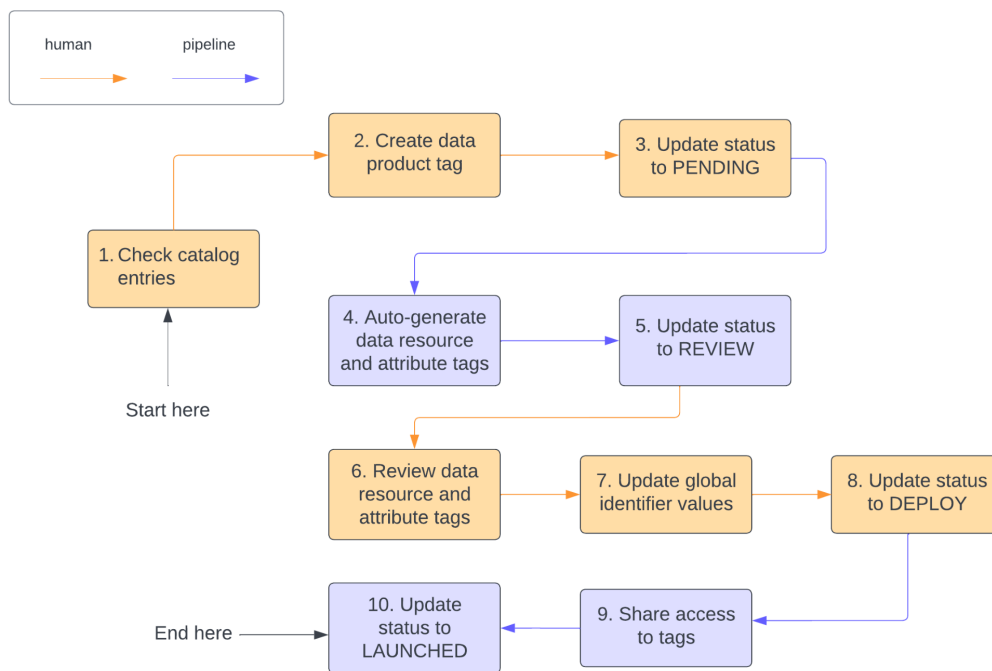
In addition to improving data discovery, the data governance tags that are discussed in this document are also machine readable. Machine-readable tags can help you to automate privacy requirements, and achieve other privacy policy goals. The following are examples of how you can use machine-readable tags to help protect data:

- Deletion policies can use the data retention tag field and technical metadata to decide when to archive or purge a data resource.
- Column-level security policies can use the sensitivity tags to control the access to the sensitive data in those fields.

Registration workflow

When a data producer wants to launch a data product, they follow a process to register their product and all of its resources in the central catalog. This process is the registration workflow. To reduce the manual work involved in this registration workflow, we recommend that the workflow include a data enrichment pipeline that auto-generates the data attribute and data resource tags. You can find [an example](#) (https://github.com/GoogleCloudPlatform/datacatalog-tag-engine/tree/main/examples/product_registration_pipeline) of this enrichment pipeline in the GitHub repository for [datacatalog-tag-engine](#) (<https://github.com/GoogleCloudPlatform/datacatalog-tag-engine>).

The following diagram illustrates the registration workflow from the initial creation of the tags through tag reviews to the actual sharing of the tags with data consumers:



As the preceding diagram shows, the steps in the process are as follows:

1. The data producer checks the catalog entries and creates any entries that are missing. They then do the following:
 - If the data resources for this data product are stored either in BigQuery or Pub/Sub, the data producer verifies that a catalog entry exists for each data resource belonging to the product. Real-time sync in Data Catalog processes automatically create those entries.
 - Depending on where the data resources are created and managed, the following occurs:
 - If the data resources for this product are stored in Cloud Storage, the data producer checks which, if any, catalog entries have been created.
 - If the Cloud Storage files are managed by Dataplex, the Dataplex discovery process creates an entry for each folder in Cloud Storage. Dataplex assumes that all the files in a folder share the same schema and are part of the same logical file. If the files represent distinct assets, however, then you must create separate catalog entries for each file.
 - If the Cloud Storage files are not managed by Dataplex, then the data producer must create catalog entries for both the folder and the files.

