Source: cloud-api-reference.md

Dremio Cloud API Reference

The Dremio REST API provides programmatic access to manage your data infrastructure, execute queries, and configure data sources.

Base URLs

All REST endpoints build on one of the supported base URLs:

- **Dremio API**: `https://api.dremio.cloud/v0/`
- **Dremio Login**: `https://login.dremio.cloud/`
- **Iceberg Catalog REST**: `https://catalog.dremio.cloud/api/iceberg/v1/`

Prerequisites

- **Access Token**: Create a personal access token (PAT) for authenticating each of your API calls.
- **Project ID**: Most Dremio API operations manage or use project resources. You can find your Project ID in the Project Settings.

Common Concepts

- **UUIDs**: Many Dremio entities use Universally Unique Identifiers (UUIDs) as identifiers (36 characters).
- **Idempotent Requests**: Add a `requestId` parameter (unique UUID) to POST requests to safely retry them.
- **Timestamps**: Dremio timestamps use ISO 8601 format in UTC: `YYYY-MM-DDTHH:mm:ss.sssZ`.

Query Parameters

Common query parameters supported by many endpoints:

- `pageToken`: Token for retrieving the next page of results.
- `maxChildren` / `maxResults`: Maximum number of items to return.
- `include`: Include non-default response fields (e.g., `permissions`).
- `exclude`: Exclude fields from the response (e.g., `children`).
- `filter`: Filter results.
- `orderBy`: Sort results.

Catalog

Catalog Attributes

˜data`: Array of container objects in the catalog.

˜id`: UUID of the object.

˜path`: Array of strings representing the path.

˜tag`: UUID version tag.

˜type`: `CONTAINER`.

˜containerType`: `SOURCE`, `FOLDER`, or `FUNCTION`.

˜stats`: Object containing `datasetCount` and `datasetCountBounded`.

Retrieve a Catalog

```
GET /v0/projects/{project_id}/catalog
```

Example

```
curl -X GET "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

Source

Source Attributes

˜entityType`: `source`.

˜id`: UUID of the source.

˜type`: Source type (e.g., `AWSGLUE`, `S3`, `SNOWFLAKE`).

˜name`: User-defined name.

˜config`: Configuration options for the source.

˜metadataPolicy`: Metadata update policies.

Create a Source

```
POST /v0/projects/{project_id}/catalog
```

Example

```
curl -X POST "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog" \
  -H "Authorization: Bearer $DREMIO_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "entityType": "source",
    "name": "MySource",
    "type": "S3",
    "config": {
      "bucketName": "my-bucket",
      "authenticationType": "ACCESS_KEY",
      "accessKey": "MY_ACCESS_KEY",
      "accessSecret": "MY_SECRET_KEY"
    }
  }'
```

Retrieve a Source

By ID:

```
GET /v0/projects/{project_id}/catalog/{id}
```

Example (By ID)

```
curl -X GET "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$SOURCE_ID" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

By Path:

```
GET /v0/projects/{project_id}/catalog/by-path/{path}
```

Example (By Path)

```
curl  -X  GET  "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/by-path/MySource"
\
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

Update a Source

```
PUT /v0/projects/{project_id}/catalog/{id}
```

Example

```
curl -X PUT "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$SOURCE_ID" \
  -H "Authorization: Bearer $DREMIO_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "entityType": "source",
    "id": "'"$SOURCE_ID"'",
    "type": "S3",
    "name": "MySourceRenamed",
    "tag": "'"$TAG_VERSION"'",
    "config": {
      "bucketName": "my-new-bucket"
    }
  }'
```

Delete a Source

```
DELETE /v0/projects/{project_id}/catalog/{id}
```

Example

```
curl -X DELETE "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$SOURCE_ID" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

Folder

Folder Attributes

˜entityType`: `folder`.

˜id`: UUID of the folder.

˜path`: Array of strings.

˜children`: List of items inside the folder.

˜accessControlList`: User/role privileges.

Add a Folder

```
POST /v0/projects/{project_id}/catalog
```

Body: `{"entityType": "folder", "path": ["source", "newFolder"]}`

Example

```
curl -X POST "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog" \
  -H "Authorization: Bearer $DREMIO_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "entityType": "folder",
    "path": ["MySource", "MyNewFolder"]
  }'
```

Retrieve a Folder

By ID:

```
GET /v0/projects/{project_id}/catalog/{id}
```

Example (By ID)

```
curl -X GET "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$FOLDER_ID" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

By Path:

```
GET /v0/projects/{project_id}/catalog/by-path/{path}
```

Example (By Path)

```
curl -X GET "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/by-path/MySource/MyFolder" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

Update a Folder

```
PUT /v0/projects/{project_id}/catalog/{id}
```

Example

```
curl -X PUT "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$FOLDER_ID" \
  -H "Authorization: Bearer $DREMIO_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "entityType": "folder",
    "id": "'"$FOLDER_ID"'",
    "path": ["MySource", "MyRenamedFolder"],
    "tag": "'"$TAG_VERSION"'"
```

```
  }'
```

Delete a Folder

```
DELETE /v0/projects/{project_id}/catalog/{id}
```

Example

```
curl -X DELETE "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$FOLDER_ID" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

File

File Attributes

˜type`: `FILE`.

˜id`: Text path of the file (e.g., `dremio:/path/to/file.csv`).

˜path`: Array of strings.

Retrieve a File by Path

```
GET /v0/projects/{project_id}/catalog/by-path/{path}
```

Example

```
curl -X GET "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/by-path/MySource/path/to/file.csv" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

Table

Table Attributes

˜entityType`: `dataset`.

˜type`: `PHYSICAL_DATASET`.

˜id`: UUID of the table.

˜path`: Array of strings.

˜format`: Table format attributes (e.g., Parquet, Text).

˜accelerationRefreshPolicy`: Reflection refresh policy.

˜fields`: Table schema.

Format a File or Folder as a Table

```
POST /v0/projects/{project_id}/catalog
```

Body: `{"entityType": "dataset", "type": "PHYSICAL_DATASET", ...}`

Example

```
curl -X POST "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog" \
  -H "Authorization: Bearer $DREMIO_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "entityType": "dataset",
    "type": "PHYSICAL_DATASET",
    "path": ["MySource", "path", "to", "file.csv"],
    "format": {
      "type": "Text",
      "fieldDelimiter": ","
    }
  }'
```

Retrieve a Table

By ID:

```
GET /v0/projects/{project_id}/catalog/{id}
```

Example (By ID)

```
curl -X GET "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$TABLE_ID" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

By Path:

```
GET /v0/projects/{project_id}/catalog/by-path/{path}
```

Example (By Path)

```
curl -X GET "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/by-path/MySource/MyTable" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

Update a Table

```
PUT /v0/projects/{project_id}/catalog/{id}
```

Example

```
curl -X PUT "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$TABLE_ID" \
  -H "Authorization: Bearer $DREMIO_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "entityType": "dataset",
    "id": "'"$TABLE_ID"'",
    "type": "PHYSICAL_DATASET",
    "path": ["MySource", "MyTable"],
    "tag": "'"$TAG_VERSION"'",
    "accelerationRefreshPolicy": {
      "refreshPeriodMs": 3600000,
      "gracePeriodMs": 10800000
    }
  }'
```

Refresh Reflections on a Table

```
POST /v0/projects/{project_id}/catalog/{id}/refresh
```

Example

```
curl -X POST "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$TABLE_ID/refresh" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

Revert a Table to a File or Folder

```
DELETE /v0/projects/{project_id}/catalog/{id}
```

Example

```
curl -X DELETE "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$TABLE_ID" \
```

```
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

Lineage

Lineage Attributes

`sources`: Array of source objects used by the dataset.

`parents`: Array of parent objects (e.g., joined datasets).

`children`: Array of child objects referencing the dataset.

Retrieve Lineage

```
GET /v0/projects/{project_id}/catalog/{id}/graph
```

Example

```
curl -X GET "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$DATASET_ID/graph"
\
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

User-Defined Function (UDF)

UDF Attributes

`entityType`: `function`.

`id`: UUID of the function.

`functionBody`: The SQL statement of the function.

`functionArgList`: Arguments and data types.

`returnType`: Return data type(s).

`isScalar`: Boolean (true for scalar, false for tabular).

Create a UDF

```
POST /v0/projects/{project_id}/catalog
```

Body: `{"entityType": "function", ...}`

Example

```
curl -X POST "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog" \
  -H "Authorization: Bearer $DREMIO_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "entityType": "function",
    "path": ["MySource", "my_udf"],
    "functionBody": "SELECT 1",
    "returnType": "INTEGER",
    "functionArgList": []
  }'
```

Retrieve a UDF

By ID:

```
GET /v0/projects/{project_id}/catalog/{id}
```

Example (By ID)

```
curl -X GET "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$UDF_ID" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

By Path:

```
GET /v0/projects/{project_id}/catalog/by-path/{path}
```

Example (By Path)

```
curl -X GET
"https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/by-path/MySource/my_udf" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

Update a UDF

```
PUT /v0/projects/{project_id}/catalog/{id}
```

Example

```
curl -X PUT "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$UDF_ID" \
  -H "Authorization: Bearer $DREMIO_TOKEN" \
  -H "Content-Type: application/json" \
```

```
  -d '{
    "entityType": "function",
    "id": "'"$UDF_ID"'",
    "tag": "'"$TAG_VERSION"'",
    "path": ["MySource", "my_udf"],
    "functionBody": "SELECT 2",
    "returnType": "INTEGER",
    "functionArgList": []
  }'
```

Delete a UDF

```
DELETE /v0/projects/{project_id}/catalog/{id}
```

Example

```
curl -X DELETE "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$UDF_ID" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

Tag

Tag Attributes

`tags`: Array of strings.

`version`: UUID version of the tag set.

Create/Modify Tags

```
POST /v0/projects/{project_id}/catalog/{id}/collaboration/tag
```

Example

```
curl -X POST "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$DATASET_ID/collaboration/tag" \
  -H "Authorization: Bearer $DREMIO_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "tags": ["tag1", "tag2"]
  }'
```

Retrieve Tags

```
GET /v0/projects/{project_id}/catalog/{id}/collaboration/tag
```

Example

```
curl                                                -X                                                GET
"https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$DATASET_ID/collaboration/tag"
\
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

Delete Tags

Send an empty array to the create/modify endpoint.

Wiki

Wiki Attributes

˜text`: Markdown text content.

˜version`: Version number.

Create/Update Wiki

```
POST /v0/projects/{project_id}/catalog/{id}/collaboration/wiki
```

Example

```
curl                                               -X                                               POST
"https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$DATASET_ID/collaboration/wiki
" \
  -H "Authorization: Bearer $DREMIO_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "text": "# My Wiki Content\nThis is a description."
  }'
```

Retrieve Wiki

```
GET /v0/projects/{project_id}/catalog/{id}/collaboration/wiki
```

Example

```
curl -X GET "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$DATASET_ID/collaboration/wiki" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

Delete Wiki

Send an empty string to the create/update endpoint.

View

View Attributes

˜entityType`: `dataset`.

˜type`: `VIRTUAL_DATASET`.

˜id`: UUID of the view.

˜sql`: SQL query defining the view.

˜sqlContext`: Context for the SQL query.

˜fields`: View schema.

Create a View

```
POST /v0/projects/{project_id}/catalog
```

Body: `{"entityType": "dataset", "type": "VIRTUAL_DATASET", ...}`

Example

```
curl -X POST "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog" \
  -H "Authorization: Bearer $DREMIO_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "entityType": "dataset",
    "type": "VIRTUAL_DATASET",
    "path": ["MySpace", "MyView"],
    "sql": "SELECT * FROM MySource.MyTable"
  }'
```

Retrieve a View

By ID:

```
GET /v0/projects/{project_id}/catalog/{id}
```

Example (By ID)

```
curl -X GET "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$VIEW_ID" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

By Path:

```
GET /v0/projects/{project_id}/catalog/by-path/{path}
```

Example (By Path)

```
curl -X GET "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/by-path/MySpace/MyView" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

Update a View

```
PUT /v0/projects/{project_id}/catalog/{id}
```

Example

```
curl -X PUT "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$VIEW_ID" \
  -H "Authorization: Bearer $DREMIO_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "entityType": "dataset",
    "id": "'"$VIEW_ID"'",
    "type": "VIRTUAL_DATASET",
    "path": ["MySpace", "MyView"],
    "tag": "'"$TAG_VERSION"'",
    "sql": "SELECT * FROM MySource.MyTable WHERE id > 100"
  }'
```

Refresh Reflections on a View

```
POST /v0/projects/{project_id}/catalog/{id}/refresh
```

Example

```
curl -X POST "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$VIEW_ID/refresh"
\
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

Delete a View

```
DELETE /v0/projects/{project_id}/catalog/{id}
```

Example

```
curl -X DELETE "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$VIEW_ID" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

Grants

Grants Attributes

˜availablePrivileges`: List of available privileges.

˜grants`: Array of grant objects (privileges, granteeType, id).

Create or Update Grants

```
PUT /v0/projects/{project_id}/catalog/{id}/grants
```

Example

```
curl                                        -X                                                PUT
"https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$CATALOG_ID/grants" \
  -H "Authorization: Bearer $DREMIO_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "grants": [
      {
        "granteeType": "USER",
        "id": "'"$USER_ID"'",
        "privileges": ["SELECT", "ALTER"]
      }
    ]
  }'
```

Retrieve Grants

```
GET /v0/projects/{project_id}/catalog/{id}/grants
```

Example

```
curl -X GET "https://api.dremio.cloud/v0/projects/$PROJECT_ID/catalog/$CATALOG_ID/grants" \
  -H "Authorization: Bearer $DREMIO_TOKEN"
```

Source: dremio-cloud-about.md

What is Dremio Cloud? | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/about/

On this page

Dremio Cloud is the agentic lakehouse—a fully managed platform built for AI agents and managed by AI agents. It unifies data across lakes, warehouses, and databases while automating platform and data operations through autonomous workflows.

Unlike traditional systems that require constant tuning and manual operations, Dremio Cloud continuously learns, adapts, and optimizes without human intervention. This creates a self-managing environment where AI agents, applications, and users can seamlessly access consistent, governed data.

Core Capabilities

AI Agent: Dremio Cloud includes a native AI agent integrated with the catalog and governance controls. The agent can discover datasets, generate queries, and help both technical and non-technical users move from questions to repeatable assets quickly. It reduces complexity by automating routine analytics tasks while operating within enterprise security and governance.

AI Semantic Layer: The semantic layer acts as a living dictionary for business entities, metrics, and relationships so agents and analysts share the same context. This reduces ambiguity, prevents hallucinations, and improves accuracy in both AI-generated and human-authored queries. Because definitions travel with the data, consistency is enforced across tools and users.

Unified Data Access: Dremio Cloud provides live access to data across lakes, warehouses, and databases without ETL pipelines or copies. Structured, semi-structured, and unstructured data can be queried together in real time, and built-in AI functions let you process formats such as PDFs and images directly in SQL workflows. At the foundation is Dremio's Open Catalog, built on Apache Polaris and co-created by Dremio. This ensures interoperability with engines like Spark and Trino, integration with catalogs such as AWS Glue and Unity Catalog, and fine-grained role-based access controls.

Autonomous Management: The platform delivers consistent sub-second performance

without manual tuning. Features such as Autonomous Reflections act as an intelligent cache that adapts to workload patterns, while Iceberg clustering and automatic table optimization continuously organize data to address skew and small files. These optimizations happen transparently—no changes to applications or hand-tuning required.

Why It Matters

Dremio Cloud gives AI agents and humans a unified, governed foundation for analytics and decision-making. The AI Semantic Layer ensures a consistent business context, while Autonomous Management capabilities deliver fast, reliable performance across all data. Built on open standards—Apache Iceberg, Apache Polaris, and Apache Arrow—Dremio avoids lock-in while supporting the scale, speed, and trust required for agentic workloads.

Was this page helpful?

Supported Regions | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/about/regions

On this page

When you create a Dremio project, you select the region in which your resources will be created and run. This determines where your metadata is stored and where your compute resources are provisioned.

When selecting a region for your Dremio project, consider the following factors:

Network Connectivity: Select regions with close geographic proximity to your data sources and end users to minimize latency and align with your existing cloud infrastructure

Data Residency: Your region choice determines where your data is stored and processed, so select regions that align with regulatory obligations

The table below lists the supported regions and availability zones:

Region	Code	Availability Zones
US East (N. Virginia)	us-east-1	us-east-1a, us-east-1b, us-east-1c, us-east-1d, us-east-1e, and us-east-1f
US West (Oregon)	us-west-2	us-west-2a, us-west-2b, us-west-2c, us-west-2d

Connection Endpoints

Use these URLs to connect to Dremio:

Interface	Endpoint

Arrow Flight (includes JDBC and ODBC)	data.dremio.cloud
Dremio console	app.dremio.cloud
Dremio JDBC (Legacy)	sql.dremio.cloud
MCP	mcp.dremio.cloud
OAuth	login.dremio.cloud
Open Catalog	catalog.dremio.cloud
REST API	api.dremio.cloud
SCIM	scim.dremio.cloud

Was this page helpful?

Connection Endpoints

Key Concepts | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/about/concepts

On this page

This page defines core Dremio concepts.

Platform

The platform provides the foundational organizational structure for Dremio. It establishes the account hierarchy through organizations and projects, and enables administrators to control user access and allocate resources.

Organizations and Projects

An organization is the top-level account within Dremio where authentication, roles, AI configuration for Model Providers, and billing are managed. An organization can contain multiple projects.

A project isolates compute, data, and resources for team-based data analysis. Projects provide the primary boundary for resource allocation and access control.

When creating a project, you can choose between Dremio-managed storage or provide your own object storage as the project store. This is where Dremio stores materializations, metadata, and Iceberg tables created in your Open Catalog.

Roles and Permissions

Roles define what actions users can perform within Dremio. Permissions control access to specific resources like projects, catalogs, tables, and views. Administrators assign roles to users to manage who can view, create, modify, or delete data objects and configurations.

Catalog

Enables unified data access across heterogeneous sources without requiring data movement or ETL processes.

Open Catalog

Dremio's Open Catalog is a metadata and data management layer built on Apache Polaris. It provides a unified namespace for organizing and accessing data across your Dremio environment with Apache Iceberg support.

Namespaces and Folders

A **namespace** is the top-level container within the Open Catalog that organizes data objects. The catalog name corresponds to your project name, and namespaces are the primary organizational boundary for tables, views, and folders within that catalog.

Folders are directories that contain tables, views, and other folders. Use folders to organize your data into common themes, such as data quality (raw, enrichment, and presentation layers), business units, or geographic regions. Folders can be organized hierarchically for better data governance.

Tables and Views

Tables contain data from your sources, formatted as rows and columns. Tables in the Open Catalog use the Iceberg table format, and Dremio automates maintenance processes including compaction and garbage collection.

Views are virtual tables based on SQL queries. Views do not contain data but provide logical abstractions over tables, other views, or combinations of both. Views leverage the Iceberg view specification for portability across different query engines.

Data Sources

Dremio connects to external systems through data sources without data movement. Supported sources include:

Iceberg Catalogs: AWS Glue Data Catalog, Snowflake Open Catalog, Unity Catalog, and Iceberg REST Catalogs

Object Storage: Amazon S3 and Azure Storage for data lake workloads

Relational Databases: PostgreSQL, MySQL, SQL Server, and other RDBMS systems

Paths

Paths are dot-separated identifiers that specify the location of an object, starting with the source or catalog name, followed by any folders, and ending with the name of the dataset, table, or view. Paths are used to qualify objects when referencing them in queries.

For example, in the path `my_catalog.usage.onprem_deployment.daily_usage`:

`my_catalog` is the catalog name

`usage` and `onprem_deployment` are folders within the catalog

-

`daily_usage` is the table or view name

AI Semantic Layer

Dremio provides multiple ways to discover and understand your data across all connected sources.

Wikis and Labels

Wikis provide detailed descriptions and context for your datasets, like a README for your data. Wikis support Markdown formatting and can include dataset descriptions, source information, and example queries.

Labels enable easy categorization of datasets. For example, add a `PII` label to indicate personally identifiable information, or `Finance` to group financial datasets.

Semantic Search

Semantic search enables you to find objects and entities across your data catalog using natural language queries. It searches object names, metadata, wikis, and labels to return relevant results including sources, folders, tables, views, user-defined functions, Reflections, scripts, and jobs.

Dremio's AI Agent

Dremio's AI Agent enables natural language data exploration and analysis. You can ask questions about your data in natural language, and the AI Agent generates SQL queries and provides insights based on your datasets. The AI Agent works with data from all connected sources and can help create views and analyze patterns across your data catalog.

Query Engine

A Dremio-managed compute engine that automatically starts, scales, and stops based on query demand. Each query engine consists of one or more replicas made up of executor instances that process queries. Every project includes a default preview query engine, which remains available for essential operations and automatically scales down when idle.

Engines

An engine processes jobs that run queries issued by users (either through a client application or through the user interface) or by Dremio (as, for example, when Dremio creates a Reflection that a user has defined). Compute resources for an engine are allocated in the cloud associated with the project. All engines in a project are associated with the same cloud.

Engines are automatically started and stopped by the Dremio control plane and can be configured to have multiple replicas for scaling. For more information, see Manage Engines.

Workload Management

Workload Management enables you to control how compute resources are allocated and prioritized across different types of queries and users to optimize performance for your specific workloads.

Reflections

Reflections accelerate query performance by providing precomputed and optimized copies of source data or query results. They can be Autonomous or manually managed. For more details, see Optimize Performance.

Was this page helpful?

Platform

Organizations and Projects

Roles and Permissions

Catalog

Open Catalog

Data Sources

Paths

AI Semantic Layer

Wikis and Labels

Semantic Search

Dremio's AI Agent

Query Engine

Engines

Workload Management

Reflections

Architecture | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/about/architecture

On this page

At a high level, Dremio's architecture is divided into three planes: data, execution, and control. Dremio is fully hosted with the control and execution planes running on Dremio's tenant.

Data

Dremio's primary data plane is Amazon S3. You can use Dremio-managed storage or bring your own bucket. Dremio can also federate across relational sources, so you can pull data from wherever it resides.

Execution

The execution plane follows a massively parallel processing (MPP) model, where workloads are divided into fragments and spread across a cluster of executors. To minimize repeated reads from S3, Dremio uses caching layers to make queries as fast as possible.

Control

The control plane is where metadata is managed, queries are planned, and security is defined.

How Queries Flow Through Dremio

With an understanding of the layers, we can follow a query through Dremio. A SQL query will start in your organization's slice of our control plane, whether submitted via the web console or via a client connection. The metadata of the datasets being queried informs Dremio how it should plan to access and transform your data. This plan is iterated over, with each iteration applying optimization. This plan, separated into fragments, is passed to a query engine. The query engine will read and transform the data amongst its constituent executors, delivering the results back up to the point of origin.

Was this page helpful?

Source: dremio-cloud-admin.md

Administration | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/

On this page

Dremio administration covers organization-wide and project-level management. Use these tools to configure your environment, manage users and resources, and monitor system performance.

Organization Management

Manage Your Subscription – Upgrade from a free trial to a paid subscription, manage billing and payment methods, and track your organization's usage and costs.

Manage Users – Add users to your organization, configure authentication methods (local or SSO), manage user roles and privileges, and control access to Dremio resources.

Configure Model Providers – Configure AI model providers for Dremio's AI Agent, enabling natural language queries and data exploration across your organization.

Project Management

Manage Projects – Create new projects to isolate compute and data resources for different teams. Configure storage options (Dremio-managed or your own S3 bucket) and manage project-level settings.

Manage Engines – Set up and configure query engines that provide the compute resources for running queries. Choose engine sizes, configure auto-scaling, and manage multiple engine replicas for your projects.

Configure External Engines – Connect industry-standard engines like Apache Spark, Trino, and Apache Flink directly to Dremio without vendor lock-in or proprietary protocols.

Monitor Jobs and Audit Logs – Monitor system health, query performance, and resource utilization. View metrics, logs, and alerts to ensure your Dremio environment is running optimally.

Optimize Performance – Improve query performance and resource efficiency through Reflection management and the results cache.

Shared Responsibility Model

Dremio operates on a shared responsibility model. For detailed information about responsibilities in each area, download the Dremio Shared Responsibility Model.

Was this page helpful?

Organization Management

Project Management

Shared Responsibility Model

Manage Users | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/users

On this page

Manage user access to your Dremio organization through internal authentication or external identity providers. This page covers user types, account management, and administrative tasks.

All users in Dremio are identified by their email address, which serves as their username. Invitations are sent to users' email addresses to set up their accounts.

User Types

Dremio supports two user types with different authentication and management workflows:

Feature	Local Users	SSO Users
Authentication	Password set in Dremio	Identity Provider (IdP) credentials
Credential Management	Within Dremio	Through your IdP
Provisioning	Manual invitation	Manual invitation or SCIM automated
Password Reset	Self-service or admin-initiated	Through IdP only

Local Users

Local users authenticate with passwords managed directly in Dremio. These users must be invited manually. Use local users when you need standalone accounts for contractors, external partners, or testing and development environments.

SSO Users

SSO users authenticate through your organization's identity provider (IdP) like Microsoft Entra ID or Okta, or through social identity providers like Google or GitHub. These users can be invited manually or provisioned automatically via System for Cross-domain Identity Management (SCIM).

What is SCIM?

SCIM is an open standard protocol that automates user provisioning between your identity provider and Dremio. Instead of manually creating and managing users in multiple systems, SCIM keeps everything synchronized automatically. When you add, update, or remove a user in your IdP, those changes propagate to Dremio without manual intervention.

SCIM Provisioning Benefits

When SCIM is configured, Dremio stays synchronized with your IdP. Deleting a user in your IdP automatically reflects in Dremio. Additional benefits of SCIM integration include:

- Automatic user creation and deactivation
- Synchronized user attributes
- Centralized access management

To learn more:

- Configure SCIM with Microsoft Entra ID

- Configure SCIM with Okta
- Configure SCIM with a generic OIDC provider

Manage Your Account

Update Your Password

Local users can reset passwords using either method:

If locked out:

1. On the login screen, enter your email.
2. Click **Forgot Password?**.
3. Check your email for the reset link.

If logged in:

1. Hover over the user icon at the bottom of the navigation sidebar.
2. Select **Account Settings**.
3. Click **Reset Password**.
4. Check your email for the reset link.

Changing your password ends all existing Dremio web sessions.

SSO users must reset passwords through their organization's identity provider. Contact your authentication administrator for assistance.

Update Your Name

You can change your display name at any time:

1. Click the user icon on the side navigation bar.
2. Select **Account Settings**.
3. On the **General Information** page, edit **First Name** and **Last Name**.
4. Click **Save**.

Administrative Tasks

The following tasks require administrator privileges or the CREATE USER privilege.

View All Users

1. Click !Settings on the left navigation bar and choose **Organization settings**.

Select **Users** in the organization settings sidebar.

The table displays all local and SSO users with access to your Dremio instance.

Add a User

SSO users are added automatically when you configure SCIM provisioning.

To add a local user:

Click !Settings on the left navigation bar and choose **Organization settings**.

Select **Users**.

Click **Add Users**.

In the **Email address(es)** field, enter one or more email addresses separated by commas, spaces, or line breaks.

For **Dremio Role**, select the roles where the user will be a member. All users are members of the PUBLIC role by default.

Click **Add**.

Each user receives an invitation email to set up their account. You can configure additional roles after users accept their invitations.

A user's email address serves as their unique identifier and cannot be changed after account creation. If a user's email changes, you must create a new account with the new email address.

If invited users don't receive the email, check spam folders and verify the email addresses are correct.

Edit a User

You can modify a user's name and role assignments. Email addresses cannot be edited—if a user's email changes, you must create a new account.

Click !Settings on the left navigation bar and choose **Organization settings**.

Select **Users**.

Hover over the user's row and click !Edit icon to edit the user.

Details tab: Edit **First Name** and **Last Name**, then click **Save**.

Roles tab: Manage role assignments:

Add roles: Search for and select roles, then click **Add Roles**.

Remove roles: Hover over a role and click **Remove**.

Click **Save**.

Reset a User's Password

This option is only available for local users. SSO users must reset passwords through their identity provider. To send a password reset email to a local user:

- Click !Settings on the left navigation bar and choose **Organization settings**.
- Select **Users**.
- Click the user's name.
- Click **Send Password Reset**.

The user receives an immediate email with reset instructions.

Remove a User

To remove an SSO user:

- First, remove the user from your external identity provider.
- Then follow the steps below to remove them from Dremio.

To remove a local user:

- Click !Settings on the left navigation bar and choose **Organization settings**.
- Select **Users**.
- Click the user's name.
- Click !Remove icon to remove.
- Confirm the deletion.

Related Topics

- Roles
- Privileges
- Configure Identity Providers

Was this page helpful?

- User Types
  - Local Users
  - SSO Users
- Manage Your Account
  - Update Your Password
  - Update Your Name

- Administrative Tasks
- View All Users
- Add a User
- Edit a User
- Reset a User's Password
- Remove a User
- Related Topics

Manage Engines | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/engines/

On this page

An engine is a Dremio entity that manages compute resources. Each engine has one or more replicas that are created for executing queries. An engine replica consists of a group of executor instances defined by the engine capacity.

When you signed up for Dremio, an organization and a project were automatically created. Each new project has a preview engine. The preview engine, by default, will scale down after 1 hour without a query. As the name suggests, it provides previews of queries and datasets. Unlike other engines, the preview engine cannot be disabled.

If an engine is created with a minimum replica of 0, it remains idle until the first query runs. No executor instances run initially. When you run a query, Dremio allocates executors to your project and starts the engine. Engines automatically start and stop based on query load.

Sizes

Dremio provides a standard executor, which is used in all of our query engine sizes. Query engine sizes are differentiated by the number of executors in a replica. For each size, Dremio provides a default query concurrency, as shown in the table below.

Replica Size	Executors per Replica	DCUs	Default Concurrency	Max Concurrency
2XSmall	1	14	2	20
XSmall	1	30	4	40
Small	2	60	6	60
Medium	4	120	8	80
Large	8	240	10	100
XLarge	16	480	12	120
2XLarge	32	960	16	160
3XLarge	64	1920	20	200

States

An engine can be in one of the following states.

State	Icon	Description
Running		Represents an enabled engine (replicas are provisioned automatically or running as per the minimum number of replicas configured). You can use this engine for running queries.
Adding Replica		Represents an engine that is scaling up (adding a replica).
Removing Replica		Represents an engine that is scaling down (removing a replica).
Disabling		Represents an engine that is being disabled.
Disabled		Represents a disabled engine (no engine replicas have been provisioned dynamically or there are no active replicas). You cannot use this engine for running queries.
Starting Engine		Represents an engine that is starting (transitioning from the disabled state to the enabled state).
Stopping Engine		Represents an engine that is stopping (transitioning from the enabled state to the disabled state).
Stopped		Represents an enabled engine that has been stopped (zero replicas running).
Deleting		Represents an engine that is being deleted.

Autoscaling

The autoscaling capability dynamically manages query workload for you based on parameters that you set for the engine. Engine replicas are started and stopped as required to provide a seamless query execution by monitoring the engine replica health.

The following table describes the engine parameters along with their role in autoscaling.

Parameter	Description
Size	The number of executors that make up an engine replica.
Max Concurrency	Maximum number of jobs that can be run concurrently on an engine replica.
Last Replica Auto-Stop	Time to wait before deleting the last replica if the engine is not in use. Not valid when the minimum engine replicas is 1 or higher. The default value is 2 hours.
Enqueued Time Limit	If there are no available resources, the query waits for a period of time that is set by this parameter. When this time limit exceeds, the query gets canceled. You are notified with the timeout during slot reservation error if the query gets canceled due to the query time limit being exceeded. The default value is 5 minutes.
Query Runtime Limit	Time a query can run before it is canceled. The default value is 5 minutes.
Drain Time Limit	Time until an engine replica continues to run after the engine is resized, disabled, or deleted before it is terminated and the running queries fail. The default value is 30 minutes. If there are no queries running on a replica, the engine is terminated without waiting for the drain time limit.

For a query that is submitted to execute on an engine, the control plane assigns an engine replica to that query. Replicas are dynamically created and assigned to queries based on the query workload. The control plane observes the query workload and

current active engine replicas to determine whether to scale up or scale down replicas. Replica is assigned to the query until the query execution is done. For a given engine, Dremio Cloud does not scale up replicas beyond the configured maximum replicas and it does not scale them down below the configured minimum replicas.

Monitor Engine Health

The Dremio Cloud control plane monitors the engines health and manages unhealthy replicas to provide a seamless query execution experience. The replica nodes send periodic heartbeats to the control plane, which determines their liveness. If a periodic heartbeat is not returned from a replica node, the control plane marks that node as unhealthy and replaces it with a healthy one.

View All Engines

To view engines:

In the Dremio Cloud application, click the Project Settings !This is the icon that represents the Project Settings. icon in the side navigation bar.

Select **Engines** in the project settings sidebar to see the list of engines in the project. On the **Engines** page, you can also see engines as per the status. Click the **Status** dropdown list to see the different statuses.

Add an Engine

To add a new engine:

On the **Project Settings** page, select **Engines** in the project settings sidebar. The **Engines** page lists the engines created for the project. Every engine created in a project is created in the cloud account associated with that project.

Click the **Add Engine** button on the top-right of the **Engines** page to create a new engine.

In the **Add Engine** dialog, for **Engine**, enter a name.

(Optional) For **Description**, enter a description.

(Optional) For **Size**, select the size of the engine. The size designates the number of executors.

(Optional) For **Max Concurrency per Replica**, enter the maximum number of jobs that can be run concurrently on this engine.

The following parameters are for **Engine Replicas**:

For **Min Replicas**, enter the minimum number of engine replicas that Dremio Cloud has running at any given time. For auto-stop, set it to 0. To guarantee low-latency query execution, set it to 1 or higher. The default number of minimum replicas is 0.

For **Max Replicas**, enter the maximum number of engine replicas that Dremio Cloud scales up to. The default number of maximum replicas is 1.

tip

You can use these settings to control costs and ensure that excessive replicas are not spun up.

Under **Advanced Configuration**. For **Last Replica Auto-Stop**, enter the time to wait before deleting the last replica if engine is not in use. The default value is 2 hours, and the minimum value is 1 minute.

note

The last replica auto stop is not valid when the minimum number of engine replicas is 1 or higher.

The following parameters are for **Time Limit**:

For **Enable Enqueued Time Limit**, check the box.

For **Enqueued Time Limit**, enter the time a query waits before being cancelled. The default value is 5 minutes.

caution

You should not set the enqueued time limit to less than one minutes, which is the typical time to start a new replica. Changing this setting does not affect queries that are currently running or queued.

(Optional) For **Enable Query Time Limit**, check the box to enable the query time limit for making a query run before it is canceled.

(Optional) For **Query Runtime Limit**, enter the time a query can run before it is canceled. The default query runtime limit is 5 minutes.

For **Drain Time Limit**, enter the time (in minutes) that an engine replica continues to run after the engine is resized, disabled, or deleted before it is terminated and the running queries fail. The default value is 30 minutes. If there are no queries running on a replica, the engine is terminated without waiting for the drain time limit.

Click **Save and Launch**. This action saves the configuration, enables this engine, and allocates the executors.

Edit an Engine

To edit an engine:

On the **Project Settings** page, select **Engines** in the project settings sidebar.

On the **Engines** page, hover over the row of the engine that you want to edit and click on the Edit Engine !This is the icon that represents the Edit Engine settings. icon that appears next to the engine. The **Edit Engine** dialog is opened.

Alternatively, you can click the engine to go to the engine's page. Click the **Edit Engine** button on the top-right of the page.

note

You cannot edit the **Engine name** parameter.

For **Description**, enter a description.

For **Size**, select the size of the engine. The size designates the number of executors.

For **Max Concurrency per Replica**, enter the maximum number of jobs that can be run concurrently on this engine.

The following parameters are for **Engine Replicas**:

For **Min Replicas**, enter the number of engine replicas that Dremio has running at any given time. Set this value to 0 to enable auto-stop, or to 1 or higher to ensure low-latency query execution.

For **Max Replicas**, enter the maximum number of engine replicas that Dremio scales up to.

Under **Advanced Configuration**. **Last Replica Auto-Stop**, enter the time to wait before deleting the last replica if the engine is not in use. The default value is 2 hours.

note

The last replica auto-stop is not valid when the minimum number of engine replicas is 1 or higher.

The following parameters are for **Time Limit**:

For **Enable Enqueued Time Limit**, check the box.

For **Enqueued Time Limit**, enter the time a query waits before being canceled. The default value is 5 minutes.

caution

You should not set the enqueued time limit to less than one minutes, which is the typical time to start a new replica. Changing this setting does not affect queries that are currently running or queued.

(Optional) For **Enable Query Time Limit**, check the box to enable the query time limit for making a query run before it is canceled.

(Optional) For **Query Runtime Limit**, enter the time a query can run before it is canceled. The default query runtime limit is 5 minutes.

For **Drain Time Limit**, enter the time (in minutes) that an engine replica continues to run after the engine is resized, disabled, or deleted before it is terminated and any running queries fail. The default value is 30 minutes. If no queries are running on a replica, the engine is terminated without waiting for the drain time limit.

Click **Save**.

Disable an Engine

You can disable an engine that is not being used:

To disable the engine:

On the **Project Settings** page, select **Engines** in the project settings sidebar. The list

of engines in this project are displayed.

Disable the engine by using the toggle in the **Enabled** column.

Confirm that you want to disable the engine.

Enable an Engine

To enable a disabled engine:

On the **Project Settings** page, select **Engines** in the project settings sidebar. The list of engines in this project are displayed.

Enable the engine by using the toggle in the **Enabled** column.

Confirm that you want to enable the engine.

Delete an Engine

You can permanently delete an engine if it is not in use (this action is irreversible). If queries are running on the engine, then Dremio waits for the drain-time-limit for the running queries to complete before deleting the engine.

caution

An engine that has a routing rule associated with it cannot be deleted. Delete the rules before deleting the engine.

To delete an engine:

On the **Project Settings** page, select **Engines** in the project settings sidebar. The list of engines in this project are displayed.

On the **Engines** page, hover over the row of the engine that you want to delete and click the Delete !This is the icon that represents the Delete settings. icon that appears next to the engine.

Confirm that you want to delete the engine.

Troubleshoot

If your engines are not scaling up or down as expected, you can reference the engine events to see the error that is causing the issue.

To view engine events:

On the **Project Settings** page, select **Engines** in the project settings sidebar. The list of engines in this project are displayed.

On the **Engines** page, click on the engine that you want to investigate.

In the engine details page, click on the **Events** tab to view the scaling events and status of each event.

If any scaling problems persist, contact Dremio Support.

Was this page helpful?

- Sizes
- States
- Autoscaling
- Monitor Engine Health
- View All Engines
- Add an Engine
- Edit an Engine
- Disable an Engine
- Enable an Engine
- Delete an Engine
- Troubleshoot

Monitor | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/monitor/

On this page

As an administrator, you can monitor catalog usage and jobs in the Dremio console. You can also use the Dremio APIs and SQL to retrieve information about jobs and events for the projects in your organization.

Monitor the Dremio Console

The Monitor page in the Dremio console allows you to monitor usage across your project, making it easier to observe patterns, analyze the resources being consumed by your data platform, and understand the impact on your users. You must be a member of the `ADMIN` role to access the Monitor page.

Catalog Usage

The data visualizations on the Monitor page point you to the most queried data and folders in a catalog.

Go to **Settings** > **Monitor** to view your catalog usage. When you open the Monitor page, you are directed to the Catalog Usage tab by default where you can see the following metrics:

- A table of the top 10 most queried datasets within the specified time range, including for each the number of linked jobs, the percentage of linked jobs in which the dataset was accelerated, and the total number of Reflections defined on the dataset
- A table of the top 10 most queried source folders within the specified time range,

including for each the number of linked jobs and the top users of that folder

note

A source can be listed in the top 10 most queried source folders if the source contains a child dataset that was used in the query (for example, `postgres.accounts`). Queries of datasets in sub-folders (for example, `s3.mybucket.iceberg_table`) are classified by the sub-folder and not the source.

All datasets are assessed in the metrics on the Monitor page except for datasets in the system tables and the information schema.

The metrics on the Monitor page analyze only user queries. Refreshes of data Reflections and metadata refreshes are excluded.

Jobs

The data visualizations on the Monitor page show the metrics for queries executed in your project, including statistics about performance and utilization.

Go to **Settings** > **Monitor** > **Jobs** to open the Jobs tab and see an aggregate view of the following metrics for the jobs that are running in your project:

- A report of today's job count and failed/canceled rate in comparison to yesterday's metrics
- A list of the top 10 most active users within the specified time range, including the number of linked jobs for each user
- Total jobs accelerated, total job time saved, and average job speedup from Autonomous Reflections over the past month
- Total number of jobs accelerated by autonomous and manual Reflections over time
- A graph showing the total number of completed and failed jobs over time (aggregated hourly or daily)
- A graph of all completed and failed jobs according to their engine (aggregated hourly or daily)
- A graph of all job states showing the percentage of time consumed for each state (aggregated hourly or daily)
- A table of the top 10 longest running jobs within the specified time range, including the linked ID, duration, user, query type, and start time of each job

To examine all jobs and the details of specific jobs, see Viewing Jobs.

You can create reports of jobs in other BI tools by leveraging the `sys.project.history.jobs` table.

Monitor with Dremio APIs and SQL

Administrators can use the Dremio APIs and SQL to retrieve information about the jobs and events in every project in the organization. This information is useful for further

monitoring and analysis.

Before you begin, make sure that you are assigned to the ADMIN role for the organization whose information you want to retrieve. You also need a personal access token (PAT) to make the necessary API requests.

The code examples in this section are written in Python.

The procedure below provides individual code examples for retrieving project IDs, retrieving information for jobs and events, saving query results to Parquet files, and uploading the Parquet files to an AWS S3 bucket. See the combined example for a single code example that combines all of the steps.

Get the IDs for all projects in the organization. In the code example for this step, the `get_projects` method uses the Projects API to get the project IDs.

note

In the following code example, replace `<personal_access_token>` with your PAT.

To use the API control plane for the EU rather than the US, replace `https://api.dremio.cloud/` with `https://api.eu.dremio.cloud/`.

Get the IDs for all projects

```
import requests
import json

dremio_server = "https://api.dremio.cloud/"
personal_access_token = "<personal_access_token>"

headers = {
   'Authorization': "Bearer " + personal_access_token,
   'Content-Type': "application/json"
}

def api_get(endpoint: str) -> Response:
   return requests.get(f'{dremio_server}/{endpoint}', headers=headers)

def get_projects() -> dict:
   """
   Get all projects in the Dremio Cloud organization
   :return: Dictionary of project IDs and project names
   """
   projects = dict()
   projects_response = api_get('v0/projects')
   for project in projects_response.json():
       projects[project['id']] = project['name']
   return projects
```

Run a SQL query to get the jobs or events for the project. The code examples for this step show how to use the SQL API to submit a SQL query, get all jobs during a specific period with the `get_jobs` method, and get all events in the `sys.project.history.events` system table during a specific period with the `get_events` method.

Submit SQL query using the API

```
def api_post(endpoint: str, body=None) -> Response:
   return requests.post(f'{dremio_server}/{endpoint}',
                        headers=headers, data=json.dumps(body))

def run_sql(project_id: str, query: str) -> str:
   """
   Run a SQL query
   :param project_id: project ID
   :param query: SQL query
   :return: query job ID
   """
   query_response = api_post(f'v0/projects/{project_id}/sql', body={'sql': query})
   job_id = query_response.json()['id']
   return job_id
```

Get all jobs in the project during a specific period

```
def api_post(endpoint: str, body=None) -> Response:
   return requests.post(f'{dremio_server}/{endpoint}',
                        headers=headers, data=json.dumps(body))

def get_jobs(project_id: str, start_time: str, end_time: str) -> str:
   """
   Run SQL query to get all jobs in a project during the specified time period
   :param project_id: project ID
   :param start_time: start timestamp (inclusive)
   :param end_time: end timestamp (exclusive)
   :return: query job ID
   """
   query_response = api_post(f'v0/projects/{project_id}/sql', body={'sql': query})
   job_id = run_sql(project_id, f'SELECT * FROM sys.project.history.jobs '
                                f'WHERE "submitted_ts" >= \'{start_time}\' '
                                f'AND "submitted_ts" < \'{end_time}\'')
   return job_id
```

Get all events during a specific period

```
def get_events(project_id: str, start_time: str, end_time: str) -> str:
   """
    Run SQL query to get all events in sys.project.history.events during the specified
time period
   :param project_id: project ID
   :param start_time: start timestamp (inclusive)
   :param end_time: end timestamp (exclusive)
   :return: query job ID
   """
   job_id = run_sql(project_id, f'SELECT * FROM sys.project.history.events '
                                f'WHERE "timestamp" >= \'{start_time}\' '
                                f'AND "timestamp" < \'{end_time}\'')
   return job_id
```

Check the status of the query to get jobs or events. In the code example for this step, the `wait_for_job_complete` method periodically checks and returns the query job state and prints out the final job status when the query is complete.

Check status of the query to get jobs or events

```
def wait_for_job_complete(project_id: str, job_id: str) -> str:
   """
   Wait for a query job to complete
   :param project_id: project ID
   :param job_id: job ID
   :return: if the job completed successfully, True; otherwise, False
   """
   while True:
       time.sleep(1)
       job = api_get(f'v0/projects/{project_id}/job/{job_id}')
       job_state = job.json()["jobState"]
       if job_state == 'COMPLETED':
           print("Job complete.")
           break
       elif job_state == 'FAILED':
           print("Job failed.", job.json()['errorMessage'])
           break
       elif job_state == 'CANCELED':
           print("Job canceled.")
           break

   return job_state
```

Download the result for the query to get jobs or events and save it to a Parquet file. In the code example for this step, the `save_job_results_to_parquet` method downloads the query result and, if the result contains at least one row, saves the result to a single Parquet file.

Download query result and save to a Parquet file

```
def save_job_results_to_parquet(project_id: str, job_id: str,
                                parquet_file_name: str) -> bool:
   """
   Download the query result and save it to a Parquet file
   :param project_id: project ID
   :param job_id: query job ID
   :param parquet_file_name: file name to save the job result
    :return: if the query returns more than 0 rows and parquet file is saved, True;
otherwise False
   """
   offset = 0
   rows_downloaded = 0
   rows = []
   while True:
       job_result = api_get(f'v0/projects/{project_id}/job/{job_id}/'
                            f'results/?offset={offset}&limit=500')
       job_result_json = job_result.json()
       row_count = job_result_json['rowCount']
```

```
        rows_downloaded += len(job_result_json['rows'])
        rows += job_result_json['rows']
        if rows_downloaded >= row_count:
            break
        offset += 500

    print(rows_downloaded, "rows")
    if rows_downloaded > 0:
        py_rows = pyarrow.array(job_result_json['rows'])
        table = pyarrow.Table.from_struct_array(py_rows)
        pyarrow.parquet.write_table(table, parquet_file_name)
        return True

    return False
```

If desired, you can use the Boto3 library to upload the Parquet file to an AWS S3 bucket.

Upload Parquet file to AWS S3 with Boto3 library

```
def upload_file(file_name: str, bucket: str, folder: str):
    """Upload Parquet file to an S3 bucket with Boto3
    :param file_name: File to upload
    :param bucket: Bucket to upload to
    :param folder: Folder to upload to
    :return: True if file was uploaded, else False
    """

    # Upload the file
    s3_client = boto3.client('s3')
    try:
        response = s3_client.upload_file(file_name, bucket, f'{folder}/{file_name}')
    except ClientError as e:
        print(e)
        return False
    return True
```

Combined Example

The following code example combines the steps above to get all jobs and events from all projects during a specific period, save the query results to Parquet files, and upload the Parquet files to an AWS S3 bucket. The parameter `start` is the start timestamp (inclusive) and the parameter `end` is the end timestamp (exclusive).

All jobs in each project during the specified time period are saved in an individual Parquet file with file name `jobs_<project_id><start>.parquet`. All events in each project during the specified time period are saved in one Parquet file with file name `events_<project_id><start>.parquet`.

Combine all steps in a single code example

```
def main(start: str, end: str):
    """
```

```
   Get all jobs and events from all projects during the specified time period, save the
results in Parquet files, and upload the files to an AWS S3 bucket.
    :param start: start timestamp (inclusive, in format "YYYY-MM-DD" or "YYYY-MM-DD
hh:mm:ss"
   :param end: end timestamp (exclusive, in format "YYYY-MM-DD" or "YYYY-MM-DD hh:mm:ss"

   """
   projects = get_projects()
   print("Projects in organization:")
   print(projects)

   # Get jobs for each project
   for project_id in projects:
       print("Get jobs for project", projects[project_id])
       # run query
       job_id = get_jobs(project_id, start, end)
       # check job status
       job_state = wait_for_job_complete(project_id, job_id)
       if job_state == "COMPLETED":
           file_name = f'jobs_{project_id}{start}.parquet'
           if save_job_results_to_parquet(project_id, job_id, file_name):
               upload_file(file_name, 'S3_BUCKET_NAME', 'dremio/jobs')

   for project_id in projects:
       print("Get events for project", projects[project_id])
       # run query
       job_id = get_events(project_id, start, end)
       # check job status
       job_state = wait_for_job_complete(project_id, job_id)
       if job_state == "COMPLETED":
           file_name = f'events_{project_id}{start}.parquet'
           if save_job_results_to_parquet(project_id, job_id, file_name):
               upload_file(file_name, 'S3_BUCKET_NAME', 'dremio/events')

if __name__ == "__main__":
   parser = argparse.ArgumentParser(
       description='Demo of collecting jobs and events from Dremio Cloud Projects')
   parser.add_argument('start',
                       help='start timestamp (inclusive, in format "YYYY-MM-DD" or
"YYYY-MM-DD hh:mm:ss")')
   parser.add_argument('end',
                         help='end timestamp (exclusive, in format "YYYY-MM-DD" or
"YYYY-MM-DD hh:mm:ss")')
   args = parser.parse_args()

   main(args.start, args.end)
```

Was this page helpful?

Monitor the Dremio Console

Monitor with Dremio APIs and SQL

Manage Projects | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/projects/

On this page

A project isolates the compute, data, and other resources a team needs for data analysis. An organization may contain multiple projects. Your first project is created during the sign-up process.

Each project in Dremio has its own storage. This is used to store metadata and Reflections and serves as the default storage location for the project's Open Catalog. You can choose between two storage options:

- Dremio-managed storage – No setup or configuration required. Usage is priced per TB, billed monthly.
- Your own storage – Use your own Amazon S3 storage. However, this requires you to manage this infrastructure.

For details on pricing, see How Storage Usage Is Calculated.

Each project in your organization contains a preview engine. Each new project has a preview engine. The preview engine, by default, will scale down after 1 hour without a query. As the name suggests, it provides previews of queries and datasets. Unlike other engines, the preview engine cannot be disabled, ensuring that many core Dremio functions that require an engine can always run.

View All Projects

To view all projects:

- In the Dremio console, hover over !This is the Dremio Settings icon. in the side navigation bar and select **Organization settings**.
- Select **Projects** in the organization settings sidebar.

The Projects page displays the status of all projects in your organization. Possible statuses include:

- Creating
- Active
- Inactive
- Deactivating
- Activating
- Archiving
- Archived
- Restoring

Grant Access to a Project

New projects are private by default. In the projects page, users can see only the projects for which they have USAGE or OWNERSHIP privileges. The projects page is empty for users without USAGE or OWNERSHIP privileges on any projects. The projects dropdown list shares this behavior.

Similarly, the Projects API returns an HTTP 403 Forbidden error for requests from users who do not have USAGE or OWNERSHIP privileges on the project. Also, users must have USAGE or OWNERSHIP privileges on a project before they can make API requests or run SQL queries on any objects in the project, even if they have object-level privileges on sources, folders, or other objects in the project.

To allow users to access a project, use the `GRANT TO ROLE` or `GRANT TO USER` SQL command or the Grants API to grant them the USAGE privilege on the project. For users who do not own the project, USAGE is the minimum privilege required to perform any operation on the project and the objects the project contains. For example, if you are using `GRANT TO USER`, you can run `GRANT USAGE ON PROJECT TO USER <username>`.

Obtain the ID of a Project

A BI client application might require the ID of a project as part of the information for creating a connection to Dremio. You can obtain the ID from the General Information page of a project's settings.

To obtain a project ID:

In the Dremio console, hover over !This is the Dremio Settings icon. in the side navigation bar and select **Project settings**.

Select **General Information** in the project settings sidebar.

Copy the value in the **Project ID** field.

Set the Default Project

When your data consumers connect to Dremio from BI tools, they must connect to the projects where their datasets reside. They can either connect to the default project or select a different project.

If an organization administrator does not set this value, Dremio automatically sets the default project to the oldest project in your organization.

You can change the default project at any time.

note

Data consumers who do not have access to the default project must select an alternative project ID when connecting to Dremio from their BI tools.

To specify the default project for your organization:

Hover over !This is the Dremio Settings icon. in the side navigation bar and select

Organization settings.

Select **General Information** in the organization settings sidebar.

In the **Default Project** field, select the project that you want data consumers to connect to by default through their BI tools.

Click **Save**.

Create a Project

If you're planning on using your own bucket, you will need create a role for Dremio granting access, this must be done prior to creating a project, see Bring Your Own Project Store for instructions. To avoid having to do this simply use Dremio-managed storage.

To add a project:

In the Dremio console, hover over !This is the Dremio Settings icon. in the side navigation bar and select **Organization settings**.

Select **Projects** option in the organization settings sidebar.

In the top-right corner of the Projects page, click **Create**.

For **Project name**, specify a name that is unique within the organization.

For **Region**, select the AWS Region where you wish the project to reside.

Select one of the two **Storage** options:

For **Dremio managed storage**, Dremio will create and manage object storage for your use.

For **your own storage**, you will need to provide Dremio the bucket URI and Role ARN previously created.

Activate a Project

Dremio automatically deactivates any project that has not been accessed in the last 15 days. Dremio sends a courtesy email to project owners three days prior to deactivation. Inactive projects are displayed in the project selector in the side navigation bar and on the Projects page. An inactive project will be activated automatically when any user tries to access it via the Dremio console, an ODBC or JDBC connection, or an API call.

note

Inactive projects do not consume any compute resources.

You can activate an inactive project on the Projects page, or by clicking the project in the project selector. It takes a few minutes to activate a project.

To activate a project from the Projects page:

Hover over !This is the Dremio Settings icon. in the side navigation bar and select **Organization settings**.

Select **Projects** in the organization settings sidebar.

Click the ellipsis menu to the far right of the inactive project, and then click **Activate Project**.

The project status will change to *Activating* while the project is activated. You can access the project after the status changes to *Active*.

Archive a Project

Users with OWNERSHIP privileges or users assigned to the ADMIN role can archive a project. Archived projects are displayed only on the Projects page.

note

Archived projects do not consume any compute resources.

To archive a project:

In the Dremio console, hover over !This is the Dremio Settings icon. in the side navigation bar and select **Organization settings**.

Select **Projects** in the organization settings sidebar.

Click the ellipsis menu to the far right of an active or inactive project, and then click **Archive Project**.

The project status will change to *Archiving* while the project is archived. When archiving is complete, the status changes to *Archived*.

Restore an Archived Project

An archived project will not be restored automatically if a user tries to access it and can only be restored manually by a user with OWNERSHIP privileges on the project or users assigned to the ADMIN role. It takes a few minutes to restore an archived project.

To restore an archived project:

In the Dremio console, hover over !This is the Dremio Settings icon. in the side navigation bar and select **Organization settings**.

Select **Projects** in the organization settings sidebar.

Click the ellipsis menu to the far right of an archived project and select **Restore Project**.

The project status will change to *Restoring* while the project is restored. You can access the project after the status changes to *Active*.

Delete a Project

Default projects cannot be deleted. If you want to delete the default project, you must first set another project as the default. See Set the Default Project.

To delete a project:

In the Dremio console, hover over !This is the Dremio Settings icon. in the side navigation bar and select **Organization Settings**.

Select **Projects** in the organization settings sidebar.

Click the ellipsis menu to the far right and select **Delete Project**.

Confirm that you want to delete the project.

Was this page helpful?

View All Projects

Grant Access to a Project

Obtain the ID of a Project

Set the Default Project

Create a Project

Activate a Project

Archive a Project

Restore an Archived Project

Delete a Project

Audit Logs | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/monitor/logs

On this page

The creation and modification of Dremio resources are tracked and traceable via the `sys.project.history.events` table. Audit logging is enabled by default and available to users with administrative permissions on the project.

An event can take up to three hours to propagate to the system table. There is currently no maximum retention policy for audit events.

note

This is a subset of the events that Dremio supports.

Organization Events

Dremio supports audit logging for the following organization event types and actions. The `sys.project.history.events` table contains these events in the default project.

Event Type	Actions	Description
BILLING_ACCOUNT	BILLING_ACCOUNT_ADD_PROJECT	Dremio added a new

project to the billing account during project creation. |
BILLING_ACCOUNT	BILLING_ACCOUNT_CREATE	A user created a billing account.
BILLING_ACCOUNT	BILLING_ACCOUNT_REMOVE_PROJECT	Dremio removed a project from the billing account during project deletion.
BILLING_ACCOUNT	BILLING_ACCOUNT_UPDATE	A user modified the billing account, such as the notification email address.
BILLING_TRANSACTION	TRANSACTION_CHARGE	Dremio recorded Dremio Consumption Unit (DCU) usage charges for the period.
BILLING_TRANSACTION	TRANSACTION_CREDIT_LOAD	Dremio loaded DCU credits into the billing account.
CATALOG	CREATE	A user created a new Open Catalog. Catalog creation is included with project creation.
CATALOG	DELETE	A user deleted an Open Catalog. Project deletion also deletes its primary Open Catalog.
CATALOG	UPDATE	A user updated an Open Catalog configuration.
CLOUD	CREATE_STARTED CREATE_COMPLETED	A user created a cloud. Clouds provide resources for running engines and storing metadata in a project.
CLOUD	DELETE_STARTED DELETE_COMPLETED	A user deleted a cloud.
CLOUD	UPDATE	A user updated a cloud.
CONNECTION	FORCE_LOGOUT	A user changed their password or deactivated another user, ending all of that user's sessions.
CONNECTION	LOGIN	A user logged in.
CONNECTION	LOGOUT	A user logged out.
EDITION	DOWNGRADE	A user downgraded the billing edition in the Dremio organization.
EDITION	UPGRADE	A user upgraded the billing edition in the Dremio organization.
IDENTITY_PROVIDER	CREATE	A user configured a new OpenID Connect (OIDC) identity provider integration.
IDENTITY_PROVIDER	DELETE	A user deleted an OIDC identity provider.
IDENTITY_PROVIDER	UPDATE	A user updated an OIDC identity provider configuration.
MODEL_PROVIDER_CONFIG	CREATE	A user created a new model provider in the Dremio organization.
MODEL_PROVIDER_CONFIG	UPDATE	A user updated a model provider in the Dremio organization.
MODEL_PROVIDER_CONFIG	DELETE	A user deleted a model provider in the Dremio organization.
MODEL_PROVIDER_CONFIG	SET_DEFAULT	A user set a new default model provider in the Dremio organization.
ORGANIZATION	CREATE_STARTED CREATE_COMPLETED	A user created the Dremio organization.
ORGANIZATION	DELETE_STARTED DELETE_COMPLETED	A user closed and deleted the Dremio organization.
ORGANIZATION	UPDATE	A user updated the Dremio Cloud organization.
PERSONAL_ACCESS_TOKEN	CREATE	A user created a new personal access token in their account.
PERSONAL_ACCESS_TOKEN	DELETE	A user deleted a personal access token.
PROJECT	CREATE_STARTED CREATE_COMPLETED	A user created a project in the Dremio organization.
PROJECT	DELETE_STARTED DELETE_COMPLETED	A user deleted a project.
PROJECT	HIBERNATE_STARTED HIBERNATE_COMPLETED	A user archived a project.
PROJECT	UNHIBERNATE_STARTED UNHIBERNATE_COMPLETED	A user activated an archived project.

PROJECT	UPDATE	A user updated the configuration of a project.
ROLE	CREATE	A user created a custom role.
ROLE	DELETE	A user deleted a role.
ROLE	MEMBERS_ADDED	A user added users or roles as members of a role.
ROLE	MEMBERS_REMOVED	A user removed users or roles as members of a role.
ROLE	UPDATE	A user updated the metadata of a custom role, such as the description.
USER_ACCOUNT	CREATE	A user added a user account.
USER_ACCOUNT	DELETE	A user deleted a user account.
USER_ACCOUNT	PASSWORD_CHANGE	A user updated their account password.
USER_ACCOUNT	UPDATE	A user updated user account metadata.

Project Events

Event Type	Actions	Description
AI_AGENT	REQUEST RESPONSE	A user sent a request to the AI Agent and received a response.
ENGINE	CREATE_STARTED CREATE_COMPLETED	A user created an engine.
ENGINE	DELETE_STARTED DELETE_COMPLETED	A user deleted an engine.
ENGINE	DISABLE_STARTED DISABLE_COMPLETED	A user disabled an engine.
ENGINE	ENABLE_STARTED ENABLE_COMPLETED	A user enabled an engine.
ENGINE	UPDATE_STARTED UPDATE_COMPLETED	A user updated an engine configuration.
ENGINE_SCALING	SCALE_DOWN_STARTED SCALE_DOWN_COMPLETED	Dremio scaled down an engine by stopping one or more running replicas.
ENGINE_SCALING	SCALE_UP_STARTED SCALE_UP_COMPLETED	Dremio scaled up an engine by starting one or more additional replicas.
LABEL	UPDATE	A user created a label on a dataset, source, or other object.
PIPE	CREATE	A user created an autoingest pipe for Apache Iceberg.
PIPE	DELETE	A user dropped an autoingest pipe.
PIPE	UPDATE	A user updated the configuration of an existing autoingest pipe.
PRIVILEGE	DELETE	A user deleted a privilege from a user or role.
PRIVILEGE	UPDATE	A user granted a privilege to a user or role.
REFLECTION	CREATE	A user created a new raw or aggregate Reflection.
REFLECTION	DELETE	A user deleted a Reflection.
REFLECTION	UPDATE	A user updated the content or configuration of a Reflection.
REPLICA	CREATE_STARTED CREATE_COMPLETED	Dremio started a replica during an ENGINE_SCALING scale-up event.
REPLICA	DELETE_STARTED DELETE_COMPLETED	Dremio stopped a replica during an ENGINE_SCALING scale-down event.
ROUTING_RULESET	UPDATE	A user modified an engine routing rule.
SUPPORT_SETTING	RESET	A user reset an advanced configuration or diagnostic setting.
SUPPORT_SETTING	SET	A user set an advanced configuration or diagnostic setting.
UDF	CREATE	A user created a user-defined function.
UDF	DELETE	A user deleted a user-defined function.
UDF	UPDATE	A user modified the SQL definition of a user-defined function.
WIKI	EDIT	A user created or updated a wiki.

Open Catalog Events

These events appear in the `sys.project.history.events` table of the project where the catalog is designated as the primary catalog.

Event Type	Actions	Description
FOLDER	CREATE	A user created a folder in the catalog.
FOLDER	DELETE	A user deleted a folder in the catalog.
TABLE	CREATE	A user created a table in the catalog.
TABLE	DELETE	A user deleted a table in the catalog.
TABLE	READ	A user read table information or data from the catalog.
TABLE	REGISTER	A user registered a new table in the catalog.
TABLE	UPDATE	A user updated a table definition in the catalog.
VIEW	CREATE	A user created a view in the catalog.
VIEW	DELETE	A user deleted a view in the catalog.
VIEW	READ	A user read a view in the catalog.
VIEW	UPDATE	A user updated a view definition in the catalog.

Source Events

These events appear in the `sys.project.history.events` table for any source in the project.

Event Type	Actions	Description
SOURCE	CREATE	A user created a data source.
SOURCE	DELETE	A user deleted a source connection. Any tables from the source were removed.
SOURCE	UPDATE	A user updated a source configuration.
FOLDER	CREATE	A user created a folder.
FOLDER	DELETE	A user deleted a folder.
TABLE	CREATE	A user created a non-catalog table.
TABLE	DELETE	A user deleted a non-catalog table.
TABLE	UPDATE	A user updated a table, or Dremio performed a metadata refresh on a non-Parquet table.

Was this page helpful?

Optimize Performance | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/performance/

Dremio uses a variety of tools to help you autonomously optimize your lakehouse. These tools apply at four stages: (1) source files, (2) intermediate transformations, (3) final or production transformations, and (4) client queries. Dremio also offers tools that allow you to manually fine-tune performance. Both approaches can coexist, enabling Dremio to manage most optimizations automatically while still giving you the flexibility

to take direct action when desired.

For details on how Dremio autonomously manages your tables, see Automatic Optimization, which focuses on Iceberg table management.

This section focuses instead on accelerating views and SQL queries, including those from clients such as AI agents and BI dashboards. The principal method for this acceration is Dremio's patterned materialization and query-rewriting, known as Reflections.

Autonomous Reflections – Learn how Dremio automatically learns your query patterns and manages Reflections to optimize performance accordingly. This capability is available for Iceberg tables, UniForm tables, Parquet datasets, and any views built on these datasets.

Manual Reflections – Use this option primarily for data formats not supported by Autonomous Reflections. Learn how to define your own Reflections and the best practices for using and managing them.

Results Cache – Understand how Dremio caches the results of queries from AI agents and BI dashboards.

Was this page helpful?

Jobs | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/monitor/jobs/

On this page

All jobs run in Dremio are listed on a separate page, showing the job ID, type, status, and other attributes.

To navigate to the Jobs page, click !This is the icon that represents the Jobs page. in the side navigation bar.

Search Filters and Columns

By default, the Jobs page lists the jobs run within the last 30 days and the jobs are filtered by **UI, External Tools** job types. To change these defaults for your account, you can filter on values and manage columns directly on the Jobs page, as shown in this image:

!This is a screenshot showing the main components of the Jobs page.

a. **Search Jobs** by typing the username or job ID.

b. **Start Time** allows you to pick the date and time at which the job began.

c. **Status** represents one or more job states. For descriptions, see Job States and Statuses.

d. **Type** includes Accelerator, Downloads, External Tools, Internal, and UI. For descriptions, see Job Properties.

e. **User** can be searched by typing the username or checking the box next to the username in the dropdown.

f. **Manage Columns** by checking the boxes next to additional columns that you want to see in the Jobs list. The grayed out checkboxes show the columns that are required by default. You can also rearrange the column order by clicking directly on a column to drag and drop.

Job Properties

Each job has the following properties, which can appear as columns in the list of jobs on the Jobs page or as details on the Job Overview page:

Property	Description
Accelerated	A purple lightning bolt in a row indicates that the job ran a query that was accelerated by one or more Reflections.
Attribute	Represents at least one of the following query types: * **UI** - queries issued from the SQL Runner in the Dremio console. * **External Tools** - queries from client applications, such as Microsoft Power BI, Superset, Tableau, other third-party client applications, and custom applications. * **Accelerator** - queries related to creating, maintaining, and removing Reflections. * **Internal** - queries that Dremio submits for internal operations. * **Downloads** - queries used to download datasets. * **AI** – queries issued from the Dremio AI Agent.
CPU Used	Provides statistics about the actual cost of the query operations in terms of CPU processing.
Dataset	The queried dataset, if one was queried. Hover over the dataset to see a metadata card appear with details about the dataset. For more information, see Discover Data.
Duration	The length of time (in seconds) that a job required from start to completion.
Engine	The engine used to run the query.
Input	The number of bytes and the number of rows considered for the job.
Job ID	A universally unique identifier.
Output	The number of bytes and the number of rows resulted as output from the job.
Planner Cost Estimate	A cost estimate calculated by Dremio based on an evaluation of the resources that to be used in the execution of a query. The number is not in units, and is intended to give a an idea of the cost of executing a query relative to the costs of executing other queries. Values are derived by adding weighted estimates of required I/O, memory, and CPU load. In reported values, K = thousand, M = million, B = billion, and T = trillion. For example, a value of 12,543,765,321 is reported as 12.5B.
Planning Time	The length of time (in seconds) in which the query optimizer planned the execution of the query.
Rows Returned	Number of output records.
Rows Scanned	Number of input records.
SQL	The SQL query that was submitted for the job.
Start Time	The date and time which the job began.
Status	Represents one or more job states. For descriptions, see Job States and Statuses.
Total Memory	Provides statistics about the actual cost of the query operations in terms of memory.
User	Username of the user who ran the query and initiated the job.
Wait on Client	The length of time (in seconds) that is waiting on the client.

Job States and Statuses

Each job passes through a sequence of states until it is complete, though the sequence can be interrupted if a query is canceled or if there is an error during a state. In this diagram, the states that a job passes through are in white, and the possible end states are in dark gray.

This table lists the statuses that the UI lets you filter on and shows how they map to the states:

Icon	Status	State	Description
	Setup	Pending	Represents a state where the query is waiting to be scheduled on the query pool.
Metadata Retrieval	Represents a state where metadata schema is retrieved and the SQL command is parsed.
Planning	Represents a state where the following are done: * Physical and logical planning * Reflection matching * Partition metadata retrieval * Mapping the query to an engine-based workload management rule * Pick the engine associated with the query to run the query.
	Engine Start	Engine Start	Represents a state where the engine starts if it has stopped. If the engine is stopped, it takes time to restart for the executors to be active. If the engine is already started, then this state does not have a duration.
	Queued	Queued	Represents a state where a job is queued. Each engine has a limit of concurrent queries. If the queries in progress exceed the concurrency limit, the query should wait until the jobs in progress complete.
	Running	Execution Planning	Represents a state where executor nodes are selected from the chosen engine to run the query, and work is distributed to each executor.
Running	Represents a state where executor nodes execute and complete the fragments assigned to them. Typically, most queries spend more time in this state.
Starting	Represents a state where the query is starting up.
	Canceled	Canceled	Represents a terminal state that indicates that the query is canceled by the user or an intervention in the system.
	Completed	Completed	Represents a terminal state that indicates that the query is successfully completed.
	Failed	Failed	Represents a terminal state that indicates that the query has failed due to an error.

View Job Details

You can view the details of a specific job by viewing the Job Overview, SQL, Visual Profile, and Raw Profile pages.

To navigate to the job details:

- Click !This is the icon that represents the Jobs page. in the side navigation bar.
- On the Jobs page, click a job that you would like to see the job overview for.
- The Job Overview page then replaces the list of jobs.

Explain SQL

Use the **Explain SQL** option in the SQL Runner to analyze and optimize your SQL queries with assistance from the AI Agent. In the SQL Runner, highlight the SQL you want to review, right-click, and select **Explain SQL**. This prompts the AI Agent to examine the query, datasets, and underlying architecture to identify potential optimizations. The AI Agent uses Dremio's SQL Parser—the same logic used during query execution—to identify referenced tables, schemas, and relationships. Based on this analysis, the Agent provides insights and recommendations to improve query performance and structure. You can continue interacting with the AI Agent to refine the analysis and iterate on the SQL. The AI Agent applies SQL best practices when suggesting improvements and may execute revised queries to validate quality before presenting recommendations.

Explain Job

Use the **Explain Job** option on the Job Details page to analyze job performance and identify opportunities for optimization. From the Job Details page, click **Explain Job** to prompt the AI Agent to review the job's query profile, planning, and execution details to compare with the AI Agents's internal understanding of optimal performance characteristics. The AI Agent generates a detailed analysis that highlights key performance metrics such as data skew, memory usage, threading efficiency, and network utilization. Based on this assessment, it recommends potential optimizations to improve performance and resource utilization. You can continue the conversation with the AI Agent to explore the job in greater depth or reference additional job IDs to extend the investigation and compare results.

Job Overview

You can view the details of a specific job on the Job Overview page.

To navigate to a job overview:

- Click !This is the icon that represents the Jobs page. in the side navigation bar.
- On the Jobs page, click a job that you would like to see the job overview for. The Job Overview page then replaces the list of jobs.

The main components of the Job Overview page are numbered below:

!This is a screenshot showing the main components of the Job Overview page.

1. Summary

Each job is summarized.

2. Total Execution Time

The total execution time is the length of time for the total execution and the job state durations in the order they occur. Only the duration of the Engine Start state is in minutes and seconds. If the engine is stopped, it takes time to restart for the executors

to be active. If the engine is already started, then Engine Start duration does not have a value. For descriptions, see Job States and Statuses.

3. Download Profile

To download the query profile, click the **Download Profile** button in the bottom-left corner of the Job Overview page. The profile will help you see more granular details about the job.

The profile downloads as a **ZIP** file. When you extract the **ZIP** file, you will see the following JSON files:

profile_attempt_0.json: This file helps with troubleshooting out of memory and wrong results issues. Note that the start and end time of the query is provided in EPOCH format. See the Epoch Converter utility for converting query time.

header.json: This file provides the full list of Dremio coordinators and executors, data sets, and sources.
This information is useful when you are using REST calls.

4. Submitted SQL

The SQL query for the selected job.

5. Queried Datasets

The datasets queried for the selected job. These can be views or tables.

6. Scans

Scan details include the source type, scan thread count, IO wait time (in milliseconds), and the number of rows scanned.

7. Acceleration

Only if the job was accelerated, the Acceleration section appears and Reflections data is provided. See Optimize Performance for more information.

8. Results

To see the job results, click the **Open Results** link in the top-right corner of the Job Overview page. As long as the engine that ran the job is up, the **Open Results** link is visible in the UI. It disappears when the engine that ran the job shuts down and is only visible for the jobs that are run through the UI.

Job SQL

Next to the Job Overview page is a tab for the SQL page, which shows the Submitted SQL and Dataset Graph.

You can view the SQL statement that was used for the selected job. Although the SQL statement is in read-only mode on the SQL Details page, the statement can be copied from the page and pasted into the SQL editor.

A dataset graph only appears if there is a queried dataset for the selected job. The dataset graph is a visual representation of the datasets used in the SQL statement.

Related Topics

Profiles – See the visual profiles and raw profiles of jobs.

Was this page helpful?

Search Filters and Columns

Job Properties

Job States and Statuses

View Job Details

Explain SQL

Explain Job

Job Overview

Job SQL

Related Topics

Manage Your Subscription | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/subscription/

On this page

Dremio offers multiple payment options for users to upgrade their organization after conclusion of the free trial:

Pay-as-you-go (PAYG): Provide credit card details via the upgrade steps in the Dremio console.

Commit-based offerings: Prepaid contracts for your organization's usage. Dremio invoices you directly, and a variety of payment options are available. Please contact Dremio Sales for more details.

AWS Marketplace: A commit-based contract paid with AWS credits. Check out the Dremio Cloud AWS Marketplace listing on the AWS Marketplace. Once ready to proceed, contact Dremio Sales for more details.

Note that your organization can be moved to a commit-based contract after upgrading to PAYG.

Upgrade

At any point during your free trial of Dremio, an organization can be upgraded by entering your credit card details. If the free trial concludes, your organization will become partially inaccessible for 30 days. During this time, you can still log in to upgrade your account, but if you do not upgrade your account before then, your organization and all of its contents may be deleted.

Pay-as-you-go Billing Cycles

Your billing cycle starts from the day of your organization's upgrade and ends one month later. At the conclusion of the billing period, we will immediately attempt to charge your card for the outstanding balance.

If for any reason payment fails (or is only partially successful), we will attempt the charge again. If these subsequent attempts fail, your organization will become partially inaccessible. You can still log in but only to update your payment method. If a new payment method is not provided before the end of this billing period, your organization and all of its contents may be deleted.

Organizations

A Dremio organization can have one or more projects. Usage across projects is aggregated for billing purposes, meaning that when the PAYG bill is paid for an organization, the balance is paid for all projects. Only users who are members of the ADMIN role within the organization can manage billing details within the Dremio console.

Find Your Organization ID

The ID of your organization can be helpful during communication with Dremio Sales or Support. To find your organization's ID:

In the Dremio console, click !Settings in the side navigation bar and select **Organization settings** to open the Organization settings page.

On the General Information tab, copy your organization's ID.

Delete Your Organization

Please contact Dremio's Support team if you would like to have your organization deleted.

Was this page helpful?

Delete Your Organization

Configure Model Providers | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/model-providers

On this page

You configure model providers for your organization for AI features when deploying Dremio. After you configure at least one model provider, you must set a default model provider and optionally set an allowlist of available models. Dremio uses this default provider for all Dremio's AI Agent interactions, whereas the allowlist models can be used by anyone writing AI functions. By default CALL MODEL is granted to all users for all new model providers so if the default changes users can continue to use the AI Agent without interruption.

Dremio-Provided LLM

Dremio provides all organizations with an out-of-the-box model provider so that all users can begin engaging with the AI Agent and AI functions without any other configuration required. Once you have added your own model provider and set it as the new default, the Dremio-Provided LLM will no longer be used. If you delete all other model providers, then the Dremio-Provided LLM will revert to the organization's default model provider. This model provider cannot be deleted.

Supported Model Providers

Dremio supports configuration of the following model providers and models. Dremio recommends using enterprise-grade reasoning models for the best performance and experience.

Category	Models	Connection Method(s)
OpenAI	* gpt-5-2025-08-07 * gpt-5-mini-2025-08-07 * gpt-5-nano-2025-08-07 * gpt-4.1-2025-04-14 * gpt-4o-2024-11-20 * gpt-4-turbo-2024-04-09 * gpt-4.1-mini-2025-04-14 * o3-mini-2025-01-31 * o4-mini-2025-04-16 * o3-2025-04-16	* Access Key
Anthropic	* claude-sonnet-4-5-20250929 * claude-opus-4-1-20250805 * claude-opus-4-20250514 * claude-sonnet-4-20250514	* Access Key
Google Gemini	* gemini-2.5-pro	* Access Key
AWS Bedrock	* specify Model ID(s) * AWS Bedrock Supported Models	* Access Key * IAM Role
Azure OpenAI	* specify Deployment Name(s) * Azure Supported Models	Combination of 1. Resource Name 2. Directory ID 3. Application ID 4. Client Secret Value

Rate Limiting and Quotas

AWS Bedrock Rate Limits

When using AWS Bedrock model providers, you may encounter rate limiting errors such as "429 Too Many Tokens (Rate Limit Exceeded)". This is particularly common with new AWS accounts that start with lower or fixed quotas.

If you experience rate limiting issues, you can contact AWS Support and request a quota increase by providing:

- Quota name
- Model ID
- AWS region
- Use case description
- Projected token and request usage

For more information about AWS Bedrock quotas and limits, see the AWS Bedrock User Guide.

Default Model Provider

To delete the model provider, you must assign a new default unless you are deleting the last available model provider that you have configured. To update the default model provider to a new one, you must have MODIFY privilege on both the current default and the new proposed default model provider.

Add Model Provider

To add a model provider in the Dremio console:

- Click !This is the Settings icon. in the side navigation bar to go to the Settings page.
- Select **Organization Settings**.
- Select the **AI Configuration** setting.
- Click **Add model provider**.

Was this page helpful?

External Engines | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/external-engines

On this page

Dremio's Open Catalog is built on Apache Polaris, providing a standards-based, open approach to data catalog management. At its core is the Iceberg REST interface, which enables seamless integration with any query engine that supports the Apache Iceberg REST catalog specification. This open architecture means you can connect industry-standard engines such as Apache Spark, Trino, and Apache Flink directly to Dremio.

Engine	Best For	Key Features
Apache Spark	Data engineering, ETL	Token exchange, nested folders, views
Trino	Interactive analytics	Fast queries, BI workloads
Apache Flink	Real-time streaming	Event-driven, continuous pipelines

By leveraging the Iceberg REST standard, the Open Catalog acts as a universal catalog layer that query engines can communicate with using a common language. This allows organizations to build flexible data architectures where multiple engines can work together, each accessing and managing the same Iceberg tables through Dremio's centralized catalog.

Apache Spark

Apache Spark is a unified analytics engine for large-scale data processing, widely used for ETL, batch processing, and data engineering workflows.

Prerequisites

This example uses Spark 3.5.3 with Iceberg 1.9.1. For other versions, ensure compatibility between Spark, Scala, and Iceberg runtime versions. Additional prerequisites include:

The following JAR files downloaded to your local directory:

`authmgr-oauth2-runtime-0.0.5.jar` from Dremio Auth Manager releases. This open-source library handles token exchange, automatically converting your personal access token (PAT) into an OAuth token for seamless authentication. For more details about Dremio Auth Manager's capabilities and configuration options, see Introducing Dremio Auth Manager for Apache Iceberg.

`iceberg-spark-runtime-3.5_2.12-1.9.1.jar` (from Apache Iceberg releases)

`iceberg-aws-bundle-1.9.1.jar` (from Apache Iceberg releases)

Docker installed and running.

Your Dremio catalog name – The default catalog in each project has the same name as the project.

If authenticating with a PAT, you must generate a token. See Personal Access Tokens for step-by-step instructions.

If authenticating with an identity provider (IDP), your IDP or other external token

provider must be configured as a trusted OAuth external token provider in Dremio.

You must have an OAuth2 client registered in your IDP configured to issue tokens that Dremio accepts (matching audience and scopes) and with a client ID and client secret provided by your IDP.

Authenticate with a PAT

You can authenticate your Apache Spark session with a Dremio personal access token using the following script. Replace `<personal_access_token>` with your Dremio personal access token and replace `<catalog_name>` with your catalog name.

In addition, you can adjust the volume mount paths to match where you've downloaded the JAR files and where you want your workspace directory. The example uses `$HOME/downloads` and `$HOME/workspace`.

Spark with PAT Authentication

```
#!/bin/bash
export CATALOG_NAME="<catalog_name>"
export DREMIO_PAT="<personal_access_token>"

docker run -it \
  -v $HOME/downloads:/opt/jars \
  -v $HOME/workspace:/workspace \
  apache/spark:3.5.3 \
  /opt/spark/bin/spark-shell \
                                                                                --jars
/opt/jars/authmgr-oauth2-runtime-0.0.5.jar,/opt/jars/iceberg-spark-runtime-3.5_2.12-1.9.
1.jar,/opt/jars/iceberg-aws-bundle-1.9.1.jar \
                                                                                --conf
spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions \

  --conf spark.sql.catalog.polaris=org.apache.iceberg.spark.SparkCatalog \
  --conf spark.sql.catalog.polaris.type=rest \
  --conf spark.sql.catalog.polaris.cache-enabled=false \
  --conf spark.sql.catalog.polaris.warehouse=$CATALOG_NAME \
  --conf spark.sql.catalog.polaris.uri=https://catalog.dremio.cloud/api/iceberg \
  --conf spark.sql.catalog.polaris.io-impl=org.apache.iceberg.aws.s3.S3FileIO \
   --conf spark.sql.catalog.polaris.header.X-Iceberg-Access-Delegation=vended-credentials
\
                                                                                --conf
spark.sql.catalog.polaris.rest.auth.type=com.dremio.iceberg.authmgr.oauth2.OAuth2Manager
\
                                                                                --conf
spark.sql.catalog.polaris.rest.auth.oauth2.token-endpoint=https://login.dremio.cloud/oau
th/token \
  --conf spark.sql.catalog.polaris.rest.auth.oauth2.grant-type=token_exchange \
  --conf spark.sql.catalog.polaris.rest.auth.oauth2.client-id=dremio-catalog-cli \
  --conf spark.sql.catalog.polaris.rest.auth.oauth2.scope=dremio.all \
                                                                                --conf
spark.sql.catalog.polaris.rest.auth.oauth2.token-exchange.subject-token="$DREMIO_PAT" \

                                                                                --conf
```

```
spark.sql.catalog.polaris.rest.auth.oauth2.token-exchange.subject-token-type=urn:ietf:params:oauth:token-type:dremio:personal-access-token
```

note

In this configuration, `polaris` is the catalog identifier used within Spark. This identifier is mapped to your actual Dremio catalog via the `spark.sql.catalog.polaris.warehouse` property.

Authenticate with an IDP

You can authenticate your Apache Spark session using an external token provider that has been integrated with Dremio.

Using this configuration:

- Spark obtains a user-specific JWT from the external token provider.
- Spark connects to Dremio and exchanges the JWT for an access token.
- Spark connects to the Open Catalog using the access token.

Using the following script, replace `<catalog_name>` with your catalog name, `<idp_url>` with the location of your external token provider, `<client_id>` and `<client_secret>` with the credentials issued by the external token provider.

In addition, you can adjust the volume mount paths to match where you've downloaded the JAR files and where you want your workspace directory. The example uses `$HOME/downloads` and `$HOME/workspace`.

Spark with IDP Authentication

```
#!/bin/bash
export CATALOG_NAME="<catalog_name>"
export IDP_URL="<idp_url>"
export CLIENT_ID="<idp_client_id>"
export CLIENT_SECRET="<idp_client_secret>"

docker run -it \
  -v $HOME/downloads:/opt/jars \
  -v $HOME/workspace:/workspace \
  apache/spark:3.5.3 \
  /opt/spark/bin/spark-shell \
                                                                                --jars
/opt/jars/authmgr-oauth2-runtime-0.0.5.jar,/opt/jars/iceberg-spark-runtime-3.5_2.12-1.9.1.jar,/opt/jars/iceberg-aws-bundle-1.9.1.jar \
                                                                                --conf
spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions \

  --conf spark.sql.catalog.polaris=org.apache.iceberg.spark.SparkCatalog \
  --conf spark.sql.catalog.polaris.type=rest \
  --conf spark.sql.catalog.polaris.cache-enabled=false \
  --conf spark.sql.catalog.polaris.warehouse=$CATALOG_NAME \
  --conf spark.sql.catalog.polaris.uri=https://catalog.dremio.cloud/api/iceberg \
```

```
  --conf spark.sql.catalog.polaris.io-impl=org.apache.iceberg.aws.s3.S3FileIO \
  --conf spark.sql.catalog.polaris.header.X-Iceberg-Access-Delegation=vended-credentials
\
                                                                                --conf
spark.sql.catalog.polaris.rest.auth.type=com.dremio.iceberg.authmgr.oauth2.OAuth2Manager
\
  --conf spark.sql.catalog.polaris.rest.auth.oauth2.issuer-url=$IDP_URL \
  --conf spark.sql.catalog.polaris.rest.auth.oauth2.grant-type=device_code \
  --conf spark.sql.catalog.polaris.rest.auth.oauth2.client-id=$CLIENT_ID \
  --conf spark.sql.catalog.polaris.rest.auth.oauth2.client-secret=$CLIENT_SECRET \
  --conf spark.sql.catalog.polaris.rest.auth.oauth2.scope=dremio.all \
  --conf spark.sql.catalog.polaris.rest.auth.oauth2.impersonation.enabled=true \
                                                                                --conf
spark.sql.catalog.polaris.rest.auth.oauth2.impersonation.token-endpoint=https://login.dr
emio.cloud/oauth/token \
  --conf spark.sql.catalog.polaris.rest.auth.oauth2.impersonation.scope=dremio.all \
                                                                                --conf
spark.sql.catalog.polaris.rest.auth.oauth2.token-exchange.subject-token-type=urn:ietf:pa
rams:oauth:token-type:jwt
```

Usage Examples

With these configurations, `polaris` is the catalog identifier used within Spark. This identifier is mapped to your actual Dremio catalog via the `spark.sql.catalog.polaris.warehouse` property. Once Spark is running and connected to your Dremio catalog:

List namespaces

```
spark.sql("SHOW NAMESPACES IN polaris").show()
```

Query a table

```
spark.sql("SELECT * FROM polaris.your_namespace.your_table LIMIT 10").show()
```

Create a table

```
spark.sql("""
  CREATE TABLE polaris.your_namespace.new_table (
    id INT,
    name STRING
  ) USING iceberg
""")
```

Trino

Trino is a distributed SQL query engine designed for fast analytic queries against data sources of all sizes. It excels at interactive SQL analysis, ad hoc queries, and joining

data across multiple sources.

Prerequisites

- Docker installed and running.
- A valid Dremio personal access token – See Personal Access Tokens for instructions to generate a personal access token.
- Your Dremio catalog name – The default catalog in each project has the same name as the project.

Configuration

To connect Trino to Dremio using Docker, follow these steps:

- Create a directory for Trino configuration and add a catalog configuration:

```
mkdir -p ~/trino-config/catalog
```

In `trino-config/catalog`, create a catalog configuration file named `polaris.properties` with the following values:

Trino polaris.properties

```
connector.name=iceberg
iceberg.catalog.type=rest
iceberg.rest-catalog.uri=https://catalog.dremio.cloud/api/iceberg
iceberg.rest-catalog.oauth2.token=<personal_access_token>

iceberg.rest-catalog.warehouse=<catalog_name>
iceberg.rest-catalog.security=OAUTH2

iceberg.rest-catalog.vended-credentials-enabled=true
fs.native-s3.enabled=true
s3.region=<region>
```

Replace the following:

- `<personal_access_token>` with your Dremio personal access token.
- `<catalog_name>` with your catalog name.
- `<region>` with the AWS region where your data is stored, such as `us-west-2`.

note

In this configuration, `polaris` (from the filename `polaris.properties`) is the catalog identifier used in Trino queries. The `iceberg.rest-catalog.warehouse` property maps this identifier to your actual Dremio catalog.

In `oauth2.token`, you provide your Dremio personal access token directly. Dremio's catalog API accepts PATs as bearer tokens without requiring token exchange.

Pull and start the Trino container:

```
docker run --name trino -d -p 8080:8080 trinodb/trino:latest
```

Verify that Trino is running:

```
docker ps
```

You can access the web UI at `http://localhost:8080` and log in as `admin`.

Restart Trino with the configuration:

```
docker stop trino
docker rm trino

# Start with mounted configuration
docker run --name trino -d -p 8080:8080 -v ~/trino-config/catalog:/etc/trino/catalog
trinodb/trino:latest

# Verify Trino is running
docker ps

# Check logs
docker logs trino -f
```

In another window, connect to the Trino CLI:

```
docker exec -it trino trino --user admin
```

You should see the Trino prompt:

```
trino>
```

Verify the catalog connection:

```
trino> show catalogs;
```

Usage Examples

Once Trino is running and connected to your Dremio catalog:

List namespaces

```
trino> show schemas from polaris;
```

Query a table

```
trino> select * from polaris.your_namespace.your_table;
```

Create a table

```
trino> CREATE TABLE polaris.demo_namespace.test_table (
  id INT,
  name VARCHAR,
  created_date DATE,
  value DOUBLE
);
```

Limitations

Case sensitivity: Namespace and table names must be in lowercase. Trino will not list or access tables in namespaces that begin with an uppercase character.

View compatibility: Trino cannot read views created in Dremio due to SQL dialect incompatibility. Returns error: "Cannot read unsupported dialect 'DremioSQL'."

Apache Flink

Apache Flink is a distributed stream processing framework designed for stateful computations over bounded and unbounded data streams, enabling real-time data pipelines and event-driven applications.

To connect Apache Flink to Dremio using Docker Compose, follow these steps:

Prerequisites

You'll need to download the required JAR files and organize them in a project directory structure.

Create the project directory structure:

```
mkdir -p flink-dremio/jars
cd flink-dremio
```

Download the required JARs into the `jars/` directory:

Iceberg Flink Runtime 1.20:

```
wget -P jars/ https://repo1.maven.org/maven2/org/apache/iceberg/iceberg-flink-runtime-1.20/1.9.1/iceberg-flink-runtime-1.20-1.9.1.jar
```

Iceberg AWS Bundle for vended credentials:

```
wget -P jars/ https://repo1.maven.org/maven2/org/apache/iceberg/iceberg-aws-bundle/1.9.1/iceberg-aws-bundle-1.9.1.jar
```

Hadoop dependencies required by Flink:

```
wget                                        -P                                          jars/
https://repo.maven.apache.org/maven2/org/apache/flink/flink-shaded-hadoop-2-uber/2.8.3-1
0.0/flink-shaded-hadoop-2-uber-2.8.3-10.0.jar
```

Create the Dockerfile.

Create a file named `Dockerfile` in the `flink-dremio` directory:

Flink Dockerfile

```
FROM flink:1.20-scala_2.12

# Copy all required JARs
COPY jars/*.jar /opt/flink/lib/
```

Create the `docker-compose.yml` file in the `flink-dremio` directory:

Flink docker-compose.yml

```
services:
  flink-jobmanager:
    build: .
    ports:
      - "8081:8081"
    command: jobmanager
    environment:
      - |
        FLINK_PROPERTIES=
        jobmanager.rpc.address: flink-jobmanager
        parallelism.default: 2
      - AWS_REGION=us-west-2

  flink-taskmanager:
    build: .
    depends_on:
      - flink-jobmanager
    command: taskmanager
    scale: 1
    environment:
      - |
        FLINK_PROPERTIES=
        jobmanager.rpc.address: flink-jobmanager
        taskmanager.numberOfTaskSlots: 4
        parallelism.default: 2
      - AWS_REGION=us-west-2
```

Build and start the Flink cluster:

```
# Build and start the cluster
docker-compose build --no-cache
```

```
docker-compose up -d

# Verify the cluster is running
docker-compose ps

# Verify required JARs are present
docker-compose exec flink-jobmanager ls -la /opt/flink/lib/ | grep -E "(iceberg|hadoop)"
```

You should see the JARs you downloaded in the previous step.

Connect to the Flink SQL client:

```
docker-compose exec flink-jobmanager ./bin/sql-client.sh
```

You can also access the Flink web UI at `http://localhost:8081` to monitor jobs.

Create the Dremio catalog connection in Flink:

```
CREATE CATALOG polaris WITH (
  'type' = 'iceberg',
  'catalog-impl' = 'org.apache.iceberg.rest.RESTCatalog',
  'uri' = 'https://catalog.dremio.cloud/api/iceberg',
  'token' = '<personal_access_token>',
  'warehouse' = '<catalog_name>',
  'header.X-Iceberg-Access-Delegation' = 'vended-credentials',
  'io-impl' = 'org.apache.iceberg.aws.s3.S3FileIO'
);
```

Replace the following:

`<personal_access_token>` with your Dremio personal access token.

`<catalog_name>` with your catalog name.

note

In this configuration, `polaris` is the catalog identifier used in Flink queries. The `CREATE CATALOG` command maps this identifier to your actual Dremio catalog.

In `token`, you provide your Dremio personal access token directly. Dremio's catalog API accepts PATs as bearer tokens without requiring token exchange.

Verify the catalog connection:

```
Flink SQL> show catalogs;
```

Usage Examples

Once Apache Flink is running and connected to your Dremio catalog:

List namespaces

```
Flink SQL> show databases in polaris;
```

Query a table

```
Flink SQL> select * from polaris.your_namespace.your_table;
```

Create a table

```
Flink SQL> CREATE TABLE polaris.demo_namespace.test_table (
  id INT,
  name STRING,
  created_date DATE,
  `value` DOUBLE
);
```

Limitations

Reserved keywords: Column names that are reserved keywords, such as `value`, `timestamp`, and `date`, must be enclosed in backticks when creating or querying tables.

Was this page helpful?

Usage | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/subscription/usage

On this page

There are multiple forms of billable Dremio usage within an organization:

- Dremio Consumption Units (DCUs) represent the usage of Dremio engines. DCUs are only consumed when your engines are running.
- Large-language model (LLM) tokens are billed when you use Dremio's AI features via the Dremio-Provided LLM.
- Storage usage is billed in terabyte-months and only applies to projects that use Dremio-hosted storage. If your projects use an object storage bucket in your account with a cloud provider as the catalog store, storage fees do not apply.

How DCUs are Calculated

The number of DCUs consumed by an engine depends on two factors:

- The size of the engine
- How long the engine and its replicas have been running for

DCU consumption for an engine is calculated as `(Total uptime for the engine and its replicas) * (DCU consumption rate for that engine size)`.

Uptime is measured in seconds and has a 60-second minimum.

The DCU consumption rate for each engine size supported in Dremio is listed in Manage Engines.

DCU Examples

Example 1

An organization has two Dremio Cloud engines defined: Engine A and Engine B, where Engine A is a 2XSmall engine, and Engine B is a Medium engine.

Suppose that between 8 a.m. and 9 a.m. one day:

- Engine A had 2 replicas running for 40 minutes each, so it accumulates a total of 80 minutes of engine uptime.
- Engine B had 5 replicas running for 50 minutes each, so it accumulates a total of 250 minutes of engine uptime.

The total usage for Engine A for this hour is `(80/60) * (16 DCUs/hour) = 21.33 DCUs`.

The total usage for Engine B for this hour is `(250/60) * (128 DCUs/hour) = 533.33 DCUs`.

Example 2

An organization has one Dremio Cloud engine defined: Engine A, where Engine A is a Medium engine.

Suppose that between 8 a.m. and 9 a.m. one day:

- Engine A had 1 replica running for the entire hour (60 minutes).
- Engine A needed to spin up an additional replica for 30 minutes to tackle a workload spike.

Engine A accumulated a total of 90 minutes of engine uptime, so the total usage for Engine A for this hour is `(90/60) * (128 DCUs/hour) = 192 DCUs`.

How AI Usage Is Calculated

If you use the Dremio-Provided LLM, you pay directly for the cost of both the input and output tokens used. If you connect to another LLM via your own model provider, you are not currently charged for this usage.

AI Examples

Example 1

Say that you use an external model provider as well as the Dremio-Provided LLM to use Dremio's AI features, resulting in a usage footprint like the below:

- External model provider 500K input tokens used.
- External model provider: 30K output tokens used.
- Dremio-Provided: 200K input tokens used.
- Dremio-Provided: 20K output tokens used.

You are not charged for using Dremio's AI features via an external model. Instead, you are only charged for the tokens consumed by the Dremio-Provided LLM:

- (200K input tokens)*($1.25/1 million tokens) = $0.25
- (20K output tokens)*($10.00/1 million tokens) = $0.20

In this scenario, you would be billed for $0.45 of AI feature usage.

In order to simplify the billing experience for AI features Dremio may explore the addition of an AI specific credit, similar to DCUs, in the future.

How Storage Usage Is Calculated

Storage is calculated through the collection of periodic snapshots of the Dremio-hosted bucket. These snapshots throughout a billing period are averaged through the billing period to calculate a number of billable terabyte-months.

Storage Usage Examples

Example 1

Suppose an organization has one Dremio project in a region where the price of a terabyte-month is $23.00, and that in a given month this project:

- Stores 1 terabyte of data for the entire 30 days of the billing period

Then the total amount charged for the storage would be (1) * ($23.00) = $23.00

Example 2

Suppose your organization has a project in a region where the price of a terabyte-month is $23.00, and that in a given period this project:

- Stores 1 terabyte of data for the first 15 days of the month
- Stores 2 terabyte of data for the last 15 days of the month

On average throughout the month, the project was storing 1.5Tb of data. So the bill would be (1.5) * ($23.00) = $34.5.

Was this page helpful?

- How DCUs are Calculated
- DCU Examples
- How AI Usage Is Calculated
- AI Examples
- How Storage Usage Is Calculated
- Storage Usage Examples

Project Preferences | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/projects/preferences

Preferences let you customize the behavior of specific features in the Dremio console.

To view the available preferences:

- In the Dremio console, hover over in the side navigation bar and select **Project settings**.
- Select **Preferences** in the project settings sidebar.

This opens the Preferences page, showing the Dremio console settings that can be modified.

- Use the toggle next to the setting to enable or disable for all users.

If any preferences are modified, users must refresh their browsers to see the change.

These preferences and their descriptions are listed in the table below.

Setting	Default	Enabled	Disabled	Details
SQL Autocomplete	Enabled	Autocomplete provides suggestions for SQL keywords, catalog objects, and functions while you are constructing SQL statements. is visible in the SQL editor, although users can switch the button off within their own accounts.	The button is hidden from the SQL editor and suggestions are not provided.	See how this works in the SQL editor.
Copy or Download Results	Enabled	and are visible above the results table, because users are allowed to copy or download the results in the SQL editor.	The buttons are hidden and users cannot copy or download results in the SQL editor.	See how this works in result set actions.
Query Dataset on Click	Enabled	Clicking on a dataset opens the SQL Runner with a `SELECT` statement on the dataset.	If you would rather click directly on a dataset to see or edit the definition, disable this preference. Clicking on a dataset opens the Datasets page, showing a `SELECT` statement on the dataset or the dataset's definition that you can view or edit depending on your dataset privileges.	
Autonomous Reflections	Enabled	Dremio automatically creates and drops Reflections based on query patterns from the last 7 days to seamlessly accelerate performance.	Dremio will provide recommendations to create and drop Reflections based on query patterns from the last 7 days to accelerate query performance.	See how this works in the Autonomous Reflections.
AI Features	Enabled	When enabled, users can interact with the Dremio's AI Agent and AI functions. The AI Agent enables agentic workflows, allowing analysts to work with the agent to generate SQL queries, find insights, and create visualizations. The AI functions allow engineers to query unstructured data and use LLMs during SQL execution.	The AI Agent and AI functions will not work.	See how this works in Explore with AI Agent and AI Functions.
Generate wikis and labels	Enabled	In the Details panel, both **Generate wiki** and **Generate labels** links will be visible for generating wikis and labels.	The links for **Generate wiki** and **Generate labels** will be hidden, making these features unavailable.	See how this works in Wikis and Labels.

Was this page helpful?

Profiles | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/monitor/jobs/profiles

On this page

Visual profiles and raw profiles are available for jobs that have run queries.

Visual Profiles

You can view the operations in visual profiles to diagnose performance or cost issues and to see the results of changes that you make, either to queries themselves or their environment, to improve performance or reduce costs.

A query profile details the plan that Dremio devised for running a query and shows statistics from the query's execution. A visual representation of a query profile is located on the Visual Profile tab. This visual profile consists of operators that are arranged as a tree, where each operator has one or two upstream operators that represent a specific action, such as a table scan, join, or sort. At the top of the tree, a

single root operator represents the query results, and at the bottom, the leaf operators represent scan or read operations from datasets.

Data processing begins with the reading of datasets at the bottom of the tree structure, and data is sequentially processed up the tree. A query plan can have many branches, and each branch is processed separately until a join or other operation connects it to the rest of the tree.

Phases

A query plan is composed of query phases (also called major fragments), and each phase defines a series of operations that are running in parallel. A query phase is depicted by the same colored boxes that are grouped together in a visual profile.

Within the query phases are multiple, single-threaded instances (also called minor fragments) running in parallel. Each thread is processing a different set of data through the same series of operations, and this data is exchanged from one phase to another. The number of threads for each operator can be found in the Details section (right panel) of a visual profile.

Use Visual Profiles

To navigate to the visual profile for a job:

Click !This is the icon that represents the Jobs page. in the side navigation bar.

On the Jobs page, click a job that you would like to see the visual profile for.

At the top of the next page, click the Visual Profile tab to open.

The main components of a visual profile are shown below:

Location	Description
1	The Visual Profile tab shows a visual representation of a query profile.
2	The left panel is where you can view the phases of the query execution or single operators, sorting them by runtime, total memory used, or records produced. Operators of the same color are within the same phase. Clicking the Collapse This button hides a panel from view. hides the left panel from view.
3	The tree graph allows you to select an operator and find out where it is in relation to the rest of the query plan.
4	The zoom controls the size of the tree graph so it's easier for you to view.
5	The right panel shows the details and statistics about the selected operator. Clicking the Collapse This button hides a panel from view. hides the right panel from view.

Use Cases

Improve the Performance of Queries

You may notice that a query is taking more time than expected and want to know if something can be done to reduce the execution time. By viewing its visual profile, you can, for example, quickly find the operators with the highest processing times.

You might decide to try making simple adjustments to cause Dremio to choose a different plan. Some of the possible adjustments include:

- Adding a filter on a partition column to reduce the amount of data scanned
- Changing join logic to avoid expanding joins (which return more rows than either of the inputs) or nested-loop joins
- Creating a Reflection to avoid some of processing-intensive work done by the query

Reduce Query-Execution Costs

If you are an administrator, you may be interested in tuning the system as a whole to support higher concurrency and lower resource usage across the system, because you want to identify the most expensive queries in the system and then see what can be done to lower the cost of these queries. Such an investigation is often important even if individual users are happy with the performance of their own queries.

On the Jobs page, you can use the columns to find the queries with the highest cost, greatest number of rows scanned, and more. You can then study the visual profiles for these queries, identifying system or data problems, and mismatches between how data is stored and how these queries retrieve it. You can try repartitioning data, modifying data types, sorting, creating views, creating Reflections, and other changes.

Raw Profiles

Click **Raw Profile** to open a raw profile of the job in a separate dialog, which includes a job summary, state durations, threads, resource allocation, operators, visualized plan, acceleration, and other details.

A raw profile is a UI-generated profile that is a subset of the data that you can download and provides a summary of metrics collected for each executed query that can be used to monitor and analyze query performance.

To navigate to a raw profile:

- Click !This is the icon that represents the Jobs page. in the side navigation bar to open the Jobs page.
- On the Jobs page, click a job that you would like to see the raw profile for.
- At the top of the next page, click the Raw Profile tab to open a raw profile of the job in a separate dialog. The associated raw profile dialog shows a variety of information for review.

Views

Within the Raw Profile dialog, you can analyze the Job Metrics based on the following views:

View	Description
Query	Shows the selected query statement and job metrics. See if your SQL query is what you were expecting and the query is run against the source data.
Visualized Plan	Shows a visualized diagram and job metrics. This view is useful in understanding the flow of the query and for analyzing out of memory issues and incorrect results. The detailed visualized pan diagram is always read from the bottom up.
Planning	Shows planning metrics, query output schema, non default options, and job metrics. This view shows how query planning is executed, because it provides statistics about the actual cost of the query operations in terms of memory, input/output, and CPU processing. You can use this view to identify which operations consumed the majority of the resources during a query and to address the cost-intensive operations. In particular, the following information is useful: * Non Default Options – See if non-default parameters are being used. * Metadata Cache Hits and Misses with times * Final Physical Transformation – Look for pushdown queries for RDBMS, MongoDB, or Elasticsearch, filter pushdowns or partition pruning for parquet, and view usage of stripes for ORC. * Compare the estimated row count versus the actual scan, join, or aggregate result. * Row Count – See if row count (versus rows) is used. Row count can cause an expensive broadcast. * Build – See if build (versus probe) is used. Build loads data into memory.
Acceleration	Shows Reflection outcome, canonicalized user query alternatives, Reflection details, and job metrics. * Multiple substitutions – See if the substitutions are excessive. * System activity – See if `sys.project.reflections`, `sys.project.materializations`, and `sys.project.refreshes` are excessive. * Comparisons – Compare cumulative cost (found in Best Cost Replacement Plan) against Logical Planning, which is in the Planning view. This view is useful for determining whether exceptions or matches are occurring. The following considerations determines the acceleration process: * Considered, Matched, Chosen – The query is accelerated. * Considered, Matched, Not Chosen – The query is not accelerated because either a costing issue or an exception during substitution occurred. * Considered, Not Matched, Not Chosen – The query is not accelerated because the Reflection does not have the data to accelerate.
Error	(If applicable) Shows information about an error. The Failure Node is always the coordinator node and the server name inside the error message is the actual affected node.

Job Metrics

Each view displays the following metrics:

Job Summary

Time in UTC

State Durations

Context

Threads

Resource Allocation

Nodes

- **Operators**

Job Summary

The job summary information includes:

- State
- Coordinator
- Threads
- Command Pool Wait
- Total Query Time

Joins in user query

Joins in final plan

- Considered Reflections
- Matched Reflections
- Chosen Reflections

Time in UTC

The Time in UTC section lists the job's start and end time, in UTC format.

State Durations

The State Durations section lists the length of time (in milliseconds) for each of the job states:

- Pending
- Metadata Retrieval
- Planning
- Engine Start
- Queued
- Execution Planning
- Starting
- Running

For descriptions of the job states, see Job States and Statuses.

Context

If you are querying an Iceberg catalog object, the Context section lists the Iceberg catalog and branch that is referenced in the query. Otherwise, the Context section is not populated. Read Iceberg Catalogs in Dremio for more information.

Threads

The Threads section provides an overview table and a major fragment block for each major fragment. Each row in the Overview table provides the number of minor fragments that Dremio parallelized from each major fragment, as well as aggregate time and memory metrics for the minor fragments.

Major fragment blocks correspond to a row in the Overview table. You can expand the blocks to see metrics for all of the minor fragments that were parallelized from each major fragment, including the host on which each minor fragment ran. Each row in the major fragment table presents the fragment state, time metrics, memory metrics, and aggregate input metrics of each minor fragment.

In particular, the following metrics are useful:

- Setup – Time opening and closing of files.
- Waiting – Time waiting on the CPU.
- Blocked on Downstream – Represents completed work whereas the next phase is not ready to accept work.
- Blocked on Upstream – Represents the phase before it is ready to give work though the cloud phase is not ready.
- Phase Metrics – Displays memory used per node (Phases can run in parallel).

Resource Allocation

The Resource Allocation section shows the following details for managed resources and workloads:

- Engine Name
- Queue Name
- Queue Id
- Query Cost
- Query Type

Nodes

The Nodes section includes host name, resource waiting time, and peak memory.

Operators

The Operators section shows aggregate metrics for each operator within a major fragment that performed relational operations during query execution.

Operator Overview Table

The following table lists descriptions for each column in the Operators Overview table:

Column Name	Description
SqlOperatorImpl ID	The coordinates of an operator that performed an operation during a particular phase of the query. For example, 02-xx-03 where 02 is the major fragment ID, xx corresponds to a minor fragment ID, and 03 is the Operator ID.
Type	The operator type. Operators can be of type project, filter, hash join, single sender, or unordered receiver.
Min Setup Time, Avg Setup Time, Max Setup Time	In general, the time spent opening and closing files. Specifically, the minimum, average, and maximum amount of time spent by the operator to set up before performing the operation.
Min Process Time, Avg Process Time, Max Process Time	The shortest amount of time the operator spent processing a record, the average time the operator spent in processing each record, and the maximum time that the operator spent in processing a record.
Wait (min, avg, max)	In general, the time spent waiting on Disk I/O. These fields represent the minimum, average, and maximum times spent by operators waiting on disk I/O.
Avg Peak Memory	Represents the average of the peak direct memory allocated across minor fragments. Relates to the memory needed by operators to perform their operations, such as hash join or sort.
Max Peak Memory	Represents the maximum of the peak direct memory allocated across minor fragments. Relates to the memory needed by operators to perform their operations, such as hash join or sort.

Operator Block

The Operator Block shows time and memory metrics for each operator type within a major fragment. Examples of operator types include:

- SCREEN
- PROJECT
- WRITER_COMMITTER
- ARROW_WRITER

The following table describes each column in the Operator Block:

Column Name	Description
Thread	The coordinate ID of the minor fragment on which the operator ran. For example, 04-03-01 where 04 is the major fragment ID, 03 is the minor fragment ID, and 01 is the Operator ID.
Setup Time	The amount of time spent by the operator to set up before performing its operation. This includes run-time code generation and opening a file.
Process Time	The amount of time spent by the operator to perform its operation.
Wait Time	The cumulative amount of time spent by an operator waiting for external resources. such as waiting to send records, waiting to receive records, waiting to write

to disk, and waiting to read from disk. |
Max Batches	The maximum number of record batches consumed from a single input stream.
Max Records	The maximum number of records consumed from a single input stream.
Peak Memory	Represents the peak direct memory allocated. Relates to the memory needed by the operators to perform their operations, such as hash join and sort.
Host Name	The hostname of the Executor the minor fragment is running on.
Record Processing Rate	The rate at which records in the minor fragment are being processed. Combined with the Host Name, the Record Processing Rate can help find hot spots in the cluster, either from skewed data or a noisy query running on the same cluster.
Operator State	The status of the minor fragment.
Last Schedule Time	The last time at which work related to the minor fragment was scheduled to be executed.

Operator blocks also contain three drop-down menus: Operator Metrics, Operator Details, and Host Metrics. Operator Metrics and Operator Details are unique to the type of operator and provide more detail about the operation of the minor fragments. Operator Metrics and Operator Details are intended to be consumed by Dremio engineers. Depending on the operator, both can be blank. Host Metrics provides high-level information about the host used when executing the operator.

Was this page helpful?

Visual Profiles

Phases

Use Visual Profiles

Use Cases

Raw Profiles

Views

Job Metrics

Results Cache | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/performance/results-cache

On this page

Results cache improves query performance by reusing results from previous executions of the same deterministic query, provided that the underlying dataset remains unchanged and the previous execution was by the same user. The results cache feature works out of the box, requires no configuration, and automatically caches and reuses results. Regardless of whether a query uses results cache, it always returns the same results.

Results cache is client-agnostic, meaning a query executed in the Dremio console will result in a cache hit even if it is later re-run through other clients like JDBC, ODBC, REST, or Arrow Flight. For a query to use the cache, its query plan must remain identical to the original cached version. Any changes to the schema or dataset generate a new

query plan, invalidating the cache.

Results cache also supports seamless coordinator scale-out, allowing newly added coordinators to benefit immediately from previously cached results.