2. The data producer creates the data product tag and attaches it to the dataset or top-level folder of the data product. The data producer defines the values for the fields in the tag, and the producer sets the `data_product_status` field to **DRAFT**.
3. When the data producer is ready to share the data product tag, they update the `data_product_status` field to **PENDING**. This status change triggers the data enrichment pipeline.
4. The data enrichment pipeline auto-generates the data resource and the data attribute tags.
5. Upon completion, the enrichment pipeline updates the `data_product_status` field to **REVIEW**.
6. The data producer reviews the data resource and the data attribute tags for accuracy.
7. If a global identifier is missing from the data resource tag, the data producer updates the tag. If an identifier does not belong in the tag, they remove it.
8. Once they have completed their review, the data producer updates the `data_product_status` field to **DEPLOY**.
9. The **DEPLOY** status triggers an access control list (ACL) pipeline that shares access to all the tags for the data product with the data consumers group. The pipeline grants metadata viewer permissions to the BigQuery dataset or Cloud Storage folder that contains the resources for the data product. If the data resources are in BigQuery, the pipeline uses the predefined role of `roles/bigquery.metadataViewer`. If the resources are in Cloud Storage, the pipeline uses a custom role as no predefined role exists for this purpose.
10. After the appropriate ACLs have been updated, the pipeline updates the `data_product_status` field to **LAUNCHED**.

Tooling

To help scale the tagging activities, we recommend that your organization establishes the following:

- A data enrichment pipeline similar to the one described earlier in this document. You can find [an example](https://github.com/GoogleCloudPlatform/datacatalog-tag-engine/tree/main/examples/product_registration_pipeline) (https://github.com/GoogleCloudPlatform/datacatalog-tag-engine/tree/main/examples/product_registration_pipeline)

of this pipeline in the [datacatalog-tag-engine](https://github.com/GoogleCloudPlatform/datacatalog-tag-engine)

(<https://github.com/GoogleCloudPlatform/datacatalog-tag-engine>) GitHub repository.

- A central tagging service that enables independent teams across the organization to create data product, data resource and data attribute tags only on their authorized datasets. A team in a marketing department, for example, should not be allowed to create tags on the data resources owned by a team in a finance department.
- A tag template lifecycle management tool that enables the data governance team to edit a published tag template. This tool must allow the team to add new fields and enum values when required so that they don't need to repeatedly recreate the tag template and all of its tags. You can find [an example](https://github.com/GoogleCloudPlatform/datacatalog-templates/blob/master/evolve_template.py) (https://github.com/GoogleCloudPlatform/datacatalog-templates/blob/master/evolve_template.py)

of this tool in the [GitHub repository](https://github.com/GoogleCloudPlatform/datacatalog-templates/blob/master/README.md).

(<https://github.com/GoogleCloudPlatform/datacatalog-templates/blob/master/README.md>)

for [datacatalog-templates](https://github.com/GoogleCloudPlatform/datacatalog-templates) (<https://github.com/GoogleCloudPlatform/datacatalog-templates>).

What's next

- Check out the [suggested tag templates](https://github.com/GoogleCloudPlatform/datacatalog-templates) (<https://github.com/GoogleCloudPlatform/datacatalog-templates>) and other sample tag templates.
- Learn how to [create bulk tags in Data Catalog](/architecture/tag-engine-and-data-catalog) (/architecture/tag-engine-and-data-catalog) using Tag Engine, an open source extension to Data Catalog.
- For more reference architectures, diagrams, tutorials, and best practices, explore the [Cloud Architecture Center](/architecture) (/architecture).

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (<https://www.apache.org/licenses/LICENSE-2.0>). For details, see the [Google Developers Site Policies](https://developers.google.com/site-policies) (<https://developers.google.com/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2022-12-07 UTC.