Cases Supported By Results Cache

Query result are cached in the following cases:

- The SQL statement is a `SELECT` statement.
- The query reads from an Iceberg, Parquet dataset, or from a raw Reflection defined on other Dremio supported data sources and formats, such as relational databases, `CSV`, `JSON`, or `TEXT`.
- The query does not contain dynamic functions such as `QUERY_USER`, `IS_MEMBER`, `RAND`, `CURRENT_DATE`, or `NOW`.
- The query does not reference `SYS` or `INFORMATION_SCHEMA` tables, or use external query.
- The result set size, when stored in Arrow format, is less than or equal to 20 MB.
- The query is not executed in Dremio console as a preview.

View Whether Queries Used Results Cache

You can view the list of jobs on the Jobs page to determine if queries from data consumers were accelerated by the results cache.

To find whether a query was accelerated by a results cache:

- Find the job that ran the query and look for !This is the icon that indicates a Reflection was used. next to it, which indicates that the query was accelerated using either Reflections or the results cache.
- Click on the row representing the job that ran the query to view the job summary. The summary, displayed in the pane to the right, provides details on whether the query was accelerated using results cache or Reflections.

!Results cache on the Job Overview page

Storage

Cached results are stored in the project store alongside all project-specific data, such as metadata and Reflections. Executors write cache entries as Arrow data files and read them when processing `SELECT` queries that result in a cache hit. Coordinators are responsible for managing the deletion of expired cache files.

Deletion

A background task running on one of the Dremio coordinators handles cache expiration. This task runs every hour to mark cache entries that have not been accessed in the

past 24 hours as expired and subsequently deletes them along with their associated cache files.

Considerations and Limitations

SQL queries executed through the Dremio console or a REST client that access the cache will rewrite the cached query results to the job results store to enable pagination.

Was this page helpful?

- Cases Supported By Results Cache
- View Whether Queries Used Results Cache
- Storage
- Deletion
- Considerations and Limitations

Workload Management | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/engines/workload-management

On this page

This topic covers how to manage resources and workloads by routing queries to particular engines through rules.

Overview

You can manage Dremio workloads via routing rules, which are evaluated at runtime (before query planning) to decide which query engine to use for a given query. In projects with only one engine, all queries share the same execution resources and route to the same single engine. However, when multiple engines are provisioned, rules determine the engine to be used.

You must arrange the rules in the order that you want them to be evaluated. In the case that multiple rules evaluate to true for a given query, the first rule that returns true will be used to select the engine.
The following diagram shows a series of rules that are evaluated when a job gets submitted.

- Rule1 routes jobs to Engine1
- Rule2 routes jobs to Engine2
- Rule3 routes jobs to the default engine that was created on project start up
- Rule4 rejects the jobs that evaluate to true

Rules

You can use Dremio SQL syntax to specify rules to target particular jobs.

The following are the types of rules that can be created along with examples.

User

Create a rule that identifies the user that triggers the job.

Create rule that identifies user

```
USER in ('JRyan','PDirk','CPhillips')
```

Group Membership

Create a rule that identifies if the user that triggers the job is part of a particular group.

Create rule that identifies whether user belongs to a specified group

```
is_member('MarketingOps') OR
is_member('Engineering')
```

Job Type

Create a rule depending on the type of job. The types of jobs can be identified by the following categories:

- Flight
- JDBC
- Internal Preview
- Internal Run
- Metadata Refresh
- ODBC
- Reflections
- REST
- UI Download
- UI Preview
- UI Run

Create rule based on type of job

```
query_type() IN ('JDBC', 'ODBC', 'UI Run', 'Flight')
```

Query Label

Labels enable rules that route queries running named commands to specific engines. Dremio supports the following query labels:

Query Label	Description
COPY	Assigned to all queries running a COPY INTO SQL command
CTAS	Assigned to all queries running a CREATE TABLE AS SQL command
DML	Assigned to all queries running an INSERT, UPDATE, DELETE, MERGE, or TRUNCATE SQL command
OPTIMIZATION	Assigned to all queries running an OPTIMIZE SQL command

Here are two example routing rules:

Create a routing rule for queries running a COPY INTO command

```
query_label() IN ('COPY')
```

Create a routing rule for queries running the DML commands INSERT, UPDATE, DELETE, MERGE, or TRUNCATE

```
query_label() IN ('DML')
```

Query Attributes

Query attributes enable routing rules that direct queries to specific engines based on their characteristics.

Dremio supports the following query attributes:

Query Attribute	Description
`DREMIO_MCP`	Set when the job is submitted via the Dremio MCP Server.
`AI_AGENT`	Set when the job is submitted via the Dremio AI Agent.
`AI_FUNCTIONS`	Set when the job contains AI functions.

You can use the following functions to define routing rules based on query attributes:

Function	Applicable Attribute	Description
`query_has_attribute(<attr>)`	`DREMIO_MCP`, `AI_AGENT`, `AI_FUNCTIONS`	Returns true if the specified attribute is present.
`query_attribute(<attr>)`	`DREMIO_MCP`, `AI_AGENT`, `AI_FUNCTIONS`	Returns the value of the attribute (if present), otherwise NULL.
`query_calls_ai_functions()`	NA	Returns true if the job has an AI function in the query.

Examples:

Create a routing rule for queries that use AI functions and are executed by a user

```
query_calls_ai_functions() AND USER = 'JRyan'
```

Create a routing rule for queries with `DREMIO_MCP` and `AI_FUNCTION`

```
query_has_attribute('DREMIO_MCP') AND query_has_attribute('AI_FUNCTIONS')
```

Tag

Create a rule that routes jobs based on a routing tag.

Create rule that routes jobs based on routing tag

```
tag() = 'ProductionDashboardQueue'
```

Date and Time

Create a rule that routes a job based on the time it was triggered. Use Dremio SQL Functions.

Create rule that routes jobs based on time triggered

```
EXTRACT(HOUR FROM CURRENT_TIME)
BETWEEN 9 AND 18
```

Combined Conditions

Create rules based on multiple conditions.

The following example routes a job depending on user, group membership, query type, query cost, tag, and the time of day that it was triggered.

Create rule based on user, group, job type, query cost, tag, and time triggered

```
(
USER IN ('JRyan', 'PDirk', 'CPhillips')
OR  is_member('superadmins')
)
AND query_type IN ('ODBC')
AND EXTRACT(HOUR FROM CURRENT_TIME)
BETWEEN 9 AND 18
```

Default Rules

Each Dremio project has its own set of rules. When a project is created, Dremio automatically creates rules for the default and preview engines. You can edit these rules as needed.

Order	Rule Name	Rule	Engine
1	UI Previews	query_type() = 'UI Preview'	preview

| 2 | Reflections | query_type() = 'Reflections' | default |
| 3 | All Other Queries | All other queries | default |

View All Rules

To view all rules:

- Click the Project Settings !This is the icon that represents the Project Settings. icon in the side navigation bar.

- Select **Engine Routing** in the project settings sidebar to see the list of engine routing rules.

Add a Rule

To add a rule:

- On the Engine Routing page, click the **Add Rule** button at the top-right corner of the screen.

- In the **New Rule** dialog, for **Rule Name**, enter a name.

- For **Conditions**, enter the routing condition. See Rules for supported conditions.

- For **Action**, complete one of the following options:

a. If you want to route the jobs that meet the conditions to a particular engine, select the **Route to engine** option. Then use the engine selector to choose the engine.

b. If you want to reject the jobs that meet the conditions, select the **Reject** option.

- Click **Add**.

Edit a Rule

To edit a rule:

- On the Engine Routing page, hover over the rule and click the Edit Rule !This is the icon that represents the Edit Rule settings. icon that appears next to the rule.

- In the **Edit Rule** dialog, for **Rule Name**, enter a name.

- For **Conditions**, enter the routing condition. See Rules for supported conditions.

- For **Action**, complete one of the following options:

a. If you want to route the jobs that meet the conditions to a particular engine, select the **Route to engine** option. Then use the engine selector to choose the engine.

b. If you want to reject the jobs that meet the conditions, select the **Reject** option.

- Click **Save**.

Delete a Rule

To delete a rule:

On the Engine Routing page, hover over the rule and click the Delete Rule !This is the icon that represents the Delete Rule settings. icon that appears next to the rule.

caution

You must have at least one rule per project to route queries to a particular engine.

In the **Delete Rule** dialog, click **Delete** to confirm.

Set and Reset Engines

The `SET ENGINE` SQL command is used to specify the exact execution engine to run subsequent queries in the current session. When using `SET ENGINE`, WLM rules and direct routing connection properties are bypassed, and queries are routed directly to the specified queue. The `RESET ENGINE` command clears the session-level engine override, reverting query routing to follow the Workload Management (WLM) rules or any direct routing connection property if set.

SET TAG

The `SET TAG` SQL command is used to specify routing tag for subsequent queries in the current session. If a `ROUTING_TAG` connection property is already set for the session, `SET TAG` will override it. When using `SET TAG`, you must have a previously defined Workload Management (WLM) routing rule that routes queries based on that routing tag. The `RESET TAG` command clears the session-level routing tag override, reverting query routing to follow the Workload Management (WLM) rules or any direct routing connection property if set.

Connection Tagging and Direct Routing Configuration

Routing tags are configured by setting the `ROUTING_TAG = <Tag Name>` parameter for a given session to the desired tag name.

JDBC Session Configuration

To configure JDBC sessions add the `ROUTING_TAG` parameter to the JDBC connection URL. For example: `jdbc:dremio:direct=localhost;ROUTING_TAG='TagA'`.

ODBC Session Configuration

Configure ODBC sessions as follows:

Windows Sessions

Add the `ROUTING_TAG` parameter to the `AdvancedProperties` parameter in the ODBC DSN field.

Mac OS Sessions

- Add the `ROUTING_TAG` parameter to the `AdvancedProperties` parameter in the system `odbc.ini` file located at `/Library/ODBC/odbc.ini`. After adding the parameter, an example Advanced Properties configuration might be: `AdvancedProperties=CastAnyToVarchar=true;HandshakeTimeout=5;QueryTimeout=180;TimestampTZDisplayTimezone=utc;NumberOfPrefetchBuffers=5;ROUTING_TAG='TagA';`

- Add the `ROUTING_TAG` parameter to the `AdvancedProperties` parameter in the user's DSN located at `~/Library/ODBC/odbc.ini`

Best Practices for Workload Management

Because every query workload is different, engine sizing often depends on several factors, such as the complexity of queries, number of concurrent users, data sources, dataset size, file and table formats, and specific business requirements for latency and cost. Workload management (WLM) ensures reliable query performance by choosing adequately sized engines for each workload type, configuring engines, and implementing query routing rules to segregate and route query workload types to appropriate engines.

This section describes best practices for adding and using Dremio engines, as well as configuring WLM to achieve reliable query performance in Dremio. This section also includes tips for migrating from self-managed Dremio Software to fully managed Dremio and information about using the system table `sys.project.history.jobs`, which stores metadata for historical jobs executed in a project, to assess the efficacy of WLM settings and make adjustments.

Set Up Engines

As a fully managed offering, Dremio is the best deployment model for Dremio in production because it allows you to achieve high levels of reliability and durability for your queries and maximize resource efficiency with engine autoscaling and does not require you to manually create and manage engines.

Segregating workload types into separate engines is vital for mitigating noisy neighbor issues, which can jeopardize performance reliability. You can segregate workloads by type, such as ad hoc, dashboard, and lakehouse (COPY INTO, DML, and optimization), as well as by business unit to facilitate cost distribution.

Metadata and Reflection refresh workloads should have their own engines for executing metadata and Reflection refresh queries. These internal queries can use a substantial amount of engine bandwidth, so assigning separate engines ensures that they do not interfere with user-initiated queries. Initial engine sizes should be XSmall and Small, but these sizes may change depending on the number and complexity of Reflection refresh and metadata jobs.

Dremio recommends the following engine setup configurations:

- Dremio offers a range of engine sizes. Experiment with typical queries, concurrency, and engine sizes to establish the best engine size for each workload type based on your organization's budget constraints and latency requirements.

- Maximum concurrency is the maximum number of jobs that Dremio can execute

concurrently on an engine replica. Dremio provides an out-of-the-box value for maximum concurrency based on engine size, but we recommend testing with typical queries directed to specific engines to determine the best maximum concurrency values for your query workloads.

Dremio offers autoscaling to meet the demands of dynamic workloads with engine replicas. It is vital to assess and configure each engine's autoscaling parameters based on your organization's budget constraints and latency requirements for each workload type. You can choose the minimum and maximum number of replicas for each engine and specify any advanced configuration as needed. For example, dashboard workloads must meet stringent low-latency requirements and are prioritized for performance rather than cost. Engines added and assigned to execute the dashboard workloads may therefore be configured to autoscale using replicas. On the other hand, an engine for ad hoc workloads may have budget constraints and therefore be configured to autoscale with a maximum of one replica.

Route Workloads

Queries are routed to engines according to routing rules. You may use Dremio's out-of-the-box routing rules that route queries to the preview engines that are established by default, but Dremio recommends creating custom routing rules based on your workloads and business requirements. Custom rules can include factors such as user, group membership, job type, date and time, query label, and tag. Read Rules for examples.

The following table lists example routing rules based on query_type, query_label, and tags:

Order	Rule Name	Rule	Engine
1	Reflections	`query_type() = 'Reflections'`	Reflection
2	Metadata	`query_type() = 'Metadata Refresh'`	Metadata
3	Dashboards	`tag() = 'dashboard'`	Dashboard
4	Ad hoc Queries	`query_type() IN ( 'UI Run' , 'REST') OR tag() = 'ad hoc'`	Ad hoc
5	Lakehouse Queries	`query_label() IN ('COPY','DML','CTAS', 'OPTIMIZATION')`	Lakehouse
6	All Other Queries	All other queries	Preview

Use the `sys.project.history.jobs` System Table

The `sys.project.history.jobs` system table contains metadata for recent jobs executed in a project, including time statistics, cost, and other relevant information. You can use the data in the `sys.project.history.jobs` system table to evaluate the effectiveness of WLM settings and make adjustments based on job metadata.

Use Job Analyzer

Job Analyzer is a package of useful query and view definitions that you may create over the `sys.projects.history.jobs` system table and use to analyze job metadata. Job Analyzer is available in a public GitHub repository.

Was this page helpful?

Overview

Rules

User

Group Membership

Job Type

Query Label

Query Attributes

Tag

Date and Time

Combined Conditions

Default Rules

View All Rules

Add a Rule

Edit a Rule

Delete a Rule

Set and Reset Engines

SET TAG

Connection Tagging and Direct Routing Configuration

JDBC Session Configuration

ODBC Session Configuration

Best Practices for Workload Management

Set Up Engines

Route Workloads

Use the `sys.project.history.jobs` System Table

Use Job Analyzer

Manual Reflections | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/performance/manual-reflections/

On this page

With Autonomous Reflections reducing the need for manual work, you no longer need

to create or manage Reflections. However, when Autonomous Reflections are not enabled or for situations that require manual control, this page provides guidance on getting Reflection recommendations and how to manage raw Reflections, aggregation Reflections, and external Reflections in Dremio.

note

For non-duplicating joins, Dremio can accelerate queries that reference only some of the joins in a Reflection, eliminating the need to create separate Reflections for every table combination.

Reflection Recommendations

When Autonomous Reflections are not enabled, Dremio automatically provides recommendations to add and remove Reflections based on query patterns to optimize performance for queries on Iceberg tables, UniForm table, Parquet datasets, and any views built on these datasets.

Recommendations to add Reflections are sorted by overall effectiveness, with the most effective recommendations shown on top. Effectiveness relates to metrics such as the estimated number of accelerated jobs, potential increase in query execution speedup, and potential time saved during querying. These are rough estimates based on past data that can give you insight into the potential benefits of each recommendation.
Reflections created using these recommendations refresh automatically when source data changes on:

- Iceberg tables – When the table is modified through Dremio or other engines. Dremio polls tables every 10 seconds.
- Parquet datasets ––– When metadata is updated in Dremio.

To view and apply the Reflection recommendations:

- In the Dremio console, hover over in the side navigation bar and select **Project Settings**.
- Select **Reflections** from the project settings sidebar.
- Click **Reflections Recommendations** to access the list of suggested Reflections.
- To apply a recommendation, click at the end of the corresponding row.

Reflections created using usage-based recommendations are only used when fully synchronized with their source data to ensure up-to-date query results.

To generate recommendations for default raw and aggregation Reflections, you can obtain the job IDs by looking them up on the Jobs page. Then, use either the `SYS.RECOMMEND_REFLECTIONS` table function or the Recommendations API to submit job IDs to accelerate specific SQL queries.

Raw Reflections

Retain the same number of records as its anchor while allowing a subset of columns. It enhances query performance by materializing complex views, transforming data from

non-performant sources into the Iceberg table format optimized for large-scale analytics, and utilizing partitioning and sorting for faster access. By precomputing and storing data in an optimized format, raw Reflections significantly reduce query latency and improve overall efficiency.

You can use the Reflections editor to create two types of raw Reflection:

- A default raw Reflection that includes all of the columns of the anchor, but does not sort or horizontally partition on any columns
- A raw Reflection that includes all or a subset of the columns of the anchor, and that does one or both of the following things:
  - Sorts on one or more columns
  - Horizontally partitions the data according to the values in one or more columns

note

For creating Reflections on views and tables with row-access and column-masking policies, see Use Reflections on Datasets with Policies.

Prerequisites

- If you want to accelerate queries on unoptimized data or data in slow storage, create a view that is itself created from a table in a non-columnar format or on slow-scan storage. You can then create your raw Reflection from that view.
- If you want to accelerate "needle-in-a-haystack" queries, create a view that includes a predicate to include only the rows that you want to scan. You can then create your raw Reflection from that view.
- If you want to accelerate queries that perform expensive transformations, create a view that performs those transformations. You can then create your raw Reflection from that view.
- If you want to accelerate queries that perform joins, create a view that performs the joins. You can then create your raw Reflection from that view.

Create Default Raw Reflections

In the **Basic** view of the Reflections editor, you can create a raw Reflection that includes all of the fields that are in a table or view. Creating a basic raw Reflection ensures that Dremio never runs user queries against the underlying table or view when the raw Reflection is enabled.

To create a raw Reflection in the **Basic** view of the Reflections editor:

- In the Dremio console, click !This is the icon that represents the Datasets page. in the side navigation bar to go to the Datasets page.
- In the catalog or folder in which the anchor is located, hover over the anchor name and click .
- Select **Reflections** in the table or view settings sidebar.

- Click the toggle switch on the left side of the **Raw Reflections** bar.

- Click **Save**.

Restrictions of the **Basic** View

- You cannot select fields to sort or create horizontal partitions on.
- The name of the Reflection that you create is restricted to "Raw Reflection".
- You can create only one raw Reflection. If you want to create multiple raw Reflections at a time, use the **Advanced** view.

Create Customized Raw Reflections

In the **Advanced** view of the Reflections editor, you can create one or more raw Reflections that include all or a selection of the fields that are in the anchor or supported anchor. You can also choose sort fields and fields for partitioning horizontally.

Dremio recommends that you follow the best practices listed in Operational Excellence when you create customized raw Reflections.

If you make any of the following changes to a raw Reflection when you are using the **Advanced** view, you cannot switch to the **Basic** view:

- Deselect one or more fields in the **Display** column. By default, all of the fields are selected.
- Select one or more fields in the **Sort**, **Partition**, or **Distribute** column.

To create a raw Reflection in the **Advanced** view of the Reflections editor:

- In the Dremio console, click !This is the icon that represents the Datasets page. in the side navigation bar to go to the Datasets page.
- In the catalog or folder in which the anchor is located, hover over the anchor name and click  .
- If the **Advanced** view is not already displayed, click the **Advanced View** button in the top-right corner of the editor.
- Click the toggle switch in the table labeled **Raw Reflection** to enable the raw Reflection.

Queries do not start using the Reflection, however, until after you have finished editing the Reflection and click **Save** in a later step.

- (Optional) Click in the label to rename the Reflection.

The purpose of the name is to help you understand, when you read job reports, which

Reflections the query optimizer considered and chose when planning queries.

In the columns of the table, follow these steps, which you don't have to do in any particular order:

note

Ignore the **Distribution** column. Selecting fields in it has no effect on the Reflection.

Click in the **Display** column to include fields in or exclude them from your Reflection.

Click in the **Sort** column to select fields on which to sort the data in the Reflection. For guidance in selecting a field on which to sort, see Sort Reflections on High-Cardinality Fields.

Click in the **Partition** column to select fields on which to horizontally partition the rows in the Reflection. For guidance in selecting fields on which to partition, and which partition transforms to apply to those fields, see Horizontally Partition Reflections that Have Many Rows.

note

If the Reflection is based on an Iceberg table, a filesystem source, an AWS Glue source, or a Hive source, and that table is partitioned, recommended partition columns and transforms are selected for you. If you change the selection of columns, then this icon appears at the top of the table: !This is the Recommendations icon.. You can click it to revert to the recommended selection of partition columns.

(Optional) Optimize the number of files used to store the Reflection. You can optimize for fast refreshes or for fast read performance by queries. Follow these steps:

a. Click the in the table in which you are defining the Reflection.

b. In the field **Reflection execution strategy**, select either of these options:

Select **Minimize Time Needed To Refresh** if you need the Reflection to be created as fast as possible. This option can result in the data for the Reflection being stored in many small files. This is the default option.

Select **Minimize Number Of Files** when you want to improve the read performance of queries against the Reflection. With this option, there tend to be fewer seeks performed for a given query.

Click **Save** when you are finished.

Edit Raw Reflections

You can edit an existing raw Reflection. You might want to do so if you are iteratively designing and testing a raw Reflection, if the definition of the view that the Reflection was created from was changed, or if the schema of the underlying table was changed.

If you created a raw Reflection in the **Basic** view of the Reflections editor, you must use the **Advanced** view to edit it.

Dremio runs the job or jobs to recreate the Reflection after you click **Save**.

To edit a raw Reflection in the **Advanced** view of the Reflections editor:

In the Dremio console, hover over in the side navigation bar and select **Project settings**.

Select **Reflections** in the project settings sidebar.

Click the name of the Reflection. This opens the Acceleration dialog with the Reflections editor.

Click the **Advanced View** button in the top-right corner of the editor.

In the **Raw Reflections** section of the **Advanced** view, locate the table that shows the definition of your Reflection.

(Optional) Click in the label to rename the Reflection.

The purpose of the name is to help you understand, when you read job reports, which Reflections the query optimizer considered and chose when planning queries.

In the columns of the table, follow these steps, which you don't have to do in any particular order:

Click in the **Display** column to include fields in or exclude them from your Reflection.

Click in the **Sort** column to select fields on which to sort the data in the Reflection. For guidance in selecting a field on which to sort, see Sort Reflections on High-Cardinality Fields.

Click in the **Partition** column to select fields on which to horizontally partition the rows in the Reflection. For guidance in selecting fields on which to partition, and which partition transforms to apply to those fields, see Horizontally Partition Reflections that Have Many Rows.

If the Reflection is based on an Iceberg table, a filesystem source, an AWS Glue source, or a Hive source, and that table is partitioned, partition columns and transforms are recommended for you. Hover over !This is the Recommendations icon. at the top of the table to see the recommendation. Click the icon to accept the recommendation.

note

Ignore the **Distribution** column. Selecting fields in it has no effect on the Reflection.

(Optional) Optimize the number of files used to store the Reflection. You can optimize for fast refreshes or for fast read performance by queries. Follow these steps:

a. Click the in the table in which you are defining the Reflection.

b. In the field **Reflection execution strategy**, select either of these options:

Select **Minimize Time Needed To Refresh** if you need the Reflection to be created as fast as possible. This option can result in the data for the Reflection being stored in many small files. This is the default option.

Select **Minimize Number Of Files** when you want to improve read performance of queries against the Reflection. With this option, there tend to be fewer seeks performed for a given query.

Click **Save** when you are finished.

Aggregation Reflections

Accelerate BI-style queries that involve aggregations (`GROUP BY` queries) by precomputing results (like `SUM`, `COUNT`, `AVG`, `GROUP BY`) across selected dimensions and measures. By precomputing expensive computations, they significantly improve query performance at runtime. These Reflections are ideal for analytical workloads with frequent aggregations on large datasets.

Create Default Aggregation Reflections

You can use the **Basic** view of the Reflections editor to create one aggregation Reflection that includes fields, from the anchor or supported anchor, that are recommended for use as dimensions or measures. You can add or remove dimensions and measures, too.

To create an aggregation Reflection in the **Basic** view of the Reflections editor:

In the Dremio console, click !This is the icon that represents the Datasets page. in the side navigation bar to go to the Datasets page.

In the catalog or folder in which the anchor is located, hover over the anchor name and click .

In the **Aggregations Reflections** section of the editor, click **Generate** to get recommended fields to use as dimensions and measures. This will override any previously selected dimensions and measures. If you wish to proceed, click **Continue** in the confirmation dialog that follows.

In the **Aggregation Reflection** section of the editor, modify or accept the recommended fields for dimensions and measures.

To make the Reflection available to the query optimizer after you create it, click the toggle switch on the left side of the **Aggregation Reflections** bar.

Click **Save**.

Restrictions

You can create only one aggregation Reflection in the **Basic** view. If you want to create multiple aggregations Reflections at a time, use the **Advanced** view.

You cannot select fields for sorting or horizontally partitioning.

The name of the Reflection is restricted to "Aggregation Reflection".

Create Customized Aggregation Reflections

You can use the **Advanced** view of the Reflections editor to create one or more

aggregation Reflections that select which fields in the anchor or supporting anchor to use as dimensions and measures. For each field that you use as a measure, you can use one or more of these SQL functions: `APPROX_DISTINCT_COUNT`, `COUNT`, `MAX`, and `MIN`. You can also choose sort fields and fields for partitioning horizontally.

Before you create customized aggregation Reflections, Dremio recommends that you follow the best practices listed in Operational Excellence when you create customized aggregation Reflections.

To create an aggregation Reflection in the **Advanced** view of the Reflections editor:

In the Dremio console, click !This is the icon that represents the Datasets page. in the side navigation bar to go to the Datasets page.

In the catalog or folder in which the anchor is located, hover over the anchor name and click .

Click the **Advanced View** button in the top-right corner of the editor.

Click **Aggregation Reflections**.

The Aggregation Reflections section is displayed, and one table for refining the aggregation Reflection that appeared in the **Basic** view is ready.

(Optional) Click in the name to rename the Reflection.

The purpose of the name is to help you understand, when you read job reports, which Reflections the query optimizer considered and chose when planning queries.

In the columns of the table, follow these steps, which you don't have to do in any particular order:

Click in the **Dimension** column to include or exclude fields to use as dimensions.

Click in the **Measure** column to include or exclude fields to use as measures. You can use one or more of these SQL functions for each measure: `APPROX_DISTINCT_COUNT`, `COUNT`, `MAX`, and `MIN`.

If you want to include a computed measure, first create a view with the computed column to use as a measure, and then create the aggregation Reflection on the view.

The full list of SQL aggregation functions that Dremio supports is not supported in the Reflections editor. If you want to create a Reflection that aggregates data by using the `AVG`, `CORR`, `HLL`, `SUM`, `VAR_POP`, or `VAR_SAMP` SQL functions, you must create a view that uses the function, and then create a raw Reflection from that view.

Click in the **Sort** column to select fields on which to sort the data in the Reflection. For guidance in selecting a field on which to sort, see Sort Reflections on High-Cardinality Fields.

Click in the **Partition** column to select fields on which to horizontally partition the rows in the Reflection. For guidance in selecting fields on which to partition, and which partition transforms to apply to those fields, see Horizontally Partition Reflections that Have Many Rows.

If the Reflection is based on an Iceberg table, a filesystem source, an AWS Glue source,

or a Hive source, and that table is partitioned, recommended partition columns and transforms are selected for you. If you change the selection of columns, then this icon appears at the top of the table: !This is the Recommendations icon.. You can click it to revert back to the recommended selection of partition columns.

note

Ignore the **Distribution** column. Selecting fields in it has no effect on the Reflection.

(Optional) Optimize the number of files used to store the Reflection. You can optimize for fast refreshes or for fast read performance by queries. Follow these steps:

a. Click the in the table in which you are defining the Reflection.

b. In the field **Reflection execution strategy**, select either of these options:

- Select **Minimize Time Needed To Refresh** if you need the Reflection to be created as fast as possible. This option can result in the data for the Reflection being stored in many small files. This is the default option.

- Select **Minimize Number Of Files** when you want to improve the read performance of queries against the Reflection. With this option, there tend to be fewer seeks performed for a given query.

- Click **Save** when you are finished.

Edit Aggregation Reflections

You might want to edit an aggregation Reflection if you are iteratively designing and testing an aggregation Reflection, if the definition of the view that the Reflection was created from was changed, if the schema of the underlying table was changed, or if you want to revise one or more aggregations defined in the Reflection.

If you created an aggregation Reflection in the **Basic** view of the Reflections editor, you can edit that Reflection either in the **Basic** view or in the **Advanced** view.

Dremio runs the job or jobs to recreate the Reflection after you click **Save**.

Use the Basic View

To edit an aggregation Reflection in the **Basic** view of the Reflections editor:

- In the Dremio console, hover over in the side navigation bar and select **Project settings**.

- Select **Reflections** in the project settings sidebar.

- Click the name of the Reflection. This opens the Acceleration dialog with the Reflections editor.

- In the Aggregation Reflection section of the editor, modify or accept the recommendation for **Dimension** and **Measure** columns.

- Click **Save**.

Use the Advanced View

To edit an aggregation Reflection in the **Advanced** view of the Reflections editor:

In the Dremio console, hover over in the side navigation bar and select **Project settings**.

Select **Reflections** in the project settings sidebar.

Click the name of the Reflection. This opens the Acceleration dialog with the Reflections editor.

Click the **Advanced View** button in the top-right corner of the editor.

Click **Aggregation Reflections**.

(Optional) Click in the name to rename the Reflection.

The purpose of the name is to help you understand, when you read job reports, which Reflections the query optimizer considered and chose when planning queries.

In the columns of the table, follow these steps, which you don't have to do in any particular order:

Click in the **Dimension** column to include or exclude fields to use as dimensions.

Click in the **Measure** column to include or exclude fields to use as measures. You can use one or more of these SQL functions for each measure: `APPROX_DISTINCT_COUNT`, `COUNT`, `MAX`, and `MIN`.

The full list of SQL aggregation functions that Dremio supports is not supported in the Reflections editor. If you want to create a Reflection that aggregates data by using the `AVG`, `CORR`, `HLL`, `SUM`, `VAR_POP`, or `VAR_SAMP` SQL functions, you must create a view that uses the function, and then create a raw Reflection from that view.

Click in the **Sort** column to select fields on which to sort the data in the Reflection. For guidance in selecting a field on which to sort, see Sort Reflections on High-Cardinality Fields.

Click in the **Partition** column to select fields on which to horizontally partition the rows in the Reflection. For guidance in selecting fields on which to partition, and which partition transforms to apply to those fields, see Horizontally Partition Reflections that Have Many Rows.

If the Reflection is based on an Iceberg table, a filesystem source, an AWS Glue source, or a Hive source, and that table is partitioned, partition columns and transforms are recommended for you. Hover over !This is the Recommendations icon. at the top of the table to see the recommendation. Click the icon to accept the recommendation.

note

Ignore the **Distribution** column. Selecting fields in it has no effect on the Reflection.

(Optional) Optimize the number of files used to store the Reflection. You can optimize for fast refreshes or for fast read performance by queries. Follow these steps:

a. Click the in the table in which you are defining the

Reflection.

b. In the field **Reflection execution strategy**, select either of these options:

- Select **Minimize Time Needed To Refresh** if you need the Reflection to be created as fast as possible. This option can result in the data for the Reflection being stored in many small files. This is the default option.

- Select **Minimize Number Of Files** when you want to improve the read performance of queries against the Reflection. With this option, there tend to be fewer seeks performed for a given query.

- Click **Save** when you are finished.

External Reflections

Reference precomputed tables in external data sources instead of materializing Reflections within Dremio, eliminating refresh overhead and storage costs. You can use an external Reflection by defining a view in Dremio that matches the precomputed table and mapping the view to the external data source. The data in the precomputed table is not refreshed by Dremio. When querying the view, Dremio's query planner leverages the external Reflection to generate optimal execution plans, improving query performance without additional storage consumption in Dremio.

Create External Reflections

To create an external Reflection:

- Follow these steps in the data source:

a. Select your source table.

b. Create a table that is derived from the source table, such as an aggregation table, if you do not already have one.

- Follow these steps in Dremio:

a. Define a view on the derived table in the data source. The definition must match that of the derived table.

b. Define a new external Reflection that maps the view to the derived table.

note

The data types and column names in the external Reflection must match those in the view that the external Reflection is mapped to.

Suppose you have a data source named `mySource` that is connected to Dremio. In that data source, there are (among all of your other tables) these two tables:

- `sales`, which is a very large table of sales data.

- `sales_by_region`, which aggregates by region the data that is in `sales`.

You want to make the data in `sales_by_region` available to data analysts who use Dremio. However, because you already have the `sales_by_region` table created, you

do not see the need to create a Dremio table from `sales`, then create a Dremio view that duplicates `sales_by_region`, and finally create a Reflection on the view. You would like instead to make `sales_by_region` available to queries run from BI tools through Dremio.

To do that, you follow these steps:

1. Create a view in Dremio that has the same definition as `sales_by_region`. Notice that the `FROM` clause points to the `sales` table that is in your data source, not to a Dremio table.

Example View

```
CREATE VIEW "myWorkspace"."sales_by_region" AS
SELECT
    AVG(sales_amount) average_sales,
    SUM(sales_amount) total_sales,
    COUNT(*) sales_count,
    region
FROM mySource.sales
GROUP BY region
```

2. Create an external Reflection that maps the view above to `sales_by_region` in `mySource`.

Example External Reflection

```
ALTER DATASET "myWorkspace"."sales_by_region"
CREATE EXTERNAL Reflection "external_sales_by_region"
USING "mySource"."sales_by_region"
```

The external Reflection lets Dremio's query planner know that there is a table in `mySource` that matches the Dremio view `myWorkplace.sales_by_region` and that can be used to satisfy queries against the view. When Dremio users query `myWorkspace.sales_by_region`, Dremio routes the query to the data source `mySource`, which runs the query against `mySource.sales_by_region`.

Edit External Reflections

If you have modified the DDL of a derived table in your data source, follow these steps in Dremio to update the corresponding external Reflection:

1. Replace the view with one that has a definition that matches the definition of the derived table. When you do so, the external Reflection is dropped.

2. Define a new external Reflection that maps the view to the derived table.

Test Reflections

You can test whether the Reflections that you created are used to satisfy a query without actually running the query. This practice can be helpful when the tables are very large and you want to avoid processing large queries unnecessarily.

To test whether one or more Reflections are used by a query:

In the Dremio console, click !The SQL Runner icon in the side navigation bar to open the SQL Runner.

In the SQL editor, type `EXPLAIN PLAN FOR` and then type or paste in your query.

Click **Run**.

When the query has finished, click the **Run** link found directly above the query results to view the job details. Any Reflections used will be shown on the page.

View Whether Queries Used Reflections

You can view the list of jobs on the Jobs page to find out whether queries were accelerated by Reflections. The Jobs page lists the jobs that ran queries, both queries from your data consumers and queries run within the Dremio user interface.

To find whether a query used a Reflection:

Find the job that ran the query by looking below the details in each row.

Look for !This is the icon that indicates a Reflection was used. next to the job to indicate that one or more Reflections were used.

View the job summary by clicking the row that represents the job that ran the query. The job summary appears in the pane to the right of the list of jobs.

Relationship between Reflections and Jobs

The relationship between a job and a Reflection can be one of the following types:

CONSIDERED – The Reflection is defined on a dataset that is used in the query but was determined not to cover the query (for example, the Reflection did not have a field that is used by the query).

MATCHED – A Reflection could have been used to accelerate the query, but Dremio determined that it would not provide any benefits or another Reflection was determined to be a better choice.

CHOSEN – A Reflection is used to accelerate the query. Note that multiple Reflections can be used to accelerate queries.

Disable Reflections

Disabled Reflections become unavailable for use by queries and will not be refreshed manually or according to their schedule.

note

Dremio does not disable external Reflections.

To disable a Reflection:

In the Dremio console, hover over in the side navigation

bar and select **Project Settings**.

Select **Reflections** in the project settings sidebar.

This opens the Reflections editor for the Reflection's anchor or supporting anchor.

Follow one of these steps:

If there is only one raw Reflection for the table or view, in the **Basic** view, click the toggle switch in the **Raw Reflections** bar.

If there are two or more raw Reflections for the table or view, in the **Advanced** view, click the toggle switch for the individual raw Reflection that you want to disable.

If there is only one aggregation Reflection for the table or view, in the **Basic** view, click the toggle switch in the **Raw Reflections** bar.

If there are two or more aggregation Reflections for the table or view, in the **Advanced** view, click the toggle switch for the individual aggregation Reflection that you want to disable.

Click **Save**. The changes take effect immediately.

Delete Reflections

You can delete Reflections individually, or all of the Reflections on a table or view. When you delete a Reflection, its definition, data, and metadata are entirely deleted.

To delete a single raw or aggregation Reflection:

In the Dremio console, hover over in the side navigation bar and select **Project settings**.

Select **Reflections** in the project settings sidebar.

This opens the Reflections editor for the Reflection's anchor or supporting anchor.

Open the **Advanced** view, if it is not already open.

If the Reflection is an aggregation Reflection, click **Aggregation Reflections**.

Click for the Reflection that you want to delete.

Click **Save**. The deletion takes effect immediately.

To delete all raw and aggregation Reflections on a table or view:

In the Dremio console, hover over in the side navigation bar and select **Project Settings**.

Select **Reflections** in the project settings sidebar.

This opens the Reflections editor for the Reflection's anchor or supporting anchor.

Click the in the top right corner of the Reflections page.

Click **Delete all reflections**.

Click **Save**.

To delete an external Reflection, or to delete a raw or aggregation Reflection without using the Reflections editor, run this SQL command:

Delete a Reflection

```
ALTER DATASET <DATASET_PATH> DROP Reflection <REFLECTION_NAME>
```

`DATASET_PATH`: The path of the view on which the external Reflection is based.

`REFLECTION_NAME`: The name of the external Reflection.

Related Topics

Data Reflections Deep Dive – Enroll in this Dremio University course to learn more about Reflections.

Operational Excellence - Follow best practices in Dremio's Well-Architected Framework for creating and managing Reflections.

Was this page helpful?

Reflection Recommendations

Raw Reflections

Prerequisites

Create Default Raw Reflections

Create Customized Raw Reflections

Edit Raw Reflections

Aggregation Reflections

Create Default Aggregation Reflections

Create Customized Aggregation Reflections

Edit Aggregation Reflections

External Reflections

Create External Reflections

Edit External Reflections

Test Reflections

View Whether Queries Used Reflections

Relationship between Reflections and Jobs

Disable Reflections

Delete Reflections

Related Topics

Bring Your Own Project Store | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/projects/your-own-project-storage

On this page

To enable secure access between Dremio and your AWS environment, you must create an AWS Identity and Access Management (IAM) role with specific permissions and a trust relationship that allows Dremio's AWS account to assume that role. The IAM policy and trust configuration are detailed bellow.

Create Your IAM Role

You will create an IAM Role in your AWS account that grants Dremio the permissions it needs to access your S3 bucket.

Attach the following policy to the role and replace `<bucket-name>` with the name of your own S3 bucket.

IAM Policy

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "s3:GetBucketLocation",
               "s3:ListAllMyBuckets"
           ],
           "Resource": "*"
       },
       {
           "Effect": "Allow",
           "Action": [
               "s3:PutObject",
               "s3:GetObject",
               "s3:ListBucket",
               "s3:DeleteObject"
           ],
           "Resource": [
               "arn:aws:s3:::<bucket-name>",
               "arn:aws:s3:::<bucket-name>/*"
           ]
       }
   ]
}
```

The first statement allows Dremio to find buckets in your account.

ListAllMyBuckets – Allow Dremio to discover your buckets when validating connectivity.

GetBucketLocation - Allow Dremio to discover your bucket's location.

The second statement allows Dremio to work with the data in your bucket.

PutObject / GetObject / DeleteObject – Allow Dremio to read, write, and delete data within the bucket.

ListBucket – Allow Dremio to enumerate objects in the bucket.

Define the Trust Relationship

The trust relationship determines which AWS account (in this case, Dremio's) is permitted to assume your IAM role.

Attach the following policy to the role.

Dremio's US trust account ID is `894535543691`.

Trust Relationship

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::894535543691:root"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

AssumeRole - Allows Dremio to assume the provided role.

TagSession - Allows Dremio to pass identifying tags during role assumption, enabling improved tracking and auditing across accounts.

Validate Role Configuration

In the AWS Console, navigate to **IAM → Roles → [Your Role Name]**.

Confirm that:

The permissions policy matches the example above.

- The trust relationship allows the Dremio AWS account as the trusted principal.
- Both `sts:AssumeRole` and `sts:TagSession` actions are present.
- If Dremio provided an AWS account ID or specific region endpoint, ensure these match your configuration.

Provide Role ARN to Dremio

Once your role is created and validated:

- Copy the Role ARN (e.g. `arn:aws:iam::<your-account-id>:role/<role-name>`).
- Provide this ARN to Dremio via the Create Project flow.

This allows Dremio to assume the role securely and begin reading/writing data to your S3 bucket.

(Optional) Enable PrivateLink Connectivity

To enhance security and keep data traffic within AWS's private network, Dremio supports integration via AWS PrivateLink with DNS-based endpoint resolution.

To enable:

- Ensure your AWS environment has PrivateLink endpoints configured for the required services.
- Verify that DNS resolution is enabled so that Dremio can route traffic to your private endpoints.
- Confirm connectivity by testing the endpoint using your VPC configuration.

Was this page helpful?

Autonomous Reflections | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/performance/autonomous-reflections/

On this page

Dremio automatically creates and manages Reflections based on query patterns to optimize performance for queries on Iceberg tables, UniForm tables, Parquet datasets,

and any views built on these datasets. With Autonomous Reflections, management and maintenance are fully automated, reducing manual effort and ensuring queries run efficiently. This eliminates the need for manual performance tuning while maintaining query correctness.

note

For data sources and formats not supported by Autonomous Reflections, you can create manual Reflections to optimize query performance.

What Is a Reflection?

A Reflection is a precomputed and optimized copy of a query result, designed to speed up query performance. It is derived from an existing table or view, known as its anchor.

Dremio's query optimizer uses Reflections to accelerate queries by avoiding the need to scan the original data. Instead of querying the raw source, Dremio automatically rewrites queries to use Reflections when they provide the necessary results, without requiring you to reference them directly.

When Dremio receives a query, it first determines whether any Reflections have at least one table in common with the tables and views referenced by the query. If any Reflections do, Dremio evaluates them to determine whether they satisfy the query. Then, if any Reflections do satisfy the query, Dremio generates a query plan that uses them.

Dremio then compares the cost of the plan to the cost of executing the query directly against the tables, and selects the plan with the lower cost. Finally, Dremio executes the selected query plan. Typically, plans that use one or more Reflections are less expensive than plans that run against raw data.

How Workloads Are Autonomously Accelerated

Dremio autonomously creates Reflections to accelerate queries on existing views, queries with joins written directly on base tables (not referencing any views), and queries that summarize data, typically submitted by AI Agents and BI dashboards.

Reflections are automatically generated based on query patterns without user intervention. Dremio continuously collects metadata from user queries, and the Autonomous Algorithm runs daily at midnight UTC to analyze recent query patterns from the last 7 days and create Autonomous Reflections that accelerate frequent and expensive queries.

Query Qualification

Only queries meeting the following criteria are considered:

- Based on Iceberg tables, Uniform table, Parquet datasets, or views built on them. Queries referencing non-Iceberg or non-Parquet datasets, either directly or via a view, are excluded.
- Execution time longer than one second.

Dremio may create system-managed views to anchor raw or aggregation Reflections that cannot be modified or referenced by users. Admins can drop these views, which also deletes the associated Reflection.

Reflection Limits

Dremio can create up to 100 Reflections total, with a maximum of 10 new Reflections created per day. The actual number depends on query patterns.

How Autonomous Reflections Are Maintained

Autonomous Reflections refresh automatically when source data changes:

Iceberg tables: Refreshed when the table is modified via Dremio (triggered immediately) or other engines (Dremio polls tables every 10 seconds to detect changes).

Uniform tables: Refreshed when the table is modified via Dremio (triggered immediately) or other engines (Dremio polls tables every 10 seconds to detect changes).

Parquet datasets: Refreshed when metadata updates occur in Dremio.

Refresh Engine: When a project is created, Dremio automatically provisions a Small internal refresh engine dedicated to executing Autonomous Reflection refresh jobs. This ensures Reflections are always accurate and up-to-date without manual refresh. The engine automatically shuts down after 30 seconds of idle time to optimize resource usage and costs.

Usage and Data Freshness

Dremio only uses Reflections in query plans when they refresh with the most recent data in tables on which they are based. If a Reflection is not yet refreshed, queries automatically fall back to the raw data source, ensuring query correctness is never compromised.

Monitor Reflections

To view Autonomous Reflections created for your project and their metadata (including status, score, footprint, and queries accelerated), see View Reflection Details.

To view the history of changes to Autonomous Reflections in the last 30 days:

Go to **Project Settings** > **Reflections**.

Click **History Log**.

Remove Reflections

Autonomous Reflections can be removed in two ways:

Automatic Removal – When a Autonomous Reflection's score falls below the

threshold, it is disabled for 7 days before being automatically dropped. Admins can view disabled Autonomous Reflections in the history log.

Manual Removal – Admins can manually drop Autonomous Reflections at any time. Autonomous Reflections cannot be modified by users. If an admin manually drops a Autonomous Reflection three times, Dremio will not recreate it for 90 days.

Disable Reflections

Every project created in Dremio is automatically accelerated with Autonomous Reflections. To disable Autonomous Reflections for a project:

Go to **Project Settings** > **Preferences**.

Toggle the **Autonomous Reflections** setting to off.

Related Topics

Data Product Fundamentals – Enroll in this Dremio University course to learn more about Autonomous Reflections.

Was this page helpful?

What Is a Reflection?

How Workloads Are Autonomously Accelerated

Query Qualification

Reflection Limits

How Autonomous Reflections Are Maintained

Usage and Data Freshness

Monitor Reflections

Remove Reflections

Disable Reflections

Related Topics

View Reflection Details | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/performance/manual-reflections/reflection-details

On this page

The Reflections page lists all raw and aggregation Reflections in Dremio.

To view this page, follow these steps:

-

In the Dremio console, hover over in the side navigation bar and select **Project Settings**.

Select **Reflections** in the project settings sidebar.

For any particular Reflection, the Reflections page presents information that answers these questions:

Question	Column with the answer
What is the status of this Reflection?	Name
Is this a raw or aggregation Reflection?	Type
Which table or view is this Reflection defined on?	Dataset
How valuable is this Reflection?	Reflection Score
How Reflection was created and managed?	Mode
How can I see a list of the jobs that created and refreshed this Reflection?	Refresh Job History
How many times has the query planner chosen this Reflection?	Acceleration Count
How many times has the query planner considered using this Reflection?	Considered Count
How many times did the query planner match a query to this Reflection?	Matched Count
How do I find out how effective this Reflection is?	Acceleration Count
When was this Reflection last refreshed?	Last Refresh From Table
Is this Reflection being refreshed now?	Refresh Status
What type of refreshes are used for this Reflection?	Refresh Method
Are refreshes scheduled for this Reflection, or do they need to be triggered manually?	Refresh Status
How much time did the most recent refresh of this Reflection take?	Last Refresh Duration
How many records are in this Reflection?	Record Count
How much storage is this Reflection taking up?	Current Footprint
When does this Reflection expire?	Available Until

Columns

Acceleration Count

Shows the number of times within the last 30 days that the query planner considered using a Reflection defined on a dataset referenced by a query, determined the Reflection could be used to satisfy the query, and chose to use the Reflection to satisfy the query.

If this count is low relative to the numbers in the **Considered Count** and **Matched Count**, the Reflection is not effective in reducing the execution times of queries on the dataset.

Available Until

Shows the date and time when this Reflection expires, based on the refresh policy of the queried dataset.

If a Reflection is set to expire soon and you want to continue using it, you can take either of these actions:

- Change the expiration setting on the table which the Reflection is either directly or indirectly defined on. A Reflection is indirectly defined on a table when it is defined on a view that is derived from that table. When you change the setting by using this method, the change goes into effect after the next refresh.

- Change the expiration setting on the data source where the table is located.

For the steps, see Set the Reflection Expiration Policy.

Mode

Shows how Reflection was created and managed.

- **autonomous**: Created and managed by Dremio

- **manual**: Created and managed by user

Considered Count

Shows the number of queries, within the last 30 days, that referenced the dataset that a Reflection is defined on. Whenever a query references a dataset on which a Reflection is defined, the query planner considers whether to use the Reflection to help satisfy the query.

If the query planner determines that the Reflection can do that (that the Reflection matches the query), the query planner compares the Reflection to any others that might also be defined on the same dataset.

If the query planner does not determine this, it ignores the Reflection.

Reflections with high considered counts and no match counts are contributing to high logical planning times. Consider deleting them.

Reflections with a considered count of 0 should be removed. They are merely taking up storage and, during refreshes, resources on compute engines.

Current Footprint

Shows the current size, in kilobytes, of a Reflection.

Dataset

Shows the name of the table or view that a Reflection is defined on.

Last Refresh Duration

Shows the length of time required for the most recent refresh of a Reflection.

Last Refresh From Table

Shows the date and time that the Reflection data was last refreshed. If the refresh is running, failing, or disabled, the value is `12/31/1969 23:59:59`.

Matched Count

Shows the number of times, within the last 30 days, that the query planner both considered a Reflection for satisfying a query and determined that the Reflection would in fact satisfy the query. However, the query planner might have decided to use a different Reflection that also matched the query. For example, a different query plan that did not include the Reflection might have had a lower cost.

This number does not show how many times the query planner used the Reflection to satisfy the query. For that number, see Acceleration Count.

If the matched count is high and the accelerating count is low, the query planner is more often deciding to use a different Reflection that also matches a query. In this case, consider deleting the Reflection.

Name

Shows the name of the Reflection and its status. The tooltip on the icon represents a combination of the status of the Reflection (which you can filter on through the values in the **Acceleration Status** field above the list) and the value in the **Refresh Status** column.

Record Count

Shows the number of records in the Reflection.

Reflection Score

Shows the score for a Reflection on a scale of 0 (worst) to 100 (best). The score indicates the value that the Reflection provides to your workloads based on the jobs that have been executed in the last 7 days. Reflection scores are calculated once each day. Factors considered in the score include the number of jobs accelerated by the Reflection and the expected improvement in query run times due to the Reflection.

To help you interpret the scores, the scores have the following labels:

Good: The score is more than 75.

Fair: The score is 25 to 75.

Poor: The score is less than 25.

New: The score is blank because the Reflection was created within the past 24 hours.

note

If a Reflection's score is listed as **-**, the score needs to be recalculated due to an error or

an upgraded instance.

Refresh Job History

Opens a list of all of the jobs that created and refreshed a Reflection.

Refresh Method

Shows which type of refresh was last used for a Reflection.

Full: All of the data in the Reflection was replaced. The new data is based on the current data in the underlying dataset.

Incremental:

- For Reflections defined on Apache Iceberg tables: Either snapshot-based incremental refresh was used (if the changes were appends only) or partition-based incremental refresh was used (if the changes included DML operations).
- For Reflections defined on Delta Lake tables: This value does not appear. Only full refreshes are supported for these Reflections.
- For Reflections defined on all other tables: Data added to the underlying dataset since the last refresh of the Reflection was appended to the existing data in the Reflection.

None: Incremental refreshes were selected in the settings for the table. However, Dremio has not confirmed that it is possible to refresh the Reflection incrementally. Applies only to Reflections that are not defined on Iceberg or Delta Lake tables.

For more information, see Refresh Reflections.

Refresh Status

Shows one of these values:

Manual: Refreshes are not run on a schedule, but must be triggered manually. See Trigger Reflection Refreshes.

Pending: If the Reflection depends on other Reflections, the refresh will begin after the refreshes of the other Reflections are finished.

Running: The Reflection is currently being refreshed.

Scheduled: Refreshes run on a schedule, but a refresh is not currently running.

Auto: All of the Reflection's underlying tables are in Iceberg format, and the Reflection automatically refreshes when new snapshots are created after an update to an underlying table, but a refresh is not currently running.

Failed: Multiple attempts to refresh a Reflection have failed. You must disable and enable the Reflection to rebuild it and continue using it. Reflections in this state will not be considered to accelerate queries.

For more information, see Refresh Reflections.

Total Footprint

Shows the current size, in kilobytes, of all of the existing materializations of the Reflection. More than one materialization of a Reflection can exist at the same time, so that refreshes do not interrupt running queries that are being satisfied by the Reflection.

Type

Shows whether the Reflection is a raw or aggregation Reflection.

Was this page helpful?

- Columns
- Acceleration Count
- Available Until
- Mode
- Considered Count
- Current Footprint
- Dataset
- Last Refresh Duration
- Last Refresh From Table
- Matched Count
- Name
- Record Count
- Reflection Score
- Refresh Job History
- Refresh Method
- Refresh Status
- Total Footprint
- Type

Refresh Reflections | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/admin/performance/manual-reflections/reflection-refresh

On this page

The data in a Reflection can become stale and may need to be refreshed. Refreshing a Reflection triggers two updates:

- The data stored in the Apache Iceberg table for the Reflection is updated.
- The metadata that stores details about the Reflection is updated.

note

Dremio does not refresh the data that external Reflections are mapped to.

Types of Reflection Refresh

How Reflections are refreshed depend on the format of the base table.

Apache Iceberg Tables, Filesystem Sources, AWS Glue Sources, and Hive Sources

There are two methods that can be used to refresh Reflections that are defined either on Iceberg tables or on these types of datasets in filesystem, AWS Glue, and Hive sources:

- Parquet datasets in Filesystem sources (on S3, Azure Storage, Google Cloud Storage, or HDFS)
- Parquet datasets, Avro datasets, or non-transactional ORC datasets on AWS Glue or Hive (Hive 2 or Hive 3) sources

Iceberg tables in all supported file-system sources (Amazon S3, Azure Storage, Google Cloud Storage, and HDFS) and non-file-system sources (AWS Glue, Hive, and Nessie) can be refreshed with either of these methods.

- Incremental refreshes
- Full refreshes

Incremental Refreshes

There are two types of incremental refreshes:

- Incremental refreshes when changes to an anchor table are only append operations
- Incremental refreshes when changes to an anchor table include non-append operations

note

- Whether an incremental refresh can be performed depends on the outcome of an algorithm.
- The initial refresh of a Reflection is always a full refresh.

Incremental Refreshes When Changes to an Anchor Table Are Only Append Operations

note

Optimize operations on Iceberg tables are also supported for this type of incremental refresh.

This type of incremental refresh is used only when the changes to the anchor table are appends and do not include updates or deletes. There are two cases to consider:

When a Reflection is defined on one anchor table

When a Reflection is defined on an anchor table or on a view that is defined on one anchor table, an incremental refresh is based on the differences between the current snapshot of the anchor table and the snapshot at the time of the last refresh.

When a Reflection is defined on a view that joins two or more anchor tables

When a Reflection is defined on a view that joins two or more anchor tables, whether an incremental refresh can be performed depends on how many anchor tables have changed since the last refresh of the Reflection:

If just one of the anchor tables has changed since the last refresh, an incremental refresh can be performed. It is based on the differences between the current snapshot of the one changed anchor table and the snapshot at the time of the last refresh.

If two or more tables have been refreshed since the last refresh, then a full refresh is used to refresh the Reflection.

Incremental Refreshes When Changes to an Anchor Table Include Non-append Operations

For Iceberg tables, this type of incremental refresh is used when the changes are DML operations that delete or modify the data (UPDATE, DELETE, etc.) made either through the Copy-on-Write (COW) or the Merge-on-Read (MOR) storage mechanism. For more information about COW and MOR, see Row-Level Changes on the Lakehouse: Copy-On-Write vs. Merge-On-Read in Apache Iceberg.

For sources in filesystems or AWS Glue, non-append operations can include, for example:

In filesystem sources, files being deleted from Parquet datasets

In AWS Glue sources, DML-equivalent operations being performed on Parquet datasets, Avro datasets, or non-transactional ORC datasets

Both the anchor table and the Reflection must be partitioned, and the partition transforms that they use must be compatible.

There are two cases to consider:

When a Reflection is defined on one anchor table

When a Reflection is defined on an anchor table or on a view that is defined on one anchor table, an incremental refresh is based on Iceberg metadata that is used to identify modified partitions and to restrict the scope of the refresh to only those partitions.

-

When a Reflection is defined on a view that joins two or more anchor tables

When a Reflection is defined on a view that joins two or more anchor tables, whether an incremental refresh can be performed depends on how many anchor tables have changed since the last refresh of the Reflection:

If just one of the anchor tables has changed since the last refresh, an incremental refresh can be performed. It is based on Iceberg metadata that is used to identify modified partitions and to restrict the scope of the refresh to only those partitions.

If two or more tables have been refreshed since the last refresh, then a full refresh is used to refresh the Reflection.

note

Dremio uses Iceberg tables to store metadata for filesystem and AWS Glue sources.

For information about partitioning Reflections and applying partition transforms, see the section Horizontally Partition Reflections that Have Many Rows.

For information about partitioning Reflections in ways that are compatible with the partitioning of anchor tables, see Partition Reflections to Allow for Partition-Based Incremental Refreshes.

Full Refreshes

In a full refresh, a Reflection is dropped, recreated, and loaded.

note

Whether a full refresh is performed depends on the outcome of an algorithm.

The initial refresh of a Reflection is always a full refresh.

Algorithm for Determining Whether an Incremental or a Full Refresh Is Used

The following algorithm determines which refresh method is used:

If the Reflection has never been refreshed, then a full refresh is performed.

If the Reflection is created from a view that uses nested group-bys, unions, window functions, or joins other than inner or cross joins, then a full refresh is performed.

If the Reflection is created from a view that joins two or more anchor tables and more than one anchor table has changed since the previous refresh, then a full refresh is performed.

If the Reflection is based on a view and the changed anchor table is used multiple times in that view, then a full refresh is performed.

If the changes to the anchor table are only appends, then an incremental refresh based on table snapshots is performed.

If the changes to the anchor table include non-append operations, then the compatibility of the partitions of the anchor table and the partitions of the Reflection is

checked:

If the partitions of the anchor table and the partitions of the Reflection are not compatible, or if either the anchor table or the Reflection is not partitioned, then a full refresh is performed.

If the partition scheme of the anchor table has been changed since the last refresh to be incompatible with the partitioning scheme of a Reflection, and if changes have occurred to data belonging to a prior partition scheme or the new partition scheme, then a full refresh is performed.
To avoid a full refresh when these two conditions hold, update the partition scheme for Reflection to match the partition scheme for the table. You do so in the **Advanced** view of the Reflection editor or through the `ALTER DATASET` SQL command.

If the partitions of the anchor table and the partitions of the Reflection are compatible, then an incremental refresh is performed.

Because this algorithm is used to determine which type of refresh to perform, you do not select a type of refresh for Reflections in the settings of the anchor table.

However, no data is read in the `REFRESH REFLECTION` job for Reflections that are dependent only on Iceberg, Parquet, Avro, non-transactional ORC datasets, or other Reflections and that have no new data since the last refresh based on the table snapshots. Instead, a "no-op" Reflection refresh is planned and a materialization is created, eliminating redundancy and minimizing the cost of a full or incremental Reflection refresh.

Delta Lake tables

Only full refreshes are supported. In a full refresh, the Reflection being refreshed is dropped, recreated, and loaded.

All Other Tables

Incremental refreshes

Dremio appends data to the existing data for a Reflection. Incremental refreshes are faster than full refreshes for large Reflections, and are appropriate for Reflections that are defined on tables that are not partitioned.

There are two ways in which Dremio can identify new records:

For directory datasets in file-based data sources like S3 and HDFS:
Dremio can automatically identify new files in the directory that were added after the prior refresh.

For all other datasets (such as datasets in relational or NoSQL databases):
An administrator specifies a strictly monotonically increasing field, such as an auto-incrementing key, that must be of type BigInt, Int, Timestamp, Date, Varchar, Float, Double, or Decimal. This allows Dremio to find and fetch the records that have been created since the last time the acceleration was incrementally refreshed.

caution

Use incremental refreshes only for Reflections that are based on tables and views that

are appended to. If records can be updated or deleted in a table or view, use full refreshes for the Reflections that are based on that table or view.

Full refreshes

In a full refresh, the Reflection being refreshed is dropped, recreated, and loaded.

Full refreshes are always used in these three cases:

- A Reflection is partitioned on one or more fields.
- A Reflection is created on a table that was promoted from a file, rather than from a folder, or is created on a view that is based on such a table.
- A Reflection is created from a view that uses nested group-bys, joins, unions, or window functions.

Specify the Reflection Refresh Policy

In the settings for a data source, you specify the refresh policy for refreshes of all Reflections that are on the tables in that data source. The default policy is period-based, with one hour between each refresh. If you select a schedule policy, the default is every day at 8:00 a.m. (UTC).

In the settings for a table that is not in the Iceberg or Delta Lake format, you can specify the type of refresh to use for all Reflections that are ultimately derived from the table. The default refresh type is **Full refresh**.

For tables in all supported table formats, you can specify a refresh policy for Reflection refreshes that overrides the policy specified in the settings for the table's data source. The default policy is the schedule set at the source of the table.

To set the refresh policy on a data source:

- In the Dremio console, right-click a data lake or external source.
- Select **Edit Details**.
- In the sidebar of the Edit Source window, select **Reflection Refresh**.
- When you are done making your selections, click **Save**. Your changes go into effect immediately.

To edit the refresh policy on a table:

- Locate the table.
- Hover over the row in which it appears and click !The Settings icon to the right.
- Select **Reflection Refresh** in the dataset settings sidebar.
- When you are done making your selections, click **Save**. Your changes go into effect immediately.

Types of Refresh Policies

Datasets and sources can set Reflections to refresh according to the following policy types:

Refresh policy type	Description
Never	Reflections are not refreshed.
Period (default)	Reflections refresh at the specified number of hours, days, or weeks. The default refresh period is one hour.
Schedule	Reflections refresh at a specific time on the specified days of the week, in UTC. The default is every day at 8:00 a.m. (UTC).
Auto refresh when Iceberg table data changes	Reflections automatically refresh for underlying Iceberg tables whenever new updates occur. Reflections under this policy type are known as Live Reflections. Live Reflections are also updated based on the minimum refresh frequency defined by the source-level policy. This refresh policy is only available for data sources that support the Iceberg table format.

Set the Reflection Expiration Policy

Rather than delete a Reflection manually, you can specify how long you want Dremio to retain the Reflection before deleting it automatically.

note

Dremio does not allow expiration policies to be set on external Reflections or Reflections that automatically refresh when Iceberg data changes according to the refresh policy.

To set the expiration policy for all Reflections derived from tables in a data source:

1. Right-click a data lake or external source.
2. Select **Edit Details**.
3. Select **Reflection Refresh** in the edit source sidebar.
4. After making your changes, click **Save**. The changes take effect on the next refresh.

To set the expiration policy on Reflections derived from a particular table:

note

The table must be based on more than one file.

1. Locate a table.
2. Click the !The Settings icon to its right.
3. Select **Reflection Refresh** in the dataset settings sidebar.
4. After making your changes, click **Save**. The changes take effect on the next refresh.

View the Reflection Refresh History

You can find out whether a refresh job for a Reflection has run, and how many times refresh jobs for a Reflection have been run.

To view the refresh history:

- In the Dremio console, go to the catalog or folder that lists the table or view from which the Reflection was created.
- Hover over the row for the table or view.
- In the **Actions** field, click !The Settings icon.
- Select **Reflections** in the dataset settings sidebar.
- Click **History** in the heading for the Reflection.

The Jobs page is opened with the ID of the Reflection in the search box, and only jobs related to that ID are listed.

When a Reflection is refreshed, Dremio runs a single job with two steps:

- The first step writes the query results as a materialization to the distributed acceleration storage by running a `REFRESH REFLECTION` command.
- The second step registers the materialization table and its metadata with the catalog so that the query optimizer can find the Reflection's definition and structure.

The following screenshot shows the `REFRESH REFLECTION` command used to refresh the Reflection named `Super-duper reflection`:

!Reflection refresh job listed on the Jobs page in the Dremio console

The Reflection refresh is listed as a single job on the Jobs page, as shown in the example below:

!Reflection refresh job listed on the Jobs page in the Dremio console

To find out which type of refresh was performed:

- Click the ID of the job that ran the `REFRESH REFLECTION` command.
- Click the **Raw Profile** tab.
- Click the **Planning** tab.
- Scroll down to the **Refresh Decision** section.

Retry a Reflection Refresh Policy

When a Reflection refresh job fails, Dremio retries the refresh according to a uniform policy. This policy is designed to balance resource consumption with the need to keep Reflection data up to date. It prioritizes newly failed Reflections to reduce excessive retries on persistent failures and helps ensure that Reflection data does not become overly stale.

After a refresh failure, Dremio's default is to repeat the refresh attempt at exponential intervals up to 4 hours: 1 minute, 2 minutes, 5 minutes, 15 minutes, 30 minutes, 1 hour, 2 hours, and 4 hours. Then, Dremio continues trying to refresh the Reflection every 4 hours.

There are two optimizations for special cases:

Long-running refresh jobs: The backoff interval will never be shorter than the last successful duration.

Small maximum retry attempts: At least one 4-hour backoff attempt is guaranteed to ensure meaningful coverage of the retry policy.

Dremio stops retrying after 24 attempts, which typically takes about 71 hours and 52 minutes, or when the 72-hour retry window is reached, whichever comes first.

To configure a different maximum number of retry attempts for Reflection refreshes than Dremio's default of 24 retries:

- Click !The Settings icon in the left navbar.
- Select **Reflections** in the left sidebar.
- On the Reflections page, click !The Settings icon in the top-right corner and select **Acceleration Settings**.
- In the field next to **Maximum attempts for Reflection job failures**, specify the maximum number of retries.
- Click **Save**. The change goes into effect immediately.

Dremio applies the retry policy after a refresh failure for all types of Reflection refreshes, no matter whether the refresh was triggered or set by a refresh policy.

Trigger Reflection Refreshes

You can click a button to start the refresh of all of the Reflections that are defined on a table or on views derived from that table.

To trigger a refresh manually:

- Locate the table.
- Hover over the row in which it appears and click !The Settings icon to the right.
- In the sidebar of the Dataset Settings window, click **Reflection Refresh**.
- Click **Refresh Now**. The message "All dependent Reflections will be refreshed." appears at the top of the screen.
- Click **Save**.

You can refresh Reflections by using the Reflection API, the Catalog API, and the SQL commands `ALTER TABLE` and `ALTER VIEW`.

- With the Reflection API, you specify the ID of a Reflection. See Refresh a Reflection.
- With the Catalog API, you specify the ID of a table or view that the Reflections are defined on. See Refresh the Reflections on a Table and Refresh the Reflections on a View.
- With the `ALTER TABLE` and `ALTER VIEW` commands, you specify the path and name

of the table or view that the Reflections are defined on.

The refresh action follows this logic for the Reflection API:

- If the Reflection is defined on a view, the action refreshes all Reflections that are defined on the tables and on downstream/dependent views that the anchor view is itself defined on.
- If the Reflection is defined on a table, the action refreshes the Reflections that are defined on the table and all Reflections that are defined on the downstream/dependent views of the anchor table.

The refresh action follows similar logic for the Catalog API and the SQL commands:

- If the action is started on a view, it refreshes all Reflections that are defined on the tables and on downstream/dependent views that the view is itself defined on.
- If the action is started on a table, it refreshes the Reflections that are defined on the table and all Reflections that are defined on the downstream/dependent views of the anchor table.

For example, suppose that you had the following tables and views, with Reflections R1 through R5 defined on them:

```
         View2(R5)
         /       \
     View1(R3) Table3(R4)
    /       \
Table1(R1) Table2(R2)
```

- Refreshing Reflection R5 through the API also refreshes R1, R2, R3, and R4.
- Refreshing Reflection R4 through the API also refreshes R5.
- Refreshing Reflection R3 through the API also refreshes R1, R2, and R5.
- Refreshing Reflection R2 through the API also refreshes R3 and R5.
- Refreshing Reflection R1 through the API also refreshes R3 and R5.

Obtain Reflection IDs

You will need one or more Reflection IDs for some of the Reflection hints. Reflection IDs can be found in three places: the Acceleration section of the raw profile of the job that ran a query using the Reflection, the `SYS.PROJECT.REFLECTIONS` system table, and the Reflection summary objects that you retrieve with the Reflection API.

To find the ID of a Reflection in Acceleration section of the raw profile of job that ran a query that used the Reflection:

- In the Dremio console, click !The Jobs icon in the side navigation bar.
- In the list of jobs, locate the job that ran the query. If the query was satisfied by a Reflection, !This is the icon that indicates a Reflection was used. appears after the name of the user who ran the query.

Click the ID of the job.

Click **Raw Profile** at the top of the page.

Click the **Acceleration** tab.

In the Reflection Outcome section, locate the ID of the Reflection.

To find the ID of a Reflection in the `SYS.PROJECT.REFLECTIONS` system table:

In the Dremio console, click !The SQL Runner icon in the left navbar.

Copy this query and paste it into the SQL editor:

Query for listing info about all existing Reflections

```
SELECT * FROM SYS.PROJECT.REFLECTIONS
```

Sort the results on the `dataset_name` column.

In the `dataset_name` column, locate the name of the dataset that the Reflection was defined on.

Scroll the table to the right to look through the display columns, dimensions, measures, sort columns, and partition columns to find the combination of attributes that define the Reflection.

Scroll the table all the way to the left to find the ID of the Reflection.

To find the ID of a Reflection by using REST APIs:

Obtain the ID of the table or view that the Reflection was defined on by using retrieving either the table or view by its path.

Use the Reflections API to retrieve a list of all of the Reflections that are defined on the table or view.

In the response, locate the Reflection by its combination of attributes.

Copy the Reflection's ID.

Was this page helpful?

Types of Reflection Refresh

Apache Iceberg Tables, Filesystem Sources, AWS Glue Sources, and Hive Sources

Delta Lake tables

All Other Tables

Specify the Reflection Refresh Policy

Types of Refresh Policies

Set the Reflection Expiration Policy

View the Reflection Refresh History

- Retry a Reflection Refresh Policy
- Trigger Reflection Refreshes
- Obtain Reflection IDs

Source: dremio-cloud-bring-data.md

Bring Your Data | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/

On this page

Dremio enables you to connect to, load, and prepare data from a wide variety of sources to enable unified data access. Whether your data resides in catalogs, object storage, or databases, Dremio provides the tools you need to bring your data into the lakehouse and prepare it for analytics.

Load Data into Tables

Load data from CSV, JSON, or Parquet files into Apache Iceberg tables for faster, more efficient queries. Dremio supports several methods for loading data:

- **Copy Data Into Tables** – Run the `COPY INTO` command to execute a one-time data load from files into existing Iceberg tables.
- **Autoingest Data** – Set up pipe that keep your tables up to date by automatically loading new files into Iceberg tables as they arrive in Amazon S3.
- **Upload Local Files** – Upload local files to create Iceberg tables in an Open Catalog for experimentation.

Connect to Your Data

Connect Dremio to your existing data sources to query data in place without moving it. Dremio supports a wide range of data sources including catalogs, object storage, and databases.

Prepare Your Data

Transform and enhance your data to make it ready for analytics. Dremio provides tools for data preparation including:

- **Views** – Create virtual tables based on SQL queries to transform, join, and aggregate data without moving or duplicating it.
- **Data Transformations** – Apply column aliasing, type casting, data cleansing, and create derived columns.
- **AI Functions** – Use AI-powered functions to enhance and analyze your data.

Was this page helpful?

Load Data into Tables

Connect to Your Data

Prepare Your Data

Load Data into Tables | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/load

On this page

Dremio supports three main methods for loading data into Apache Iceberg tables. You can copy data from CSV, JSON, or Parquet files directly into an existing table, set up autoingest pipes to continuously ingest new data as it arrives, or upload local files as Iceberg tables in a catalog.

Copy Data into Tables

Querying large datasets stored in CSV or JSON files can be inefficient. For faster performance and scalability, load your data into Apache Iceberg tables, which use the columnar Parquet format for optimized storage and retrieval. Even queries on Parquet files perform faster when the data is stored in Iceberg tables, enabling you to take full advantage of Dremio’s Iceberg capabilities.

Prerequisites

At least one column in the target table must match a column in every data file.

Files cannot contain duplicate column names.

CSV data files must have a header line at the start of the file.

Supported storage locations: Azure Storage or Amazon S3.

Copy Operation

Use the `COPY INTO` SQL command to load data from CSV, JSON, and Parquet files into existing Iceberg tables. The operation matches columns in the files to columns in the target table and loads data accordingly.

The copy operation supports Iceberg tables in the Open Catalog, AWS Glue Data Catalog, and catalogs that implement the Iceberg REST Catalog specification.

The copy operation verifies that at least one column in the target table matches a column represented in the data files. It then follows these rules:

If a match is found, the values in the data files are loaded into the column or columns.

If additional non-matching columns are present in the data files, the values in these columns are not loaded.

If additional non-matching columns are present in the target table, the operation inserts NULL values into these columns.

If no column in the target table matches any column represented in the data files, the operation fails.

The copy operation ignores case when comparing column names.

Type Coercion

For a list of the type coercions used by the copy operation when copying data from CSV and JSON files, see Type Coercion When Copying Data from CSV or JSON Files Into Apache Iceberg Tables.

For the type coercions used by the copy operation when copying data from Parquet files, refer to this table:
!Supported and Unsupported Coercions for File-formatted Sources

Column Nullability Constraints

A column's nullability constraint defines whether the column can contain `NULL` values, because you can specify that each column is either:

˜NULL` — Allows `NULL` values, which is useful for optional or unknown data.

˜NOT NULL` — Requires a value for every row; `NULL` values are not allowed.

When running `COPY INTO` with `ON_ERROR` set to 'continue' or 'skip_file', the command will not fail on nullability conflicts. Instead, it skips the problematic file or record.

However, if `ON_ERROR` is set to 'abort' (or left unspecified), the command will fail if any row violates the table’s `NOT NULL` constraints.

Autoingest Data Preview

You can use autoingest pipes to automatically ingest data into Apache Iceberg tables.

Autoingest pipes are objects in Dremio that represent event-driven ingestion pipelines, which collect and load data into a centralized data repository for further analysis and utilization. Event-driven ingestion, or autoingestion, occurs when a new file is added to a specified cloud storage location, which sets off an event in Dremio to copy the new file into an Iceberg table. Dremio automatically ingests the file with micro-batch processing, loading files in small, predefined batches at regular intervals.

Autoingest pipes remove the complexity and operational overhead of setting up, running, and monitoring data pipelines by providing:

˜**Single-Command Setup**: Dremio provides a streamlined process for setting up and running autoingest pipes. Create an autoingest pipe using the `CREATE PIPE` SQL command to specify the parameters, and run a cloud-specific CLI command to set up the required infrastructure and connect your cloud storage to Dremio for autoingestion.

-

File Deduplication: File deduplication eliminates copies of files and enhances storage utilization. Dremio's autoingest pipes provide file deduplication by ensuring that your pipe loads semantics only once and preventing files with the same name from loading in a given time period (the `DEDUPE_LOOKBACK_PERIOD`).

Event-Based Execution of `COPY INTO`: After a new file is added in the source location, an event is sent to Dremio to run a `COPY INTO` statement. Ingested files are processed in micro-batches to optimize user resources and ensure that the data is efficiently loaded into Dremio.

Execution Monitoring and Error Handling: For common issues that can occur with data pipelines, such as single rows that do not conform to the target table schema or read permissions being revoked on the source, Dremio takes the appropriate action to alert the user and suggest potential solutions.

Efficient Batching for Optimal Engine Utilization: When implementing an event-based loading system, users often execute a load command for every file immediately after the file is added to cloud storage. This frequent loading increases the likelihood that Iceberg file sizes will be smaller than the optimal size and the engine will be overutilized. Both of these issues result in higher total cost of ownership because they require running `OPTIMIZE TABLE` jobs more frequently to compact Iceberg tables, which uses engine resources inefficiently. Dremio's autoingest pipes efficiently organize requests into micro-batches that minimize the cost of loading data and maintain acceptable latency in a data lifecycle.

note

Autoingest pipes can only ingest data from Amazon S3 sources in Dremio.

Upload Local Files

You can upload an individual local file to Dremio if the file is 500 MB or smaller and in CSV, JSON, or Parquet format. During the upload process, Dremio formats the file into an Iceberg table.

To upload a file:

In the Dremio console, click !This is the icon that represents the Datasets page. in the side navigation bar to go to the Datasets page.

Click **Add Data** in the bottom left corner of the Datasets page.

Upload the file either by:

a. Dragging the file from your local machine and dropping it in the Add Data dialog.

b. Clicking **Choose file to upload** and navigating to the file on your local machine.

If the file is large, it may take a few moments to upload, depending on your connection speed.

(Optional) During the upload process, configure the file settings. For example, configure how the file is delimited.

Click **Save**.

Limits

- Uploaded files are copies of your local file. Updates to your local file are not automatically reflected in Dremio.
- Bulk upload of multiple files is not supported.

Case Sensitivity

Dremio does not support case-sensitive data file names, table names, or column names.

For example, if you have three file names that have the same name, but with different cases (such as, `MARKET`, `Market`, and `market`), Dremio is unable to discern the case differences, resulting in unanticipated data results.

For column names, if two columns have the same name using different cases (such as `Trip_Pickup_DateTime` and `trip_pickup_datetime`) exist in the table, one of the columns may disappear when the header is extracted.

Was this page helpful?

Prepare Your Data | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/prepare

On this page

You can bring curated data into Dremio by:

- Landing it in Dremio's Open Catalog as Iceberg tables using your tool of choice, such as dbt, Fivetran, Confluent, or others.
- Connecting your existing data sources to Dremio

Once this data is in Dremio, you can further cleanse, combine, transform, and aggregate your data using SQL functions in Dremio and create virtual representations of your data in the form of views.

Unified Data Access

You can query and combine data across multiple sources and formats in Dremio. Dremio's Query Engine can federate queries across sources in real time without requiring ETL. This allows you to pull together Iceberg tables from Dremio's Open Catalog with existing data in other catalogs, object stores, and databases. For more information on supported data sources, see Connect to Your Data. You can then create views from the SQL that queries across your sources into your Open Catalog.

Process Unstructured Data

Dremio allows you to process and combine structured, semi-structured, and unstructured data. Examples of unstructured data include PDFs, images, and videos that are stored in object storage.

Dremio natively offers AI functions to enable you to extract and process unstructured data:

- `AI_GENERATE` – Process unstructured data, primarily for complex data extraction requiring multiple fields from source files.
- `AI_CLASSIFY` – Categorize documents or analyze sentiment as `VARCHAR` values, using a provided classification list.
- `AI_COMPLETE` – Generate text or create summaries as `VARCHAR` values.

These functions are processed using Dremio's Query Engine and the AI model provider of your choice. For more information on how to configure your model provider, see Configure Model Providers.

Create a View

To create a virtual representation of your transformed and aggregated data, you can create a view by following these steps in the Dremio console:

1. Click !The SQL Runner icon in the side navigation bar to open the SQL Runner.
2. Write a SQL query that transforms your data and click **Run** to validate the query.
3. After the query has finished running, click the arrow next to **Save as View** in the top right of the SQL editor, and select **Save as View...** from the dropdown menu.
4. In the Save View As dialog, name the new view and select from a list of folders where it will be stored.

You can also run the `CREATE VIEW` from the SQL Runner or your tool of choice to achieve the same results.

Related Topics

- Explore and Analyze – Run queries on prepared data.
- Manage and Govern Data – Organize and secure your data in the AI Semantic Layer.

Data Privacy – Learn more about Dremio's data privacy practices.

Was this page helpful?

Unified Data Access

Process Unstructured Data

Create a View

Related Topics

Connect to Your Data | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/

On this page

You can connect to and access data from a variety of sources, including lakehouse catalogs, object storage, and databases. These sources are supported across Dremio projects, providing a unified foundation for data exploration and analytics.

Catalogs

Every Dremio project includes a native Open Catalog, powered by Apache Polaris. You can connect additional Open Catalogs from other projects and the following external catalogs:

AWS Glue Data Catalog

Snowflake Open Catalog

Unity Catalog

Object Storage

Dremio supports Amazon S3 and Azure Storage as object storage services.

Databases

You can run queries directly on the data in databases, which are referred to as external sources. In addition, you can run external queries:

That use the native syntax of the relational database.

To process SQL statements that are not supported by Dremio or are too complex to convert.

Network Connectivity

When connecting Dremio to data sources, the connection must use public networking. Ensure that the data source endpoint is publicly accessible and that network egress

rules, firewall settings, and IAM permissions are properly configured to allow outbound connectivity from Dremio to the source.

Was this page helpful?

- Catalogs
- Object Storage
- Databases
- Network Connectivity

Dremio | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/dremio

On this page

Connect your project to one or more Dremio Software clusters to create a federated data architecture that combines the best of both environments.

This configuration enables:

- **Reduced query latency** – Queries or portions of queries that utilize tables on the Software cluster are pushed down to maximize performance and reduce latency compared to transporting large raw tables.
- **Cross-cluster data federation** – Join data across multiple Dremio Software clusters and expose unified views through Dremio.
- **Enhanced security and data isolation** – Expose only a single Dremio port to the cloud instead of opening multiple source connections from your data center. Administrators of the Software cluster control what data is visible to the managing Dremio environment, allowing isolation of highly sensitive data on the Software cluster while exposing only aggregations or derived datasets to the managing project.
- **Simplified data access** – Access all data sources connected to Software clusters as schemas within Dremio without managing individual source connections.
- **Centralized semantic layer** – Build views and virtual datasets on top of federated clusters for consistent business logic across your organization.

When you connect a Dremio Software cluster as a data source, all sources on the Software cluster can be available from your project. You can create Reflections, build views, and query across the federated environment just as you would with any connected source.

Example Configuration

When you add a Dremio Software cluster as a source to your Dremio project:

- The Software cluster appears under Sources > Databases in your project.

- Data sources connected to the Software cluster appear as folders/schemas.
- You can promote tables from both directly connected sources and federated sources.
- Create views and Reflections on any promoted tables, regardless of source type.
- Query and join data across all sources—direct and federated.

Deployment Considerations

If your Dremio project and the source Dremio Software cluster are in different cloud regions or cloud vendors, your deployment design may be influenced by network latency and egress costs.

Network Latency

Cross-region or cross-cloud queries can experience increased latency. To minimize impact:

Use Reflections – Create Reflections in your Dremio project of frequently queried data from the Software cluster. Queries use the Reflections instead of fetching data across regions.

- **Push down filters and aggregations** – Write queries that leverage Dremio's query pushdown to perform filtering and aggregation on the Software cluster before returning results.
- **Colocate when possible** – If latency is critical, deploy the Software cluster in the same region as your Dremio organization.

Cloud Egress Costs

Data transfer between cloud regions or cloud vendors can incur significant egress charges. To control costs:

- **Create Reflections for frequently used data** – Reflection data is stored in your Dremio region, eliminating repeated egress charges for frequently accessed datasets.

Use aggregated views – Expose only aggregated or summarized data from the Software cluster rather than raw tables, reducing data transfer volume.

Limit full table scans – Ensure queries include appropriate filters to minimize the amount of data transferred across regions.

Monitor query patterns – Use Dremio's query history to identify expensive cross-region queries and optimize them with Reflections.

Security

Configure full TLS wire encryption on Software clusters to protect data in transit across regions and cloud boundaries.

User Impersonation

When you connect your project to a Dremio Software cluster, you provide the username and password of an account on the cluster. By default, queries that run from the project against the Dremio Software cluster run under the username of that account.

Alternatively, you can utilize user impersonation, which allows users running queries from your project to run them under their own usernames on the Dremio Software cluster. Users in your project must have accounts on the Dremio Software cluster, and the usernames must match. User impersonation (also known as *Inbound Impersonation*) must be set up on the Dremio Software cluster. The policy for user impersonation would look like this:

Example policy

```
ALTER SYSTEM SET "exec.impersonation.inbound_policies"='[
   {
      "proxy_principals":{
         "users":[
            "User_1"
         ]
      },
      "target_principals":{
         "users":[
            "User_1"
         ]
      }
   }
]'
```

Prerequisites

You must have a username and password for the account on the Dremio Software cluster to use for connections from your project.

Configure a Dremio Software Cluster as a Source

In the bottom-left corner of the Datasets page, click **Add Data**.

Under **Databases** in the Add Data Source dialog, select **Dremio**.

General Options

In the **Name** field, specify the name by which you want the data-source cluster to appear in the **Databases** section. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.

Under **Connection**, specify how you want to connect to the data-source cluster:

Direct: Connect directly to a coordinator node of the cluster.

-

ZooKeeper: Connect to an external ZooKeeper instance that is coordinating the nodes of the cluster.

In the **Host** and **Port** fields, specify the hostname or IP address and the port number of the coordinator node or ZooKeeper instance.

If the data-source cluster is configured to use TLS for connections to it, select the **Use SSL** option.

Under **Authentication**, specify the username and password for the project to use when connecting to the data-source cluster.

Advanced Options

On the Advanced Options page, you can set values for these optional parameters:

Maximum Idle Connections – The total number of connections allowed to be idle at a given time. The default is 8.

Connection Idle Time – The amount of time (in seconds) allowed for a connection to remain idle before the connection is terminated. The default is 60 seconds.

Query Timeout – The amount of time (in seconds) allowed to wait for the results of a query. If this time expires, the connection being used is returned to an idle state.

User Impersonation – Allows users to run queries on the data-source cluster under their own user IDs, not the user ID for the account used to authenticate with the data-source cluster. Inbound impersonation must be configured on the data-source cluster.

Reflection Refresh Options

On the Reflection Refresh page, set the policy that controls how often Reflections are scheduled to be refreshed automatically, as well as the time limit after which Reflections expire and are removed.

Never refresh – Select to prevent automatic Reflection refresh. The default is to automatically refresh.

Refresh every – How often to refresh Reflections, specified in hours, days, or weeks. This option is ignored if **Never refresh** is selected.

Never expire – Select to prevent Reflections from expiring. The default is to automatically expire after the time limit below.

Expire after – The time limit after which Reflections expire and are removed from Dremio, specified in hours, days, or weeks. This option is ignored if **Never expire** is selected.

Metadata Options

On the Metadata page, you can configure settings to refresh metadata and handle datasets.

Dataset Handling

Remove dataset definitions if underlying data is unavailable – By default, Dremio removes dataset definitions if underlying data is unavailable. This is useful when files are temporarily deleted and added back in the same location with new sets of files.

Metadata Refresh

These are the optional **Metadata Refresh** parameters:

Dataset Discovery: The refresh interval for fetching top-level source object names such as databases and tables. Set the time interval using this parameter.

Fetch every (Optional) – You can choose to set the frequency to fetch object names in minutes, hours, days, or weeks. The default is 1 hour.

Dataset Details: The metadata that Dremio needs for query planning, such as information required for fields, types, shards, statistics, and locality. These are the parameters to fetch the dataset information:

Fetch mode – You can choose to fetch only from queried datasets, which is set by default. Dremio updates details for previously queried objects in a source. Fetching from all datasets is deprecated.

Fetch every – You can choose to set the frequency to fetch dataset details in minutes, hours, days, or weeks. The default is 1 hour.

Expire after – You can choose to set the expiry time of dataset details in minutes, hours, days, or weeks. The default is 3 hours.

Privileges

On the Privileges page, you can grant privileges to specific users or roles. See Access Control for additional information about user privileges.

(Optional) For **Privileges**, enter the username or role name that you want to grant access to and click the **Add to Privileges** button. The added user or role is displayed in the **Users** table.

(Optional) For the users or roles in the **Users** table, toggle the green checkmark for each privilege you want to grant on the Dremio source that is being created.

Click **Save** after setting the configuration.

Update a Dremio Source

To edit a Dremio source:

On the Datasets page, under **Databases**, find the name of the source you want to edit.

Right-click the source name and select **Settings** from the list of actions.

In the Source Settings dialog, edit the settings you wish to update. Dremio does not

support updating the source name. For information about the settings options, see Configure a Dremio Software Cluster as a Source.

Click **Save**.

Remove a Dremio Source

To remove a Dremio source, perform these steps:

On the Datasets page, under **Databases**, find the name of the source you want to remove.

In the list of data sources, hover over the name of the source you want to remove and right-click.

From the list of actions, click **Delete**.

In the Delete Source dialog, click **Delete** to confirm that you want to remove the source.

Predicate Pushdowns

Projects offload these operations to data-source clusters. Data-source clusters either process these operations or offload them to their connected data sources.

`&&`, `||`, `!`, `AND`, `OR`
`+`, `-`, `/`, `*`, `%`
`<=`, `<`, `>`, `>=`, `=`, `<>`, `!=`
`ABS`
`ADD_MONTHS`
`AVG`
`BETWEEN`
`CASE`
`CAST`
`CEIL`
`CEILING`
`CHARACTER_LENGTH`
`CHAR_LENGTH`
`COALESCE`
`CONCAT`
`CONTAINS`
`COUNT`
`COUNT_DISTINCT`
`COUNT_DISTINCT_MULTI`
`COUNT_FUNCTIONS`
`COUNT_MULTI`
`COUNT_STAR`
`CURRENT_DATE`
`CURRENT_TIMESTAMP`
`DATE_ADD`
`DATE_DIFF`
`DATE_SUB`
`DATE_TRUNC`
`DATE_TRUNC_DAY`

`DATE_TRUNC_HOUR`
`DATE_TRUNC_MINUTE`
`DATE_TRUNC_MONTH`
`DATE_TRUNC_QUARTER`
`DATE_TRUNC_WEEK`
`DATE_TRUNC_YEAR`
`DAYOFMONTH`
`DAYOFWEEK`
`DAYOFYEAR`
`EXTRACT`
`FLATTEN`
`FLOOR`
`ILIKE`
`IN`
`IS DISTINCT FROM`
`IS NOT DISTINCT FROM`
`IS NOT NULL`
`IS NULL`
`LAST_DAY`
`LCASE`
`LEFT`
`LENGTH`
`LIKE`
`LOCATE`
`LOWER`
`LPAD`
`LTRIM`
`MAX`
`MEDIAN`
`MIN`
`MOD`
`NEXT_DAY`
`NOT`
`NVL`
`PERCENTILE_CONT`
`PERCENTILE_DISC`
`PERCENT_RANK`
`POSITION`
`REGEXP_LIKE`
`REPLACE`
`REVERSE`
`RIGHT`
`ROUND`
`RPAD`
`RTRIM`
`SIGN`
`SQRT`
`STDDEV`
`STDDEV_POP`
`STDDEV_SAMP`
`SUBSTR`
`SUBSTRING`
`SUM`
`TO_CHAR`

`TO_DATE`
`TRIM`
`TRUNC`
`TRUNCATE`
`UCASE`
`UPPER`
`VAR_POP`
`VAR_SAMP`

Limitations

You cannot query columns that use complex data types, such as `LIST`, `STRUCT`, and `MAP`. Columns of complex data types do not appear in result sets.

Was this page helpful?

Catalogs | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/catalogs/

On this page

A catalog is a metadata repository that maintains information about datasets and their structure. Catalogs enable consistent discovery, organization, and query execution by storing metadata independently of the underlying data.

A catalog typically includes:

Tables and views – definitions, schemas, columns, and data types

Data locations – where files are stored in object storage

Metadata – partitioning, file formats, and statistics

Dremio can connect to external catalogs to provide a unified metadata layer across platforms. This allows users to query existing datasets in place, without data movement, while preserving a single source of truth for metadata management.

Catalog Comparison

Catalog	Provider	Read	Write	Vended Credentials	Best For
Dremio's Open Catalog	Dremio	✔	✔	✔	Open lakehouses
Iceberg REST Catalog	Various	✔	✔	Varies	Third-party Iceberg catalogs
Snowflake Open Catalog	Snowflake	✔	✔*	✔	Snowflake-managed Iceberg tables
Unity Catalog	Databricks	✔		✔	Databricks Delta Lake with UniForm
AWS Glue Data Catalog	AWS	✔	✔	✔	AWS-native Iceberg environments

*Write supported for external catalogs only

Dremio's Open Catalog

Every Dremio project includes a built-in Open Catalog for managing your Iceberg tables. You can also connect to catalogs from other projects in your organization for cross-project collaboration.

Key features:

- Open and standards-based catalog for Apache Iceberg
- Automatic table maintenance with compaction and vacuuming
- Built-in access controls enforced at the catalog level
- Multi-engine compatibility via Iceberg REST API

Best for: Teams working with Apache Iceberg who want automated maintenance and multi-engine access without vendor lock-in.

AWS Glue Data Catalog

Connect to AWS Glue's managed metadata catalog for accessing Iceberg tables stored in Amazon S3.

Key features:

- Native integration with AWS ecosystem
- Managed metadata storage and schema management
- Support for both read and write operations
- Integration with AWS Lake Formation for fine-grained access control

Best for: AWS-native environments that use Glue for metadata management and want to query Iceberg tables with Dremio.

Iceberg REST Catalog

Connect to any Iceberg Catalog implementing the REST API specification, including Apache Polaris, AWS Glue Data Catalog, Snowflake Open Catalog, Amazon S3 tables, and Confluent Tableflow.

Key features:

- Universal compatibility with REST-compliant Iceberg catalogs
- Support for multiple authentication mechanisms
- Flexible storage credential management
- Connect to on-premises Dremio clusters for hybrid cloud analytics

Best for: Connecting to Iceberg catalogs from other vendors or on-premises systems.

Snowflake Open Catalog

Connect to Snowflake's managed service for Apache Polaris to read and write Iceberg tables across Snowflake and other compatible engines.

Key features:

- Read from internal and external Snowflake Open Catalogs
- Write to external Snowflake Open Catalogs
- Credential vending for secure storage access
- Support for AWS, Azure, and GCS storage

Best for: Organizations using Snowflake that want to query Iceberg tables with Dremio while leveraging Snowflake's catalog management.

Unity Catalog

Connect to Databricks Unity Catalog to query Delta Lake tables through the UniForm Iceberg compatibility layer.

Key features:

- Read Delta Lake tables via UniForm Iceberg metadata layer

- Integration with Databricks governance and security
- Support for AWS, Azure, and GCS storage
- Credential vending for secure access

Best for: Databricks users who want to query Delta Lake tables with Dremio using the UniForm compatibility layer.

Was this page helpful?

- Catalog Comparison
- Dremio's Open Catalog
- AWS Glue Data Catalog
- Iceberg REST Catalog
- Snowflake Open Catalog
- Unity Catalog

Databases | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/databases/

On this page

Databases are external sources that you can connect to and query data from. This includes relational databases, data warehouses, and NoSQL databases, with some sources also supporting external queries for advanced use cases.

note

For the Dremio trial, database sources are not supported.

Relational Databases

These relational databases store data in tables with predefined schemas and use SQL for queries:

- IBM Db2
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL
- SAP HANA

Data Warehouses

These data warehouses are optimized for large-scale analytical queries rather than transactional workloads:

- Amazon Redshift
- Google BigQuery
- Microsoft Azure Synapse Analytics
- Snowflake
- Vertica

NoSQL Databases

These NoSQL databases use flexible schemas and are designed for unstructured or semi-structured data:

- Apache Druid
- MongoDB

External Queries

External queries use native syntax for a source and can be used for SQL statements that are not supported directly in Dremio or are too complex to convert. To run external queries, use the `SELECT` SQL command with the `EXTERNAL_QUERY` parameter.

You can run external queries against the following databases: Amazon Redshift, Microsoft SQL Server, MySQL, and PostgreSQL. To run these queries, you need the EXTERNAL QUERY privilege. Keep in mind, however, when you update the metadata for the data source, Dremio clears permissions, formats, and data Reflections for all datasets created from external queries.

When external querying is not enabled, all queries must use standard SQL syntax. Likewise, all user or role read and write access to tables or views within a source are governed by privileges.

Was this page helpful?

Data Type Mapping

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/databases/mysql

On this page

MySQL is a managed database service for deploying cloud-native applications.

Prerequisites

Ensure that you have the following details before configuring MySQL as a source:

- Hostname or IP address
- Port
- Outbound port (3306 is the default port) open in your AWS or Azure security group

Configure MySQL as a Source

Perform these steps to configure MySQL as a source:

- On the Datasets page, you can see a truncated list of **Sources** at the bottom-left of the page. Click **Add Source**.

Alternatively, select **Databases** in the Data panel. The page displays all database sources. Click the **Add database** button at the top-right corner of that page.

- In the **Add Data Source** dialog, click **MySQL**.

The following section describes the source configuration settings in the New MySQL Source dialog.

note

Sources containing a large number of files or tables may take longer to be added. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being added. Once complete, the source becomes accessible.

General

Under **General** in the sidebar, complete the following steps:

- For **Name**, enter a name. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.
- For **Host**, enter the MySQL host name.
- For **Port**, enter the MySQL port number. The default port is 3306.
- (Optional) For **Database**, enter the MySQL database name.
- For **Authentication**, choose one of the following options:

a. Select **No Authentication** to not provide credentials.

b. Select **Master Credentials** to provide the username and password of a master database user with permissions to read required objects:

- For **Username**, enter your MySQL database username.
-

For **Password**, enter your MySQL database password.

Advanced Options

(Optional) Select **Advanced Options** in the sidebar and change any of the following settings:

Advanced Option	Description
Net write timeout (in seconds)	Seconds to wait for data from the server before aborting the connection. The default timeout is 60 seconds.
Record fetch size	Number of records to fetch at once. Set to 0 (zero) to have Dremio automatically decide. The default record fetch size is 200.
Maximum Idle Connections	The total number of connections allowed to be idle at a given time. The default maximum idle connections is 8.
Connection idle time (s)	The amount of time (in seconds) allowed for a connection to remain idle before the connection is terminated. The default connection idle time is 60 seconds.
Query timeout (s)	The amount of time (in seconds) for the query to be executed before it is canceled.
Properties	Custom key value pairs for the connection relevant to the source. To add a connection property, click **Add property** and add the property name and value.

Reflection Refresh

Select **Reflection Refresh** in the sidebar and set time intervals for Reflections to refresh or expire.

Metadata

(Optional) Select **Metadata** in the sidebar and complete the following steps:

Select **Remove dataset definitions if underlying data is unavailable**. By default, Dremio removes dataset definitions if underlying data is unavailable. This can be useful when files are temporarily deleted and added back in the same location with new sets of files.

Set the following **Metadata Refresh** parameters:

Parameter	Description	Field	Setting
Dataset Discovery	The refresh interval for fetching top-level source object names such as databases and tables.	**Fetch every**	You can choose to set the frequency to fetch object names in minutes, hours, days, or weeks. The default frequency to fetch object names is 1 hour.
Dataset Details	The metadata that Dremio needs for query planning such as information required for fields, types, shards, statistics, and locality. These are the parameters to fetch the dataset information.	**Fetch mode**	You can choose to fetch only from queried datasets that are set by default. Dremio updates details for previously queried objects in a source. Fetching from all datasets is deprecated.
		Fetch every	You can choose to set the frequency to fetch dataset details in

minutes, hours, days, or weeks. The default frequency to fetch dataset details is 1 hour. |
| **Expire after** | You can choose to set the expire time of dataset details in minutes, hours, days, or weeks. The default expiry time of dataset details is 3 hours. |

Privileges

(Optional) Select **Privileges** in the sidebar and grant privileges to specific users or roles by completing the following steps:

- For **Privileges**, enter the user name or role name that you want to grant access to and click the **Add to Privileges** button. The added user or role is displayed in the **Users** table.
- For the users or roles in the **Users** table, toggle the green checkmark for each privilege you want to grant to the MySQL source that is being created.
- Click **Save**.

Edit a MySQL Source

To edit a MySQL source:

- On the Database page, click **Databases**. A list of databases is displayed.
- Hover over the database and click the Settings !This is the icon that represents the Database settings. icon that appears next to the source.
- In the Source Settings dialog, you cannot edit the name. Editing other parameters is optional.
- Click **Save**.

Remove a MySQL Source

To remove a MySQL source, perform these steps:

- On the Datasets page, click **Databases**. A list of sources is displayed.
- Hover over the source and click the More (...) icon that appears next to the source.
- From the list of actions, click **Remove Source**. Confirm that you want to remove the source.

caution

Removing a source causes all downstream views dependent on objects in this source to break.

note

Sources containing a large number of files or tables may take longer to be removed. During this time, the source name is grayed out and shows a spinner icon, indicating

the source is being removed. Once complete, the source disappears.

Predicate Pushdowns

Dremio offloads these operations to MySQL.

`*`, `+`, `-`, `/`, `%`
`<`, `<=`, `<>`, `=`, `>`, `>=`, `!=`
`AND`, `NOT`, `NOT` `LIKE`, `OR`, `||`
`ABS`
`ACOS`
`ADD_MONTHS`
`ASIN`
`ATAN`
`ATAN2`
`AVG`
`CAST`
`CEIL`
`CEILING`
`CHAR_LENGTH`
`CHARACTER_LENGTH`
`CONCAT`
`COS`
`COT`
`CURRENT_DATE`
`CURRENT_TIME`
`CURRENT_TIMESTAMP`
`DATE_ADD`
`DATE_SUB`
`DATE_TRUNC_DAY`
`DATE_TRUNC_DECADE`
`DATE_TRUNC_HOUR`
`DATE_TRUNC_MINUTE`
`DATE_TRUNC_MONTH`
`DATE_TRUNC_SECOND`
`DATE_TRUNC_WEEK`
`DATE_TRUNC_YEAR`
`DEGREES`
`E`
`EXP`
`EXTRACT_DAY`
`EXTRACT_DOW`
`EXTRACT_DOY`
`EXTRACT_HOUR`
`EXTRACT_MINUTE`
`EXTRACT_MONTH`
`EXTRACT_QUARTER`
`EXTRACT_SECOND`
`EXTRACT_WEEK`
`EXTRACT_YEAR`
`FLOOR`
`IS DISTINCT FROM`
`IS NOT DISTINCT FROM`

`IS NOT NULL`
`IS NULL`
`LAST_DAY`
`LCASE`
`LEFT`
`LENGTH`
`LIKE`
`LN`
`LOCATE`
`LOG`
`LOG10`
`LOWER`
`LPAD`
`LTRIM`
`MAX`
`MIN`
`MOD`
`PI`
`POSITION`
`POW`
`POWER`
`RADIANS`
`RAND`
`REPLACE`
`REVERSE`
`RIGHT`
`ROUND`
`RPAD`
`RTRIM`
`SIGN`
`SIN`
`SQRT`
`STDDEV`
`STDDEV_POP`
`STDDEV_SAMP`
`SUBSTR`
`SUBSTRING`
`SUM`
`TAN`
`TIMESTAMPADD_DAY`
`TIMESTAMPADD_HOUR`
`TIMESTAMPADD_MINUTE`
`TIMESTAMPADD_MONTH`
`TIMESTAMPADD_QUARTER`
`TIMESTAMPADD_SECOND`
`TIMESTAMPADD_YEAR`
`TIMESTAMPDIFF_DAY`
`TIMESTAMPDIFF_HOUR`
`TIMESTAMPDIFF_MINUTE`
`TIMESTAMPDIFF_MONTH`
`TIMESTAMPDIFF_QUARTER`
`TIMESTAMPDIFF_SECOND`
`TIMESTAMPDIFF_WEEK`
`TIMESTAMPDIFF_YEAR`

`TO_DATE`
`TRIM`
`TRUNC`
`TRUNCATE`
`UCASE`
`UPPER`
`VAR_POP`
`VAR_SAMP`

Data Type Mapping

Dremio supports MySQL data types, as shown in the following table which provides the mappings from MySQL to Dremio data types. If there are additional MySQL types not listed in the table, then those types are not supported in Dremio.

MySQL Data Type	Dremio Type
BIT	BOOLEAN
BIGINT	BIGINT
BIGINT UNSIGNED	BIGINT
BINARY	VARBINARY
BLOB	VARBINARY
CHAR	VARCHAR
DATE	DATE
DATETIME	TIMESTAMP
DECIMAL UNSIGNED	DECIMAL
DOUBLE	DOUBLE
DOUBLE PRECISION	DOUBLE
ENUM	VARCHAR
FLOAT	FLOAT
INT	INTEGER
INT UNSIGNED	BIGINT
INTEGER	INTEGER
INTEGER UNSIGNED	BIGINT
LONG VARBINARY	VARBINARY
LONG VARCHAR	VARCHAR
LONGBLOB	VARBINARY
LONGTEXT	VARCHAR
MEDIUMBLOB	VARBINARY
MEDIUMINT	INTEGER
MEDIUMINT UNSIGNED	INTEGER
MEDIUMTEXT	VARCHAR
NUMERIC	NUMERIC
REAL	DOUBLE
SET	VARCHAR
SMALLINT	INTEGER
SMALLINT UNSIGNED	INTEGER
TEXT	VARCHAR
TIME	TIME
TIMESTAMP	TIMESTAMP
TINYBLOB	VARBINARY
TINYINT	INTEGER
TINYINT UNSIGNED	INTEGER

TINYTEXT	VARCHAR
VARBINARY	VARBINARY
VARCHAR	VARCHAR
YEAR	INTEGER

Was this page helpful?

- Prerequisites
- Configure MySQL as a Source
- General
- Advanced Options
- Reflection Refresh
- Metadata
- Privileges
- Edit a MySQL Source
- Remove a MySQL Source
- Predicate Pushdowns

Object Storage | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/object-storage/

On this page

Object storage provides scalable, cost-effective storage for data lakes. Dremio connects directly to your Amazon S3 bucket or Azure Storage to query data in place without moving or copying it.

Amazon S3

Dremio supports Amazon S3, which stores data in Amazon S3 buckets on AWS. Use data source credentials or project data credentials to enable Dremio to access Amazon S3 using the IAM role that is associated with your Dremio project.

Azure Storage

Dremio supports Azure Storage, which stores data in Azure Blob Storage and Azure Data Lake Storage Gen2.

Table Formatting

Dremio can query data in object storage across multiple formats:

- **Apache Iceberg** – Open table format designed for petabyte-scale analytics

Delta Lake – Open-source storage framework that brings reliability and performance to data lakes

Delimited files (CSV, TSV, etc.) - Text files with configurable delimiters

JSON - Structured and semi-structured JSON data

Parquet - Columnar format optimized for analytics

Excel (XLSX, XLS) - Spreadsheet files

For information about formatting files and folders as tables, see Table Formatting.

Was this page helpful?

- Amazon S3
- Azure Storage
- Table Formatting

Data Type Mapping

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/databases/oracle

On this page

Oracle is an object-relational database management system that is widely used in enterprise applications.

Prerequisites

Ensure that you have the following details before configuring Oracle as a source:

- Hostname or IP address
- Port
- Service Name
- Outbound port (1521 is the default port) open in your AWS or Azure security group

User Impersonation

The Oracle database username provided in the source configuration is the default username that is used for running queries. When queries are run against Oracle in Dremio, users use the privileges associated with the Oracle database username and run queries under that username.

You can change this default in Dremio by enabling user impersonation in the Advanced Options, which allows users to run queries under their own usernames and restricts their access. For example, `user_1` can run queries as `user_1` rather than `oracle_svc`. Before enabling user impersonation, some setup is required in Oracle to

allow one user to impersonate another user, because the username of the user in Dremio must be the same as their username in Oracle and the user must be able to connect through the Oracle database username.

To set up user impersonation, follow these steps:

- Ensure the user's username in Oracle matches their username in Dremio. If the usernames do not match, modify one of the usernames or create a new user account with a matching username.

- Run a ALTER USER command in Oracle to allow the user to connect through the Oracle database username:

Example of altering the user in Oracle

```
ALTER USER testuser1 GRANT CONNECT THROUGH proxyuser;
```

In this example, the user can log in as `testuser1` in Dremio and in Oracle, and they can connect through the `proxyuser`. The `proxyuser` is the Oracle database username provided in the source configuration.

- Log in as an admin to Dremio.

- Follow the steps for Configure Oracle as a Source using the Oracle database username and enable **User Impersonation** in the **Advanced Options**.

- Grant source privileges to the user.

Now that you have enabled user impersonation, a user logging in to Dremio with their username can access the Oracle source and its datasets according to their privileges. The user also runs queries against Oracle under their username.

Configure Oracle as a Source

Perform these steps to configure Oracle:

- On the Datasets page, you can see a truncated list of **Sources** at the bottom-left of the page. Click **Add Source**.

Alternatively, click **Databases**. The page displays all database sources. Click the **Add database** button at the top-right of that page.

- In the **Add Data Source** dialog, click **Oracle**.

The following section describes the source configuration tabs.

note

Sources containing a large number of files or tables may take longer to be added. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being added. Once complete, the source becomes accessible.

General

The **General** tab contains the required fields to create an Oracle source.

Perform these steps in the **General** tab:

In the **General** tab, for **Name**, enter a name. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.

For **Host**, enter the Oracle host name.

For **Port**, enter the Oracle port number. The default port is 1521.

For **Service Name**, enter the service name of your Oracle database.

(Optional) For **Enable TLS encryption**, enable encrypted connections to Oracle using TLS.

note

You can only enable TLS encryption or Oracle native encryption for a given source.

(Optional) For **Oracle Native Encryption**, the default encryption is **Accepted**. The other values are **Required**, **Rejected**, and **Requested**.

To enable Oracle native encryption, you should also modify the **SQLNET.Ora** file by adding the following lines.

```
SQLNET.ENCRYPTION_SERVER = required. //Set the value to required or request
SQLNET.ENCRYPTION_TYPES_SERVER = (AES256) //Set the value to the appropriate encryption and the value can be different.
SQLNET.CRYPTO_CHECKSUM_SERVER = required
```

The Oracle native encryption values are described in the following table.

Oracle Native Encryption Values	Description
Accepted	The client or server allows both encrypted and non-encrypted connections. This is the default if the parameter is not set.
Rejected	The client or server refuses encrypted traffic.
Requested	The client or server requests encrypted traffic if it is possible, but accepts non-encrypted traffic if encryption is not possible.
Required	The client or server only accepts encrypted traffic.

For **Authentication**, you must choose one of the following authentication options:

Master Authentication, this is the default option. Provide the username and password of a master database user with permissions to read required objects:

For **Username**, enter your Oracle database username.

For **Password**, enter your Oracle database password.

Secret Resource Url:

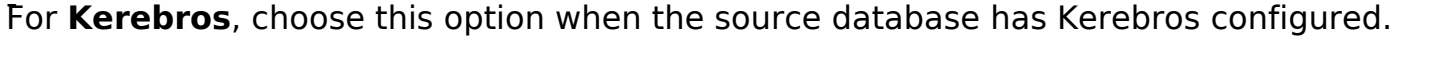
For **Username**, enter your Oracle database username.

For **Secret Resource Url**, enter the Secret Resource URL that allows Dremio to fetch the password from AWS Secrets Manager. The Secret Resource URL is the Amazon Resource Name (ARN) for the secret (for example, `arn:aws:secretsmanager:us-west-2:123456789012:secret:my-rds-secret-VNenFy`).

For **Kerebros**, choose this option when the source database has Kerebros configured.

Advanced Options

Click **Advanced Options** in the sidebar.



note

All advanced options are optional.

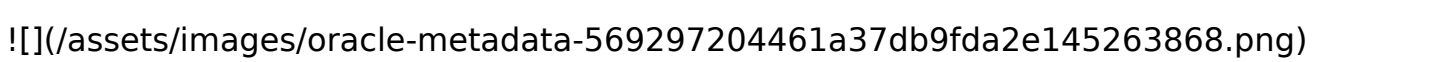
Advanced Option	Description
Use timezone as connection region	If selected, uses timezone to set connection region.
Include synonyms	If selected, includes synonyms as datasets.
Map Oracle DATE columns to TIMESTAMP	If checked, Oracle `DATE` columns are exposed as `TIMESTAMP`.
Record fetch size	Number of records to fetch at once. Set to 0 (zero) to have Dremio automatically decide. The default record fetch size is 200.
Maximum Idle Connections	The total number of connections allowed to be idle at a given time. The default maximum idle connections is 8.
Connection Idle Time	The amount of time (in seconds) allowed for a connection to remain idle before the connection is terminated. The default connection idle time is 60 seconds.
Use LDAP Naming Services	To use LDAP in the authentication of the external sources that can be used rather than locally configured users within Oracle. If checked, enter a domain name in the **Set DN for LDAP Naming Services** text box.
User Impersonation	Allows users to run queries using their credentials rather than the username provided in the source credentials. Some setup is required in Oracle to allow one user to impersonate another user. See User Impersonation.
Encryption	Provide the **SSL/TLS server certificate distinguished name**, otherwise, leave it blank to disable the DN match.
Connection Properties	Custom key value pairs for the connection relevant to the source. To add a connection property, click **Add property** and add the property name and value.

Reflection Refresh

The **Reflection Refresh** tab in the sidebar allows you to set time intervals for Reflections to refresh or expire.

Metadata

You can configure settings to refresh metadata and handle datasets. Click **Metadata** in the sidebar.



You can configure Dataset Handling and Metadata Refresh parameters.

Dataset Handling

These are the **Dataset Handling** parameters.

note

All **Dataset Handling** parameters are optional.

Parameter	Description
Remove dataset definitions if underlying data is unavailable	By default, Dremio removes dataset definitions if underlying data is unavailable. Useful when files are temporarily deleted and added back in the same location with new sets of files.

Metadata Refresh

These are the **Metadata Refresh** parameters:

Dataset Discovery: The refresh interval for fetching top-level source object names such as databases and tables. Set the time interval using this parameter.

Parameter	Description
(Optional) **Fetch every**	You can choose to set the frequency to fetch object names in minutes, hours, days, or weeks. The default frequency to fetch object names is 1 hour.

Dataset Details: The metadata that Dremio needs for query planning such as information required for fields, types, shards, statistics, and locality. These are the parameters to fetch the dataset information.

note

All **Dataset Details** parameters are optional.

Parameter	Description
Fetch mode	You can choose to fetch only from queried datasets that are set by default. Dremio updates details for previously queried objects in a source. Fetching from all datasets is deprecated.
Fetch every	You can choose to set the frequency to fetch dataset details in minutes, hours, days, or weeks. The default frequency to fetch dataset details is 1 hour.
Expire after	You can choose to set the expiry time of dataset details in minutes, hours, days, or weeks. The default expiry time of dataset details is 3 hours.

Privileges

You can grant privileges to specific users or roles.

(Optional) For **Privileges**, enter the user name or role name that you want to grant access to and click the **Add to Privileges** button. The added user or role is displayed in the **Users** table.

(Optional) For the users or roles in the **Users** table, toggle the green checkmark for each privilege you want to grant to the Oracle source that is being created.

Click **Save** after setting the configuration.

Edit an Oracle Source

To edit an Oracle source:

On the Datasets page, click **Databases**. A list of databases is displayed.

Hover over the database and click the Settings !This is the Settings icon.

In the Source Settings dialog, you cannot edit the name. Editing other parameters is optional.

Click **Save**.

Remove an Oracle Source

To remove an Oracle source, perform these steps:

On the Datasets page, click **Databases**. A list of sources is displayed.

Hover over the source and click the More (...) icon that appears next to the source.

From the list of actions, click **Remove Source**. Confirm that you want to remove the source.

caution

Removing a source causes all downstream views dependent on objects in this source to break.

note

Sources containing a large number of files or tables may take longer to be removed. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being removed. Once complete, the source disappears.

Predicate Pushdowns

Dremio offloads these operations to Oracle.

`*`, `+`, `-`, `/`, `%`
`<`, `<=`, `<>`, `=`, `>`, `>=`, `!=`
`AND`, `NOT`, `OR`, `||`

`ABS`
`ACOS`
`ADD_MONTHS`
`ASIN`
`ATAN`
`ATAN2`
`AVG`
`CAST`
`CEIL`
`CEILING`
`CHAR_LENGTH`
`CHARACTER_LENGTH`
`CONCAT`
`COS`
`COSH`
`COT`
`COVAR_POP`
`COVAR_SAMP`
`DATE_ADD`
`DATE_SUB`
`DATE_TRUNC_DAY`
`DATE_TRUNC_HOUR`
`DATE_TRUNC_MINUTE`
`DATE_TRUNC_MONTH`
`DATE_TRUNC_QUARTER`
`DATE_TRUNC_WEEK`
`DATE_TRUNC_YEAR`
`DEGREES`
`E`
`EXP`
`EXTRACT_CENTURY`
`EXTRACT_DAY`
`EXTRACT_DOW`
`EXTRACT_DOY`
`EXTRACT_HOUR`
`EXTRACT_MINUTE`
`EXTRACT_MONTH`
`EXTRACT_QUARTER`
`EXTRACT_SECOND`
`EXTRACT_WEEK`
`EXTRACT_YEAR`
`FLOOR`
`IS DISTINCT FROM`
`IS NOT DISTINCT FROM`
`IS NOT NULL`
`IS NULL`
`LAST_DAY`
`LCASE`
`LEFT`
`LENGTH`
`LIKE`
`LN`
`LOCATE`
`LOG`

`LOG10`
`LOWER`
`LPAD`
`LTRIM`
`MAX`
`MEDIAN`
`MIN`
`MOD`
`PERCENT_CONT`
`PERCENT_DISC`
`PI`
`POSITION`
`POW`
`POWER`
`RADIANS`
`REGEXP_LIKE`
`REPLACE`
`REVERSE`
`RIGHT`
`ROUND`
`RPAD`
`RTRIM`
`SIGN`
`SIN`
`SINH`
`SQRT`
`STDDEV`
`STDDEV_POP`
`STDDEV_SAMP`
`SUBSTR`
`SUBSTRING`
`SUM`
`TAN`
`TANH`
`TO_CHAR`
`TO_DATE`
`TRIM`
`TRUNC`
`TRUNCATE`
`UCASE`
`UPPER`
`VAR_POP`
`VAR_SAMP`

Data Type Mapping

Dremio supports Oracle data types, as shown in the following table which provides the mappings from Oracle to Dremio data types. If there are additional Oracle types not listed in the table, then those types are not supported in Dremio.

Oracle Data Type	Dremio Type
BINARY_DOUBLE	DOUBLE

BINARY_FLOAT	FLOAT
BLOB	VARBINARY
CHAR	VARCHAR
DATE	DATE
FLOAT	DOUBLE
INTERVALDS	INTERVAL (day to seconds)
INTERVALYM	INTERVAL (years to months)
LONG RAW	VARBINARY
LONG	VARCHAR
NCHAR	VARCHAR
NUMBER	DECIMAL
NVARCHAR2	VARCHAR
RAW	VARBINARY
TIMESTAMP	TIMESTAMP
VARCHAR2	VARCHAR

Was this page helpful?

IBM Db2 | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/databases/ibm-db2

On this page

You can add Db2 databases as sources to Dremio.

Limitations

Only IBM Db2 for Linux, UNIX, and Windows is supported.

Configure IBM Db2 as a Source

On the Datasets page, to the right of **Sources** in the left panel, click !This is the Add Source icon..

In the Add Data Source dialog, under **Databases**, select **IBM Db2**.

General Options

In the **Name** field, specify the name by which you want the Db2 source to appear in the **Databases** section. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.

Under **Connection**, follow these steps:

In the **Host** field, specify the hostname or IP address of the database to connect to.

In the **Port** field, specify the port to use when connecting. The default is 50000.

In the **Database** field, specify the name of the database.

Under **Authentication**, follow these steps:

For **Username**, enter your database username.

For **Password**, enter your database password.

note

Sources containing a large number of files or tables may take longer to be added. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being added. Once complete, the source becomes accessible.

Advanced Options

On the Advanced Options page, you can set values for these non-required options:

Option	Description
Record fetch size	Number of records to fetch at once. Set to 0 (zero) to have Dremio automatically decide. By default, this is set to **200**.
Maximum Idle Connections	The total number of connections allowed to be idle at a given time. The default number of maximum idle connections is 8.
Connection Idle Time	The amount of time (in seconds) allowed for a connection to remain idle before the connection is terminated. The default connection idle time is 60 seconds.
Query Timeout	The amount of time (in seconds) allowed to wait for the results of a query. If this time expires, the connection being used is returned to an idle state.
Enable external authorization plugin	When enabled, authorizes an external plugin.
Connection Properties	Connection properties and values for the data source.

Reflection Refresh Options

On the Reflection Refresh page, set the policy that controls how often Reflections are

scheduled to be refreshed automatically, as well as the time limit after which Reflections expire and are removed.

Option	Description
Never refresh	Select to prevent automatic Reflection refresh, default is to automatically refresh.
Refresh every	How often to refresh Reflections, specified in hours, days or weeks. This option is ignored if **Never refresh** is selected.
Never expire	Select to prevent Reflections from expiring, default is to automatically expire after the time limit below.
Expire after	The time limit after which Reflections expire and are removed from Dremio, specified in hours, days or weeks. This option is ignored if **Never expire** is selected.

Metadata Options

On the Metadata page, you can configure settings to refresh metadata and handle datasets.

Dataset Handling

These are the optional **Dataset Handling** parameters.

Parameter	Description
Remove dataset definitions if underlying data is unavailable	By default, Dremio removes dataset definitions if underlying data is unavailable. Useful when files are temporarily deleted and added back in the same location with new sets of files.

Metadata Refresh

These are the optional **Metadata Refresh** parameters:

Dataset Discovery: The refresh interval for fetching top-level source object names such as databases and tables. Set the time interval using this parameter.

Parameter	Description
(Optional) **Fetch every**	You can choose to set the frequency to fetch object names in minutes, hours, days, or weeks. The default frequency to fetch object names is 1 hour.

Dataset Details: The metadata that Dremio needs for query planning such as information required for fields, types, shards, statistics, and locality. These are the parameters to fetch the dataset information.

Parameter	Description
Fetch mode	You can choose to fetch only from queried datasets that are set by default. Dremio updates details for previously queried objects in a source. Fetching from all datasets is deprecated.
Fetch every	You can choose to set the frequency to fetch dataset details in minutes, hours, days, or weeks. The default frequency to fetch dataset details is 1 hour.

|
| **Expire after** | You can choose to set the expiry time of dataset details in minutes, hours, days, or weeks. The default expiry time of dataset details is 3 hours. |

Privileges

On the Privileges tab, you can grant privileges to specific users or roles. See Privileges for additional information about privileges.

note

All privileges are optional.

For **Privileges**, enter the user name or role name that you want to grant access to and click the **Add to Privileges** button. The added user or role is displayed in the **USERS/ROLES** table.

For the users or roles in the **USERS/ROLES** table, toggle the checkmark for each privilege you want to grant on the Dremio source that is being created.

Click **Save** after setting the configuration.

Update an IBM Db2 Source

To update an IBM Db2 source:

On the Datasets page, under **Databases** in the panel on the left, find the name of the source you want to update.

Right-click the source name and select **Settings** from the list of actions. Alternatively, click the source name and then the !The Settings icon at the top right corner of the page.

In the **Source Settings** dialog, edit the settings you wish to update. Dremio does not support updating the source name. For information about the settings options, see Configure IBM Db2 as a Source.

Click **Save**.

Delete an IBM Db2 Source

note

If the source is in a bad state (for example, Dremio cannot authenticate to the source or the source is otherwise unavailable), only users who belong to the ADMIN role can delete the source.

To delete an IBM Db2 source, perform these steps:

On the Datasets page, click **Sources** > **Databases** in the panel on the left.

In the list of data sources, hover over the name of the source you want to remove and right-click.

-

From the list of actions, click **Delete**.

In the Delete Source dialog, click **Delete** to confirm that you want to remove the source.

caution

Deleting a source causes all downstream views that depend on objects in the source to break.

note

Sources containing a large number of files or tables may take longer to be deleted. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being deleted. Once complete, the source disappears.

Predicate Pushdowns

Dremio delegates the execution of these expressions and functions to the database being queried, often dramatically improving query performance. It can also offload entire SQL queries that include one or more of these expressions and functions.

`||`, `AND`, `OR`
`=`, `+`, `-`, `/`, `*`
`<=`, `<`, `>`, `>=`, `=`, `<>`, `!=`
ABS
ADD_MONTHS
AVG
BETWEEN
CASE
CAST
CEIL
CEILING
CHAR_LENGTH
CHARACTER_LENGTH
COALESCE
CONCAT
DATE_ADD
DATE_DIFF
DATE_TRUNC
DATE_TRUNC_CENTURY
DATE_TRUNC_DAY
DATE_TRUNC_DECADE
DATE_TRUNC_HOUR
DATE_TRUNC_MICROSECOND
DATE_TRUNC_MILLENIUM
DATE_TRUNC_MILLISECOND
DATE_TRUNC_MINUTE
DATE_TRUNC_MONTH
DATE_TRUNC_QUARTER
DATE_TRUNC_SECOND
DATE_TRUNC_WEEK
DATE_TRUNC_YEAR
DAYOFMONTH

DAYOFWEEK
DAYOFYEAR
EXTRACT
FLOOR
ILIKE
IN
IS DISTINCT FROM
IS NOT DISTINCT FROM
IS NOT NULL
IS NULL
LAST_DAY
LEFT
LENGTH
LIKE
LOCATE
LOWER
LPAD
LTRIM
MAX
MIN
MOD
NOT
OR
PERCENT_RANK
POSITION
REPLACE
RIGHT
ROUND
RPAD
RTRIM
SIGN
SQRT
STDDEV
STDDEV_POP
STDDEV_SAMP
SUBSTR
SUBSTRING
SUM
TO_CHAR
TO_DATE
TRIM
TRUNC
TRUNCATE
UPPER

Data Type Mapping

The following table shows the mappings from Db2 to Dremio data types.

note

If a type is not present in the table, it is not currently supported.

| Db2 Database Type | Dremio Type |

| --- | --- |
BIGINT	BIGINT
BLOB	VARCHAR
BOOLEAN	BOOLEAN
CHAR	VARCHAR
CHAR () FOR BIT DATA	VARCHAR
CLOB	VARCHAR
DATE	DATE
DBCLOB	VARCHAR
DECFLOAT	DECIMAL
DECIMAL	DECIMAL
DOUBLE	DOUBLE
GRAPHIC	VARCHAR
INTEGER	INTEGER
LONG VARCHAR	VARCHAR
LONG VARCHAR FOR BIT DATA	VARCHAR
LONG VARGRAPHIC	VARCHAR
REAL	DOUBLE
SMALLINT	INTEGER
TIME	TIME
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR
VARCHAR () FOR BIT DATA	VARCHAR
VARGRAPHIC	VARCHAR

Was this page helpful?

Limitations

Configure IBM Db2 as a Source

General Options

Advanced Options

Reflection Refresh Options

Metadata Options

Privileges

Update an IBM Db2 Source

Delete an IBM Db2 Source

Predicate Pushdowns

Data Type Mapping

Data Type Mapping

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/databases/mongodb

On this page

Supported Versions

Dremio supports MongoDB 3.6 through 8.0.

Prerequisites

Ensure that that you have the outbound port (27017 is the default port) open in your AWS or Azure security group.

Configure MongoDB as a Source

General

Under **Name**, enter the name to identify the data source in Dremio. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.

Connection

Name	Description
Hosts	A list of Mongo hosts. If MongoDB is sharded, enter the mongos hosts. Otherwise, enter the mongod host.
Port	A list of Mongo port numbers. Defaults to 27017.

Connection Scheme Select how to connect to the source.

Encrypt connection Forces an encrypted connection over SSL.

Read from secondaries only Disables reading from primaries. Might degrade performance.

Authentication

No authentication method

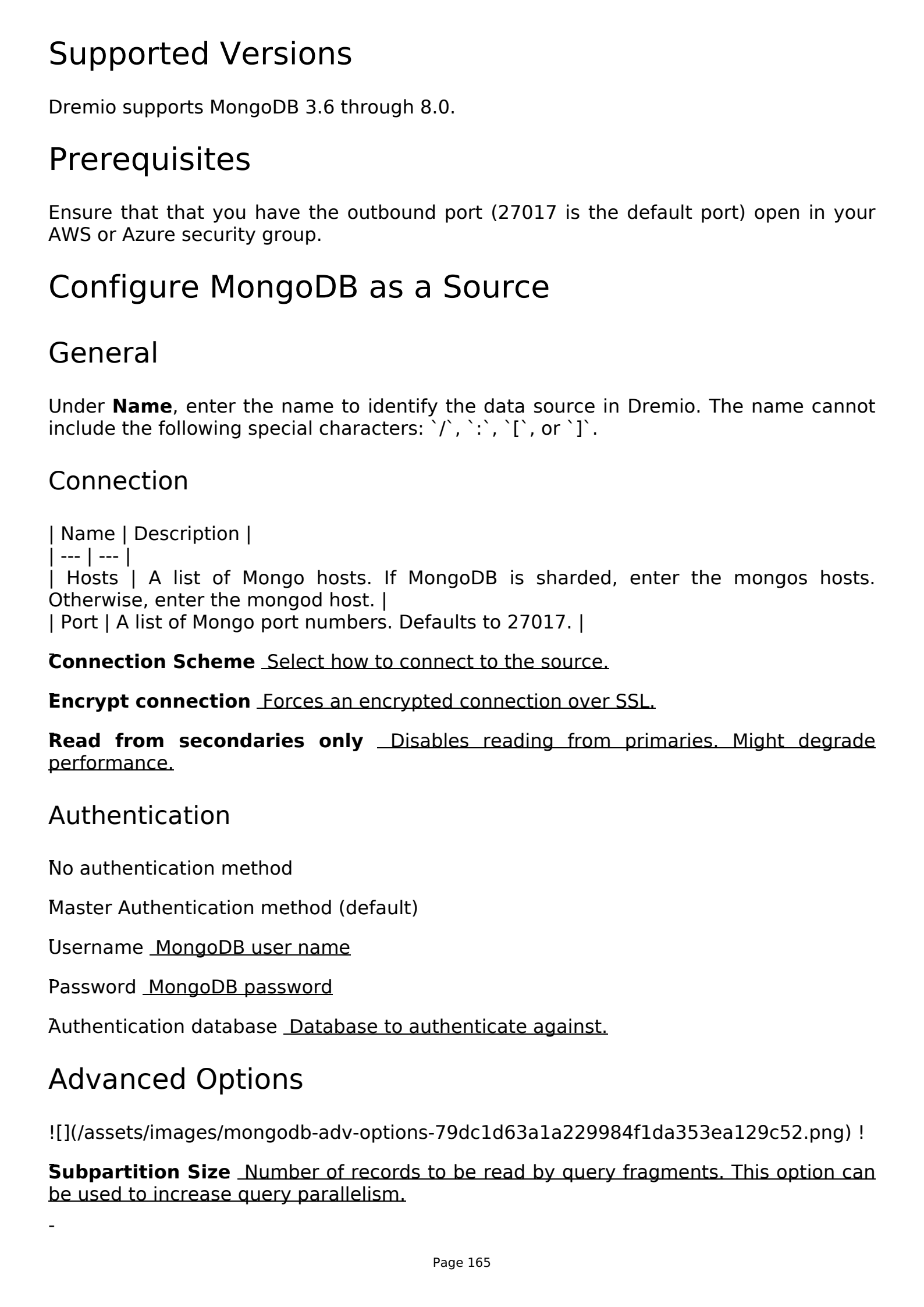
Master Authentication method (default)

Username MongoDB user name

Password MongoDB password

Authentication database Database to authenticate against.

Advanced Options



Subpartition Size Number of records to be read by query fragments. This option can be used to increase query parallelism.

-

Sample Size Number of records to be read when sampling to determine the schema for a collection. If zero the sample size is unlimited.

Sample Method The method (First or Last) by which records should be read when sampling a collection to determine the schema.

Auth Timeout (millis) Authentication timeout in milliseconds.

Field names are case insensitive When enabled, Dremio reads all known variations of a field name when determining the schema, ignoring any value set for Sample Size. All field name variations are then used when pushing an operation down to Mongo.

Connection Properties A list of additional MongoDB connection parameters.

Reflection Refresh

 !

Never refresh Specifies how often to refresh based on hours, days, weeks, or never.

Never expire Specifies how often to expire based on hours, days, weeks, or never.

Metadata

!

Dataset Handling

Remove dataset definitions if underlying data is unavailable (Default).
If this box is *not* checked and the underlying files under a folder are removed or the folder/source is not accessible,
Dremio does not remove the dataset definitions.
This option is useful in cases when files are temporarily deleted and put back in place with new sets of files.

Metadata Refresh

Dataset Discovery Refresh interval for top-level source object names such as names of DBs and tables.

Fetch every Specify fetch time based on minutes, hours, days, or weeks. Default: 1 hour

Dataset Details The metadata that Dremio needs for query planning such as information needed for
fields, types, shards, statistics, and locality.

Fetch mode Specify either Only Queried Datasets, All Datasets, or As Needed. Default: Only Queried Datasets

Only Queried Datasets Dremio updates details for previously queried objects in a source.
This mode increases query performance because less work is needed at query time for

these datasets.

All Datasets Dremio updates details for all datasets in a source.
This mode increases query performance because less work is needed at query time.

As Needed Dremio updates details for a dataset at query time.
This mode minimized metadata queries on a source when not used,
but might lead to longer planning times.

Fetch every Specify fetch time based on minutes, hours, days, or weeks. Default: 1 hour

Expire after Specify expiration time based on minutes, hours, days, or weeks. Default: 3 hours

Share

GlNbgJA9fy5csnZcuWlQ0bNuBUbrvtNl2eIQx9UX1VqlTJgyxwHfe2aNHCZPH7C4K6++67P
dKV9iAHDhzQppN6YMjy5cvlrrvuchMGMoMo0A8KESACIrZoGgbI2NhYPdkx4bGkiiVY+Buw
ejJq1Ci59dZbdVY4Q42GYe7FxARpQLuAwKmKdzomTJggyuTQjlSTFxPdCCZ67ty5A2oAyJc3
b95UhAOtBiSWkpKif7dv304HrQGXv0TABwK2kQZU+hw5crgnLyZomzZtpHXr1lq1Hzx4sD
Rp0kSyZbtUJSaxt2DympWXHTt2yJ133ikgov79+0vFihUlLi5OvvzyS7c24n1/Rs5BWoY8QC
AUIkAEfCNgi3minJ/SuHFjSUpKSlULJuODDz6oJyQ0h/QCGXSpElSbWpVWbZsmTYpqlevLuX
Ll5e6detqn0N6yrnSPPBvwPzBapC3GLLzvs5zIhBpCNhCGjAP8JRWqxo+8Vu1apXP62ldPHL
kiJQsWVKsDlSQDhyV8fHxad161WkguFq1amltBsu9VoGzlEIEiIBNPPo0iRYrIlSy+9JG+99ZY2
He6//37Nfzv7CYHhELXfq1Qn4Mcx7E/7Ax8tgmLxwbA4aNEi/+wGzBr4OmDh4hwKrGvCLZM
+e3V8x7uu+zCB3ouXA5IMTFC+ode3aVbDSglUerKRA84H5RSECkY6AbT6Nzp07S4UKFbT
D88knn3TjCjCp8p8EJjJ+kQornHEaGuyuTQjlSTFxPdCCZ67ty5A2oAyJc3
GTAp/mge0lOjoaF1eevPBMQqiw2rJww8/rO8FSfXp00feffddjxfCK/iECEIWDLa+TemMEn
AI0CkxZP6owINApDECCgUAq0G5AciGzhwoXSr18/mT9/fob7FMo+sC4iYDcCtvg0vBuFuFt
/R0YJA+VCQ0BZoSIM/I9M8+bN9b/2o14QBhy8Q4cOlOYNGvjVarwx4DkRCFcEbDNPwUg
mDrQkEAceL8E5g3eQAVp4X9WoHlQiEAkIxAU8ySSwvAzAU8ySzAwrzasGCBaL+P0Uv72JFBT4ZvHtCIQ
KRjgBJI9L/Ath/InCFCATFp3GFbWB2IkAEMhECttABDIReAiSNTDR
YbCoRcAICJA0njALbQAQyEQIkjUw0WGqwEXACAiQNQ
SeMAttABDIRArb87wk2EqYQASIQGQjwNfLIGGf2kgjYhgDNE9ugZEFEIDIQIGlExjizl0TANgRI
GrZByYKIQGQgQNKIjHFmL4mAbQiQGePMXhIB2xAgadgGJQsiApGB
AEkjMsaZvSQCtiFA0rANShZEBCIDAZJGwze0kEbEOApGEblCyICEQSNyBhn9pII2IYAS
cM2KFkQEYgMBEgakTHO7CURsA0BkoZtULIgIhAZCJA0BkoZtULIgIhAZCJA0mOc2UsiYBsCJA3boGRBRCAyECBpRMY4s5dEwDYEAHB
pRMY4s5dEwDYEYESBq2QcmCiEBkIEDSiIxxZi+JgG0IkDRsg5IFEYHIQICkERnjzF4SAdsQIGkWCXui
bU1B5LPytRte2XZgaOy62SyTiqVECc3Fs4nHcoVk8JxdbsPCYCRCCCTIWCrpjFj5365KcS
kQ+l9fWea1u0l/cIxrSEOejEhSUpL8/vvn4nK5xqckLD8dd79a+HLH3/8ZfLH3/8oOAdtIA
2Tw+ebd8n7T2tKjWlmpkidBorNE6Q+Oce0DlYY86SEOkMKzzz4rw4YN80Bnz5498pzJ
YD32yMQTWbBggTz//9dff13/1m369ff7/Bs4kKG2nG6Xb6wmmb/VK
x6dhJmbhpl6w8mKgbVr9QXnm8cilB2uD610m3X/6QBupaWqbKgQMHZNGiRRIbGys9e/JAQ
PHny6LKio6Mla9asqYEwvTF8c37tyCD+TRRx+V8+fP6+N9+/bJE088Ecp+ERfXMSKqNwAbd3
AleCgC2aBnwY95UvrkkBhNHj17WyeN8ROXvhov7gGNeQBq0DeXFPwjPyHnwCTWFJJ//nypl6eOlClCpVSp
YuXZWpWVo80aCgjRoyQb7/yXdzEDnF5z02c
OFFfr1/vmc/5Cf0+Ra/mNnnbt2smDfDhg3ac2uo05b2sj3Qs3Mq/1gMTA3Zfm8dy5dP1QAbrQTZPrUhupRjlLTDhbw65802c
OFFfr1+/vqB+CJ7YrVq2smDDz4oFy5Ma/b69ev1/x0nMzJ+2/pcPeNb/IYnkT91B/w4ZQ9A33h/ngHx4/mA2d
O/eXdfTsWNHQXtMPZjcXr10mpx36Cf0z91nbi+pdffQqnvRXuMkOMXQ1XGdhf5ovWpfsMLWyMr5DM1nbj8aC5kGhSHpLbu3WkeQjS2F
GzfKypUrZdasmdwbPdFQnTQ4MknLvi6e3a1ft1sNNgTdmRB9wUoTHcoMd5cG8/vKwb8gEq5XqzqPKGtttnaraZA2+6qJ8IMWdyNHzU9scSmy5QOVCWfNv8ccEqmjWdqQDubxaMxzZI8pXRnJQFaXDvb2L
Nmz8JHDh0wLk08+qcjPHYpj9HN58SKxe/MuxYKnLCbPkmCTnd7ycWvEfJiBhaLvD7tWXWYRz/4bAz7Oe2E1YHVRgYppSBzXnhG/b+RIOSCUl4MG
bG7wwTXnr3b9dWp5EFCvZnjL+PI/1EzrmqvsHaYG9Y
kzQUaNGaUL6888/3XhAJPxgQcekbLZt+fHHH1VjO0zbr45tANDyfbpRWEzdz98cnrONZcw8hHBhT1f3J/0/evzaDhelCO2v
b6qQ8iQj1r1qyR2rVr677DhAAheYu/dq5evVpGjhwywyE2QfyYTYB5bk2oTeSEz9aLzI7J9XFhdyQ7zL23pNU3tTFFpl0eQ9zlBXXH8WcXdSsakmzyR36BzbRzw54I7bjO5CBt1p8SnvW0lHAXM9HkyFFfwyFYsZTvfYVXkqgY95+EVG45DjJ8pIHB3w4lYHfXbsEysh7xm47DNs
BB8EshAleKgC3mCVZJyue69FQzJomvhpg05DUrK77ybdq0STs5a9SooSc7JXmpXQ7A4ZvZrpbkQJ2EmsJ5DDUWU57Uj5U9
3ktS5bUfIjJefbsWcmZM6UK1ZMEwSIASQFsoAPAJO2SpUqMmnSnn3+0FlG2bFmtHaBMzmbI4MIkpuJEpQqJHzVp2bFmtHaBMz
MXIOJlCtXLqlevbpgEhSARmFPKLJlyyZvZv22Lh9m1SeffCJxcnXHaNICmgPtBGNAU
UM8jjzwiqKdr165y0e4y5MgRj3JBFL7IdBIu6oCgqNJF6e8y5My51UaOe6eRfjBpr4DWP+Nlb
hBIPbPSc1eQ80ALwAT57rvvZPr06ZI7ZV0yIn2A7aBOrv4lSrX7tZpU01q4Y4Gs6bB27oRx4JjVHRsvVVvFNGrYS3J8ByT41p2wxF8LsoHd
oDlcf/310qhRI61lgFC85Y477PZ9d4azz9QLmVqBs7ihHVnhY67TxdJlo96Qw+u5nl0W+bVtw0Z3/GNlsxN4B5oZdmKB
WLVpokweaFMSXbwHE5C3QVHy1c8eOHdrkAonD4qXBfrLBs+xaY1F+MmERG9WYeK/K/gKl/g8KuZowH6CHuHjVKGHTu8pOVK1Ph8O+
GvLiHl+Cpy6IAhMLExK+iObNm+snRqRITE5Mq+vGc9u9Zcx6SWJxLWZaDheMpJAXJY+gKk2Fj0vQ0l6aNaK99
8U9cHVR+mwlNPPZWqDDhocR3pH330kV+Pz9kPZXvFX+8DpAlcw5fD5IwCCoaT20ioDuFgQyXI3ixTnVlgr4sUrGYPa+J47W
kuoXmoe3GAewdzuhlUB7gLllxOQ159bfHDlyWE95TASuCAFbSAMvbTb2055Vk7xDBF4N7bXAIpIv5vX2fpy1c9qo5
SIMgL+7xJXgHA4KlVnwwyfBJTk4Wb9+Br/txDRMIT3RMUjgAV6xYoKB59ASZs+eLWeLdAwYF
5gQkMDueWWW2TwsxDMLxHzAT4ED0FvhaYAKgLJAO/BMwe6xSrVo1b0HZ5w0b7
QKBVGL8MNBj4GMaMGWO9cnCxv3bCnIOjFn4WvRiuBsr/66iu3I9RaMLjXE8PR9MaxHFE
4RCBNKLgC0+DbzpziYu/iRe34dev6qdqIP2ZoGZ07d9Z8iRzJ+A5M5MKvOMvd9AOm5S2Tjv3d9Z+A5MBT0Z4
eEYfPzxx8X9FpLqQexfaxAwBHZpEkT9T7AB4xfqD+FmxDSNDb7qIFAwAr5ZYPnEG7SxInqRaiAgVIaNNKxH4ID/BgTpX9S8JDtOV88K8J
2bp0qVly5YtWzWOphQ3/ZDkHq4f9yqUVB0FB8jAx8Ot+5fqCSNk6Yl6Oewaqg0wKkjm
nT+r+Wh2TvkwWXPPXTvgqsLoD7IzA0QqxlgVTCaR6++23a8IYO3as3HfffvZoAzX38JQJjR

CVlHJJD07IIH2Yl7teV+9h4H0MX7JREcbAlRvk4Uol5Y7Sns5DX/kzeg1aBCaIddKgTDzxobmA
cLzTYMLABwIHZlqCsmECpZUP9cCkgoZgFX/XrXkCHftrp7/r3uXBN+LdLu88PCcCvhCwjTR
QOIgDL3nhPQxoHWZFBT4MmCTQMF6sVSEkhOGrs7xGBIhAxhGwlTTQHP7DWsYHhSUQA
ScjYDtpOLmzbBsRIAIZRyD1MkfGy2QJRIAIhDECJI0wHlx2jQgEAwGSRjBQZZlEIIwRIGmE8e
Cya0QgGAiQNIKBKsskAmGMAEkjjAeXXSMCwUCApBEMVFkmEQhjBEgaYTy47BoRCAYCJI1
goMoyiUAYI0DSCOPBZdeIQDAQIGkEA1WWSQTCGAGSRhgPLrtGBIKBQNqbRlxFjdinAntWY
lu8U6cubQGIrfQLFy4sZdTGwNjfgkIEiEDmRcDW/3LFjuHYkg+7amOHKOzHCcG+n9gzE9vc
YYdxxDJJj2CjGOyqldZGN+kpJ715sNX/8uXL9fZ92LIPO3dRiAAR8ETANtIAYWB7f+zybaKhe
VZ1KR7IqlWrpGLFimkSB/b1RHwSkAYEG/ni3Owq7l3u1Z4jXCG22nvjjTc0sWHvT2hHlSpVk
uLFi+vdyb139rraungfEQgXBGwxT8ymvwjG448wABjSQCrYDRz7dfoyVaCRIGxgv379BAG
Kdu/eLYg2ho2AEWjITrGGK0SkNWyVt2zZspBpNnb2hWURgVAhYIsjFD4MmCRpEYbpEPIgL
+7xJdgJHHtXYuNb5IU5g8hh2P0b92CX8CFDhujwg9j814Q1RFmI/YoARAiF+N5777l3/4bZA
U0F1xEKEcGFICZcIcrABsHQOpo2bar9MdbdyP2FWNSFqC9oRIiHYnYvx3mnTp3cYQ+t7UL
gJLN7OfpjQkQi2DX6iM2VsUs52om+ILocMMFGyGg/+owdxylE4FohYAtpwOkJH0Z6BXlxjy9
BGhynmDS//fabDgGA2KSY9JhsiIUCMwWxWuFzwO7k2LwXExsTb8CAAToeCSKRIVQBNvFF
Huw0/v333+tASNixGwRhwhVit3CESTDhCkFa0HCweTDy+AqxiHKNYKIjLADyQ3AOjcn8ol0
o//PPP9ehBRCsCWYQdg/HjumIgwI8ELIB9+zcuVPvgg5tCyQxevRoHXIA7QeZIs4JiIRCBK4FA
raYJ5gAxumZnk4gr1lZ8c6PkIkgBEQ7M8GKsG0/TBQIJjTOkQ9Bjlu2bKkdrN988400a9ZMWr
durcMBYKLD1MA2/tAk5s6dq0MJYAIjLojxl6BM7EpuDVcIMwUfiL8QizoxHV8w3SCoA1oTose
hbjhcISA0kFXPnj014YEYUTf63qFDB00iIC7kgYmG9iN8AncS1/Dx6xogYAtp2N3u/Pnza+cko
r0jHCNUcxAN/CEQo96DOBBLJDExUT95YQY0bNjQ3RwEQ4IGgAlm/CdYicFTP70CjcFXiMX0
3o+ASoh1gkDWEJgtiIeCKG7QdhBJzgjaaTQI4/SFIxYhHaFpwXRCHsQq8RVBzpTDXyIQTARs
IQ2YE1hWTY9PA51BXtzjS/AkRkT1d999Vz+db7vtNk0KX3zxhdSpU0ffYpZg8cSGxgLzAk9n
rH7A34FjkweRxBBVDGo/BL/wCVx33XX6PNCXrxCLCKAEQjIT25CYqdMaEhF+ihtvvFGHZcS
SM8gDGgMCKgGvefPm6dgr5h5TpmkX2gtyQf9hhs2aNUtrG8AI5VCIQKgRsMWngRe38ER
OryAv7vElJUuW1A7NadOm6ckC38Knn36qnYKYUCAKhFzEL3wDCJSMlRio8ojOZtT+cePGy
ddff60JpVChQoJzmAoLFiwQrJoY/4OvNhgSQJq/EIvW+0AW0GRgDqEOmEr4hZaAGK7wQ8B
JCn8NyAZtB5HA7IAphvoWLlyoNRCrrwR1gDQQbxbaEcoD2YEEIdCiEHUO90DL+vvvv3WZ
5prOxC8iYDcCiLCGT0ZETUDXzJkzXcpMCFgM8iAv7vEnyiRx1atXz6V8APqjnJsu9bR1qSe8+
xrSVDhEl4rTqotRE8c1ceJEd7qKzepSDkWdtmPHDpfyd+g03LN06VJ9XZGKS2k0+hhlq5CM
LjWBXSr+qkv5SlzKhNBpimjc5bZp08alwjLq69YvlGnai3vRfvQV7Xr//ffdacrB61JOT32rWnp2
X0e7cI78vXv3dilHrrt4tM20H3VMmTJF51Mak64HWCrnqGvChAmuDRs26GvKzHHfzwMiYC
cCjny5yxAjnsjWN0Lh0IRPAE9I2PTQPIxJYO7BU9dXKESkQ7uARnA1L2ylVW566obJpAbOpw
MT/UR/ArUL7YdPw5gypl7+EoFQImCLTwMNNq+G48Utu14jxwSxCsgAEwy/1mDJ1jwgEu/7
TLp3sGhzPT2/aZVr7k8rT1qOS3/tNeWa34y035TBXyKQUQRs0zRMQ2DLK3MgKP+wBqcnl
kCxjIoJSiECRCD0CNhOGqHvAmskAkQglAjwcR1KtFkXEQgDBEgaYTCI7AIRCCUCJI1Qos26iE
AYIEDSCINBZBeIQCgRIGmEEm3WRQTCAAGSRhgMIrtABEKJAEkjlGizLiIQBgiQNMJgENkFIh
BKBEgaoUSbdRGBMECApBEGg8guEIFQIkDSCCXarIsIhAECJI0wGER2gQiEEgGSRijRZl1EIA
wQIGmEwSCyC0QglAjYtglPKBvNukQOnzknW45fCrANPIrkiJUyOXMQGiIQdARIGkGH+OorS
LnoksX7jsiRs5disJiSKuSKl9PnL0jf5etFZdFyfYE8MqZxDZOFv0QgaAg42jxJPJsizacvlmnb9w
UNACcX/MzitbL5+EmfTWxUJJ88Xvm/qPYX5TJ7+MzNi0TAPgQcr2ngSXpWbfEfabL52EkpEp
ddnq5axmfXYZpM2bZX8sRGy4lzKT7z8CIRCAYCjicNf50GmfT+7S9ZfiBRskSJ3FuuuDxXo5
x+3g7+faP8tOeQVt3rF8orb9xQTanz5+WhBavUjuAix9Uke6Z6WZXnsBxTx0fPpMgZRUxV
8+aUcU1qSTQKvCxL9x+V11ZukALZY+XfU8kSkzWLdK5UUubsOqjP47NllWENq0q9gnl0fa
ZNKKF5sQlyqH4VOZmSuu5WJQpJz1/Xyt5TZ3SZA+pWlhYqvxGYHzlU2b4EhPE/S/6UGNXO
sU1qyvqjSfKrMmMoRCAUCDjaPEkLgIG/b5CVBxOle7Wy0lpNwMlb98gvauL875/bZP6/h9QT
uqy8WKuCrDqUKJM27VLaykU9eZMVOXQsX0xAJjB/9qlJe68671ShhPydmCQzd+73qBZkck
bdmzMmmwyqV0VP5P/bsFOfv3J9JZ13+OrNckGxUf/lf8sKRWJoU+fKpWTh3sMyQqWlqrtgX
un68xqllZyXfnUqaQfmwJUbNZl5VO7jxJswSsTHSduShWRog/RFjPNRJC8RgStCIFNqGngKL1
EawE1F8kuHskXVZDsvi/YekdWHjmkCACEUi88uxxUpxKvoZ+uOHFeayKWo9lD37y9fXGsk
51WMlNoFcks3dQ0eARAGSMJboHgMrFdZMEFR91trt8iYm2pInNIEVh8+LguVVpOonJXLDx
6VO8oUkYcqltBFbDyWJD+rdoHAIKZukB1WP0BqzYrll4JxMfLC0r9kkzJJGqi2+xNvwgAZ4RO
rtB8KEQgVApmSNKDuX1RPdqjkrWcudWO1Pem0nEq5IP1X/C3n1GSKVmEOUhQxZFXhDC
EwaRCwyCpZLqfhWjaLWWLNg+Pzl5cpEqKzarMg2+UQCmiHSJSavC6lbYhUzJ3gvrWN0oB

AZBBr3Umq/ZDRinzwMbLtxKk0SWOrSgdBYJUEBPbKig3SpUopKa9WUyhEIFQIZArSiPGKcQ
IygNyjtIdeNctf1g6i9IR6UT2x88REy9et62vfRIe5K7TpYCegMEW8BW0E50CDMLJCaRSXCCt
1fuQZoXwhNxTOq8kGWk9CdNrDATMEfg/4VShE4Foh4Pi/PkxE+Brw+V09tfGJVpOmXM54
vRQLR+Ue5ZdoP2e5/LDzgMYRJsRfR08IfA8Hk8/aor77IgrroGEVA1rGZ5t3Kz/KMa0FzVO+l
VrK/DFaicl/Y+F8lzQG5X/ZfTJZmVpH5I7Zv8m+02dMFimVM052KM0J72pYxRAGNCnkL5oju
zWZx0Qg6Aik/WgLevWBK4BhMWf3Qf0xuZ9VqySjbqymVx/6/LZeX8bkaaCe2hVyx0t3tSrx
7OJ1ymgQ9fSPkkPJ57QWAAKKzfrfigTIJy2TxNSHX2PieB9DwzBlvKXa1HXRH3plA/nwhuawB
lWV6XJB12/qhi9kpNIy0PbOP61GVsF7F4Xi/gtDmS82RloWLyiPL1ztNo10xstfMFPuU74Zfyss
1rw8JgJ2IpDpI6zBP4Al0uwWMgBAWFbNpcwUEEeoBW2CBZNLrbgEErQzTrXdaBCB8jOdCF
xrBDI9aVxrAFk/EYg0BBzv04i0AWF/iYDTESBpOH2E2D4i4DAESBoOGxA2hwg4HQGShtNHi
O0jAg5DgKThsAFhc4iA0xEgaTh9hNg+IuAwBEgaDhsQNocIOB0BkobTR4jtIwIOQ4Ck4bAB
YXOIgNMRIGk4fYTYPiLgMARIGg4bEDaHCDgdAZKG00eI7SMCDkOApOGwAWFziIDTESBpO
H2E2D4i4DAESBoOGxA2hwg4HQGShtNHiO0jAg5DgKThsAFhc4iA0xEgaTh9hNg+IuAwBE
gaDhsQNocIOB0BkobTR4jtIwIOQ4Ck4bABYXOIgNMR+H9Aywkjmi5+mwAAAABJRU5ErkJg
gg==) !

You can specify which users can edit. Options include:

- All users can edit.
- Specific users can edit.

Predicate Pushdowns

Dremio offloads these operations to MongoDB:

ABS
ADD
AND
CASE
CEIL
CONCAT
DAY_OF_MONTH
DIVIDE
EQUAL
EXP
FLOOR
GREATER
GREATER_OR_EQUAL
HOUR
LESS
LESS_OR_EQUAL
LN
LOG
LOG10
MAX
MIN
MINUTE
MOD
MONTH
MULTIPLY
NOT
NOT_EQUAL
OR

POW
REGEX
SECOND
SQRT
SUBSTR
SUBTRACT
TO_LOWER
TO_UPPER
TRUNC
YEAR

For More Information

For information about Dremio data types, see Data Types.

Limitations

Queries that un-nest nested fields are not allowed as they would cause incorrect schemas. This may be easily circumvented by pushing filters into the subquery or by simply not referencing the alias.

Data Type Mapping

Dremio supports selecting the following MongoDB Database types.
The following table shows the mappings from MongoDB to Dremio data types. If there are additional MongoDB types not listed in the table, then those types are not supported in Dremio.

MongoDB Database Type	Dremio Type
ARRAY	LIST
BINDATA	VARBINARY
BOOL	BOOLEAN
DATE	TIMESTAMP
DBPOINTER	{ "namespace": VARCHAR, "id": VARBINARY }
DOUBLE	DOUBLE
INT	INTEGER (or DOUBLE if store.mongo.read_numbers_as_double set)
JAVASCRIPT	VARCHAR
JAVASCRIPTWITHSCOPE	{ "code": VARCHAR, "scope": { ... } }
LONG	BIGINT (or DOUBLE if store.mongo.read_numbers_as_double set)
OBJECT	STRUCT
OBJECTID	VARBINARY
REGEX	{ "pattern": VARCHAR, "options": VARCHAR }
STRING	VARCHAR
SYMBOL	VARCHAR
TIMESTAMP	TIMESTAMP

Was this page helpful?

Supported Versions

Prerequisites

Vertica | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/databases/vertica

On this page

Vertica is an analytical database.

Prerequisites

Ensure that you have the following details before configuring Vertica as a source:

- Database name
- Hostname or IP address
- Port
- Outbound port (5433 is the default port) open in your AWS or Azure security group

Configure Vertica as a Source

- On the Datasets page, to the right of **Sources** in the left panel, click !This is the Add Source icon..
- In the Add Data Source dialog, under **Databases**, select **Vertica**.

note

Sources containing a large number of files or tables may take longer to be added. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being added. Once complete, the source becomes accessible.

General

For **Name**, enter the name to identify the data source in Dremio. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.

Connection

Name	Description
Host	Vertica host name.
Port	Vertica port number. Defaults to 5433.
Database	Service name of your database.

Authentication

For **Username**, enter your Vertica username.

For **Password**, enter your Vertica password.

Advanced Options

Specify advanced options with the following settings.

Record fetch size: Number of records to fetch at once. Set to 0 (zero) to have Dremio automatically decide. By default, this is set to **10**.

Maximum idle connections: The total number of connections allowed to be idle at a given time. By default, this is set to **8**.

Connection idle time (s): The amount of time (in seconds) allowed for a connection to remain idle before the connection is terminated. By default, this is set to **60**.

Query timeout: The amount of time (in seconds) allowed to wait for the results of a query. If this time expires, the connection being used is returned to an idle state.

Enable external authorization plugin: When enabled, authorizes an external plugin.

Connection Properties: Connection properties and values for the data source.

Reflection Refresh

You can set the policy that controls how often Reflections are scheduled to be refreshed automatically, as well as the time limit after which Reflections expire and are removed.

Option	Description
Never refresh	Select to prevent automatic Reflection refresh, default is to automatically refresh.
Refresh every	How often to refresh Reflections, specified in hours, days or weeks. This option is ignored if **Never refresh** is selected.
Set refresh schedule	Specify the daily or weekly schedule.
Never expire	Select to prevent Reflections from expiring, default is to automatically expire after the time limit below.
Expire after	The time limit after which Reflections expire and are removed from

Dremio, specified in hours, days or weeks. This option is ignored if **Never expire** is selected. |

Metadata

Specifying metadata options is handled with the following settings.

Dataset Handling

Remove dataset definitions if underlying data is unavailable (Default).

If this box is *not* checked and the underlying files under a folder are removed or the folder/source is not accessible, Dremio does not remove the dataset definitions. This option is useful in cases when files are temporarily deleted and put back in place with new sets of files.

Metadata Refresh

These are the optional **Metadata Refresh** parameters:

Dataset Discovery: The refresh interval for fetching top-level source object names such as databases and tables. Set the time interval using this parameter.

Parameter	Description
Fetch every	You can choose to set the frequency to fetch object names in minutes, hours, days, or weeks. The default frequency to fetch object names is 1 hour.

Dataset Details: The metadata that Dremio needs for query planning such as information needed for fields, types, shards, statistics, and locality. These are the parameters to fetch the dataset information.

Parameter	Description
Fetch mode	You can choose to fetch only from queried datasets. Dremio updates details for previously queried objects in a source. By default, this is set to **Only Queried Datasets**.
Fetch every	You can choose to set the frequency to fetch dataset details in minutes, hours, days, or weeks. The default frequency to fetch dataset details is 1 hour.
Expire after	You can choose to set the expiry time of dataset details in minutes, hours, days, or weeks. The default expiry time of dataset details is 3 hours.

Privileges

On the Privileges tab, you can grant privileges to specific users or roles. See Access Control for additional information about privileges.

note

All privileges are optional.

-

For **Privileges**, enter the user name or role name that you want to grant access to and click the **Add to Privileges** button. The added user or role is displayed in the **USERS/ROLES** table.

For the users or roles in the **USERS/ROLES** table, toggle the checkmark for each privilege you want to grant on the Dremio source that is being created.

Click **Save** after setting the configuration.

Update a Vertica Source

To update a Vertica source:

On the Datasets page, under **Databases** in the panel on the left, find the name of the source you want to update.

Right-click the source name and select **Settings** from the list of actions. Alternatively, click the source name and then the !The Settings icon at the top right corner of the page.

In the **Source Settings** dialog, edit the settings you wish to update. Dremio does not support updating the source name. For information about the settings options, see Configure Vertica as a Source.

Click **Save**.

Delete a Vertica Source

note

If the source is in a bad state (for example, Dremio cannot authenticate to the source or the source is otherwise unavailable), only users who belong to the ADMIN role can delete the source.

To delete a Vertica source, perform these steps:

On the Datasets page, click **Sources** > **Databases** in the panel on the left.

In the list of data sources, hover over the name of the source you want to remove and right-click.

From the list of actions, click **Delete**.

In the Delete Source dialog, click **Delete** to confirm that you want to remove the source.

caution

Deleting a source causes all downstream views that depend on objects in the source to break.

note

Sources containing a large number of files or tables may take longer to be deleted. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being deleted. Once complete, the source disappears.

Predicate Pushdowns

Dremio delegates the execution of these expressions and functions to the database being queried, often dramatically improving query performance. It can also offload entire SQL queries that include one or more of these expressions and functions.

`*`, `+`, `-`, `/`, `%`
`<`, `<=`, `<>`, `=`, `>`, `>=`, `!=`
AND, NOT, OR, `||`
ABS
ACOS
ADD_MONTHS
ASIN
ATAN
ATAN2
AVG
BTRIM
CAST
CBRT
CEIL
CEILING
CHAR_LENGTH
CHARACTER_LENGTH
CONCAT
COS
COT
DATE_ADD
DATE_PART
DATE_SUB
DATE_TRUNC_CENTURY
DATE_TRUNC_DAY
DATE_TRUNC_DECADE
DATE_TRUNC_HOUR
DATE_TRUNC_MICROSECOND
DATE_TRUNC_MILLISECOND
DATE_TRUNC_MINUTE
DATE_TRUNC_MONTH
DATE_TRUNC_QUARTER
DATE_TRUNC_SECOND
DATE_TRUNC_WEEK
DATE_TRUNC_YEAR
DEGREES
E
EXP
EXTRACT_CENTURY
EXTRACT_DAY
EXTRACT_DECADE
EXTRACT_DOW
EXTRACT_DOY
EXTRACT_EPOCH
EXTRACT_HOUR
EXTRACT_MILLENNIUM
EXTRACT_MINUTE
EXTRACT_MONTH

EXTRACT_QUARTER
EXTRACT_SECOND
EXTRACT_WEEK
EXTRACT_YEAR
FLOOR
ILIKE
IS DISTINCT FROM
IS NOT DISTINCT FROM
IS NOT NULL
IS NULL
ISNULL
LAST_DAY
LCASE
LEFT
LENGTH
LIKE
LN
LOCALTIME
LOCALTIMESTAMP
LOCATE
LOG
LOG10
LOWER
LPAD
LTRIM
MAX
MIN
MOD
NOW
NULLIF
PI
POSITION
POW
POWER
RADIANS
RANDOM
REGEXP_REPLACE
REPLACE
RIGHT
ROUND
RPAD
RTRIM
SIGN
SIN
SQRT
STDDEV
STDDEV_POP
STDDEV_SAMP
STRPOS
SUBSTR
SUBSTRING
SUM
TAN
TIMESTAMPADD

TIMESTAMPDIFF
TO_CHAR
TO_DATE
TO_NUMBER
TO_TIMESTAMP
TRIM
TRUNC
TRUNCATE
UCASE
UPPER
VAR_POP
VAR_SAMP
VARIANCE

Was this page helpful?

- Prerequisites
- Configure Vertica as a Source
- General
- Advanced Options
- Reflection Refresh
- Metadata
- Privileges
- Update a Vertica Source
- Delete a Vertica Source
- Predicate Pushdowns

Data Type Mapping

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/databases/postgres

On this page

PostgreSQL is an open source object-relational database management system that is known for its reliability and performance.

Prerequisites

Ensure that you have the following details before configuring PostgreSQL as a source:

- Database name
- Hostname or IP address
- Port

- Outbound port (5432 is the default port) open in your AWS or Azure security group

Configure PostgreSQL as a Source

Perform these steps to configure PostgreSQL:

- On the Datasets page, you can see a truncated list of **Sources** at the bottom-left of the page. Click **Add Source**.

Alternatively, click **Databases**. The page displays all database sources. Click the **Add database** button at the top-right of that page.

- In the **Add Data Source** dialog, click **PostgreSQL**.

The following section describes the source configuration tabs.

note

Sources containing a large number of files or tables may take longer to be added. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being added. Once complete, the source becomes accessible.

General

The **General** tab contains the required fields to create a PostgreSQL source.

Perform these steps in the **General** tab:

- In the **General** tab, for **Name**, enter a name. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.

- For **Host**, enter the PostgreSQL host name.

- For **Port**, enter the PostgreSQL port number. The default port is 5432.

- For **Database**, enter the PostgreSQL database name.

- (Optional) For **Encrypt connection**, enable encrypted connections to PostgreSQL using SSL. You can modify the encryption validation mode under **Advanced Options**.

- For **Authentication**, you must choose one of the following authentication options:

Master Authentication, this is the default option. Provide the username and password of a master database user with permissions to read required objects:

- For **Username**, enter your PostgreSQL database username.

- For **Password**, enter your PostgreSQL database password.

- **Secret Resource Url**:

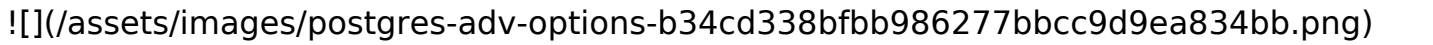
- For **Username**, enter your PostgreSQL database username.

- For **Secret Resource Url**, enter the Secret Resource URL that allows Dremio to fetch

the password from AWS Secrets Manager. The Secret Resource URL is the Amazon Resource Name (ARN) for the secret (for example, `arn:aws:secretsmanager:us-west-2:123456789012:secret:my-rds-secret-VNenFy`).

Advanced Options

Click **Advanced Options** in the sidebar.



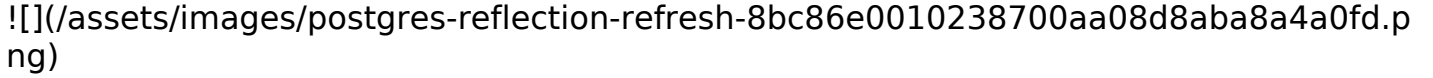
note

All advanced options are optional.

Advanced Option	Description
Record fetch size	Number of records to fetch at once. Set to 0 (zero) to have Dremio automatically decide. The default record fetch size is 200.
Maximum Idle Connections	The total number of connections allowed to be idle at a given time. The default maximum idle connections is 8.
Connection Idle Time	The amount of time (in seconds) allowed for a connection to remain idle before the connection is terminated. The default connection idle time is 60 seconds.
Encryption Validation Mode	When encryption is enabled, set the validation mode as one of the following: * **Validate certificate and hostname** (default mode) * **Validate certificate only** * **Do not validate certificate or hostname**
Connection Properties	Custom key value pairs for the connection relevant to the source. To add a connection property, click **Add property** and add the property name and value.

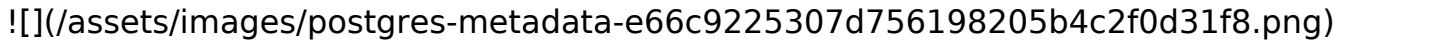
Reflection Refresh

The **Reflection Refresh** tab in the sidebar allows you to set time intervals for Reflections to refresh or expire.



Metadata

You can configure settings to refresh metadata and handle datasets. Click **Metadata** in the sidebar.



You can configure Dataset Handling and Metadata Refresh parameters.

Dataset Handling

These are the **Dataset Handling** parameters.

note

All **Dataset Handling** parameters are optional.

Parameter	Description
Remove dataset definitions if underlying data is unavailable	By default, Dremio removes dataset definitions if underlying data is unavailable. Useful when files are temporarily deleted and added back in the same location with new sets of files.

Metadata Refresh

These are the **Metadata Refresh** parameters:

Dataset Discovery: The refresh interval for fetching top-level source object names such as databases and tables. Set the time interval using this parameter.

Parameter	Description
(Optional) **Fetch every**	You can choose to set the frequency to fetch object names in minutes, hours, days, or weeks. The default frequency to fetch object names is 1 hour.

Dataset Details: The metadata that Dremio needs for query planning such as information required for fields, types, shards, statistics, and locality. These are the parameters to fetch the dataset information.

note

All **Dataset Details** parameters are optional.

Parameter	Description
Fetch mode	You can choose to fetch only from queried datasets that are set by default. Dremio updates details for previously queried objects in a source. Fetching from all datasets is deprecated.
Fetch every	You can choose to set the frequency to fetch dataset details in minutes, hours, days, or weeks. The default frequency to fetch dataset details is 1 hour.
Expire after	You can choose to set the expiry time of dataset details in minutes, hours, days, or weeks. The default expiry time of dataset details is 3 hours.

Privileges

You can grant privileges to specific users or roles.

(Optional) For **Privileges**, enter the user name or role name that you want to grant access to and click the **Add to Privileges** button. The added user or role is displayed in the **Users** table.

(Optional) For the users or roles in the **Users** table, toggle the green checkmark for each privilege you want to grant to the PostgreSQL source that is being created.

Click **Save** after setting the configuration.

Edit a PostgreSQL Source

To edit a PostgreSQL source:

- On the Datasets page, click **Databases**. A list of databases is displayed.
- Hover over the database and click the Settings ! This is the icon that represents the Database settings. icon that appears next to the source.
- In the Source Settings dialog, you cannot edit the name. Editing other parameters is optional.
- Click **Save**.

Remove a PostgreSQL Source

To remove a PostgreSQL source, perform these steps:

- On the Datasets page, click **Databases**. A list of sources is displayed.
- Hover over the source and click the More (...) icon that appears next to the source.
- From the list of actions, click **Remove Source**. Confirm that you want to remove the source.

caution

Removing a source causes all downstream views dependent on objects in this source to break.

note

Sources containing a large number of files or tables may take longer to be removed. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being removed. Once complete, the source disappears.

Predicate Pushdowns

Dremio offloads these operations to Postgres.

`*`, `+`, `-`, `/`, `%`
`<`, `<=`, `<>`, `=`, `>`, `>=`, `!=`
`AND`, `NOT`, `OR`, `||`
`ABS`
`ACOS`
`ADD_MONTHS`
`ASIN`
`ATAN`
`ATAN2`
`AVG`
`CAST`
`CBRT`
`CEIL`
`CEILING`

`CHAR_LENGTH`
`CHARACTER_LENGTH`
`CONCAT`
`COS`
`COT`
`COVAR_POP`
`COVAR_SAMP`
`DATE_ADD`
`DATE_SUB`
`DATE_TRUNC_CENTURY`
`DATE_TRUNC_DAY`
`DATE_TRUNC_DECADE`
`DATE_TRUNC_HOUR`
`DATE_TRUNC_MINUTE`
`DATE_TRUNC_MONTH`
`DATE_TRUNC_QUARTER`
`DATE_TRUNC_SECOND`
`DATE_TRUNC_WEEK`
`DATE_TRUNC_YEAR`
`DEGREES`
`E`
`EXP`
`EXTRACT_CENTURY`
`EXTRACT_DAY`
`EXTRACT_DECADE`
`EXTRACT_DOW`
`EXTRACT_DOY`
`EXTRACT_EPOCH`
`EXTRACT_HOUR`
`EXTRACT_MILLENNIUM`
`EXTRACT_MINUTE`
`EXTRACT_MONTH`
`EXTRACT_QUARTER`
`EXTRACT_SECOND`
`EXTRACT_WEEK`
`EXTRACT_YEAR`
`FLOOR`
`IS DISTINCT FROM`
`IS NOT DISTINCT FROM`
`IS NOT NULL`
`IS NULL`
`LAST_DAY`
`LCASE`
`LEFT`
`LENGTH`
`LIKE`
`LN`
`LOCATE`
`LOG`
`LOG10`
`LOWER`
`LPAD`
`LTRIM`
`MAX`

`MEDIAN`
`MIN`
`MOD`
`PERCENT_CONT`
`PERCENT_DISC`
`PI`
`POSITION`
`POW`
`POWER`
`RADIANS`
`REGEXP_LIKE`
`REPLACE`
`REVERSE`
`RIGHT`
`ROUND`
`RPAD`
`RTRIM`
`SIGN`
`SIN`
`SQRT`
`STDDEV`
`STDDEV_POP`
`STDDEV_SAMP`
`SUBSTR`
`SUBSTRING`
`SUM`
`TAN`
`TO_CHAR`
`TO_DATE`
`TRIM`
`TRUNC`
`TRUNCATE`
`UCASE`
`UPPER`
`VAR_POP`
`VAR_SAMP`

Data Type Mapping

Dremio supports PostgreSQL data types, as shown in the following table which provides the mappings from PostgreSQL to Dremio data types. If there are additional PostgreSQL types not listed in the table, then those types are not supported in Dremio.

PostgreSQL Data Type	Dremio Type
BIGSERIAL	BIGINT
BIT	BOOLEAN
BOOL	BOOLEAN
BPCHAR	VARCHAR
BYTEA	VARBINARY
CHAR	VARCHAR
DATE	DATE
DOUBLE PRECISION	DOUBLE

FLOAT4	FLOAT
FLOAT8	DOUBLE
INT2	INTEGER
INT4	INTEGER
INT8	BIGINT
INTERVAL_DS	INTERVAL (day to seconds)
INTERVAL_YM	INTERVAL (years to months)
MONEY	DOUBLE
NAME	VARCHAR
NUMERIC	DECIMAL
OID	BIGINT
SERIAL	INTEGER
TEXT	VARCHAR
TIME	TIME
TIMESTAMP	TIMESTAMP
TIMESTAMPTZ	TIMESTAMP
TIMETZ	TIME
VARCHAR	VARCHAR

Was this page helpful?

Prerequisites

Configure PostgreSQL as a Source

General

Advanced Options

Reflection Refresh

Metadata

Privileges

Edit a PostgreSQL Source

Remove a PostgreSQL Source

Predicate Pushdowns

SAP HANA | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/databases/sap-hana

On this page

Dremio supports direct connections to SAP HANA using only a username and password.

Supported Versions

The SAP HANA source supports SAP HANA 2.0. The connector was tested against HANA Express.

Configure SAP HANA as a Source

On the Datasets page, to the right of **Sources** in the left panel, click !This is the Add Source icon..

In the Add Data Source dialog, under **Databases**, select **SAP HANA**.

General

Under **Name**, enter the name to identify the data source in Dremio. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.

Connection

Under **Host**, enter the URL or IP address for the SAP HANA instance. Example: `sap://myServer:`

Under **Port**, enter the port required to access the data source. Port 39017 is the default for SAP HANA.

Under **Schema (optional)**, enter the schema within the SAP HANA instance that you would like to connect to.

Authentication

Choose an authentication method:

User Credentials - Dremio must provide a specified username and password in order to access the SAP HANA.

Username - The username with sufficient privileges to perform read/write actions on the SAP HANA instance

Password - The password associated with the username.

Advanced Options

The following settings control more advanced functionalities in Dremio.

Advanced Options

**Record fetch size ** - Number of records to fetch at once. Set to 0 (zero) to have Dremio automatically decide. The default value is `200`.

**Maximum idle connections ** - The maximum number of idle connections to keep. The default value is `8`.

**Connection idle time (s) ** - Idle time, in seconds, before a connection is considered for closure. The default value is `60`.

**Query timeout (s) ** - The timeout, in seconds, for query execution before it is canceled. Set to `0` for no timeout. The default value is `0`.

Connection Properties

[**Name ** - The unique name for any custom properties.

[**Value ** - The value associated with the custom property.

Reflection Refresh

This tab controls the frequency of Reflection refreshes or the timespan for expiration for any queries performed using this data source.

[**Never refresh ** - Prevents any query Reflections associated with this source from refreshing.

[**Refresh every ** - Sets the time interval by which Reflections for this source are refreshed. This may be set to hours, days, and weeks.

[**Set refresh schedule ** - Specify the daily or weekly schedule.

[**Never expire ** - Prevents any query Reflections associated with this source from expiring.

[**Expire after ** - Sets the time after a Reflection is created that it then expires and can no longer be used for queries. This may be set to hours, days, and weeks.

Metadata

This tab offers settings that control how dataset details are fetched and refreshed.

Dataset Handling

[**Remove dataset definitions if underlying data is unavailable ** - If this box is not checked and the underlying files under a folder are removed or the folder/source is not accessible, Dremio does not remove the dataset definitions. This option is useful in cases when files are temporarily deleted and put back in place with new sets of files.

Metadata Refresh

Dataset Discovery

[**Fetch every ** - Specifies the time interval by which Dremio fetches object names. This can be set by minutes, hours, days, and weeks.

Dataset Details

[**Fetch mode ** - Restricts when metadata is retrieved.

[**Fetch every ** - Specifies the time interval by which metadata is fetched. This can be set by minutes, hours, days, and weeks.

[**Expire after ** - Specifies the timespan for when dataset details expire after a dataset is queried. This can be set by minutes, hours, days, and weeks.

Privileges

On the Privileges tab, you can grant privileges to specific users or roles. See Access Control for additional information about privileges.

note

All privileges are optional.

For **Privileges**, enter the user name or role name that you want to grant access to and click the **Add to Privileges** button. The **USERS/ROLES** table will display the added user or role.

For the users or roles in the **USERS/ROLES** table, toggle the checkmark for each privilege you want to grant on the Dremio source that is being created.

Click **Save** after setting the configuration.

Update an SAP HANA source

To update an SAP HANA source:

On the Datasets page, under **Databases** in the panel on the left, find the name of the source you want to update.

Right-click the source name and select **Settings** from the list of actions. Alternatively, click the source name and then the !The Settings icon at the top right corner of the page.

In the **Source Settings** dialog, edit the settings you wish to update. Dremio does not support updating the source name.

Click **Save**.

Delete an SAP HANA Source

note

If the source is in a bad state (for example, Dremio cannot authenticate to the source or the source is otherwise unavailable), only users who belong to the ADMIN role can delete the source.

To delete an SAP HANA source, perform these steps:

On the Datasets page, click **Sources** > **Databases** in the panel on the left.

In the list of data sources, hover over the name of the source you want to remove and right-click.

From the list of actions, click **Delete**.

In the Delete Source dialog, click **Delete** to confirm that you want to remove the source.

caution

Deleting a source causes all downstream views that depend on objects in the source to break.

note

Sources containing a large number of files or tables may take longer to be deleted. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being deleted. Once complete, the source disappears.

Predicate Pushdowns

Dremio delegates the execution of these expressions and functions to the database being queried, often dramatically improving query performance. It can also offload entire SQL queries that include one or more of the following expressions and functions:

`%`, `*`, `+`, `-`, `/`
`<`, `<=`, `<>`, `=`, `>`, `>=`, `!=`
AND, NOT, OR, `||`
ABS
ACOS
ADD_MONTHS
ASIN
ATAN
ATAN2
AVG
CAST
CBRT
CEIL
CEILING
CHAR_LENGTH
CHARACTER_LENGTH
CONCAT
COS
COT
DATE_ADD
DATE_DIFF
DATE_SUB
DATE_TRUNC_CENTURY
DATE_TRUNC_DAY
DATE_TRUNC_DECADE
DATE_TRUNC_HOUR
DATE_TRUNC_MILLENIUM
DATE_TRUNC_MINUTE
DATE_TRUNC_MONTH
DATE_TRUNC_SECOND
DATE_TRUNC_YEAR
DEGREES
E
EXP
EXTRACT_DAY
EXTRACT_DOW
EXTRACT_DOY
EXTRACT_HOUR
EXTRACT_MINUTE
EXTRACT_MONTH
EXTRACT_QUARTER
EXTRACT_SECOND

EXTRACT_WEEK
EXTRACT_YEAR
FLOOR
IS DISTINCT FROM
IS NOT DISTINCT FROM
IS NOT NULL
IS NULL
LCASE
LEFT
LN
LOCATE
LOG
LOG10
LPAD
LTRIM
MAX
MIN
MOD
MONTH
PI
POSITION
POW
POWER
RADIANS
RAND
REPLACE
REVERSE
RIGHT
ROUND
RPAD
RTRIM
SIGN
SIN
SQRT
SUBSTR
SUBSTRING
SUM
TAN
TO_CHAR
TO_DATE
TRIM
TRUNC
TRUNCATE
UCASE
UPPER
YEAR

Was this page helpful?

Supported Versions

Configure SAP HANA as a Source

General

Advanced Options

Reflection Refresh

Metadata

Privileges

Update an SAP HANA source

Delete an SAP HANA Source

Predicate Pushdowns

Snowflake | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/databases/snowflake

On this page

Snowflake is a cloud data warehouse.

User Impersonation

Dremio supports OAuth with impersonation for Snowflake. This allows Dremio users to authenticate via external OAuth and map to Snowflake roles securely. For reference, see Snowflake's Create Security Integration (External OAuth) documentation.

Before configuring a Snowflake source with user impersonation, perform the following steps:

Run the following curl commands to obtain the Dremio OAuth parameters (issuer and public key):

To get the issuer:

```
curl --location 'https://<dremio_url>/api/v3/external-oauth/discovery/jwt-issuer' \
--header 'Authorization: Bearer <Token>' \
--header 'Content-Type: application/json' \
--data ''
```

To get the public key:

```
curl --location 'https://<dremio_url>/api/v3/external-oauth/discovery/jwks' \
--header 'Authorization: Bearer <Token>' \
--header 'Content-Type: application/json' \
--data ''
```

The above JWKS response needs to be converted to PEM format, which Snowflake accepts. We recommend using this open-source tool: rsa-jwks-to-pem.

Example conversion:

```
python rsa-jwks-to-pem.py key_jwks.json
```

Create a Snowflake external OAuth security integration in Snowflake.
Set `EXTERNAL_OAUTH_ISSUER` to the issuer obtained from Dremio, `EXTERNAL_OAUTH_RSA_PUBLIC_KEY` to the PEM-formatted key from the script, and `EXTERNAL_OAUTH_AUDIENCE_LIST` to any additional audience values for token validation beyond your Snowflake account URL.

Create Security Integration

```
CREATE OR REPLACE SECURITY INTEGRATION snowflake_imp
TYPE = EXTERNAL_OAUTH
ENABLED = TRUE
EXTERNAL_OAUTH_TYPE = CUSTOM
EXTERNAL_OAUTH_ISSUER = '<issuer-from-dremio>'
EXTERNAL_OAUTH_AUDIENCE_LIST = ('<audience-values>')
EXTERNAL_OAUTH_ALLOWED_ROLES_LIST = ('REGRESSION', 'ACCOUNTADMIN', 'PUBLIC')
EXTERNAL_OAUTH_RSA_PUBLIC_KEY = '<PEM-formatted-key>'
EXTERNAL_OAUTH_TOKEN_USER_MAPPING_CLAIM = 'sub'
EXTERNAL_OAUTH_SNOWFLAKE_USER_MAPPING_ATTRIBUTE = 'login_name';
```

To configure Snowflake source in any mode (which allows users to assume any role they have access to in Snowflake), enable `EXTERNAL_OAUTH_ANY_ROLE_MODE` for Snowflake security integration:
Alter Security Integration

```
ALTER SECURITY INTEGRATION snowflake_imp SET EXTERNAL_OAUTH_ANY_ROLE_MODE = 'ENABLE';
```

Configure Snowflake as a Source

In the bottom-left corner of the Datasets page, click **Add Source**.

Under **Databases** in the Add Data Source dialog, select **Snowflake**.

note

Sources containing a large number of files or tables may take longer to be added. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being added. Once complete, the source becomes accessible.

General

Perform these steps in the **General** tab:

For **Name**, specify the name by which you want the Snowflake source to appear in the **Databases** section. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.

For **Host**, specify the hostname of the Snowflake source in the format `LOCATOR_ID.snowflakecomputing.com`.

For **Port**, enter the port number. The default port is 443.

note

The optional connection parameters are case-sensitive. For example, if the name of a warehouse uses upper case only (e.g., WAREHOUSE1), specify it the same way in the **Warehouse** field.

(Optional) For **Database**, specify the default database to use.

(Optional) For **Role**, specify the default access-control role to use.

(Optional) For **Schema**, specify the default schema to use.

(Optional) For **Warehouse**, specify the warehouse that will provide resources for executing DML statements and queries.

Under **Authentication**, you must choose one of the following authentication methods:

Login-password authentication:

For **Username**, enter your Snowflake username.

For **Password**, enter your Snowflake password.

Key-pair authentication (see Snowflake's key-pair documentation):

For **Username**, enter your Snowflake username.

For **Private Key**, enter your generated Snowflake private key in Privacy Enhanced Mail (PEM) format.

(Optional) For **Private key passphrase**, enter the passphrase if you are using an encrypted private key.

OAuth with impersonation: This allows Dremio users to authenticate via external OAuth and map to Snowflake roles securely. If you have not already, complete the steps in User Impersonation.

Set the JWT `audience` parameter to match Snowflake's `EXTERNAL_OAUTH_AUDIENCE_LIST`. This ensures proper token validation and role mapping between Dremio and Snowflake.

Advanced

On the Advanced Options page, you can set values for these non-required options:

Option	Description
Maximum Idle Connections	The total number of connections allowed to be idle at a given time. The default maximum idle connections is 8.
Connection Idle Time	The amount of time (in seconds) allowed for a connection to remain idle before the connection is terminated. The default connection idle time is 60 seconds.
Query Timeout	The amount of time (in seconds) allowed to wait for the results of a query. If this time expires, the connection being used is returned to an idle state.

Reflection Refresh

On the Reflection Refresh page, set the policy that controls how often Reflections are scheduled to be refreshed automatically, as well as the time limit after which Reflections expire and are removed.

Option	Description
Never refresh	Select to prevent automatic Reflection refresh, default is to automatically refresh.
Refresh every	How often to refresh Reflections, specified in hours, days or weeks. This option is ignored if **Never refresh** is selected.
Never expire	Select to prevent Reflections from expiring, default is to automatically expire after the time limit below.
Expire after	The time limit after which Reflections expire and are removed from Dremio, specified in hours, days or weeks. This option is ignored if **Never expire** is selected.

Metadata Options

On the Metadata page, you can configure settings to refresh metadata and handle datasets.

Dataset Handling

These are the optional **Dataset Handling** parameters.

Parameter	Description
Remove dataset definitions if underlying data is unavailable	By default, Dremio removes dataset definitions if underlying data is unavailable. Useful when files are temporarily deleted and added back in the same location with new sets of files.

Metadata Refresh

These are the optional **Metadata Refresh** parameters:

Dataset Discovery: The refresh interval for fetching top-level source object names such as databases and tables. Set the time interval using this parameter.

Parameter	Description
(Optional) **Fetch every**	You can choose to set the frequency to fetch object names in minutes, hours, days, or weeks. The default frequency to fetch object names is 1 hour.

Dataset Details: The metadata that Dremio needs for query planning such as information required for fields, types, shards, statistics, and locality. These are the parameters to fetch the dataset information.

Parameter	Description
Fetch mode	You can choose to fetch only from queried datasets that are set by

default. Dremio updates details for previously queried objects in a source. Fetching from all datasets is deprecated. |
| **Fetch every** | You can choose to set the frequency to fetch dataset details in minutes, hours, days, or weeks. The default frequency to fetch dataset details is 1 hour. |
| **Expire after** | You can choose to set the expiry time of dataset details in minutes, hours, days, or weeks. The default expiry time of dataset details is 3 hours. |

Privileges

On the Privileges page, you can grant privileges to specific users or roles. See Access Control for additional information about user privileges.

(Optional) For **Privileges**, enter the user name or role name that you want to grant access to and click the **Add to Privileges** button. The added user or role is displayed in the **Users** table.

(Optional) For the users or roles in the **Users** table, toggle the green checkmark for each privilege you want to grant on the Dremio source that is being created.

Click **Save** after setting the configuration.

Edit a Snowflake Source

To edit a Snowflake source:

On the Datasets page, click **External Sources** at the bottom-left of the page. A list of sources is displayed.

Hover over the external source and click the Settings !This is the icon that represents the Source settings. icon that appears next to the source.

In the Source Settings dialog, you cannot edit the name. Editing other parameters is optional. For parameters and advanced options, see Configure Snowflake as a Source.

Click **Save**.

Remove a Snowflake Source

To remove a Snowflake source, perform these steps:

On the Datasets page, click **External Sources** at the bottom-left of the page. A list of sources is displayed.

Hover over the source and click the More (...) icon that appears next to the source.

From the list of actions, click **Remove Source**. Confirm that you want to remove the source.

caution

Removing a source causes all downstream views dependent on objects in this source to break.

note

Sources containing a large number of files or tables may take longer to be removed. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being removed. Once complete, the source disappears.

Predicate Pushdowns

These operations and functions are performed by Snowflake warehouses:

`||`, `AND`, `OR`
`+`, `-`, `/`, `*`
`<=`, `<`, `>`, `>=`, `=`, `<>`, `!=`
`ABS`
`ADD_MONTHS`
`AVG`
`BETWEEN`
`CASE`
`CAST`
`CEIL`
`CEILING`
`CHARACTER_LENGTH`
`CHAR_LENGTH`
`COALESCE`
`CONCAT`
`COUNT`
`COUNT_DISTINCT`
`COUNT_DISTINCT_MULTI`
`COUNT_FUNCTIONS`
`COUNT_MULTI`
`COUNT_STAR`
`DATE_ADD`
`DATE_SUB`
`DATE_TRUNC`
`DATE_TRUNC_DAY`
`DATE_TRUNC_HOUR`
`DATE_TRUNC_MINUTE`
`DATE_TRUNC_MONTH`
`DATE_TRUNC_QUARTER`
`DATE_TRUNC_WEEK`
`DATE_TRUNC_YEAR`
`DAYOFMONTH`
`DAYOFWEEK`
`DAYOFYEAR`
`EXTRACT`
`FLOOR`
`ILIKE`
`IN`
`IS DISTINCT FROM`
`IS NOT DISTINCT FROM`
`IS NOT NULL`
`IS NULL`
`LAST_DAY`
`LEFT`

`LENGTH`
`LIKE`
`LOCATE`
`LOWER`
`LPAD`
`LTRIM`
`MAX`
`MEDIAN`
`MIN`
`MOD`
`NOT`
`PERCENT_CONT`
`PERCENT_DISC`
`PERCENT_RANK`
`POSITION`
`REGEXP_LIKE`
`REPLACE`
`REVERSE`
`RIGHT`
`ROUND`
`RPAD`
`RTRIM`
`SIGN`
`SQRT`
`STDDEV`
`STDDEV_POP`
`STDDEV_SAMP`
`SUBSTR`
`SUBSTRING`
`SUM`
`TO_CHAR`
`TO_DATE`
`TRIM`
`TRUNC`
`TRUNCATE`
`UPPER`

Was this page helpful?

User Impersonation

Configure Snowflake as a Source

General

Advanced

Reflection Refresh

Metadata Options

Privileges

Edit a Snowflake Source

Remove a Snowflake Source

Predicate Pushdowns

Open Catalog | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/catalogs/open-catalog

On this page

Dremio's Open Catalog is a built-in lakehouse catalog powered by Apache Polaris. It provides centralized, secure access to your Iceberg tables while automating data maintenance to keep performance optimized.

Key Capabilities

Comprehensive Access Controls – Protect your data with Role-Based Access Control (RBAC) alongside fine-grained security policies. RBAC privileges are enforced within the catalog itself, providing complete privilege enforcement as the catalog is shared with other projects or engines. Apply row filters to limit data visibility by criteria such as region, or use column masks to obfuscate sensitive information such as Social Security numbers.

Automatic Table Maintenance – Open Catalog handles Iceberg table compaction and vacuum operations automatically, so you get optimal query performance and lower storage costs without manual intervention. These table maintenance jobs run on a dedicated engine that requires no routing rules, engine configuration, or scheduling.

Analyst-Friendly Data Discovery – Built-in data product capabilities make it easy for analysts to find and understand data. Use semantic search to discover datasets with natural language, leverage descriptions and labels to understand business context, and explore lineage graphs to trace data transformations and assess downstream impact.

Multi-Engine Compatibility – Access your catalog from any query engine or framework that supports the Iceberg REST API. Ingest data using Spark or Flink, then leverage Dremio to curate and deliver refined data products—all working from the same catalog.

Every project in your Dremio organization includes an Open Catalog by default. This catalog is automatically provisioned when you create a project and provides immediate access to your Iceberg tables with full control over security, maintenance, and data organization. Your Open Catalog is ready to use out of the box.

Create a Namespace

Namespaces help you organize tables logically within your Open Catalog. You might create namespaces by team (Engineering, Revenue), by domain (Finance, Marketing), or by use case.

To create a namespace:

In the Data panel, click !Add icon next to **Namespaces**.

Enter a namespace name.

Click **Create**.

Your namespace is now ready for tables. You can create tables within the namespace using SQL or by uploading data through the Dremio console.

Naming tips:

Use lowercase letters, numbers, and underscores for maximum compatibility across engines.

Choose descriptive names that clearly indicate the namespace's purpose (e.g., `customer_analytics`, `finance_reporting`).

Avoid spaces or special characters that may require escaping in SQL queries.

Observe Table Maintenance

Open Catalog automatically performs maintenance operations such as compaction and vacuum to optimize query performance. You can monitor these jobs to understand maintenance activity.

To view maintenance jobs:

In the Dremio console, click !Jobs icon in the side navigation bar to open the Jobs page.

Select **Internal** job type.

Review jobs with engine type **MAINTENANCE**.

Add Catalogs from Other Projects

In addition to your Open Catalog, you can connect to Open Catalogs from other projects in your organization. When you add a catalog from another project, it appears as a source in your project, enabling you to access shared data assets while maintaining consistent security and governance. All Role-Based Access Control (RBAC) privileges and fine-grained access controls are enforced at the catalog level, ensuring secure data access across projects.

Catalogs from other projects enable:

Cross-Project Collaboration – Access tables from other teams without duplicating data or managing separate copies.

Centralized Governance – Data owners maintain control over access policies while enabling broad data sharing.

Consistent Security – RBAC and fine-grained controls travel with the catalog, so permissions are enforced regardless of which project accesses the data.

Simplified Data Discovery – Users can browse and query shared data assets directly from their own project workspace.

To add a catalog from another project:

In the Datasets panel, click !Add Source icon next to **Sources**.

In the Add Data Source dialog, under **Lakehouse Catalogs**, select **Open Catalog**.

In the **Name** field, choose the project hosting the desired catalog from the dropdown menu.

Click **Save**.

The catalog now appears under **Lakehouse Catalogs** in your Sources panel. You can browse its namespaces and query tables just as you would with your catalog, with all access controls enforced automatically.

You will only see projects in the dropdown where you have been granted access to their Open Catalog. If you do not see a project you expect, contact the project owner to request access.

Catalog Settings

The default catalog configurations work well for most use cases. If you need to adjust them:

Click !Settings on the left navigation bar and choose **Project Settings**.

Select **Catalog** to view the catalog settings page.

For catalogs from other projects:

Select the catalog from the **Lakehouse Catalogs** section of **Sources** on the Datasets panel.

Select **Settings** from the dropdown menu.

Select from the available tabs for additional configurations.

Reflection Refresh

Control how often Reflections are automatically refreshed and when they expire. These settings are specific to each project using the catalog.

Refresh Settings

Never refresh: Prevent automatic Reflection refresh. By default, Reflections refresh automatically.

Refresh every: Set the refresh interval in hours, days, or weeks. Ignored if **Never refresh** is selected.

Set refresh schedule: Specify a daily or weekly refresh schedule.

Expire Settings

Never expire: Prevent Reflections from expiring. By default, Reflections expire after the configured time limit.

Expire after: The time limit after which Reflections are removed from Dremio, specified in hours, days, or weeks. Ignored if **Never expire** is selected.

Metadata

Configure how Dremio handles dataset definitions and metadata refresh. These settings are specific to each project using the catalog.

In Open Catalog, metadata refresh serves two purposes:

Cache Refresh: Dremio maintains a project-level cache of table metadata to accelerate query planning and execution. Writes from Dremio query engines automatically update this cache. However, writes from other query engines only update snapshot metadata in object storage. Metadata refresh syncs these external changes into Dremio's cache to improve subsequent query performance.

Lineage Computation: Metadata refresh recomputes lineage information to reflect the latest changes in lineage graphs.

Dataset Handling

Remove dataset definitions if the underlying data is unavailable (Default) – When selected, Dremio removes dataset definitions if the underlying files are deleted or the folder/source becomes inaccessible. When deselected, Dremio retains dataset definitions even when data is unavailable. This is useful when files are temporarily deleted and replaced with new files.

Dataset Discovery

Fetch every: How often to refresh top-level source object names (databases and tables). Set the interval in minutes, hours, days, or weeks. Default: 1 hour.

Dataset Details

Metadata Dremio needs for query planning, including field information, types, shards, statistics, and locality.

Fetch mode: Choose to fetch metadata only from queried datasets. Dremio updates details for previously queried objects in the source. Default: **Only Queried Datasets**.

Fetch every: How often to fetch dataset details, specified in minutes, hours, days, or weeks. Default: 1 hour.

Expire after: When dataset details expire, specified in minutes, hours, days, or weeks. Default: 3 hours.

Privileges

Grant access to specific users or roles. See Privileges for additional information about privileges.

To grant access:

Under **Privileges**, enter the user name or role name you want to grant access to and click **Add to Privileges**. The user or role appears in the **USERS/ROLES** table.

In the **USERS/ROLES** table, toggle the checkbox for each privilege you want to grant.

Click **Save** after configuring all settings.

Delete an Open Catalog Connection

To delete a catalog connection from another project:

On the Datasets page, click **Sources** > **Lakehouse Catalogs** in the Data panel.

In the list of data sources, hover over the source you want to remove and right-click.

From the list of actions, click **Delete**.

In the Delete Source dialog, click **Delete** to confirm removal.

You cannot delete your default Open Catalog as it is a core component of your project.

note

If the source is in a bad state (for example, Dremio cannot authenticate to the source or the source is otherwise unavailable), only users who belong to the ADMIN role can delete the source.

Was this page helpful?

Key Capabilities

Create a Namespace

Observe Table Maintenance

Add Catalogs from Other Projects

Catalog Settings

Reflection Refresh

Metadata

Privileges

Delete an Open Catalog Connection

Table Formatting | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/object-storage/format

On this page

This topic describes how to query the data from your data lake sources by creating

tables.

Overview

Dremio allows you to query the data from your data lake sources without ingesting or copying it. After configuring your data source, you can format the data in your source as a table so that it can be queried in Dremio using SQL. You can format individual files or a folder of files, which will create one table with the data from the folder. You can query the file or folder without creating a table, but performance may be impacted.

This functionality is currently only supported on object storage sources.

Supported Table and File Formats

Format	File Limit
Apache Iceberg	Unlimited
Delta Lake	Unlimited
Excel	10,000
JSON	300,000
Parquet	Unlimited
Text (delimited)	300,000
XLS	10,000

note

Formatting folders that contain a mix of file formats is not supported. All files in a folder must be the same format.

The names of files and folders cannot include the following special characters: `/`, `:`, `[`, or `]`.

Format a File or Folder as a Table

To format a file or folder as a table:

On the Datasets page, navigate to the file or folder that you want to format.

To format a file or folder:

File: Hover over the file to format and click on the !This is the icon that represents the format file action. button on the far right.

Folder: Hover over the folder to format and click on the !This is the icon that represents the format folder action. button on the far right.

In the Dataset Settings dialog, for **Format**, verify that the correct format has been detected.

(Optional) For the Parquet format, click the checkbox to enable the **Ignore other file formats** if desired. If you select this option, Dremio ignores all non-Parquet files in the related folder structure, and the promoted table works as if only Parquet files are in the folder structure.

7. (Optional) For Excel, XLS, and Text (delimited) formats, you can configure the following parameters.

Excel and XLS format:

Parameter	Description
Extract Field Names	Check this box to extract the column names from the first line of the file.
Expand Merged Cells	Check this box to expand out cells that have been merged in the Excel sheet.
Sheet Name	Specify the sheet name if there are multiple sheets within the file.

Text (delimited) format:

Parameter	Description
Field Delimiter	Select the delimiter in your text file - Comma, Tab, Pipe, or Custom. For Custom, enter the characters used for a delimiter in the text box.
Quote	Select the character that is used for quotes in your file - Single Quote, Double Quote, Custom. For Custom, enter the characters used for quotes in the text box.
Comment	Select the character that is used for comments in your file - Number Sign, Double Slash, Custom. For Custom, enter the characters used for comments in the text box.
Line Delimiter	Select the character that is used for a line delimiter in your file - CRLF, LF, Custom. For Custom, enter the characters used for a line delimiter in the text box.
Escape	Select the character that is used for to escape in your file - Double Quote, Back Quote, Backslack, Custom. For Custom, enter the characters used to escape in the text box.
Extract Field Names	Select this option to extract the column names from the first line of the file.
Skip First Line	Select this option to skip the first line of the file.
Trim Column Names	Select this option to trim whitespace from the left and right sides of the names of the columns. This option is checked by default.

8. Click **Save**. The parameter values will be auto-detected but can be altered. When you click **Save**, your table will appear in the Datasets page.

Partitioned Data

The data in a source dataset might be partitioned into one or more levels of subfolders, one level for each partition column. In such cases, when you format the source dataset as a table, Dremio appends to the table one column per partition. The data type of the appended columns is varchar.

Examples

Example 1

The source dataset `orders` is partitioned on the column `state`. Each subfolder is named `state=<abbreviation>`, where `<abbreviation>` is the two-letter abbreviation of the name of a US state.

When you format `orders` as a table, all of the columns from the Parquet files, except `state`, are included, and Dremio appends the column `dir0`, which has the data type varchar. The values in that column are `state=AK` for the rows from the file `0.parquet`, `state=AL` for the rows from the file `1.parquet`, `state=AR` for the rows from the file `2.parquet`, and so on.

Example 2

The source dataset `orders` is partitioned on the columns `state` and `zipCode`. Each first-level subfolder is named `state=<abbreviation>`, where `<abbreviation>` is the two-letter abbreviation of the name of a US state. Each second-level subfolder is named `zipCode=<zip code>`.

When you format `orders` as a table, all of the columns from the Parquet files, except `state` and `zipCode`, are included, and Dremio appends the columns `dir0` and `dir1`, which both have the data type varchar.

The values in `dir0` are `state=AK` for all rows in which the value in `dir1` is `zipCode=<zip code in AK>`, `state=AL` for all rows in which the value in `dir1` is `zipCode=<zip code in Al>`, and so on.

The values in `dir1` are `zipCode=99502` for the rows from `0.parquet`, `zipCode=99503` for the rows from `1.parquet`, and so on.

Partition Column Inference

By default, when a source dataset uses Parquet files and the data is partitioned on one or more columns, Dremio behaves as described in Partitioned Data. However, if you select the option **Enable partition column inference** in the advanced options for a data source, you change how Dremio handles partition columns.

In addition to appending a column named `dir<n>` for each partition level and using subfolder names for values in those columns, Dremio detects the name of the partition column, appends a column that uses that name, detects values in the names of subfolders, and uses those values in the appended column.

Appended columns still use the varchar data type.

Examples

Example 1

The source dataset `orders` is partitioned on the column `state`. Each subfolder is named `state=<abbreviation>`, where `<abbreviation>` is the two-letter abbreviation of the name of a US state.

When you format `orders` as a table, all of the columns from the Parquet files are included, and Dremio appends the columns `dir0` and `state`, both of which use the varchar data type.

The values in `dir0` are `state=AK` for the rows from the file `0.parquet`, `state=AL` for the rows from the file `1.parquet`, `state=AR` for the rows from the file `2.parquet`, and so on.

The values in `state` are `AK` for the rows from the file `0.parquet`, `AL` for the rows from the file `1.parquet`, `AR` for the rows from the file `2.parquet`, and so on.

Example 2

The source dataset `orders` is partitioned on the columns `state` and `zipCode`. Each first-level subfolder is named `state=<abbreviation>`, where `<abbreviation>` is the two-letter abbreviation of the name of a US state. Each second-level subfolder is named `zipCode=<zip code>`.

When you format `orders` as a table, all of the columns from the Parquet files are included, and Dremio appends the columns `dir0`, `dir1`, `state`, and `zipCode`, all of which use the varchar data type.

The values in `dir0` are `state=AK` for all rows in which the value in `dir1` is `zipCode=<zip code in AK>`, `state=AL` for all rows in which the value in `dir1` is `zipCode=<zip code in Al>`, and so on.

The values in `dir1` are `zipCode=99502` for the rows from `0.parquet`, `zipCode=99503` for the rows from `1.parquet`, and so on.

The values in `state` are `AK` for all rows in which the value in `zipCode` is `<zip code in AK>`, `AL` for all rows in which the value in `zipCode` is `<zip code in Al>`, and so on.

The values in `zipCode` are `99502` for the rows from `0.parquet`, `99503` for the rows from `1.parquet`, and so on.

Requirements

For the **Enable partition column inference** option to work correctly, ensure that the names of your subfolders meet these requirements:

- Names must be in the format `column_name=<column_value>`. `colum_name=` is a valid input.
- Names must meet Dremio's naming conventions for columns.
- Names must be unique within and across directory levels.
- Names must not be present in data files.
- All Parquet files in the source dataset must be in leaf subfolders.

How Dremio Handles Existing Tables

If you enable the **Enable partition column inference** option, and already have one or more tables that are based on sources that use Parquet files and that are partitioned, those existing tables remain as they are until you run the `ALTER TABLE` command twice on each. The first time, you run the command to cause Dremio to forget the metadata for the table. The second time, you run the command to cause Dremio to refresh the metadata. The commands are listed in ALTER Commands to Cause Dremio to Forget and to Refresh Metadata on each of those tables.

For example, before you enable the **Enable partition column inference** option, your `orders` table might have these columns:

orderID	*multiple columns*	dir0
000001	...	state=CA
000002	...	state=WA

Suppose that you enable the **Enable partition column inference** option. The columns in the table remain the same. If you want to take advantage of partition column inference, you run these two SQL commands:

SQL commands for partition column inference

```
ALTER TABLE path.to.orders FORGET METADATA
ALTER TABLE path.to.orders REFRESH METADATA
```

As a result of the second `ALTER TABLE` command, Dremio adds the column `state`:

orderID	*multiple columns*	dir0	state
000001	...	state=CA	CA
000002	...	state=WA	WA

Because Dremio appends the new column, any views that are defined on the table and that use the `dir0` column are still valid. When you define new views, you can use the appended column.

Enable Partition Column Inference

After you follow the steps in either of these procedures, Dremio uses partition column inference for all source datasets that you format to tables.

To enable partition column inference for a new source:

- On the Datasets page, click **Add Source** below the list of sources.
- Select **Advanced Options**.
- Select **Enable partition column inference**.
- Specify any other settings that you want for your new data source.
- Click **Save**.

To enable partition column inference for an existing source:

-

On the Datasets page, click the name of the data source.

In the top-right corner of the page, click the gear icon.

In the Edit Source dialog, select **Advanced Options**.

Select **Enable partition column inference**.

Click **Save**.

If there are existing tables that are based on datasets in the current data source, run the two ALTER commands described in `ALTER` Commands to Cause Dremio to Forget and to Refresh Metadata on each of those tables.

note

If you change the partitioning schema of a source dataset after enabling partition column inference, metadata refreshes of all tables defined on the source dataset fail. To resolve this problem, run the two `ALTER` commands described here on each of the affected tables.

`ALTER` Commands to Cause Dremio to Forget and to Refresh Metadata

When you enable partition column inference on a source, you might have one or more existing tables in Dremio that are based on datasets in that source. Also, you might you enable partition column inference on a source and then change the partition schema of a source dataset that is the basis of one or more tables in Dremio.

In both cases, you must run these two `ALTER` commands on each affected table:

SQL ALTER commands to forget and refresh metadata

```
ALTER TABLE <dataset_path> FORGET METADATA
ALTER TABLE <dataset_path> REFRESH METADATA
```

Enable Automatic Formatting of Data

You can configure a source to automatically format the data located in the source to tables when a user triggers a query on the data for the first time.

To configure data to be automatically formatted:

Click on the **Object Storage** header in the left panel on the **Datasets** page.

On the Object storage page, hover over the source for which you want to enable this property. Click the !This is the icon that represents additional settings. button on the far right and select **Settings**.

In the **Source Settings** dialog, navigate to the **Metadata** tab.

In the **Metadata** tab, click the checkbox to **Automatically format files into tables when users issue queries**.

-

Click **Save**.

Remove Formatting on Data

Removing the formatting on a table will revert the table to the folder or file format that it was originally in. Removing the formatting is not supported for tables in an Open Catalog.

To remove the formatting on data:

On the **Datasets** page, locate the table for which you want to remove formatting.

Hover over the table for which you want to remove formatting. Click the !This is the icon that represents additional settings. button and select **Remove Format**.

In the **Remove Format** dialog, confirm that you want to remove formatting for the selected dataset. Any views that have been created from this table will be disconnected from the parent.

Time Zone Support

Dremio does *not* apply any conversions to the `TIME` or `TIMESTAMP` entry that is in the datasource table. Dremio retrieves the time or timestamp value with the assumption that the time zone is in Coordinated Universal Time (UTC).

Time Zone Limitation

For JSON files where the time zone for the time or timestamp values are not in UTC time, Dremio assumes and processes the values as in UTC time. For such files, we recommend that you convert these values to the UTC time zone before using in Dremio.

Related Topics

Apache Iceberg – Open table format designed for petabyte-scale analytics.

Delta Lake – Open-source storage framework that brings reliability and performance to data lakes.

Parquet – Columnar storage format optimized for analytical workloads.

Was this page helpful?

Overview

Supported Table and File Formats

Format a File or Folder as a Table

Partitioned Data

Examples

Partition Column Inference

- Examples
- Requirements
- How Dremio Handles Existing Tables
- Enable Partition Column Inference
- `ALTER` Commands to Cause Dremio to Forget and to Refresh Metadata
- Enable Automatic Formatting of Data
- Remove Formatting on Data
- Time Zone Support
- Time Zone Limitation
- Related Topics

Apache Druid | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/databases/apache-druid

On this page

Apache Druid is a high performance, real-time analytics database that delivers sub-second queries on streaming and batch data at scale and under load.

Prerequisites

Ensure that that you have the outbound port (27017 is the default port) open in your AWS or Azure security group.

Configure Apache Druid as a Source

- In the bottom-left corner of the Datasets page, click **Add Source**.
- Under **Databases** in the Add Data Source dialog, select **Apache Druid**.

General

- In the **Name** field, specify the name by which you want the Druid source to appear in the list of data sources. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.
- Under **Connection**, follow these steps:
- In the **Host** field, specify the hostname or IP address of the Druid source.
- In the **Port** field, specify the port to use. The default port is 8888.
- (Optional) Select **Use SSL** to use SSL to secure connections.

Under **Authentication**, specify the username and password for Dremio to use when connecting to the Druid source.

note

Sources containing a large number of files or tables may take longer to be added. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being added. Once complete, the source becomes accessible.

Advanced Options

On the Advanced Options page, you can set values for these non-required options:

Option	Description
Maximum Idle Connections	The total number of connections allowed to be idle at a given time. The default maximum idle connections is 8.
Connection Idle Time	The amount of time (in seconds) allowed for a connection to remain idle before the connection is terminated. The default connection idle time is 60 seconds.
Query Timeout	The amount of time (in seconds) allowed to wait for the results of a query. If this time expires, the connection being used is returned to an idle state.

Reflection Refresh

On the Reflection Refresh page, set the policy that controls how often Reflections are scheduled to be refreshed automatically, as well as the time limit after which Reflections expire and are removed.

Option	Description
Never refresh	Select to prevent automatic Reflection refresh, default is to automatically refresh.
Refresh every	How often to refresh Reflections, specified in hours, days or weeks. This option is ignored if **Never refresh** is selected.
Never expire	Select to prevent Reflections from expiring, default is to automatically expire after the time limit below.
Expire after	The time limit after which Reflections expire and are removed from Dremio, specified in hours, days or weeks. This option is ignored if **Never expire** is selected.

Metadata

On the Metadata page, you can configure settings to refresh metadata and handle datasets.

Dataset Handling

These are the optional **Dataset Handling** parameters.

| Parameter | Description |

| --- | --- |
| **Remove dataset definitions if underlying data is unavailable** | By default, Dremio removes dataset definitions if underlying data is unavailable. Useful when files are temporarily deleted and added back in the same location with new sets of files. |

Metadata Refresh

These are the optional **Metadata Refresh** parameters:

Dataset Discovery: The refresh interval for fetching top-level source object names such as databases and tables. Set the time interval using this parameter.

Parameter	Description
(Optional) **Fetch every**	You can choose to set the frequency to fetch object names in minutes, hours, days, or weeks. The default frequency to fetch object names is 1 hour.

Dataset Details: The metadata that Dremio needs for query planning such as information required for fields, types, shards, statistics, and locality. These are the parameters to fetch the dataset information.

Parameter	Description
Fetch mode	You can choose to fetch only from queried datasets that are set by default. Dremio updates details for previously queried objects in a source. Fetching from all datasets is deprecated.
Fetch every	You can choose to set the frequency to fetch dataset details in minutes, hours, days, or weeks. The default frequency to fetch dataset details is 1 hour.
Expire after	You can choose to set the expiry time of dataset details in minutes, hours, days, or weeks. The default expiry time of dataset details is 3 hours.

Privileges

On the Privileges page, you can grant privileges to specific users or roles. See Privileges for additional information about user privileges.

(Optional) For **Privileges**, enter the user name or role name that you want to grant access to and click the **Add to Privileges** button. The added user or role is displayed in the **Users** table.

(Optional) For the users or roles in the **Users** table, toggle the green checkmark for each privilege you want to grant on the Dremio source that is being created.

Click **Save** after setting the configuration.

Edit a Druid Source

To edit a Druid source:

On the Datasets page, click **Sources** > **Databases** in the panel on the left. The list of data sources appears to the right.

-

Hover over the name of the Druid source.

Click !The Settings icon to the right.

In the **Source Settings** dialog, you cannot edit the name. Editing other parameters is optional. For parameters and advanced options, see Configuring Apache Druid as a Source.

Click **Save**.

Remove a Druid Source

To remove a Druid source, perform these steps:

On the Datasets page, click **Sources** > **Databases** in the panel on the left. The list of data sources appears to the right.

Hover over the name of the Druid source.

Click !The Ellipsis icon to the right.

From the list of actions, click **Remove Source**. Confirm that you want to remove the source.

warning

Removing a source causes all downstream views dependent on objects in this source to break.

note

Sources containing a large number of files or tables may take longer to be removed. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being removed. Once complete, the source disappears.

Predicate Pushdowns

These operations are performed by Druid:

`!=`
`*`
`+`
`-`
`/`
`<`
`<=`
`<>`
`=`
`>`
`>=`
abs
acos
and
asin
atan

atan2
avg
cast
ceil
concat
convert_timezone
cos
cot
degrees
floor
is not null
is null
length
like
ln
log
lower
lpad
ltrim
max
min
mod
not
or
power
radians
regexp_like
replace
reverse
round
rpad
rtrim
sign
sin
substr
substring
sum
tan
tanh
trim
upper
`||`

Was this page helpful?

- Prerequisites
- Configure Apache Druid as a Source
- General
- Advanced Options
- Reflection Refresh

-

Metadata

Privileges

Edit a Druid Source

Remove a Druid Source

Predicate Pushdowns

Amazon S3 | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/object-storage/amazon-s3

On this page

Amazon S3 is an object storage service from AWS.

Supported Formats

Dremio can query data stored in S3 in file formats (including delimited, Excel (XLSX), JSON, and Parquet) and table formats (including Apache Iceberg and Delta Lake).

Add an Amazon S3 Source

To add an S3 source:

From the Datasets page, next to Sources, click !Add Source icon..

In the Add Data Source dialog, under **Object Storage**, click **Amazon S3**.

General

To configure an S3 source:

Name – In the Name field, enter a name for the Amazon S3 source. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.

Authentication – Provide the role that Dremio will assume to gain access to the source:

Create an AWS IAM role in your AWS account that trusts Dremio.

Add an S3 Access Policy to your custom role that provides access to your S3 source.

Add the **Role ARN** to the source configuration.

Public Buckets – (Optional) Click **Add bucket** and enter the public S3 bucket URL. You can add multiple public S3 buckets. AWS credentials are not necessary if you are accessing only public S3 buckets.

Encrypt Connection – (Optional) To secure the connections between the S3 buckets and Dremio, select the **Encrypt connection** checkbox.

Advanced Options

Click **Advanced Options** in the left menu sidebar.

Apply requester-pays to S3 requests – The requester (instead of the bucket owner) pays the cost of the S3 request and the data downloaded from the S3 bucket.

Enable file status check – Enabled by default; uncheck the box to disable. Enables Dremio to check if a file exists in the S3 bucket before proceeding to handle errors gracefully. Disable this option when there are no files missing from the S3 bucket or when the file's access permissions have not changed. Disabling this option reduces the amount of communication to the S3 bucket.

Root Path – The root path for the Amazon S3 bucket. The default root path is /.

VPC-restricted S3 buckets are not supported.

Server-side encryption key ARN – Add the ARN key created in AWS Key Management Service (KMS) if you want to store passwords in AWS KMS. Ensure that the AWS credentials you share with Dremio have access to this ARN key.

Default CTAS Format – Choose the default format for tables you create in Dremio: either Parquet or Iceberg (default).

Connection Properties – Provide custom key-value pairs for the connection relevant to the source. Click **Add Property**. For Name, enter a connection property. For Value, enter the corresponding connection property value.

Allowlisted buckets – Add an approved S3 bucket in the text field. You can add multiple S3 buckets. When using this option to add specific S3 buckets, you will only be able to see those buckets and not all the buckets that may be available in the source. Buckets entered must be valid. Misspelled or nonexistent buckets will not appear in the resulting source.

Under Cache Options:

Enable local caching when possible – Selected by default, along with asynchronous access for cloud caching. Uncheck the checkbox to disable this option.

Max percent of total available cache space to use when possible – Specifies the disk quota, as a percentage, that a source can use on any single executor node only when local caching is enabled. The default is 100 percent of the total disk space available on the mount point provided for caching. You can either manually enter a percentage in the value field or use the arrows to the far right to adjust the percentage.

Reflection Refresh

Click **Reflection Refresh** in the left menu sidebar. This section allows you to manage how often reflections are refreshed and how long data can be served before expiration. To learn more about reflections, refer to Manual Reflections. All settings are optional.

You can set the following refresh policies for reflections:

Refresh period – Manage the refresh period by either enabling the option to never refresh or setting a refresh frequency in hours, days, or weeks. The default frequency to

refresh reflections is every hour.

Expiration period – Set the expiration period for the length of time that data can be served by either enabling the option to never expire or setting an expiration time in hours, days, or weeks. The default expiration time is three hours.

Metadata

Click **Metadata** in the left menu sidebar. This section allows you to configure settings to refresh metadata and enable other dataset options.

You can configure Dataset Handling and Metadata Refresh parameters.

Dataset Handling

Select from the following options. All settings are optional.

Remove dataset definitions if underlying data is unavailable – By default, Dremio removes dataset definitions if underlying data is unavailable. This option is for scenarios when files are temporarily deleted and added back in the same location with new sets of files.

Automatically format files into tables when users issue queries – Enable this option to allow Dremio to automatically format files into tables when you run queries. This option is for scenarios when the data contains CSV files with non-default options.

Metadata Refresh

The **Metadata Refresh** parameters include **Dataset Discovery** and **Dataset Details**.

Dataset Discovery – The refresh interval for fetching top-level source object names such as databases and tables. Use this parameter to set the time interval. You can choose to set the frequency to fetch object names in minutes, hours, days, or weeks. The default frequency to fetch object names is one hour.

Dataset Details – The metadata that Dremio needs for query planning, such as information required for fields, types, shards, statistics, and locality. The following describes the parameters that fetch the dataset information.

Fetch mode – You can choose to fetch only from queried datasets, which is set by default. Dremio updates details for previously queried objects in a source. Fetching from all datasets is deprecated.

Fetch every – You can choose to set the frequency to fetch dataset details in minutes, hours, days, or weeks. The default frequency to fetch dataset details is one hour.

Expire after – You can choose to set the expiry time of dataset details in minutes, hours, days, or weeks. The default expiry time of dataset details is three hours.

Privileges

Click **Privileges** in the left menu sidebar. This section allows you to grant privileges to

specific users or roles. To learn more about how Dremio allows for the implementation of granular-level privileges, see Privileges. All settings are optional.

To add a privilege for a user or role:

In the Add User/Role field, enter the user or role name to which you want to apply privileges.

Click **Add to Privileges**. The user or role is added to the Users table.

To set privileges for a user or role:

In the Users table, identify the user for which you want to set privileges and click under the appropriate column (Select, Alter, Create Table, etc.) to either enable or disable that privilege. A green checkmark indicates that the privilege is enabled.

Click **Save**.

Edit an Amazon S3 Source

To edit an S3 source:

From the Datasets page, right-click on the source to edit and select **Settings**.

In the Edit Source dialog box, make changes as needed. For information about the settings in each category, see Add an Amazon S3 Source.

Click **Save**.

Remove an Amazon S3 Source

To remove an S3 source:

From the Datasets page, right-click on the source to be removed and select **Delete**.

Confirm that you want to remove the source.

Create an AWS IAM Role

To create an AWS IAM role that provides Dremio with access to your source:

Sign in to the AWS Identity and Access Management (IAM) console.

From the left menu pane, under Access management, select **Roles**.

On the Roles page, click **Create role**.

On the Select trusted entity page:

Under **Trusted entity type**, select the radio button for **Custom Trust Policy**.

Delete the current JSON policy and paste in the custom trust policy template:

Custom Trust Policy Template

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAssumeRoleWithExternalId",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<dremio_trust_account>:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "<project_id>"
        }
      }
    },
    {
      "Sid": "AllowTagSessionFromCallerRole",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<dremio_trust_account>:root"
      },
      "Action": "sts:TagSession"
    }
  ]
}
```

Replace `<dremio_trust_account>` with your Dremio Trust Account ID.

Click !Settings in the side navigation bar and choose **Project settings** to copy your Project ID. Replace `<project_id>` with your Project ID.

Click **Next** to go to the Add permissions page. No edits are needed on this page.

Click **Next** to go to the Name, review, and create page.

In the **Role details** section, in the Role name field, enter a name for this role.

Click **Create role**.

Add an S3 Access Policy to a Custom Role

To add the required S3 access policy to your custom role:

On the Roles page, click the role name. Use the **Search** field to locate the role if needed.

From the Roles page, in the Permissions section, click **Add permissions** > **Create inline policy**.

On the Create policy page, click the **JSON** tab.

Delete the current JSON policy and copy the IAM Policy Template for S3. Replace `<bucket-name>` with the name of your S3 bucket.

- Click **Next**.
- On the Review policy page, in the Name field, enter a name for the policy.
- Click **Create policy**. The policy is created and you are returned to the Roles page.

Was this page helpful?

- Supported Formats
- Add an Amazon S3 Source
- General
- Advanced Options
- Reflection Refresh
- Metadata
- Privileges
- Edit an Amazon S3 Source
- Remove an Amazon S3 Source
- Create an AWS IAM Role
- Add an S3 Access Policy to a Custom Role

Data Type Mapping

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/databases/amazon-redshift

On this page

Amazon Redshift is a cloud data-warehouse service.

There are two different types of connection that you can make to a Redshift cluster that you add as a source:

- A secure connection to a publicly accessible Redshift cluster
- A secure connection to a private Redshift cluster

Create a Secure Connection to a Publicly Accessible Amazon Redshift Cluster

When Dremio runs queries against the Redshift cluster, the compute engines in your VPC and the cluster communicate through a connection that consists of a NAT gateway and internet gateway in the VPC used with Dremio, the internet, and the internet gateway in the VPC used for Redshift.

Prerequisites for the VPC that You Are Using for Your Dremio project

- Ensure that the compute engines are deployed in a private subnet.
- Ensure that a NAT gateway configured in a public subnet.
- Ensure that an internet gateway is attached to the VPC.

Prerequisites for the VPC that You Are Using for Amazon Redshift

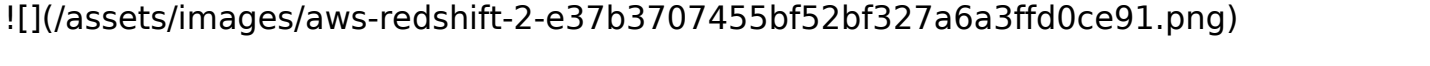
- Ensure that the Redshift cluster is in a public subnet.
- Ensure that you know the IP address of the NAT gateway that is in the VPC that you are using for Dremio.

Steps to Follow in Amazon Redshift

- In the **Clusters** table, click the name of the Redshift cluster that you plan to use. The UI console for the cluster opens.
- Make the cluster publicly accessible:
- Click **Actions** in the upper-right corner of the console.
- Select **Modify publicly accessible setting**.
- In the **Edit publicly accessible** dialog, select the check box **Turn on Publicly accessible**, and select the Elastic IP address to use for connections to the cluster.
- Create an inbound rule for the IP address of the NAT gateway that is in the VPC being used with Dremio:
- In the UI console for the cluster, scroll down to the **Network and security settings** section.
- Click the name of the VPC security group. The UI console for the security group opens.
- In the **Inbound rules** section, click **Edit Inbound Rules**.
- On the Edit inbound rules page, click **Add rule**.
- In the **Type** field, select **Redshift**.
- Specify the IP address for the NAT gateway in the field to the right of the **Source** field.
- Click **Save rules**.

Create a Secure Connection to a Private Amazon Redshift Cluster

When Dremio runs queries against the Redshift cluster, the compute engines in your VPC and the cluster communicate through a VPC peering connection.



Prerequisite for the VPC that You Are Using for Your Dremio project

Ensure that the compute engines are deployed in a private subnet.

Prerequisite for the VPC that You Are Using for Amazon Redshift

Ensure that the Redshift cluster is in a private subnet.

Steps in AWS

1. Create a VPC peering connection between the two VPCs.
2. Add the Redshift cluster VPC's CIDR block as a destination in the route tables of the VPC that you are using for Dremio:
3. In the left navigation bar of the VPC console, click **Your VPCs**.
4. Select the VPC that you are using for Dremio.
5. In the **Details** section, copy the VPC ID and copy the IPv4 CIDR.
6. In the left navigation bar, click **Route tables**.
7. In the search field at the top of the Route tables page, paste the VPC ID.
8. Select the route table.
9. In the **Routes** section, click **Edit routes**.
10. On the Edit routes page, click **Add route**.
11. In the **Destination** field, paste the IPv4 CIDR that you copied in step 3.
12. In the **Target** field, specify your VPC peering connection.
13. Click **Save changes**.
14. Add the Dremio VPC's CIDR block as a destination in the route tables of the Redshift cluster's VPC:
15. In the left navigation bar of the VPC console, click **Your VPCs**.
16. Select the VPC that you are using for Redshift.
17. In the **Details** section, copy the VPC ID and copy the IPv4 CIDR.
18. In the left navigation bar, click **Route tables**.
19. In the search field at the top of the Route tables page, paste the VPC ID.
20. Select the route table.

In the **Routes** section, click **Edit routes**.

On the Edit routes page, click **Add route**.

In the **Destination** field, paste the IPv4 CIDR.

In the **Target** field, specify your VPC peering connection.

Click **Save changes**.

In the security group for the Redshift cluster, create an inbound rule for the CIDR block of the VPC that you are using for Dremio:

In the left navigation bar of the VPC console, click **Your VPCs**.

Select the VPC that you are using for Dremio.

In the **Details** section, copy the IPv4 CIDR.

Open the Redshift console.

Select your Redshift cluster. The UI console for the cluster opens.

Under **Properties**, scroll down to the **Network and security settings** section.

Click the name of the VPC security group. The UI console for the security group opens.

In the **Inbound rules** section, click **Edit Inbound Rules**.

On the Edit inbound rules console, click **Add rule**.

In the **Type** field, select **Redshift**.

In the **Port range** field, specify `5439`.

Paste the IPv4 CIDR for the Dremio VPC into the field to the right of the **Source** field.

Click **Save rules**.

In the security group for the Dremio VPC cluster, create an inbound rule for the CIDR block of the VPC that you are using for the Redshift cluster:

In the left navigation bar of the VPC console, click **Your VPCs**.

Select the VPC that you are using for the Redshift cluster.

In the **Details** section, copy the IPv4 CIDR.

In the left navigation bar of the VPC console, click **Security groups**.

For each security group that you are using for Dremio:

Click the name of the VPC security group. The UI console for the security group opens.

In the **Inbound rules** section, click **Edit Inbound Rules**.

On the Edit inbound rules console, click **Add rule**.

In the **Type** field, select **Redshift**.

In the **Port range** field, specify `5439`.

Paste the IPv4 CIDR for the Redshift cluster VPC into the field to the right of the **Source** field.

Click **Save rules**.

Add an Amazon Redshift Cluster as a Data Source

After you create a connection between the VPC that you are using with Dremio and the VPC that hosts your Redshift cluster, you can add the cluster as a data source.

Prerequisites

Ensure that you have the JDBC connection string of the Redshift database to add it as a source. You can find the JDBC connection URL in the AWS console.

Steps

Perform these steps to configure Redshift:

On the Datasets page, you can see a truncated list of **Sources** at the bottom-left of the page. Click **Add Source**.

Alternatively, click **Databases**. The page displays all database sources. Click the **Add database** button at the top-right of that page.

In the **Add Data Source** dialog, click **Amazon Redshift**.

The following section describes the source configuration tabs.

General

The **General** tab contains the required fields to create a Redshift source.

Perform these steps in the **General** tab:

In the **General** tab, for **Name**, enter a name. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.

For **JDBC Connection String**, enter the JDBC connection string of the Redshift database.

For **Authentication**, you must choose one of the following authentication options:

Master Authentication, this is the default option. Provide the username and password of a master database user with permissions to read required objects:

For **Username**, enter your Redshift database username.

For **Password**, enter your Redshift database password.

Secret Resource Url:

For **Username**, enter your Redshift database username.

For **Secret Resource Url**, enter the Secret Resource URL that allows Dremio to fetch the password from AWS Secrets Manager. The Secret Resource URL is the Amazon Resource Name (ARN) for the secret (for example, `arn:aws:secretsmanager:us-west-2:123456789012:secret:my-rds-secret-VNenFy`).

Advanced Options

Click **Advanced Options** in the sidebar.

note

All advanced options are optional.

Advanced Option	Description
Record fetch size	Number of records to fetch at once. Set to 0 (zero) to have Dremio automatically decide. The default record fetch size is 200.
Maximum Idle Connections	The total number of connections allowed to be idle at a given time. The default maximum idle connections is 8.
Connection Idle Time	The amount of time (in seconds) allowed for a connection to remain idle before the connection is terminated. The default connection idle time is 60 seconds.
Connection Properties	Custom key value pairs for the connection relevant to the source. To add a connection property, click **Add property** and add the property name and value.

Reflection Refresh

The **Reflection Refresh** tab in the sidebar allows you to set time intervals for Reflections to refresh or expire.

Metadata

You can configure settings to refresh metadata and handle datasets. Click **Metadata** in the sidebar.

You can configure Dataset Handling and Metadata Refresh parameters.

Dataset Handling

These are the **Dataset Handling** parameters.

note

All **Dataset Handling** parameters are optional.

Parameter	Description
Remove dataset definitions if underlying data is unavailable	By default, Dremio removes dataset definitions if underlying data is unavailable. Useful when files are temporarily deleted and added back in the same location with new sets of files.

Metadata Refresh

These are the **Metadata Refresh** parameters:

Dataset Discovery: The refresh interval for fetching top-level source object names such as databases and tables. Set the time interval using this parameter.

Parameter	Description
(Optional) **Fetch every**	You can choose to set the frequency to fetch object names in minutes, hours, days, or weeks. The default frequency to fetch object names is 1 hour.

Dataset Details: The metadata that Dremio needs for query planning such as information required for fields, types, shards, statistics, and locality. These are the parameters to fetch the dataset information.

note

All **Dataset Details** parameters are optional.

Parameter	Description
Fetch mode	You can choose to fetch only from queried datasets that are set by default. Dremio updates details for previously queried objects in a source. Fetching from all datasets is deprecated.
Fetch every	You can choose to set the frequency to fetch dataset details in minutes, hours, days, or weeks. The default frequency to fetch dataset details is 1 hour.
Expire after	You can choose to set the expiry time of dataset details in minutes, hours, days, or weeks. The default expiry time of dataset details is 3 hours.

Privileges

You can grant privileges to specific users or roles.

(Optional) For **Privileges**, enter the user name or role name that you want to grant access to and click the **Add to Privileges** button. The added user or role is displayed in the **Users** table.

(Optional) For the users or roles in the **Users** table, toggle the green checkmark for each privilege you want to grant to the Redshift source that is being created.

Click **Save** after setting the configuration.

note

Sources containing a large number of files or tables may take longer to be added. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being added. Once complete, the source becomes accessible.

Edit Information About an Amazon Redshift Cluster Used as a Data Source

To edit a Redshift source:

- On the Datasets page, click **Databases**. A list of databases is displayed.
- Hover over the database and click the Settings !This is the icon that represents the Database settings. icon that appears next to the database.
- In the Source Settings dialog, you cannot edit the name. Editing other parameters is optional.
- Click **Save**.

Remove an Amazon Redshift Cluster Used as a Data Source

To remove a Redshift source, perform these steps:

- On the Datasets page, click **Databases**. A list of sources is displayed.
- Hover over the database and click the More (...) icon that appears next to the database.
- From the list of actions, click **Remove Source**. Confirm that you want to remove the source.

caution

Removing a source causes all downstream views dependent on objects in this source to break.

note

Sources containing a large number of files or tables may take longer to be removed. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being removed. Once complete, the source disappears.

Predicate Pushdowns

Dremio offloads these operations to Redshift.

`-*`, `+`, `-`, `/`
`<`, `<=`, `<>`, `=`, `>`, `>=`, `!=`
`AND`, `NOT`, `OR`, `||`
`ABS`
`ACOS`
`ADD_MONTHS`

`ASIN`
`ATAN`
`ATAN2`
`AVG`
`CAST`
`CBRT`
`CEIL`
`CEILING`
`CHAR_LENGTH`
`CHARACTER_LENGTH`
`CONCAT`
`COS`
`COT`
`DATE_ADD`
`DATE_SUB`
`DATE_TRUNC_CENTURY`
`DATE_TRUNC_DAY`
`DATE_TRUNC_DECADE`
`DATE_TRUNC_HOUR`
`DATE_TRUNC_MINUTE`
`DATE_TRUNC_MONTH`
`DATE_TRUNC_QUARTER`
`DATE_TRUNC_SECOND`
`DATE_TRUNC_WEEK`
`DATE_TRUNC_YEAR`
`DEGREES`
`E`
`EXP`
`EXTRACT_CENTURY`
`EXTRACT_DAY`
`EXTRACT_DECADE`
`EXTRACT_DOW`
`EXTRACT_DOY`
`EXTRACT_EPOCH`
`EXTRACT_HOUR`
`EXTRACT_MILLENNIUM`
`EXTRACT_MINUTE`
`EXTRACT_MONTH`
`EXTRACT_QUARTER`
`EXTRACT_SECOND`
`EXTRACT_WEEK`
`EXTRACT_YEAR`
`FLOOR`
`IS DISTINCT FROM`
`IS NOT DISTINCT FROM`
`IS NOT NULL`
`IS NULL`
`LAST_DAY`
`LCASE`
`LEFT`
`LENGTH`
`LIKE`
`LN`
`LOCATE`

`LOG`
`LOG10`
`LOWER`
`LPAD`
`LTRIM`
`MAX`
`MEDIAN`
`MIN`
`MOD`
`PERCENT_CONT`
`PERCENT_DISC`
`PI`
`POSITION`
`POW`
`POWER`
`RADIANS`
`REPLACE`
`REVERSE`
`RIGHT`
`ROUND`
`RPAD`
`RTRIM`
`SIGN`
`SIN`
`SQRT`
`STDDEV`
`STDDEV_POP`
`STDDEV_SAMP`
`SUBSTR`
`SUBSTRING`
`SUM`
`TAN`
`TIMESTAMPADD_DAY`
`TIMESTAMPADD_HOUR`
`TIMESTAMPADD_MINUTE`
`TIMESTAMPADD_MONTH`
`TIMESTAMPADD_QUARTER`
`TIMESTAMPADD_SECOND`
`TIMESTAMPADD_WEEK`
`TIMESTAMPADD_YEAR`
`TIMESTAMPDIFF_DAY`
`TIMESTAMPDIFF_HOUR`
`TIMESTAMPDIFF_MINUTE`
`TIMESTAMPDIFF_MONTH`
`TIMESTAMPDIFF_QUARTER`
`TIMESTAMPDIFF_SECOND`
`TIMESTAMPDIFF_WEEK`
`TIMESTAMPDIFF_YEAR`
`TO_CHAR`
`TO_DATE`
`TRIM`
`TRUNC`
`TRUNCATE`
`UCASE`

`UPPER`
`VAR_POP`
`VAR_SAMP`

Data Type Mapping

Dremio supports Amazon Redshift data types, as shown in the following table which provides the mappings from Amazon Redshift to Dremio data types. If there are additional Redshift types not listed in the table, then those types are not supported in Dremio.

Amazon Redshift Data Type	Dremio Type
BIT	BOOLEAN
BOOL	BOOLEAN
BOOLEAN	BOOLEAN
BPCHAR	VARCHAR
BYTEA	VARBINARY
CHAR	VARCHAR
CHARACTER VARYING	VARCHAR
DATE	DATE
FLOAT4	FLOAT
FLOAT8	DOUBLE
INT	INTEGER
INT2	INTEGER
INT4	INTEGER
INT8	BIGINT
INTEGER	INTEGER
NCHAR	VARCHAR
NUMERIC	DECIMAL
NVARCHAR	VARCHAR
TEXT	VARCHAR
TIMESTAMP WITH TIMEZONE	TIMESTAMP
TIMESTAMP WITHOUT TIMEZONE	TIMESTAMP
TIMESTAMP	TIMESTAMP
TIMESTAMPTZ	TIMESTAMP
TINYINT	INTEGER
VARBINARY	VARBINARY
VARCHAR	VARCHAR

Was this page helpful?

- Create a Secure Connection to a Publicly Accessible Amazon Redshift Cluster
  - Prerequisites for the VPC that You Are Using for Your Dremio project
  - Prerequisites for the VPC that You Are Using for Amazon Redshift
  - Steps to Follow in Amazon Redshift
- Create a Secure Connection to a Private Amazon Redshift Cluster
  - Prerequisite for the VPC that You Are Using for Your Dremio project
  - Prerequisite for the VPC that You Are Using for Amazon Redshift

- Steps in AWS
- Add an Amazon Redshift Cluster as a Data Source
- Prerequisites
- Steps
- Edit Information About an Amazon Redshift Cluster Used as a Data Source
- Remove an Amazon Redshift Cluster Used as a Data Source
- Predicate Pushdowns

Google BigQuery | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/databases/google-bigquery

On this page

Dremio supports connecting to Google BigQuery as an external source. The connector uses Google Service Account Keys as the authentication method. To know more about creating service account keys, see Create and delete service account keys.

Configure Google BigQuery as a Source

In the bottom-left corner of the Datasets page, click **Add Source**.

Under **Databases** in the Add Data Source dialog, select **Google BigQuery**.

General

In the **Name** field, specify the name by which you want the Google BigQuery source to appear in the list of data sources. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.

Under **Connection**, follow these steps:

In the **Host** field, specify the URL for the Google BigQuery source.

In the **Port** field, specify the port to use. The default port is `443`.

In the **Project Id** field, specifiy the Google Cloud Project ID.

Under **Authentication**, specify the service account **Client Email** and the **Service Account Key** (JSON).

note

This connector assumes that the **Service Account Key** is a JSON Web Key. For more information on Google Cloud service account credentials, please see Service account credentials.

Advanced Options

On the Advanced Options page, you can set values for these non-required options:

Option	Description
Record fetch size	Number of records to fetch at once. Set to `0` (zero) to have Dremio automatically decide. By default, this is set to `200`.
Maximum Idle Connections	The total number of connections allowed to be idle at a given time. The default number of maximum idle connections is `8`.
Connection Idle Time	The amount of time (in seconds) allowed for a connection to remain idle before the connection is terminated. The default connection idle time is `6` seconds.
Query Timeout	The amount of time (in seconds) allowed to wait for the results of a query. If this time expires, the connection being used is returned to an idle state. Set the Query timneout to 0 for no timeout. The default Query timeout is `0`.

Reflection Refresh

On the Reflection Refresh page, set the policy that controls how often Reflections are scheduled to be refreshed automatically, as well as the time limit after which Reflections expire and are removed.

Option	Description
Never refresh	Select to prevent automatic Reflection refresh, otherwise, the default is to refresh automatically.
Refresh every	How often to refresh Reflections, specified in hours, days or weeks. This option is ignored if **Never refresh** is selected.
Set refresh schedule	Specify the daily or weekly schedule.
Never expire	Select to prevent Reflections from expiring, otherwise, the default is to expire automatically after the time limit specified in **Expire after**.
Expire after	The time limit after which Reflections expire and are removed from Dremio, specified in hours, days or weeks. This option is ignored if **Never expire** is selected.

Metadata

On the Metadata page, you can configure settings to refresh metadata and handle datasets.

Dataset Handling

These are the optional **Dataset Handling** parameters.

Parameter	Description
Remove dataset definitions if underlying data is unavailable	By default, Dremio removes dataset definitions if underlying data is unavailable. Useful when files are temporarily deleted and added back in the same location with new sets of files.

Metadata Refresh

These are the optional **Metadata Refresh** parameters:

Dataset Discovery: The refresh interval for fetching top-level source object names such as databases and tables. Set the time interval using this parameter.

Parameter	Description
(Optional) **Fetch every**	You can choose to set the frequency to fetch object names in minutes, hours, days, or weeks. The default frequency to fetch object names is 1 hour.

Dataset Details: The metadata that Dremio needs for query planning such as information required for fields, types, shards, statistics, and locality. These are the parameters to fetch the dataset information.

Parameter	Description
Fetch mode	You can choose to fetch only from queried datasets that are set by default. Dremio updates details for previously queried objects in a source.
Fetch every	You can choose to set the frequency to fetch dataset details in minutes, hours, days, or weeks. The default frequency to fetch dataset details is `1` hour.
Expire after	You can choose to set the expiry time of dataset details in minutes, hours, days, or weeks. The default expiry time of dataset details is `3` hours.

Privileges

On the Privileges tab, you can grant privileges to specific users or roles. See Access Control for additional information about privileges.

note

All privileges are optional.

For **Privileges**, enter the user name or role name that you want to grant access to and click the **Add to Privileges** button. The added user or role is displayed in the **USERS/ROLES** table.

For the users or roles in the **USERS/ROLES** table, toggle the checkmark for each privilege you want to grant on the Dremio source that is being created.

Click **Save** after setting the configuration.

Update a Google BigQuery Source

To update a Google BigQuery source:

On the Datasets page, under **Databases** in the panel on the left, find the name of the source you want to update.

Right-click the source name and select **Settings** from the list of actions. Alternatively, click the source name and then the !The Settings icon at the top right corner of the page.

In the **Source Settings** dialog, edit the settings you wish to update. Dremio does not support updating the source name. For information about the settings options, see Configure Google BigQuery as a Source.

Click **Save**.

Delete a Google BigQuery Source

note

If the source is in a bad state (for example, Dremio cannot authenticate to the source or the source is otherwise unavailable), only users who belong to the ADMIN role can delete the source.

To delete a Google BigQuery source, perform these steps:

On the Datasets page, click **Sources** > **Databases** in the panel on the left.

In the list of data sources, hover over the name of the source you want to remove and click !The Settings icon to the right.

From the dropdown, select **Delete**.

In the Delete Source dialog, click **Delete** to confirm that you want to remove the source.

caution

Deleting a source causes all downstream views that depend on objects in the source to break.

note

Sources containing a large number of files or tables may take longer to be deleted. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being deleted. Once complete, the source disappears.

Predicate Pushdowns

Dremio pushes the following operations to Google BigQuery.

`*`, `+`, `-`, `/`, `%`
`<`, `<=`, `<>`, `=`, `>`, `>=`, `!=`
`AND`, `NOT`, `OR`, `||`
`ABS`
`ACOS`
`ADD_MONTHS`
`ASIN`
`ATAN`
`ATAN2`
`AVG`
`CAST`
`CEIL`
`CEILING`

`CHAR_LENGTH`
`CHARACTER_LENGTH`
`CONCAT`
`COS`
`COT`
`DATE_ADD`
`DATE_SUB`
`DATE_TRUNC_DAY`
`DATE_TRUNC_HOUR`
`DATE_TRUNC_MINUTE`
`DATE_TRUNC_MONTH`
`DATE_TRUNC_YEAR`
`DEGREES`
`E`
`EXP`
`EXTRACT_DAY`
`EXTRACT_DOW`
`EXTRACT_DOY`
`EXTRACT_HOUR`
`EXTRACT_MINUTE`
`EXTRACT_MONTH`
`EXTRACT_QUARTER`
`EXTRACT_SECOND`
`EXTRACT_WEEK`
`EXTRACT_YEAR`
`FLOOR`
`IS DISTINCT FROM`
`IS NOT DISTINCT FROM`
`IS NOT NULL`
`IS NULL`
`LAST_DAY`
`LCASE`
`LEFT`
`LENGTH`
`LIKE`
`LN`
`LOCATE`
`LOG`
`LOG10`
`LOWER`
`LPAD`
`LTRIM`
`MAX`
`MEDIAN`
`MIN`
`MOD`
`MONTH`
`PERCENT_CONT`
`PERCENT_DISC`
`PI`
`POSITION`
`POW`
`POWER`
`RADIANS`

`RAND`
`REPLACE`
`REVERSE`
`RIGHT`
`ROUND`
`RPAD`
`RTRIM`
`SIGN`
`SIN`
`SQRT`
`SUBSTR`
`SUBSTRING`
`SUM`
`TAN`
`TIMESTAMPADD_DAY`
`TIMESTAMPADD_HOUR`
`TIMESTAMPADD_MINUTE`
`TIMESTAMPADD_MONTH`
`TIMESTAMPADD_QUARTER`
`TIMESTAMPADD_SECOND`
`TIMESTAMPADD_YEAR`
`TIMESTAMPDIFF_DAY`
`TIMESTAMPDIFF_HOUR`
`TIMESTAMPDIFF_MINUTE`
`TIMESTAMPDIFF_MONTH`
`TIMESTAMPDIFF_QUARTER`
`TIMESTAMPDIFF_SECOND`
`TIMESTAMPDIFF_WEEK`
`TIMESTAMPDIFF_YEAR`
`TO_DATE`
`TRIM`
`TRUNC`
`TRUNCATE`
`UCASE`
`UPPER`
`YEAR`

Was this page helpful?

Configure Google BigQuery as a Source

General

Advanced Options

Reflection Refresh

Metadata

Privileges

Update a Google BigQuery Source

Delete a Google BigQuery Source

Predicate Pushdowns

Azure Storage | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/object-storage/azure-storage

On this page

The Dremio source connector for Azure Storage includes support for the following Azure Storage services:

- **Azure Blob Storage** is Microsoft's object storage solution for the cloud. Blob storage is optimized for storing massive amounts of unstructured data, such as text or binary data.

- **Azure Data Lake Storage Gen2** is a set of capabilities dedicated to big data analytics, built on top of Azure Blob storage. Features, such as file system semantics, directory, and file-level security and scale are combined with the low-cost, tiered storage, and high availability/disaster recovery capabilities of Azure Blob storage.

note

Soft delete for blobs is not supported for Azure Storage accounts. Soft delete should be disabled to establish a successful connection.

Zero-byte files created with Iceberg tables in Azure Storage can be safely ignored—they don't impact Dremio's functionality. To prevent these files from being created, enable **Hierarchical Namespace** on your storage container. See Azure Data Lake Storage Gen2 hierarchical namespace for instructions.

Grant Permissions

In order to use Azure Storage as a data source, the OAuth 2.0 application that you created in Azure must have appropriate permissions within the specified Azure Storage account.

To grant these permissions, you can use the built-in **Storage Blob Data Contributor** role by assigning roles for your storage account:

- In Step 3: Select the appropriate role, assign the **Storage Blob Data Contributor** role.

- In Step 4: Select who needs access, for **Assign access to**, select **User, group or service principal**. For **Select Members**, select the name of the application/service principal that you previously registered.

Add an Azure Storage Source

To add an Azure Storage source to your project:

- From the Datasets page, click **Object Storage** at the bottom of the Sources pane.

- From the top-right of the page, click **Add object storage**.

- In the Add Data Source dialog, under Object Storage, click **Azure Storage**.

The New Azure Storage Source dialog box appears, which contains the following sections: General, Advanced Options, Reflection Refresh, Metadata, Privileges.

Refer to the following for guidance on how to complete each section.

General

Section	Field/Option	Description
Name	Name	Provide a name to use for this Azure Storage source. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.
Connection	Account Name	The name of the Azure Storage account from the Azure portal app.
Encrypt connection	Enabled by default, this option encrypts network traffic with TLS. Dremio recommends encrypted connections.
Storage Connection Protocol (Driver)	Select the Azure Storage driver connection protocol you would like to use. The options are `WASBS (Legacy)` and `ABFSS (Recommended)`. ABFSS is the default based on Azure best practices.
Authentication	Shared access key	Select this option to authenticate using the Shared Access Key from the Azure portal App.
Microsoft Entra ID	Select this option to use Microsoft Entra ID credentials for authentication.

Microsoft Entra ID Authentication

To configure the Azure Storage source to use Microsoft Entra ID for authentication, provide the following values from the OAuth 2.0 application that you created in the Azure portal for this source:

- Application ID - The application (client) ID in Azure.

- OAuth 2.0 Token Endpoint - The OAuth 2.0 token endpoint (v1.0), which includes the tenant ID and is used by the application in order to get an access token or a refresh token.

- Application Secret - The secret key generated for the application.

Advanced Options

Advanced Options include:

Enable partition column inference – If a source dataset uses Parquet files and the data is partitioned on one or more columns, enabling this option will append a column named `dir<n>` for each partition level and use subfolder names for values in those columns. Dremio detects the name of the partition column, appends a column that uses that name, detects values in the names of subfolders, and uses those values in the appended column.

- **Root Path** – The root path for the Azure Storage location. The default root path is /.

- **Default CTAS Format** – Choose the default format for tables you create in Dremio, either Iceberg or Parquet.

-

Advanced Properties – Provide the custom key value pairs for the connection relevant to the source.

- Click **Add Property**.
- For Name, enter a connection property.
- For Value, enter the corresponding connection property value.

Blob Containers & Filesystem Allowlist – Add an approved Azure Storage account in the text field. You can add multiple accounts this way. When using this option to add specific accounts, you will only be able to see those accounts and not all accounts that may be available in the source.

Under Cache Options, review the following table and edit the options to meet your needs.

- **Enable local caching when possible** – Selected by default, along with asynchronous access for cloud caching. Uncheck the checkbox to disable this option. For more information about local caching, see the note below this table.

Max percent of total available cache space to use when possible – Specifies the disk quota, as a percentage, that a source can use on any single node only when local caching is enabled. The default is 100 percent of the total disk space available. You can either manually enter in a percentage in the value field or use the arrows to the far right to adjust the percentage.

Columnar Cloud Cache (C3) enables Dremio to achieve NVMe-level I/O performance on S3/ADLS by leveraging the NVMe/SSD built into cloud compute instances. C3 caches only the data required to satisfy your workloads and can even cache individual microblocks within datasets. If your table has 1,000 columns and you only query a subset of those columns and filter for data within a certain timeframe, C3 will cache only that portion of your table. By selectively caching data, C3 eliminates over 90% of S3/ADLS I/O costs, which can make up 10-15% of the costs for each query you run.

Reflection Refresh

These settings define how often Reflections are refreshed and how long data can be served before expiration. To learn more about Reflections, refer to Optimize Performance. All Reflection parameters are optional.

You can set the following refresh policies for Reflections:

- **Refresh period** – Manage the refresh period by either enabling the option to never refresh or setting a refresh frequency in hours, days, or weeks. The default frequency to refresh Reflections is every hour.

- **Expiration period** – Set the expiration period for the length of time that data can be served by either enabling the option to never expire or setting an expiration time in hours, days, or weeks. The default expiration time is set to three hours.

Metadata

Metadata settings include Dataset Handling options and Metadata Refresh options.

Dataset Handling

You can review each option provided in the following table to set up the dataset handling options to meet your needs.

- **Remove dataset definitions if underlying data is unavailable** (Default) – When selected, datasets are automatically removed if their underlying files/folders are removed from Azure Storage or if the folder or source are not accessible. If this option is *not* selected, Dremio will not remove dataset definitions if underlying files/folder are removed from Azure Storage. This may be useful if files are temporarily deleted and replaced with a new set of files.

- **Automatically format files into physical datasets when you issue queries** – When selected, Dremio will automatically promote a folder to a table using default options. If you have CSV files, especially with non-default formatting, it might be useful to *not* select this option.

Metadata Refresh

The **Metadata Refresh** parameters include **Dataset Details**, which is the metadata that Dremio needs for query planning such as information required for fields, types, shards, statistics, and locality. All metadata parameters are optional. The following table describes the parameters that fetch the dataset information.

- **Fetch mode** – You can choose to fetch only from queried datasets that are set by default. Dremio updates details for previously queried objects in a source. Fetching from all datasets is deprecated.

- **Fetch every** – You can choose to set the frequency to fetch dataset details in minutes, hours, days, or weeks. The default frequency to fetch dataset details is one day.

- **Expire after** – You can choose to set the expiry time of dataset details in minutes, hours, days, or weeks. The default expiry time of dataset details is three days.

Privileges

This section lets you grant privileges on the source to specific users or roles. All privilege parameters are optional. To learn more about how Dremio allows for the implementation of granular-level privileges, see Privileges.

To add a privilege for a user or to a role:

1. In the **Add User/Role** field, enter the user or role to which you want to apply privileges.

2. Click **Add to Privileges**. The user or role is added to the Users table.

To set privileges for a user or role:

1. In the Users table, identify the user to set privileges for and click under the appropriate column (Select, Alter, Create Table, etc.) to either enable or disable that privilege. A green checkmark indicates that the privilege is enabled.

2. Click **Save**.

Was this page helpful?

- Grant Permissions
- Add an Azure Storage Source
- General
- Advanced Options
- Reflection Refresh
- Metadata
- Privileges

Iceberg REST Catalog | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/catalogs/iceberg-rest-catalog

On this page

Dremio connects to any catalog supporting the Iceberg REST Catalog specification. See Supported Configurations below for ready-to-use configurations for many popular catalogs, including:

- Apache Polaris
- Project Nessie
- AWS Glue
- S3 Tables
- Confluent Tableflow
- Microsoft OneLake

In addition, Dremio provides specific connectors for:

- Snowflake Open Catalog
- Databricks Unity Catalog

Configure an Iceberg REST Catalog Source

To add an Iceberg REST Catalog source:

- On the Datasets page, to the right of **Sources** in the left panel, click !This is the Add Source icon..
- In the Add Data Source dialog, under **Lakehouse Catalogs**, select **Iceberg REST Catalog Source**.

General

To configure the source connection:

- For **Name**, enter a name for the source. The name you enter must be unique in the organization. Also, consider a name that is easy for users to reference. This name cannot be edited once the source is created. The name cannot exceed 255 characters and must contain only the following characters: 0-9, A-Z, a-z, underscore (_), or hyphen (-).

- For **Endpoint URI**, specify the catalog service URI.

- By default, **Use vended credentials** is turned on. This allows Dremio to connect to the catalog and receive temporary credentials for the underlying storage location. When this setting is enabled, you do not need to add storage authentication in **Advanced Options**. If you experience errors using vended credentials, turn the setting off and provide credentials via **Advanced Options** to establish a connection.

- (Optional) For **Allowed Namespaces**, add each namespace and check the option if you want to include their entire subtrees. Tables are organized into namespaces, which can be at the top level or nested within one another. Namespace names cannot contain periods or spaces.

Advanced Options

Storage Authentication

If you disabled vended credentials in the General tab, you must manually provide storage authentication.

Dremio supports Amazon S3 and Azure Storage as object storage services. For acceptable storage authentication configurations, see the following catalog properties and credentials for each service option.

- S3 Access Key

- S3 Assumed Role

- Azure Shared Key

- `fs.s3a.aws.credentials.provider` (property)

  - Value: `org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider`

  - Description: Required field for an Iceberg REST Catalog source

- `fs.s3a.access.key` (credential)

  - Value: `<your_access_key>`

  - Description: AWS access key ID used by the S3A file system. Omit for IAM role-based or provider-based authentication.

- `fs.s3a.secret.key` (credential)

  - Value: `<your_secret_key>`

  - Description: AWS secret access key used by the S3A file system. Omit for IAM role-based or provider-based authentication.

- `fs.s3a.assumed.role.arn` (property)

  - Value: `arn:aws:iam::*******:role/OrganizationAccountAccessRole`

  - Description: AWS ARN for the role to be assumed

- `fs.s3a.aws.credentials.provider` (property)

  - Value: `com.dremio.plugins.s3.store.STSCredentialProviderV1`

  - Description: Required field for an Iceberg REST Catalog source

- `fs.s3a.assumed.role.credentials.provider` (property)

  - Value: `org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider`

  - Description: Use only if the credential provider is `AssumedRoleCredentialProvider`; lists credential providers to authenticate with the STS endpoint and retrieve short-lived role credentials

- `fs.azure.account.key` (credential)

  - Value: `<your_account_key>`

  - Description: Storage account key

Cache Options

Enable local caching when possible: Selected by default. Along with asynchronous access for cloud caching, local caching can improve query performance.

Max percent of total available cache space to use when possible: Specifies the disk quota, as a percentage, that a source can use on any single executor node only when local caching is enabled. The default is 100 percent of the total disk space available on the mount point provided for caching. You can either manually enter a percentage in the value field or use the arrows to the far right to adjust the percentage.

Reflection Refresh

You can set the policy that controls how often reflections are scheduled to be refreshed automatically, as well as the time limit after which reflections expire and are removed.

Refresh Settings

Never refresh: Prevent automatic Reflection refresh. By default, Reflections refresh automatically.

Refresh every: Set the refresh interval in hours, days, or weeks. Ignored if **Never refresh** is selected.

Set refresh schedule: Specify a daily or weekly refresh schedule.

Expire Settings

Never expire: Prevent reflections from expiring. By default, reflections expire after the configured time limit.

- **Expire after**: The time limit after which reflections are removed from Dremio, specified in hours, days, or weeks. Ignored if **Never expire** is selected.

Metadata

Specifying metadata options is handled with the following settings.

Dataset Handling

- Remove dataset definitions if underlying data is unavailable (default).

Dataset Discovery

The refresh interval for fetching top-level source object names such as databases and tables.

- **Fetch every**: You can choose to set the frequency to fetch object names in minutes, hours, days, or weeks. The default frequency to fetch object names is 1 hour.

Dataset Details

The metadata that Dremio needs for query planning, such as information needed for fields, types, shards, statistics, and locality.

- **Fetch mode**: You can choose to fetch only from queried datasets. Dremio updates details for previously queried objects in a source. By default, this is set to **Only Queried Datasets**.
- **Fetch every**: You can choose to set the frequency to fetch dataset details in minutes, hours, days, or weeks. The default frequency to fetch dataset details is 1 hour.
- **Expire after**: You can choose to set the expiry time of dataset details in minutes, hours, days, or weeks. The default expiry time of dataset details is 3 hours.

Privileges

You can grant privileges to specific users or roles. To grant access to a user or role:

- For **Privileges**, enter the user name or role name to which you want to grant access and click **Add to Privileges**. The added user or role is displayed in the **USERS/ROLES** table.
- For the users or roles in the **USERS/ROLES** table, toggle the checkmark for each privilege you want to grant on the Dremio source being created.
- Click **Save** after setting the configuration.

Update an Iceberg REST Catalog Source

To update an Iceberg REST Catalog:

- On the Datasets page, under **Lakehouse Catalogs** in the panel on the left, find the name of the source you want to edit.
-

Right-click the source name and select **Settings** from the list of actions. Alternatively, click the source name and then the !The Settings icon at the top right corner of the page.

In the Source Settings dialog, edit the settings you wish to update. Dremio does not support updating the source name.

Click **Save**.

Delete an Iceberg REST Catalog Source

To delete an Iceberg REST Catalog source:

On the Datasets page, click **Sources** > **Lakehouse Catalogs** in the panel on the left.

In the list of data sources, hover over the name of the source you want to remove and right-click.

From the list of actions, click **Delete**.

In the Delete Source dialog, click **Delete** to confirm that you want to remove the source.

note

If the source is in a bad state (for example, Dremio cannot authenticate to the source or the source is otherwise unavailable), only users who belong to the ADMIN role can delete the source.

Supported Configurations

Select your catalog type below.

Apache Polaris OSS

Nessie Catalog

Glue Iceberg REST

S3 Tables

Tableflow Catalog

Microsoft OneLake

General Settings

Endpoint URI: `http://localhost:8181/api/catalog`

Use vended credentials: Unchecked

Advanced Options – Catalog Properties

`warehouse`: `polaris_oss_catalog`

`scope`: `PRINCIPAL_ROLE:ALL`

˜fs.s3a.aws.credentials.provider`: `org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider`

Advanced Options – Catalog Credentials

˜fs.s3a.access.key`: `<s3AccessKey>`

˜fs.s3a.secret.key`: `<s3SecretKey>`

˜credential`: `<client_id:client_secret>`

General Settings

Endpoint URI: `http://127.0.0.1:19120/iceberg/`

Use vended credentials: Unchecked

Advanced Options – Catalog Properties

˜warehouse`: `s3://mybucket/restcatalog/`

˜fs.s3a.aws.credentials.provider`: `org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider`

Advanced Options – Catalog Credentials

˜fs.s3a.access.key`: `<s3AccessKey>`

˜fs.s3a.secret.key`: `<s3SecretKey>`

Replace `region` with your AWS region (e.g., `us-west-2`). You will need your AWS account number and Table Bucket name.

General Settings

Endpoint URI: `https://glue.region.amazonaws.com/iceberg`

Use vended credentials: Unchecked

Advanced Options – Catalog Properties

˜warehouse`: `accountnumber:s3tablescatalog/tablebucketname`

˜rest.sigv4-enabled`: `true`

˜rest.signing-name`: `glue`

˜rest.signing-region`: `<region>`

˜fs.s3a.aws.credentials.provider`: `org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider`

˜dremio.bucket.discovery.enabled`: `false`

˜dremio.s3.region`: `<region>`

˜fs.s3a.audit.enabled`: `false`

˜fs.s3a.create.file-status-check`: `false`

Advanced Options – Catalog Credentials

˜rest.access-key-id`: `<s3AccessKey>`

˜rest.secret-access-key`: `<s3SecretKey>`

˜fs.s3a.access.key`: `<s3AccessKey>`

˜fs.s3a.secret.key`: `<s3SecretKey>`

Replace `region` with your AWS region (e.g., `us-west-2`). You will need your AWS account number and Table Bucket name.

General Settings

Endpoint URI: `https://s3tables.region.amazonaws.com/iceberg`

Use vended credentials: Unchecked

Advanced Options – Catalog Properties

˜warehouse`: `arn:aws:s3tables:region:accountnumber:bucket/tablebucketname`

˜rest.sigv4-enabled`: `true`

˜rest.signing-name`: `s3tables`

˜rest.signing-region`: `<region>`

˜fs.s3a.aws.credentials.provider`:
`org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider`

˜dremio.bucket.discovery.enabled`: `false`

˜dremio.s3.region`: `<region>`

˜fs.s3a.audit.enabled`: `false`

˜fs.s3a.create.file-status-check`: `false`

Advanced Options – Catalog Credentials

˜rest.access-key-id`: `<s3AccessKey>`

˜rest.secret-access-key`: `<s3SecretKey>`

˜fs.s3a.access.key`: `<s3AccessKey>`

˜fs.s3a.secret.key`: `<s3SecretKey>`

Note: Namespaces for the Tableflow Catalog are the Kafka clusters within your environment.

General Settings

Endpoint URI:
`https://tableflow.us-west-2.aws.confluent.cloud/iceberg/catalog/organizations/f140b886-a3e9-4e1d-ba9d-5b96b8bf4ea8/environments/env-7kn93o`

-

Allowed Namespaces: `<kafkaClusterID>`

Allowed Namespaces include their whole subtrees: Unchecked

Use vended credentials: Checked

Advanced Options – Catalog Credentials

`credential`: `<api_key:secret_key>`

Replace the placeholders inside `<...>` with your respective values. For example, a warehouse value could be `icy/icelake.Lakehouse`. The `fs.*` values are used to establish connections to the storage underlying your catalog in OneLake.

General Settings

Endpoint URI: `https://onelake.table.fabric.microsoft.com/iceberg`

Use vended credentials: Unchecked

Advanced Options – Catalog Properties

`rest.auth.type`: `oauth2`

`oauth2-server-uri`: `https://login.microsoftonline.com/<tenant_id>/oauth2/v2.0/token`

`scope`: `https://storage.azure.com/.default`

`warehouse`: `<catalog>`

`fs.azure.endpoint`: `dfs.fabric.microsoft.com`

`fs.azure.account.auth.type`: `OAuth`

`fs.azure.account.oauth2.client.endpoint`: `https://login.microsoftonline.com/<tenant_id>/oauth2/v2.0/token`

`fs.azure.account.oauth2.client.id`: `<oauth_client_id>`

Advanced Options – Catalog Credentials

`fs.azure.account.oauth2.client.secret`: `<oauth_client_secret>`

`credential`: `<oauth_client_id:oauth_client_secret>`

Was this page helpful?

Configure an Iceberg REST Catalog Source

General

Advanced Options

Reflection Refresh

Metadata

Privileges

Update an Iceberg REST Catalog Source

- Delete an Iceberg REST Catalog Source
- Supported Configurations

AWS Glue Data Catalog | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/catalogs/aws-glue-data-catalog

On this page

The AWS Glue Data Catalog is a metadata store that lets you store and share metadata in the AWS Cloud.

Supported Formats

Dremio can query data stored in S3 in file formats (including delimited, Excel (XLSX), and Parquet) and Apache Iceberg or Delta Lake table formats.

Add an AWS Glue Data Catalog

To add an AWS Glue Data Catalog to your project:

- From the Datasets page, to the right of **Sources** in the left panel, click !Add icon.
- In the Add Data Source dialog, under **Lakehouse Catalogs**, select **AWS Glue Data Catalog**.

General

To configure an AWS Glue Data Catalog source:

Name – Specify a name for the data source. You cannot change the name after the source is created. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.

- **AWS Region Selection** – Specify the region hosting the AWS Glue catalog.
- **Authentication** – Provide the role that Dremio will assume to gain access to the source:
  - Create an AWS IAM role in your AWS account that trusts Dremio.
  - Add an AWS Glue Access Policy to your custom role that provides access to your AWS Glue Data Catalog source.
  - Add the **Role ARN** to the source configuration.
- **Allowed Databases** – (Optional) The allowed databases configuration is a post-connection filter on the databases visible from AWS Glue. When selective access to the databases within AWS Glue is required, the allowed databases filter limits access

within Dremio to only the needed databases per source connection, improving data security and source metadata refresh performance.

When the allowed databases filter is empty, all databases from the AWS Glue source are visible in Dremio. When a database is added to or removed from the filter, Dremio performs an asynchronous update to expose new databases and remove databases not included in the filter. Each entry in the allowed databases filter must be a valid database name; misspelled or nonexistent databases are ignored.

Encrypted Connection – (Optional) To secure the connections between AWS Glue and Dremio, select the **Encrypt connection** checkbox.

Advanced Options

Click **Advanced Options** in the left menu sidebar.

Connection Properties – You can add key-value pairs to provide custom connection properties relevant to the source.

Click **Add Property**.

For Name, enter a connection property.

For Value, enter the corresponding connection property value.

Lake Formation – Lake Formation provides access controls and allows administrators to define security policies. Enabling this functionality and additional details on the configuration options below are described in AWS Lake Formation.

Enforce AWS Lake Formation access permissions on datasets – Dremio checks any datasets included in the AWS Glue source for the required permissions to perform queries.

Prefix to map Dremio users to AWS ARNs – Leave blank to default to the end user's username, or enter a regular expression.

Prefix to map Dremio groups to AWS ARNs – Leave blank to default to the end user's group, or enter a regular expression.

Under Cache Options:

Enable local caching when possible – Selected by default, along with asynchronous access for cloud caching. Uncheck the checkbox to disable this option.

Max percent of total available cache space to use when possible – Specifies the disk quota, as a percentage, that a source can use on any single executor node only when local caching is enabled. The default is 100 percent of the total disk space available on the mount point provided for caching. You can either manually enter a percentage in the value field or use the arrows to the far right to adjust the percentage.

Reflection Refresh

Click **Reflection Refresh** in the source settings sidebar. This section lets you manage how often Reflections are refreshed and how long data can be served before expiration. To learn more about Reflections, see Manual Reflections. All Reflection parameters are

optional.

You can set the following refresh policies for Reflections:

Refresh period – Manage the refresh period by either enabling the option to never refresh or setting a refresh frequency in hours, days, or weeks. The default frequency to refresh Reflections is every hour.

Expiration period – Set the expiration period for the length of time that data can be served by either enabling the option to never expire or setting an expiration time in hours, days, or weeks. The default expiration time is three hours.

Metadata

Click **Metadata** in the left menu sidebar. This section lets you configure settings to refresh metadata and enable other dataset options. All metadata parameters are optional.

You can configure Dataset Handling and Metadata Refresh parameters.

Dataset Handling

Remove dataset definitions if underlying data is unavailable – By default, Dremio removes dataset definitions if underlying data is unavailable. This option is for scenarios when files are temporarily deleted and added back in the same location with new sets of files.

Metadata Refresh

Dataset Discovery – The refresh interval for retrieving top-level source object names such as databases and tables. Use this parameter to set the time interval. You can choose to set the frequency to collect object names in minutes, hours, days, or weeks. The default frequency to fetch object names is one hour.

Dataset Details – The metadata that Dremio needs for query planning, such as information required for fields, types, shards, statistics, and locality.

Fetch mode – You can choose to fetch only from queried datasets. Dremio updates details for previously queried objects in a source. By default, this is set to **Only Queried Datasets**.

Fetch every – You can choose to set the frequency to fetch dataset details in minutes, hours, days, or weeks. The default frequency to fetch dataset details is one hour.

Expire after – You can choose to set the expiry time of dataset details in minutes, hours, days, or weeks. The default expiry time of dataset details is three hours.

Privileges

Click **Privileges** in the left menu sidebar. This section lets you grant privileges to specific users or roles. To learn more about how Dremio allows for the implementation of granular-level privileges, see Privileges.

To add a privilege for a user or role:

In the Add User/Role field, enter the user or role name to which you want to apply privileges.

Click **Add to Privileges**. The user or role is added to the Users table.

To set privileges for a user or role:

In the Users table, identify the user to set privileges for and click under the appropriate column (Select, Alter, Create Table, etc.) to either enable or disable that privilege. A green checkmark indicates that the privilege is enabled.

Click **Save**.

After you have connected Dremio to the AWS Glue Data Catalog, you will be able to edit the Data Catalog and remove it when it is no longer needed.

Update an AWS Glue Data Catalog Source

To update an AWS Glue Data Catalog source:

From the Datasets page, in the **Lakehouse Catalogs** section, right-click on the source and select **Settings**.

For information about these settings and guidance on the changes you can make, see Add an AWS Glue Data Catalog.

Click **Save**.

Delete an AWS Glue Data Catalog Source

To remove a Data Catalog source:

From the Datasets page, in the **Lakehouse Catalogs** section, right-click on the source and select **Delete**.

Click **Delete** again to confirm.

Add an AWS Glue Access Policy to a Custom Role

To add the required AWS Glue access policy to your custom role:

On the Roles page, click the role name. Use the **Search** field to locate the role if needed.

From the Roles page, in the Permissions section, click **Add permissions** > **Create inline policy**.

On the Create policy page, click the **JSON** tab.

Delete the current JSON policy and copy the IAM Policy Template for AWS Glue Catalog.

IAM Policy Template for AWS Glue Catalog

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AccessGlueCatalog",
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:GetTable",
                "glue:GetTableVersions",
                "glue:GetTables",
                "glue:GetConnection",
                "glue:GetConnections",
                "glue:GetDevEndpoint",
                "glue:GetDevEndpoints",
                "glue:GetUserDefinedFunction",
                "glue:GetUserDefinedFunctions",
                "glue:BatchGetPartition"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "ReadWriteGlueS3Buckets",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::aws-glue-*/*",
                "arn:aws:s3:::*/*aws-glue-*/*"
            ]
        },
        {
            "Sid": "ReadPublicGlueBuckets",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::crawler-public*",
                "arn:aws:s3:::aws-glue-*"
            ]
        },
        {
            "Sid": "ManageGlueServiceTags",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags",
```

```
                "ec2:DeleteTags"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "aws-glue-service-resource"
                    ]
                }
            },
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:instance/*"
            ]
        }
    ]
}
```

- Click **Next**.
- On the Review policy page, in the **Name** field, enter a name for the policy.
- Click **Create policy**. The policy is created and you are returned to the Roles page.

AWS Lake Formation

AWS Lake Formation provides access controls for datasets in the AWS Glue Data Catalog and is used to define security policies from a centralized location that may be shared across multiple tools. Dremio may be configured to refer to this service to verify user access to contained datasets.

Requirements

- Identity provider service set up
- (Recommended) SAML connection with AWS
- Permissions set up in Lake Formation
- AWS Glue Data Catalog connected to Dremio
- User and Group ARN prefixes specified and enabled

Lake Formation Workflow

When Lake Formation is properly configured, Dremio adheres to the following workflow each time an end user attempts to access, edit, or query datasets with managed privileges:

- Dremio enforces access control. See Configure Sources for Lake Formation for access control recommendations.

- Dremio checks each table to determine if those stored in the AWS Glue source are configured to use Lake Formation for security.

- If one or more datasets leverage Lake Formation, Dremio determines the user ARNs to use when checking against Lake Formation.

- Dremio queries Lake Formation to determine a user's access level to the datasets using the user/group ARNs.

- If the user has access to the datasets specified within the query's scope, the query proceeds.

- If the user lacks access, the query fails with a permission error.

Configure Sources for Lake Formation

Lake Formation integration is dependent on the mapping of user/group names in Dremio to the IAM user/group ARNs used by AWS.

To configure an existing or new AWS Glue Data Catalog source, you must set the following options:

- From your existing source or upon creating an **AWS Glue Data Catalog** source, navigate to the Advanced Options tab.

- Enable **Enforce AWS Lake Formation access permissions on datasets**.

- Fill in the user and group prefix settings as instructed in the Lake Formation Permissions Reference. For example, if you are using a SAML provider in AWS:

  - User prefix with SAML: `arn:aws:iam::<AWS_ACCOUNT_ID>:saml-provider/<PROVIDER_NAME_IN_AWS>:user/`

  - Group prefix with SAML: `arn:aws:iam::<AWS_ACCOUNT_ID>:saml-provider/<PROVIDER_NAME_IN_AWS>:group/`

note

Best Practice: On the Privileges tab, we recommend enabling the **Select** privilege for **All Users** to allow non-admin users to access the AWS Glue source from Dremio.

Lake Formation Cell-Level Security

Dremio supports AWS Lake Formation cell-level security with row-level access permissions based on AWS Lake Formation PartiQL expressions. If the user does not have read permissions on a column or cell, Dremio masks the data in that column or cell with a `NULL` value.

To speed up query planning, Dremio uses the AWS Lake Formation permissions cache for each table. By default, the cache is enabled and reuses previously loaded permissions for up to 3600 seconds (1 hour).

Limitations

VPC-restricted S3 buckets are not supported.

Was this page helpful?

Supported Formats

Add an AWS Glue Data Catalog

General

Advanced Options

Reflection Refresh

Metadata

Privileges

Update an AWS Glue Data Catalog Source

Delete an AWS Glue Data Catalog Source

Add an AWS Glue Access Policy to a Custom Role

AWS Lake Formation

Requirements

Lake Formation Workflow

Configure Sources for Lake Formation

Lake Formation Cell-Level Security

Limitations

Data Type Mapping

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/databases/microsoft-sql-server

On this page

Microsoft SQL Server is a database server for storing and retrieving data.

Prerequisites

Ensure that you have the following details before configuring Microsoft SQL Server as a source:

Hostname or IP address of the database

Port

Outbound port (1433 is the default port) open in your AWS or Azure security group

- Ensure that the database version is Microsoft SQL Server version 2012 or later

User Impersonation

The Microsoft SQL Server username provided in the source configuration is the default username that is used for running queries. When queries are run against Microsoft SQL Server in Dremio, users use the privileges associated with the Microsoft SQL Server username and run queries under that username.

You can change this default in Dremio by enabling user impersonation in the Advanced Options, which allows users to run queries under their own usernames and restricts their access. For example, `user_1` can run queries as `user_1` rather than `sqlsvr_svc`. Before enabling user impersonation, some setup is required in Microsoft SQL Server to allow one user to impersonate another user because the username of the user in Dremio must be the same as their username in Microsoft SQL Server and the user must be able to connect through the Microsoft SQL Server username.

To set up user impersonation, follow these steps:

- Ensure the user's username in Microsoft SQL Server matches their username in Dremio. If the usernames do not match, modify one of the usernames or create a new user account with a matching username.

- Run a GRANT IMPERSONATE command in Microsoft SQL Server to allow the user to connect through their Microsoft SQL Server username:

Example of granting impersonate privilege in Microsoft SQL Server

```
GRANT IMPERSONATE ON USER::testuser1 TO proxyuser;
```

In this example, the user can log in as `testuser1` in Dremio and in Microsoft SQL Server, and they can connect through the `proxyuser`. The `proxyuser` is the Microsoft SQL Server username provided in the source configuration.

- Log in to Dremio as a member of the ADMIN role.

- Follow the steps for Configure Microsoft SQL Server as a Source using the Microsoft SQL Server username `proxyuser` and enable **User Impersonation** in the **Advanced Options**.

- Grant source privileges to the user.

Now that you have enabled user impersonation, a user who logs in to Dremio with their username can access the Microsoft SQL Server source and its datasets according to their privileges. The user can also run queries against Microsoft SQL Server under their username.

Configure Microsoft SQL Server as a Source

Perform these steps to configure Microsoft SQL Server as a source:

- On the Datasets page, you can see a truncated list of **Sources** at the bottom-left of the page. Click **Add Source**.

Alternatively, click **Databases**. The page displays all database sources. Click the **Add database** button at the top-right of that page.

In the **Add Data Source** dialog, click **Microsoft SQL Server**.

The following section describes the source configuration tabs.

note

Sources containing a large number of files or tables may take longer to be added. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being added. Once complete, the source becomes accessible.

General

The **General** tab contains the required fields to create a Microsoft SQL Server source.

Perform these steps in the **General** tab:

In the **General** tab, for **Name**, enter a name. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.

For **Host**, enter the Microsoft SQL Server host name.

For **Port**, enter the Microsoft SQL Server port number. The default port is 1433.

(Optional) For **Database**, enter the Microsoft SQL Server database name.

(Optional) For **Encrypt connection**, enable encrypted connections to Microsoft SQL Server using SSL.

For **Authentication**, **Master Authentication** is the default and the only option. Provide the username and password of a master database user with permissions to read required objects:

For **Username**, enter your Microsoft SQL Server database username.

For **Password**, enter your Microsoft SQL Server database password.

Advanced Options

Click **Advanced Options** in the sidebar.

note

All advanced options are optional.

Advanced Option	Description
Show only the initial database used for connecting	If selected, hides the other databases that the credential has access to.
Record fetch size	Number of records to fetch at once. Set to 0 (zero) to have Dremio automatically decide. The default record fetch size is 200.
Maximum idle connections	The total number of connections allowed to be idle at a

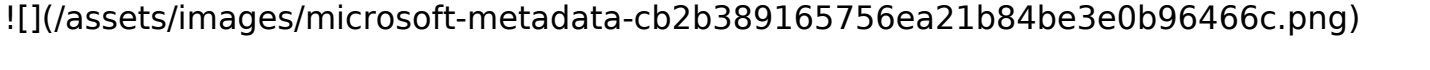
given time. The default maximum idle connections is 8. |
Connection idle time (s)	The amount of time (in seconds) allowed for a connection to remain idle before the connection is terminated. The default connection idle time is 60 seconds.
Query timeout (s)	The amount of time (in seconds) allowed to wait for the results of a query. If this time expires, the connection being used is returned to an idle state.
Enable user impersonation	Select the checkbox to allow users to run queries using their credentials rather than those of the user specified in the Authentication configuration. Some setup is required in Microsoft SQL Server to allow one user to impersonate another user. Read User Impersonation for more information.
Encryption	**Verify Server Certificate** is enabled. Add an SSL/TLS server certificate distinguished name in the text box.
Connection Properties	Custom key value pairs for the connection relevant to the source. To add a connection property, click **Add property** and add the property name and value.

Reflection Refresh

The **Reflection Refresh** tab in the sidebar allows you to set time intervals for Reflections to refresh or expire.

Metadata

You can configure settings to refresh metadata and handle datasets. Click **Metadata** in the sidebar.



You can configure Dataset Handling and Metadata Refresh parameters.

Dataset Handling

These are the **Dataset Handling** parameters.

note

All **Dataset Handling** parameters are optional.

Parameter	Description
Remove dataset definitions if underlying data is unavailable	By default, Dremio removes dataset definitions if underlying data is unavailable. Useful when files are temporarily deleted and added back in the same location with new sets of files.

Metadata Refresh

These are the **Metadata Refresh** parameters:

Dataset Discovery: The refresh interval for fetching top-level source object names

such as databases and tables. Set the time interval using this parameter.

Parameter	Description
(Optional) **Fetch every**	You can choose to set the frequency to fetch object names in minutes, hours, days, or weeks. The default frequency to fetch object names is 1 hour.

Dataset Details: The metadata that Dremio needs for query planning such as information required for fields, types, shards, statistics, and locality. These are the parameters to fetch the dataset information.

note

All **Dataset Details** parameters are optional.

Parameter	Description
Fetch mode	You can choose to fetch only from queried datasets that are set by default. Dremio updates details for previously queried objects in a source. Fetching from all datasets is deprecated.
Fetch every	You can choose to set the frequency to fetch dataset details in minutes, hours, days, or weeks. The default frequency to fetch dataset details is 1 hour.
Expire after	You can choose to set the expiry time of dataset details in minutes, hours, days, or weeks. The default expiry time of dataset details is 3 hours.

Privileges

Grant privileges on the source to specific users and roles. Read Privileges for more information about privileges.

To add source-specific privileges:

Enter a user or role name under **Add User/Role**, click to select the user or role, and click the **Add to Privileges** button. The added user or role is displayed in the **Users/Roles** table.

For each user or role listed in the **Users/Roles** table, select the checkboxes for each privilege you want to grant on the source.

Click **Save**.

Edit a Microsoft SQL Server Source

To edit a Microsoft SQL Server source:

On the Datasets page, click **Databases**. A list of databases is displayed.

Hover over the database and click Settings that appears next to the database.

In the Source Settings dialog, you cannot edit the name. Editing other parameters is optional.

Click **Save**.

Remove a Microsoft SQL Server Source

To remove a Microsoft SQL Server source, perform these steps:

On the Datasets page, click **Databases**. A list of sources is displayed.

Hover over the source and click the More (...) icon that appears next to the source.

From the list of actions, click **Remove Source**. Confirm that you want to remove the source.

caution

Removing a source causes all downstream views dependent on objects in this source to break.

note

Sources containing a large number of files or tables may take longer to be removed. During this time, the source name is grayed out and shows a spinner icon, indicating the source is being removed. Once complete, the source disappears.

Predicate Pushdowns

Dremio offloads these operations to Microsoft SQL Server.

`*`, `+`, `-`, `/`, `%`
`<`, `<=`, `<>`, `=`, `>`, `>=`, `!=`
`AND`, `NOT`, `OR`, `||`
`ABS`
`ACOS`
`ADD_MONTHS`
`ASIN`
`ATAN`
`ATAN2`
`AVG`
`CAST`
`CEIL`
`CEILING`
`CHAR_LENGTH`
`CHARACTER_LENGTH`
`CONCAT`
`COS`
`COT`
`DATE_ADD`
`DATE_SUB`
`DATE_TRUNC_DAY`
`DATE_TRUNC_HOUR`
`DATE_TRUNC_MINUTE`
`DATE_TRUNC_MONTH`
`DATE_TRUNC_YEAR`
`DEGREES`
`E`
`EXP`

`EXTRACT_DAY`
`EXTRACT_DOW`
`EXTRACT_DOY`
`EXTRACT_HOUR`
`EXTRACT_MINUTE`
`EXTRACT_MONTH`
`EXTRACT_QUARTER`
`EXTRACT_SECOND`
`EXTRACT_WEEK`
`EXTRACT_YEAR`
`FLOOR`
`IS DISTINCT FROM`
`IS NOT DISTINCT FROM`
`IS NOT NULL`
`IS NULL`
`LAST_DAY`
`LCASE`
`LEFT`
`LENGTH`
`LIKE`
`LN`
`LOCATE`
`LOG`
`LOG10`
`LOWER`
`LPAD`
`LTRIM`
`MAX`
`MEDIAN`
`MIN`
`MOD`
`MONTH`
`PERCENT_CONT`
`PERCENT_DISC`
`PI`
`POSITION`
`POW`
`POWER`
`RADIANS`
`RAND`
`REPLACE`
`REVERSE`
`RIGHT`
`ROUND`
`RPAD`
`RTRIM`
`SIGN`
`SIN`
`SQRT`
`SUBSTR`
`SUBSTRING`
`SUM`
`TAN`
`TIMESTAMPADD_DAY`

`TIMESTAMPADD_HOUR`
`TIMESTAMPADD_MINUTE`
`TIMESTAMPADD_MONTH`
`TIMESTAMPADD_QUARTER`
`TIMESTAMPADD_SECOND`
`TIMESTAMPADD_YEAR`
`TIMESTAMPDIFF_DAY`
`TIMESTAMPDIFF_HOUR`
`TIMESTAMPDIFF_MINUTE`
`TIMESTAMPDIFF_MONTH`
`TIMESTAMPDIFF_QUARTER`
`TIMESTAMPDIFF_SECOND`
`TIMESTAMPDIFF_WEEK`
`TIMESTAMPDIFF_YEAR`
`TO_DATE`
`TRIM`
`TRUNC`
`TRUNCATE`
`UCASE`
`UPPER`
`YEAR`

note

Since Microsoft SQL Server has no Boolean type, project operations that contain SQL expressions which evaluate to true or false (e.g. `SELECT username, friends > 0`), and filter operations that include Boolean literals in a filter (e.g. `WHERE currentAccount = true`) cannot be executed as pushdowns.

Data Type Mapping

Dremio supports SQL Server data types, as shown in the following table which provides the mappings from SQL Server to Dremio data types. If there are additional SQL Server types not listed in the table, then those types are not supported in Dremio.

SQL Server Data Type	Dremio Type
BIGINT IDENTITY	BIGINT
BIGINT	BIGINT
BINARY	VARBINARY
BIT	BOOLEAN
CHAR	VARCHAR
DATE	DATE
DATETIME	TIMESTAMP
DATETIME2	TIMESTAMP
DECIMAL	DECIMAL
DECIMAL() IDENTITY	DECIMAL
FLOAT	DOUBLE
IMAGE	VARBINARY
INT IDENTITY	INTEGER
INT	INTEGER
MONEY	DOUBLE
NCHAR	VARCHAR
NTEXT	VARCHAR

NUMERIC	DECIMAL
NUMERIC() IDENTITY	DECIMAL
NVARCHAR	VARCHAR
REAL	FLOAT
SMALLDATETIME	TIMESTAMP
SMALLINT IDENTITY	INTEGER
SMALLINT	INTEGER
SMALLMONEY	DOUBLE
SYSNAME	VARCHAR
TEXT	VARCHAR
TIME	TIME
TINYINT IDENTITY	INTEGER
TINYINT	INTEGER
UNIQUEIDENTIFIER	VARBINARY
VARBINARY	VARBINARY
VARCHAR	VARCHAR

Was this page helpful?

Prerequisites

User Impersonation

Configure Microsoft SQL Server as a Source

General

Advanced Options

Reflection Refresh

Metadata

Privileges

Edit a Microsoft SQL Server Source

Remove a Microsoft SQL Server Source

Predicate Pushdowns

Snowflake Open Catalog | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/catalogs/snowflake-open-catalog

On this page

Dremio supports Snowflake Open Catalog as an Iceberg catalog source. With this source connector, you can connect to and read from internal and external Snowflake Open Catalogs and write to external Snowflake Open Catalogs.

Prerequisites

You will need the catalog **Service URI**, **Client ID**, and **Client Secret** from the Snowflake setup. For a walkthrough of the Snowflake setup, refer to Query a table in Snowflake Open Catalog using a third-party engine.

Configure Snowflake Open Catalog as a Source

To add a Snowflake Open Catalog source:

- On the Datasets page, to the right of **Sources** in the left panel, click !Add Source icon.
- In the Add Data Source dialog, under **Lakehouse Catalogs**, select **Snowflake Open Catalog**.

General

To configure the source connection:

- For **Name**, enter a name for the source. The name you enter must be unique in the organization. Consider a name that is easy for users to reference. This name cannot be edited once the source is created. The name cannot exceed 255 characters and must contain only the following characters: 0-9, A-Z, a-z, underscore (_), or hyphen (-).
- Enter the name of the Snowflake Open Catalog.
- For **Endpoint URI**, specify the catalog service URI.
- In the Authentication section, use the **Client ID** and **Client Secret** created during the configuration of a service connection for the Snowflake Open Catalog.
- By default, **Use vended credentials** is enabled. This allows Dremio to connect to the catalog and receive temporary credentials to the underlying storage location. If this is enabled, you do not need to add storage authentication in Advanced Options.
- (Optional) For **Allowed Namespaces**, add each namespace and select the option if you want to include their entire subtrees. Tables are organized into namespaces, which can be at the top level or nested within one another. Namespace names cannot contain periods or spaces.

Advanced Options

Storage Authentication

If you disabled vended credentials in the General tab, you must manually provide the storage authentication.

Dremio supports Amazon S3, Azure Storage, and Google Cloud Storage (GCS) as object storage services. For acceptable storage authentication configurations, see the following catalog properties and credentials for each service option.

- S3 Access Key
- S3 Assumed Role
- Azure with Entra ID

- Azure Shared Key
- GCS Default Credentials
- GCS KeyFile

- `fs.s3a.aws.credentials.provider` (property)
  - Value: `org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider`
  - Description: Required value for a Snowflake Open Catalog source
- `fs.s3a.access.key` (credential)
  - Value: `<your_access_key>`
  - Description: AWS access key ID used by S3A file system
- `fs.s3a.secret.key` (credential)
  - Value: `<your_secret_key>`
  - Description: AWS secret key used by S3A file system
- `fs.s3a.assumed.role.arn` (property)
  - Value: `arn:aws:iam::*******:role/OrganizationAccountAccessRole`
  - Description: AWS ARN for the role to be assumed
- `fs.s3a.aws.credentials.provider` (property)
  - Value: `com.dremio.plugins.s3.store.STSCredentialProviderV1`
  - Description: Required value for a Snowflake Open Catalog source
- `fs.s3a.assumed.role.credentials.provider` (property)
  - Value: `org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider`
  - Description: Use only if the credential provider is `AssumedRoleCredentialProvider`; lists credential providers to authenticate with the STS endpoint and retrieve short-lived role credentials
- `fs.s3a.access.key` (credential)
  - Value: `<your_access_key>`
  - Description: AWS access key ID used by S3A file system
- `fs.s3a.secret.key` (credential)
  - Value: `<your_secret_key>`
  - Description: AWS secret key used by S3A file system
- `fs.azure.account.auth.type` (property)
  - Value: `OAuth`

Description: Authentication type

`fs.azure.account.oauth2.client.id` (property)

Value: `<your_client_ID>`

Description: Client ID from App Registration within Azure Portal

`fs.azure.account.oauth2.client.endpoint` (property)

Value: `https://login.microsoftonline.com/<ENTRA_ID>/oauth2/token`

Description: Microsoft Entra ID from Azure Portal

`fs.azure.account.oauth2.client.secret` (credential)

Value: `<your_client_secret>`

Description: Client secret from App Registration within Azure Portal

`fs.azure.account.key` (credential)

Value: `<your_account_key>`

Description: Storage account key

`dremio.gcs.use_keyfile` (property)

Value: `false`

Description: Required value for a Snowflake Open Catalog source

`dremio.gcs.clientId` (property)

Value: `<your_client_ID>`

Description: Client ID from GCS

`dremio.gcs.projectId` (property)

Value: `<your_project_ID>`

Description: Project ID from GCS

`dremio.gcs.clientEmail` (property)

Value: `<your_client_email>`

Description: Client email from GCS

`dremio.gcs.privateKeyId` (property)

Value: `<your_private_key_ID>`

Description: Private key ID from GCS

`dremio.gcs.use_keyfile` (property)

Value: `true`

Description: Required value for a Snowflake Open Catalog source

`dremio.gcs.privateKey` (credential)

Value: `<your_private_key>`

Description: Private key from GCS

Cache Options

Enable local caching when possible: Selected by default. Along with asynchronous access for cloud caching, local caching can improve query performance.

Max percent of total available cache space to use when possible: Specifies the disk quota, as a percentage, that a source can use on any single executor node only when local caching is enabled. The default is 100 percent of the total disk space available on the mount point provided for caching. You can either manually enter a percentage in the value field or use the arrows to the far right to adjust the percentage.

Reflection Refresh

You can set the policy that controls how often reflections are scheduled to be refreshed automatically, as well as the time limit after which reflections expire and are removed.

Refresh Settings

Never refresh: Prevent automatic Reflection refresh. By default, Reflections refresh automatically.

Refresh every: Set the refresh interval in hours, days, or weeks. Ignored if **Never refresh** is selected.

Set refresh schedule: Specify a daily or weekly refresh schedule.

Expire Settings

Never expire: Prevent Reflections from expiring. By default, Reflections expire after the configured time limit.

Expire after: The time limit after which Reflections are removed from Dremio, specified in hours, days, or weeks. Ignored if **Never expire** is selected.

Metadata

Specify metadata options with the following settings.

Dataset Handling

Remove dataset definitions if underlying data is unavailable (default).

Metadata Refresh

These are the optional **Metadata Refresh** parameters:

Dataset Discovery

The refresh interval for fetching top-level source object names such as databases and tables.

Fetch every: You can choose to set the frequency to fetch object names in minutes, hours, days, or weeks. The default frequency to fetch object names is 1 hour.

Dataset Details

The metadata that Dremio needs for query planning, such as information needed for fields, types, shards, statistics, and locality.

Fetch mode: You can choose to fetch only from queried datasets. Dremio updates details for previously queried objects in a source. By default, this is set to **Only Queried Datasets**.

Fetch every: You can choose to set the frequency to fetch dataset details in minutes, hours, days, or weeks. The default frequency to fetch dataset details is 1 hour.

Expire after: You can choose to set the expiry time of dataset details in minutes, hours, days, or weeks. The default expiry time of dataset details is 3 hours.

Privileges

You can grant privileges to specific users or roles. See Privileges for more information.

To grant access to a user or role:

For **Privileges**, enter the user name or role name to which you want to grant access and click the **Add to Privileges** button. The added user or role is displayed in the **USERS/ROLES** table.

For the users or roles in the **USERS/ROLES** table, toggle the checkmark for each privilege you want to grant on the Dremio source that is being created.

Click **Save** after setting the configuration.

Update a Snowflake Open Catalog Source

To update a Snowflake Open Catalog source:

On the Datasets page, under **Lakehouse Catalogs** in the panel on the left, find the name of the source you want to edit.

Right-click the source name and select **Settings** from the list of actions. Alternatively, click the source name and then the !Settings icon at the top right corner of the page.

In the **Source Settings** dialog, edit the settings you want to update. Dremio does not support updating the source name. For information about the settings options, see Configure Snowflake Open Catalog as a Source.

Click **Save**.

Delete a Snowflake Open Catalog Source

To delete a Snowflake Open Catalog source:

On the Datasets page, click **Sources** > **Lakehouse Catalogs** in the panel on the left.

In the list of data sources, hover over the name of the source you want to remove and right-click.

From the list of actions, click **Delete**.

In the Delete Source dialog, click **Delete** to confirm that you want to remove the source.

note

If the source is in a bad state (for example, Dremio cannot authenticate to the source or the source is otherwise unavailable), only users who belong to the ADMIN role can delete the source.

Was this page helpful?

Prerequisites

Configure Snowflake Open Catalog as a Source

General

Advanced Options

Reflection Refresh

Metadata

Privileges

Update a Snowflake Open Catalog Source

Delete a Snowflake Open Catalog Source

Databricks Unity Catalog | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/catalogs/databricks-unity-catalog

On this page

Connect to Delta Lake tables in Databricks Unity Catalog through Dremio. Unity Catalog acts as a centralized metastore for managing tables across your Databricks environment, and Dremio can query these tables when they use Delta Lake's Universal Format (UniForm).

UniForm Iceberg Required

To query Delta tables through Dremio, you must enable UniForm on your tables in Databricks. UniForm provides an Iceberg metadata layer that allows Iceberg-compatible

clients like Dremio to read Delta tables.

Requirements:

- UniForm must be enabled on your Delta tables. See Enable UniForm Iceberg in the Databricks documentation.
- Your tables must use Delta protocol minReaderVersion 2 or higher. See Features by protocol version.

UniForm tables have certain limitations. Review Limitations in the Databricks documentation before proceeding.

Configure Databricks Unity Catalog as a Source

To add a Unity Catalog source:

- On the Datasets page, to the right of **Sources** in the left panel, click !This is the Add Source icon..
- In the Add Data Source dialog, under **Lakehouse Catalogs**, select **Unity Catalog**.

General

To configure the source connection:

- For **Name**, enter a name for the source. The name you enter must be unique in the organization. Also, consider a name that is easy for users to reference. This name cannot be edited once the source is created. The name cannot exceed 255 characters and must contain only the following characters: 0-9, A-Z, a-z, underscore (_), or hyphen (-).
- For **Unity Catalog**, enter the catalog name.
- For **Endpoint URI**, specify the catalog service URI. For more information on how to find your Unity Catalog URI, see Read using the Unity Catalog Iceberg catalog endpoint.
- Under **Authentication Type**, select one of the following:

Databricks Personal Access Token: Provide a Databricks Personal Access Token (PAT). Depending on your deployment, see the following:

- AWS-based Unity deployment: See Databricks personal access tokens for service principals to create a PAT.
- Azure Databricks-based Unity deployment: See Azure Databricks personal access tokens for service principals to create a PAT.

Microsoft Entra ID: Provide the following information from your application registered in Microsoft Entra ID:

- **Application ID**: The Application ID of the application registered in Microsoft Entra ID.
- **OAuth 2.0 Token Endpoint**: The OAuth 2.0 token endpoint URL (for example, `https://login.microsoftonline.com/{tenantId}/oauth2/v2.0/token`).

- **Application Secret**: The Application Secret of the application registered in Microsoft Entra ID.

By default, **Use vended credentials** is turned on. This allows Dremio to connect to the catalog and receive temporary credentials for the underlying storage location. When this option is enabled, you don't need to add the storage authentication in **Advanced Options**.

(Optional) For **Allowed Namespaces**, add each namespace and check the option if you want to include their entire subtrees. Tables are organized into namespaces, which can be at the top level or nested within one another. Namespace names cannot contain periods or spaces.

Advanced Options

Storage Authentication

If you disabled vended credentials in the General tab, you must manually provide the storage authentication.

Dremio supports Amazon S3, Azure Storage, and Google Cloud Storage (GCS) as object storage services. For acceptable storage authentication configurations, see the following catalog properties and credentials for each service option.

- S3 Access Key

- S3 Assumed Role

- Azure Shared Key

- GCS KeyFile

- `fs.s3a.aws.credentials.provider` (property)

  - Value: `org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider`

  - Description: Required field for a Unity Catalog source

- `fs.s3a.access.key` (credential)

  - Value: `<your_access_key>`

  - Description: AWS access key ID used by S3A file system. Omit for IAM role-based or provider-based authentication.

- `fs.s3a.secret.key` (credential)

  - Value: `<your_secret_key>`

  - Description: AWS secret key used by S3A file system. Omit for IAM role-based or provider-based authentication.

- `fs.s3a.assumed.role.arn` (property)

  - Value: `arn:aws:iam::*******:role/OrganizationAccountAccessRole`

  - Description: AWS ARN for the role to be assumed

˜fs.s3a.aws.credentials.provider` (property)

Value: `com.dremio.plugins.s3.store.STSCredentialProviderV1`

Description: Required field for a Unity Catalog source

˜fs.s3a.assumed.role.credentials.provider` (property)

Value: `org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider`

Description: Use only if the credential provider is `AssumedRoleCredentialProvider`; lists credential providers to authenticate with the STS endpoint and retrieve short-lived role credentials.

˜fs.azure.account.key` (credential)

Value: `<your_account_key>`

Description: Storage account key

˜fs.AbstractFileSystem.gs.impl` (property)

Value: `com.google.cloud.hadoop.fs.gcs.GoogleHadoopFS`

Description: Required field for a Unity Catalog source

˜fs.gs.auth.service.account.enable` (property)

Value: `true`

Description: Required field for a Unity Catalog source

˜fs.gs.impl` (property)

Value: `com.google.cloud.hadoop.fs.gcs.GoogleHadoopFileSystem`

Description: Required field for a Unity Catalog source

˜fs.gs.bucket` (property)

Value: `<your_bucket>`

Description: Bucket where your data is stored for Unity Catalog in GCS

˜fs.gs.project.id` (property)

Value: `<your_project_ID>`

Description: Project ID from GCS

˜fs.gs.auth.service.account.email` (property)

Value: `<your_client_email>`

Description: Client email from GCS

˜fs.gs.auth.service.account.private.key.id` (property)

Value: `<your_private_key_id>`

Description: Private key ID from GCS

`dremio.gcs.use_keyfile` (property)

Value: `true`

Description: Required field for a Unity Catalog source

`fs.gs.auth.service.account.private.key` (credential)

Value: `<your_private_key>`

Description: Private key from GCS

Cache Options

Enable local caching when possible: Selected by default. Along with asynchronous access for cloud caching, local caching can improve query performance.

Max percent of total available cache space to use when possible: Specifies the disk quota, as a percentage, that a source can use on any single executor node only when local caching is enabled. The default is 100 percent of the total disk space available on the mount point provided for caching. You can either manually enter a percentage in the value field or use the arrows to the far right to adjust the percentage.

Reflection Refresh

You can set the policy that controls how often reflections are scheduled to be refreshed automatically, as well as the time limit after which reflections expire and are removed.

Refresh Settings

Never refresh: Prevent automatic Reflection refresh. By default, Reflections refresh automatically.

Refresh every: Set the refresh interval in hours, days, or weeks. Ignored if **Never refresh** is selected.

Set refresh schedule: Specify a daily or weekly refresh schedule.

Expire Settings

Never expire: Prevent reflections from expiring. By default, reflections expire after the configured time limit.

Expire after: The time limit after which reflections are removed from Dremio, specified in hours, days, or weeks. Ignored if **Never expire** is selected.

Metadata

Specifying metadata options is handled with the following settings.

Dataset Handling

Remove dataset definitions if underlying data is unavailable (default).

Metadata Refresh

These are the optional **Metadata Refresh** parameters:

Dataset Discovery

The refresh interval for fetching top-level source object names such as databases and tables.

Fetch every: You can choose to set the frequency to fetch object names in minutes, hours, days, or weeks. The default frequency to fetch object names is 1 hour.

Dataset Details

The metadata that Dremio needs for query planning, such as information needed for fields, types, shards, statistics, and locality.

Fetch mode: You can choose to fetch only from queried datasets. Dremio updates details for previously queried objects in a source. By default, this is set to **Only Queried Datasets**.

Fetch every: You can choose to set the frequency to fetch dataset details in minutes, hours, days, or weeks. The default frequency to fetch dataset details is 1 hour.

Expire after: You can choose to set the expiry time of dataset details in minutes, hours, days, or weeks. The default expiry time of dataset details is 3 hours.

Privileges

You can grant privileges to specific users or roles. See Privileges for additional information.

To grant access to a user or role:

For **Privileges**, enter the user name or role name that you want to grant access to and click the **Add to Privileges** button. The added user or role is displayed in the **USERS/ROLES** table.

For the users or roles in the **USERS/ROLES** table, toggle the checkmark for each privilege you want to grant on the Dremio source that is being created.

Click **Save** after setting the configuration.

Update a Databricks Unity Catalog Source

To update a Unity Catalog source:

On the Datasets page, under **Lakehouse Catalogs** in the panel on the left, find the name of the source you want to edit.

Right-click the source name and select **Settings** from the list of actions. Alternatively, click the source name, and then click the !The Settings icon at the top-right corner of

the page.

In the **Source Settings** dialog, edit the settings you wish to update. Dremio does not support updating the source name. For information about the settings options, see Configure Databricks Unity Catalog as a Source.

Click **Save**.

Delete a Databricks Unity Catalog Source

To delete a Unity Catalog source:

On the Datasets page, click **Sources** > **Lakehouse Catalogs** in the panel on the left.

In the list of data sources, hover over the name of the source you want to remove and right-click.

From the list of actions, click **Delete**.

In the Delete Source dialog, click **Delete** to confirm that you want to remove the source.

note

If the source is in a bad state (for example, Dremio cannot authenticate to the source or the source is otherwise unavailable), only users who belong to the ADMIN role can delete the source.

Was this page helpful?

UniForm Iceberg Required

Configure Databricks Unity Catalog as a Source

General

Advanced Options

Reflection Refresh

Metadata

Privileges

Update a Databricks Unity Catalog Source

Delete a Databricks Unity Catalog Source

Microsoft Azure Synapse Analytics | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/bring-data/connect/databases/microsoft-azure-synapse-analytics

On this page

Dremio supports integrations with organizations using Azure Synapse Analytics dedicated SQL pools via the external source.

Configure Microsoft Azure Synapse Analytics as a Source

On the Datasets page, to the right of **Sources** in the left panel, click !This is the Add Source icon..

In the Add Data Source dialog, under **Databases**, select **Microsoft Azure Synapse Analytics**.

General

Under **Name**, enter the name to identify the data source in Dremio. The name cannot include the following special characters: `/`, `:`, `[`, or `]`.

Connection

Under **Host**, enter the URL for your dedicated SQL pool, which typically ends in `.sql.azuresynapse.net`.

Under **Port (optional)**, enter the port required to access the data source. Port 1433 is the default for Azure Synapse dedicated pools.

Under **Database**, enter the database's name. Only this database is accessed by Dremio.

Authentication

Choose an authentication method:

No Authentication: Dremio does not attempt to provide any authentication when connecting with the SQL pool.

Master Credentials: Dremio must provide a specified username and password in order to access the SQL pool.

Username: Enter the Microsoft Azure Synapse Analytics username.

Password:

Dremio: Provide the Microsoft Azure Synapse Analytics password in plain text. Dremio stores the password.

Select the **Encrypt connection** option to encrypt the connection to Microsoft Azure Synapse Analytics. Clear the checkbox to disable encryption.

Advanced Options

The following settings control more advanced functionalities in Dremio.

Advanced Options

Record fetch size - Number of records to fetch at once. Set to 0 (zero) to have Dremio automatically decide. The default value is `200`.

Maximum idle connections - The maximum number of idle connections to keep. The default value is `8`.

Connection idle time (s) - Idle time, in seconds, before a connection is considered for closure. The default value is `60`.

Query timeout (s) - The timeout, in seconds, for query execution before it is canceled. Set to `0` for no timeout. The default value is `0`.

Encryption

Verify server certificate - Forces Dremio to verify the server's certificate using the distinguished name.

SSL/TLS server certificate distinguished name - Specifies the location for the certificate server, which must be set to `*.sql.azuresynapse.net`.

Connection Properties

Name - The unique name for any custom properties.

Value - The value associated with the custom property.

Reflection Refresh

This tab controls the frequency of Reflection refreshes or the timespan for expiration for any queries performed using this data source.

Never refresh - Prevents any query Reflections associated with this source from refreshing.

Refresh every - Sets the time interval by which Reflections for this source are refreshed. This may be set to hours, days, and weeks.

Set refresh schedule - Specify the daily or weekly schedule.

Never expire - Prevents any query Reflections associated with this source from expiring.

Expire after - Sets the time after a Reflection is created that it then expires and can no longer be used for queries. This may be set to hours, days, and weeks.

Metadata

This tab offers settings that control how dataset details are fetched and refreshed.

Dataset Handling

Remove dataset definitions if underlying data is unavailable - If this box is not checked and the underlying files under a folder are removed or the folder/source is not

accessible, Dremio does not remove the dataset definitions. This option is useful in cases when files are temporarily deleted and put back in place with new sets of files.

Metadata Refresh

Dataset Discovery

Fetch every - Specifies the time interval by which Dremio fetches object names. This can be set by minutes, hours, days, and weeks.

Dataset Details

Fetch mode - Restricts when metadata is retrieved.

Fetch every - Specifies the time interval by which metadata is fetched. This can be set by minutes, hours, days, and weeks.

Expire after - Specifies the timespan for when dataset details expire after a dataset is queried. This can be set by minutes, hours, days, and weeks.

Privileges

On the Privileges tab, you can grant privileges to specific users or roles. See Access Controls for additional information about privileges.

note

All privileges are optional.

For **Privileges**, enter the user name or role name that you want to grant access to and click the **Add to Privileges** button. The added user or role is displayed in the **USERS/ROLES** table.

For the users or roles in the **USERS/ROLES** table, toggle the checkmark for each privilege you want to grant on the Dremio source that is being created.

Click **Save** after setting the configuration.

Update a Microsoft Azure Synapse Analytics Source

To update a Microsoft Azure Synapse Analytics source:

On the Datasets page, under **Databases** in the panel on the left, find the name of the source you want to update.

Right-click the source name and select **Settings** from the list of actions. Alternatively, click the source name and then the !The Settings icon at the top right corner of the page.

In the **Source Settings** dialog, edit the settings you wish to update. Dremio does not support updating the source name. For information about the settings options, see Configure Microsoft Azure Synapse Analytics as a Source.

Click **Save**.

Delete a Microsoft Azure Synapse Analytics Source

note

If the source is in a bad state (for example, Dremio cannot authenticate to the source or the source is otherwise unavailable), only users who belong to the ADMIN role can delete the source.

To delete a Microsoft Azure Synapse Analytics source, perform these steps:

On the Datasets page, click **Sources** > **Databases** in the panel on the left.

In the list of data sources, hover over the name of the source you want to remove and right-click.

From the list of actions, click **Delete**.

In the Delete Source dialog, click **Delete** to confirm that you want to remove the source.

note

Deleting a source causes all downstream views that depend on objects in the source to break.

Predicate Pushdowns

Dremio delegates the execution of these expressions and functions to the database being queried, often dramatically improving query performance. It can also offload entire SQL queries that include one or more of these expressions and functions.

`%`, `*`, `+`, `-`, `/`
`<`, `<=`, `<>`, `=`, `>`, `>=`, `!=`
AND, NOT, OR, `||`
ABS
ACOS
ADD_MONTHS
ASIN
ATAN
ATAN2
AVG
CAST
CEIL
CEILING
CHAR_LENGTH
CHARACTER_LENGTH
CONCAT
COS
COT
DATE_ADD
DATE_SUB
DATE_TRUNC_DAY
DATE_TRUNC_HOUR
DATE_TRUNC_MINUTE
DATE_TRUNC_MONTH

DATE_TRUNC_YEAR
DEGREES
E
EXP
EXTRACT_DAY
EXTRACT_DOW
EXTRACT_DOY
EXTRACT_HOUR
EXTRACT_MINUTE
EXTRACT_MONTH
EXTRACT_QUARTER
EXTRACT_SECOND
EXTRACT_WEEK
EXTRACT_YEAR
FLOOR
IS DISTINCT FROM
IS NOT DISTINCT FROM
IS NOT NULL
IS NULL
LAST_DAY
LCASE
LEFT
LENGTH
LIKE
LN
LOCATE
LOG
LOG10
LOWER
LPAD
LTRIM
MAX
MIN
MOD
MONTH
PI
POSITION
POW
POWER
RADIANS
RAND
REPLACE
REVERSE
RIGHT
ROUND
RPAD
RTRIM
SIGN
SIN
SQRT
SUBSTR
SUBSTRING
SUM
TAN

TIMESTAMPADD_DAY
TIMESTAMPADD_HOUR
TIMESTAMPADD_MINUTE
TIMESTAMPADD_MONTH
TIMESTAMPADD_QUARTER
TIMESTAMPADD_SECOND
TIMESTAMPADD_YEAR
TIMESTAMPDIFF_DAY
TIMESTAMPDIFF_HOUR
TIMESTAMPDIFF_MINUTE
TIMESTAMPDIFF_MONTH
TIMESTAMPDIFF_QUARTER
TIMESTAMPDIFF_SECOND
TIMESTAMPDIFF_WEEK
TIMESTAMPDIFF_YEAR
TO_DATE
TRIM
TRUNC
TRUNCATE
UCASE
UPPER
YEAR

Was this page helpful?

Configure Microsoft Azure Synapse Analytics as a Source

General

Advanced Options

Reflection Refresh

Metadata

Privileges

Update a Microsoft Azure Synapse Analytics Source

Delete a Microsoft Azure Synapse Analytics Source

Predicate Pushdowns

Source: dremio-cloud-changelog.md

Changelog | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/changelog

On this page

This changelog provides a detailed record of updates and enhancements we have made to improve your Dremio Cloud experience.

November 12, 2025

What's New

General Updates

Introducing the new Dremio Cloud, featuring the built-in Open Catalog (Powered by Polaris) for seamless data governance and interoperability, Dremio's Query Engine to deliver on fast performance and Dremio's AI Agent so you can discover, analyze, and visualize data using natural language with no SQL expertise required. Try it out with our 30-day free trial by signing up at dremio.com/get-started.

DX-105656

DX-103207

DX-97048

DX-107350

DX-105441

Was this page helpful?

November 12, 2025

Source: dremio-cloud-developer.md

Developer Guide | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/developer/

On this page

You can develop applications that connect to Dremio using Arrow Flight for high-performance data access, APIs for management operations, or by integrating with development tools and frameworks.

Build Custom Applications

Use Arrow Flight and Python SDKs to build applications that connect to Dremio:

- Arrow Flight – High-performance data access for analytics applications
- Arrow Flight SQL – Standardized SQL database interactions with prepared statements
- Python – Build applications using Arrow Flight or REST APIs
- Dremio MCP Server – AI Agent integration for natural language interactions

Build Pipelines and Transformations

Use your tool of choice to build pipelines, perform transformations, and work with

Dremio:

- dbt Integration – Transform data with version control and testing
- VS Code Extension – Query Dremio from Visual Studio Code

Customize and Automate

Use APIs to power any type of customization or automation:

- API Reference – Web applications and administrative automation

For sample applications, connectors, and additional integrations, see Dremio Hub.

Supported Data Formats

For a deep dive into open table and data formats that Dremio supports, see Data Formats.

Was this page helpful?

- Build Custom Applications
- Build Pipelines and Transformations
- Customize and Automate
- Supported Data Formats

dbt | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/developer/dbt

On this page

dbt enables analytics engineers to transform their data using the same practices that software engineers use to build applications.

You can use Dremio's dbt connector `dbt-dremio` to transform data that is in data sources that are connected to a Dremio project.

Prerequisites

- Download the `dbt-dremio` package from <https://github.com/dremio/dbt-dremio>.
- Ensure that Python 3.9.x or later is installed.
- Before connecting from a dbt project to Dremio, follow these prerequisite steps:
- Ensure that you have the ID of the Dremio project that you want to use. See Obtain the ID of a Project.
- Ensure that you have a personal access token (PAT) for authenticating to Dremio. See Create a PAT.

Install

Install this package from PyPi by running this command:

Install dbt-dremio package

```
pip install dbt-dremio
```

note

`dbt-dremio` works exclusively with dbt-core versions 1.8-1.9. Previous versions of dbt-core are outside of official support.

Initialize a dbt Project

Run the command `dbt init <project_name>`.

Select `dremio` as the database to use.

Select the `dremio_cloud` option.

Provide a value for `cloud_host`.

Enter your username, PAT, and the ID of your Dremio project.

Select the `enterprise_catalog` option.

For `enterprise_catalog_namespace`, enter the name of an existing namespace within the catalog.

For `enterprise_catalog_folder`, enter the name of a folder which already exists within the namespace.

For descriptions of the configurations in the above steps, see Configurations.

After these steps are completed, you will now have a profile for your new dbt project. This file will typically be named `profiles.yml`.

This file can be edited to add multiple profiles, one for each `target` configuration of Dremio.
A common pattern is to have a `dev` target a dbt project is tested, and then another `prod` target where changes to the model are promoted after testing:

Example Profile

```
[project name]:
  outputs:
    dev:
      cloud_host: api.dremio.cloud
      cloud_project_id: 1ab23456-78c9-01d2-de3f-456g7h890ij1
      enterprise_catalog_folder: sales
      enterprise_catalog_namespace: dev
      pat: A1BCDrE2FwgH3IJkLM4123qrsT5uV6WXyza7I8bcDEFgJ9hIj0Kl1MNOPq2Rstu==
      threads: 1
      type: dremio
```

```
      use_ssl: true
      user: name@company.com
    prod:
      cloud_host: api.dremio.cloud
      cloud_project_id: 1ab23456-78c9-01d2-de3f-456g7h890ij1
      enterprise_catalog_folder: sales
      enterprise_catalog_namespace: prod
      pat: A1BCDrE2FwgH3IJkLM4123qrsT5uV6WXyza7I8bcDEFgJ9hIj0Kl1MNOPq2Rstu==
      threads: 1
      type: dremio
      use_ssl: true
      user: name@company.com
  target: dev
```

Note that the `target` value inside of the profiles.yml file can be overriden when invoking the `dbt run`.

Specify target for dbt run command

```
dbt run --target <target_name>
```

Configurations

Configuration	Required	Default Value	Description
`cloud_host`	Yes	`api.dremio.cloud`	US Control Plane: `api.dremio.cloud` EU Control Plane: `api.eu.dremio.cloud`
`cloud_project_id`	Yes	None	The ID of the Dremio project in which to run transformations.
`enterprise_catalog_namespace`	Yes	None	The namespace in which to create tables, views, etc. The dbt aliases are `datalake` (for objects) and `database` (for views).
`enterprise_catalog_folder`	Yes	None	The path in the catalog in which to create catalog objects. The dbt aliases are `root_path` (for objects) and `schema` (for views). Nested folders in the path are separated with periods.
`pat`	Yes	None	The personal access token to use for authentication. See Personal Access Tokens for instructions about obtaining a token.
`threads`	Yes	1	The number of threads the dbt project runs on.
`type`	Yes	`dremio`	Auto-populated when creating a Dremio project. Do not change this value.
`use_ssl`	Yes	`true`	The value must be `true`.
`user`	Yes	None	Email address used as a username in Dremio.

Known Limitations

Model contracts are not supported.

Was this page helpful?

Prerequisites

Install

Initialize a dbt Project

Configurations

Known Limitations

Python | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/developer/python

On this page

You can develop client applications in Python that use that use Arrow Flight and connect to Dremio's Arrow Flight server endpoint. For help getting started, try out the sample application.

Sample Python Arrow Flight Client Application

This lightweight sample Python client application connects to the Dremio Arrow Flight server endpoint. You can use token-based credentials for authentication. Any datasets in Dremio that are accessible by the provided Dremio user can be queried. You can change settings in a `.yaml` configuration file before running the client.

The Sample Python Client Application

```
"""
  Copyright (C) 2017-2021 Dremio Corporation

  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
"""
from dremio.arguments.parse import get_config
from dremio.flight.endpoint import DremioFlightEndpoint

if __name__ == "__main__":
    # Parse the config file.
    args = get_config()

    # Instantiate DremioFlightEndpoint object
    dremio_flight_endpoint = DremioFlightEndpoint(args)

    # Connect to Dremio Arrow Flight server endpoint.
    flight_client = dremio_flight_endpoint.connect()
```

```
    # Execute query
    dataframe = dremio_flight_endpoint.execute_query(flight_client)

    # Print out the data
    print(dataframe)
```

Steps

Install Python 3.

Download the Dremio Flight endpoint .whl file.

Install the `.whl` file:
Command for installing the file

```
python3 -m pip install <path to .whl file>
```

Create a local folder to store the client file and config file.

Create a file named `example.py` in the folder that you created.

Copy the contents of `arrow-flight-client-examples/python/example.py` (available here) into `example.py`.

Create a file named `config.yaml` in the folder that you created.

Copy the contents of `arrow-flight-client-examples/python/config_template.yaml` (available here) into `config.yaml`.

Uncomment the options in `config.yaml`, as needed, appending arguments after their keys (i.e., `username: my_username`). You can either delete the options that are not being used or leave them commented.

Example config file for connecting to Dremio

```
hostname: data.dremio.cloud
port: 443
pat: my_PAT
tls: true
query: SELECT * FROM Samples."samples.dremio.com"."NYC-taxi-trips" limit 10
```

Run the Python Arrow Flight Client by navigating to the folder that you created in the previous step and running this command:
Command for running the client

```
python3 example.py [-config CONFIG_REL_PATH | --config-path CONFIG_REL_PATH]
```

`[-config CONFIG_REL_PATH | config-path CONFIG_REL_PATH]`: Use either of these options to set the relative path to the config file. The default is "./config.yaml".

Config File Options

Default content of the config file

```
hostname:
port:
username:
password:
token:
query:
tls:
disable_certificate_verification:
path_to_certs:
session_properties:
engine:
```

Name	Type	Required?	Default	Description
`hostname`	string	No	`localhost`	Must be `data.dremio.cloud`.
`port`	integer	No	32010	Dremio's Arrow Flight server port. Must be `443`.
`username`	string	No	N/A	Not applicable when connecting to Dremio.
`password`	string	No	N/A	Not applicable when connecting to Dremio.
`token`	string	Yes	N/A	Either a Personal Access Token or an OAuth2 Token.
`query`	string	Yes	N/A	The SQL query to test.
`tls`	boolean	No	false	Enables encryption on a connection.
`disable_certificate_verification`	boolean	No	false	Disables TLS server verification.
`path_to_certs`	string	No	System Certificates	Path to trusted certificates for encrypted connections.
`session_properties`	list of strings	No	N/A	Key value pairs of `session_properties`. Example: ``` session_properties: - schema='Samples."samples.dremio.com"' ``` For a list of the available properties, see Manage Workloads.
`engine`	string	No	N/A	The specific engine to run against.

Was this page helpful?

Sample Python Arrow Flight Client Application

Steps

Config File Options

What You Can Do

Original URL: https://docs.dremio.com/dremio-cloud/developer/vs-code

On this page

The Dremio Visual Studio (VS) Code extension transforms VS Code into an AI-ready workspace, enabling you to discover, explore, and analyze enterprise data with natural language and SQL side by side, directly in your IDE.

What You Can Do

The VS Code extension for Dremio allows you to:

- Connect across projects – Access one or more Dremio Cloud projects from within VS Code.
- Browse & discover with context – Explore governed objects in your catalog, complete with metadata and semantic context.
- Query with intelligence – Write and run SQL with autocomplete, formatting, and syntax highlighting—or let AI agents generate SQL for you.
- Explore and get insights using natural language – Use the built-in Microsoft Copilot integration to ask questions in plain English, moving from questions to insights faster, without leaving your development environment.

Prerequisites

Before you begin, ensure you have:

- Access to a Dremio Cloud project.
- Personal access token (PAT) for connectivity to your project. For instructions, see Create a PAT.
- Visual Studio Code installed with access to the Extensions tab in the tool.

Install VS Code Extension for Dremio

1. Launch VS Code and click the Extensions button on the left navigation toolbar.
2. Search for and click on the **Dremio** extension.
3. On the Dremio extension page, click **Install**.
Once the installation is complete, you're ready to start querying Dremio from VS Code.

Connect to Dremio from VS Code

To create a connection from VS Code:

1. From the extension for Dremio, click the + button that appears when you hover over the **Connections** heading on the left panel.
2. For **Select your Dremio deployment**, select **Dremio Cloud**.
3. From the **Select a control plane** menu, select **US Control Plane** or **European Control Plane** based on where your Dremio Cloud organization is located.
4. Click **Personal Access Token** and enter the PAT that you have previously generated and press Enter.
5. The connection to your Dremio Cloud project will appear on the left under **Connections**.
6. To browse your data, click `<your_dremio_account_email>` under your connection.

Use the Copilot Integration

With Copilot in VS Code set to Agent mode, you can interact with your data through plain-language queries powered by Dremio’s semantic layer. For example, try asking:

"What curated views are available for financial analysis?"

"Summarize sales trends over the last 90 days by product category."

"Write SQL to compare revenue growth in North America vs. Europe."

Behind the scenes, Copilot taps into Dremio’s AI Semantic Layer and autonomous optimization to ensure queries run with sub-second performance — whether executed by humans or AI agents.

Was this page helpful?

Prerequisites

Install VS Code Extension for Dremio

Connect to Dremio from VS Code

Use the Copilot Integration

Dremio MCP Server | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/developer/mcp-server

On this page

The Dremio MCP Server is an open-source project that enables AI chat clients or agents to securely interact with your Dremio deployment using natural language. Connecting to the Dremio-hosted MCP Server is the fastest path to enabling external AI chat clients to work with Dremio. The Dremio-hosted MCP Server provides OAuth support, which guarantees and propagates the user identity, authentication, and authorization for all interactions with Dremio. Once connected, you can use natural language to explore and query data, perform analysis and create visualizations, create views, and analyze system performance. While you can fork the open-source Dremio MCP Server for customization or install it locally for use with a personal AI chat client account we recommend using the Dremio-hosted MCP Server available to all projects for experimentation, development and production when possible.

Configure Connectivity

Review the documentation below from AI chat client providers to verify you meet the requirements for creating custom connectors before proceeding.

Claude Custom Connector Documentation

ChatGPT Custom Connector Documentation

To configure connectivity to your Dremio-hosted MCP Server, you first need to set up a Native OAUth application and provide the redirect URLs for the AI chat client you are using.

If you are using Claude, fill in `https://claude.ai/api/mcp/auth_callback,https://claude.com/api/mcp/auth_callback,http://localhost/callback,http://localhost` as redirect URLs for the OAuth Application

If you are using ChatGPT, fill in `https://chatgpt.com/connector_platform_oauth_redirect,http://localhost` as the redirect URLs for the OAuth Application

For a custom AI chat client, you will need to speak to your administrator.

Then configure the custom connector to the Dremio-hosted MCP Server by providing the client ID from the OAuth application and the MCP endpoint for your control plane.

For Dremio instances using the US control plane, your MCP endpoint is `mcp.dremio.cloud/mcp/{project_id}`.

For Dremio instances using the European control plane, your MCP endpoint is `mcp.eu.dremio.cloud/mcp/{project_id}`.

If you are unsure of your endpoint, you can copy the **MCP endpoint** from the Project Overview page in Project Settings.

Was this page helpful?

Configure Connectivity

Arrow Flight | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/developer/arrow-flight

On this page

You can create client applications that use Arrow Flight to query data lakes at data-transfer speeds greater than speeds possible with ODBC and JDBC, without incurring the cost in time and CPU resources of deserializing data. As the volumes of data that are transferred increase in size, the performance benefits from the use of Apache Flight rather than ODBC or JDBC also increase.

You can run queries on datasets that are in the default project of a Dremio organization. Dremio is able to determine the organization and the default project from the authentication token that a Flight client uses. To query datasets in a non-default project, you can pass in the ID for the non-default project.

Dremio provides these endpoints for Arrow Flight connections:

In the US control plane: `data.dremio.cloud:443`

In the EU control plane: `data.eu.dremio.cloud:443`

All traffic within a control plane between Flight clients and Dremio go through the endpoint for that control plane. However, Dremio can scale up or down automatically to accommodate increasing and decreasing traffic on the endpoint.

Unless you pass in a different project ID, Arrow Flight clients run queries only against datasets that are in the default project or on datasources that are associated with the default project. By default, Dremio uses the oldest project in an organization as that

organization's default project.

Supported Versions of Apache Arrow

Dremio supports client applications that use Arrow Flight in Apache Arrow version 6.0.

Supported Authentication Method

Client applications can authenticate to Dremio with personal access tokens (PATs). To create a PAT, follow the steps in the section Creating a Token.

Flight Sessions

A Flight session has a duration of 120 minutes during which a Flight client interacts with Dremio. A Flight client initiates a new session by passing a `getFlightInfo()` request that does not include a Cookie header that specifies a session ID that was obtained from Dremio. All requests that pass the same session ID are considered to be in the same session.

The Flight client, having obtained a PAT from Dremio, sends a `getFlightInfo()` request that includes the query to run, the URI for the endpoint, and the bearer token (PAT). A single bearer token can be used for requests until it expires.

If Dremio is able to authenticate the Flight client by using the bearer token, it sends a response that includes FlightInfo, a Set-Cookie header with the session ID, the bearer token, and a Set-Cookie header with the ID of the default project in the organization.

FlightInfo responses from Dremio include the single endpoint for the control plane being used and the ticket for that endpoint. There is only one endpoint listed in FlightInfo responses.

Session IDs are generated by Dremio.

The client sends a `getStream()` request that includes the ticket, a Cookie header for the session ID, the bearer token, and a Cookie header for the ID of the default project.

Dremio returns the query results in one flight.

The Flight client sends another `getFlightInfo()` request using the same session ID and bearer token. If this second request did not include the session ID that Dremio sent in response to the first request, then Dremio would send a new session ID and a new session would begin.

Use a Non-Default Project

To run queries on datasets and data sources in non-default projects in Dremio, the `project_id` of the projects must be passed as a session option. The `project_id` is stored in the user session, and the server responds with a `Set-Cookie` header containing the session ID. The client must include this cookie in all subsequent requests.

To enable this behavior, a cookie middleware must be added to the Flight client. This middleware is responsible for managing cookies and will add the previous session ID to all subsequent requests.

After adding the middleware when initializing the client object, the `project_id` can be passed as a session option.

Here are examples of how to implement the `project_id` in Java and Go:

Java

Go

Pass in the ID for a non-default project in Java

```
// Create a ClientCookieMiddleware
final FlightClient.Builder flightClientBuilder = FlightClient.builder();
final ClientCookieMiddleware.Factory cookieFactory = new
ClientCookieMiddleware.Factory();
flightClientBuilder.intercept(cookieFactory);

// Add the project ID to the session options
final SetSessionOptionsRequest setSessionOptionRequest =
new SetSessionOptionsRequest(ImmutableMap.<String, SessionOptionValue>
builder().put("project_id",
SessionOptionValueFactory.makeSessionOptionValue(yourprojectid)).build());

// Close your session later once query is done
client.closeSession(new CloseSessionRequest(), bearerToken, headerCallOption);
```

Pass in the ID for a non-default project in Go

```
// Create a ClientCookieMiddleware
client, err := flight.NewClientWithMiddleware(
     net.JoinHostPort(config.Host, config.Port),
     nil,
     []flight.ClientMiddleware{flight.NewClientCookieMiddleware(),},
     grpc.WithTransportCredentials(creds),
 )
// Close the session once the query is done
defer client.CloseSession(ctx, &flight.CloseSessionRequest{})
// Add the project ID to the session options
projectIdSessionOption, err := flight.NewSessionOptionValue(projectID)
 sessionOptionsRequest := flight.SetSessionOptionsRequest{
     SessionOptions: map[string]*flight.SessionOptionValue{
         "project_id": &projectIdSessionOption,
     },
 }
response, err = client.SetSessionOptions(ctx, &sessionOptionsRequest)
```

note

In Dremio, the term catalog is sometimes used interchangeably with `project_id`. Therefore, using catalog instead of `project_id` will also work when selecting a

non-default project. We recommend using `project_id` for clarity. Throughout this documentation, we will consistently use `project_id`.

Manage Workloads

Dremio administrators can use the Arrow Flight server endpoint to manage query workloads by adding the following connection properties to Flight clients:

Flight Client Property	Description
`ENGINE`	Name of the engine to use to process all queries issued during the current session.
`SCHEMA`	The name of the schema (datasource or folder, including child paths, such as `mySource.folder1` and `folder1.folder2`) to use by default when a schema is not specified in a query.

Sample Arrow Flight Client Applications

Dremio provides sample Arrow Flight client applications in several languages at Dremio Hub.

Both sample clients use the hostname `local` and the port number `32010` by default. Make sure you override these defaults with the hostname `data.dremio.cloud` or `data.eu.dremio.cloud` and the port number `443`.

note

The Python sample application only supports connecting to the default project in Dremio.

Was this page helpful?

- Supported Versions of Apache Arrow
- Supported Authentication Method
- Flight Sessions
- Use a Non-Default Project
- Manage Workloads
- Sample Arrow Flight Client Applications

Data Formats | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/developer/data-formats/

Dremio supports the following data formats:

- File Formats
- Delimited text files, such as comma-separated values
- JSON

- ORC
- Parquet
- Table Formats
- Apache Iceberg
- Delta Lake

Was this page helpful?

Arrow Flight SQL | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/developer/arrow-flight-sql

On this page

You can use Apache Arrow Flight SQL to develop client applications that interact with Dremio. Apache Arrow Flight SQL is a new API developed by the Apache Arrow community for interacting with SQL databases. For more information about Apache Arrow Flight SQL, see the documentation for the Apache Arrow project.

Through Flight SQL, client applications can run queries, create prepared statements, and fetch metadata about the SQL dialect supported by datasource in Dremio, available types, defined tables, and more.

The requests for running queries are

- CommandExecute
- CommandStatementUpdate

The commands on prepared statements are:

- ActionClosePreparedStatementRequest: Closes a prepared statement.
- ActionCreatePreparedStatementRequest: Creates a prepared statement.
- CommandPreparedStatementQuery: Runs a prepared statement.
- CommandPreparedStatementUpdate: Runs a prepared statement that updates data.

The metadata requests that Dremio supports are:

- CommandGetDbSchemas: Lists the schemas that are in a catalog.
- CommandGetTables: Lists that tables that are in a catalog or schema.
- CommandGetTableTypes: Lists the table types that are supported in a catalog or schema. The types are Table, View, and System Table.
- CommandGetSqlInfo: Retrieves information about the datasource and the SQL dialect that it supports.

There are two clients already implemented and available in the Apache Arrow repository on GitHub for you to make use of:

- Client in C++
- Client in Java

note

At this time, you can only connect to the default project in Dremio.

Use the Sample Client

You can download and try out the sample client from <https://github.com/dremio-hub/arrow-flight-sql-clients>. Extract the content of the file and then, in a terminal window, change to the `flight-sql-client-example` directory.

Before running the sample client, ensure that you have met these prerequisites:

- Add the Samples data lake to your Dremio project by clicking the !This is the Add Source icon. icon in the **Data Lakes** section of the Datasets page.
- Ensure that Java 8 or later (up to Java 15) is installed on the system on which you run the example commands.

Command Syntax for the Sample Client

Use this syntax when sending commands to the sample client:

Sample client usage

```
Usage: java -jar flight-sql-sample-client-application.jar  -host data.dremio.cloud -port
443 ...

 -command,--command <arg>                  Method to run
 -dsv,--disableServerVerification <arg>    Disable TLS server verification.
                                           Defaults to false.
 -host,--hostname <arg>                       `data.dremio.cloud` for Dremio's US control
plane
                                             `data.eu.dremio.cloud` for Dremio's European
control plane
 -kstpass,--keyStorePassword <arg>         The jks keystore password.
 -kstpath,--keyStorePath <arg>             Path to the jks keystore.
 -pat,--personalAccessToken <arg>          Personal access token
 -port,--flightport <arg>                  443
 -query,--query <arg>                      The query to run
 -schema,--schema <arg>                    The schema to use
 -sp,--sessionProperty <arg>               Key value pairs of
                                           SessionProperty, example: -sp
                                           schema='Samples."samples.dremio.
                                           com"' -sp key=value
 -table,--table <arg>                      The table to query
 -tls,--tls <arg>                          Enable encrypted connection.
                                           Defaults to true.
```

Examples

The examples demonstrate what is returned for each of these requests:

- CommandGetDbSchemas
- CommandGetTables
- CommandGetTableTypes
- CommandExecute

note

These examples use the Flight endpoint for Dremio's US control plane: `data.dremio.cloud`. To use Dremio's European control plane, use this endpoint instead: `data.eu.dremio.cloud`.

Flight SQL Request: CommandGetDbSchemas

This command submits a `CommandGetDbSchemas` request to list the schemas in a catalog.

Example CommandGetDbSchemas request

```
java -jar flight-sql-sample-client-application.jar -tls true -host data.dremio.cloud -port 443 --pat '<personal-access-token>' -command GetSchemas
```

Example output for CommandGetDbSchemas request

```
catalog_namedb_schema_name
null        @myUserName
null        INFORMATION_SCHEMA
null        Samples
null        sys
```

Flight SQL Request: CommandGetTables

This command submits a `CommandGetTables` request to list the tables that are in a catalog or schema.

Example CommandGetTables request

```
java -jar flight-sql-sample-client-application.jar -tls true -host data.dremio.cloud -port 443 --pat '<personal-access-token>' -command GetTables -schema INFORMATION_SCHEMA
```

If you have a folder in your schema, you can escape it like this:

Example CommandGetTables request with folder in schema

```
java -jar flight-sql-sample-client-application.jar -tls true -host data.dremio.cloud -port 443 --pat '<personal-access-token>' -command GetTables -schema "Samples\
```

```
(1).samples.dremio.com"
```

Example output for CommandGetTables request

```
catalog_name  db_schema_name        table_nametable_type
null      INFORMATION_SCHEMACATALOGSSYSTEM_TABLE
null      INFORMATION_SCHEMACOLUMNS         SYSTEM_TABLE
null      INFORMATION_SCHEMASCHEMATASYSTEM_TABLE
null      INFORMATION_SCHEMATABLES          SYSTEM_TABLE
null      INFORMATION_SCHEMAVIEWS           SYSTEM_TABLE
```

Flight SQL Request: CommandGetTableTypes

This command submits a `CommandTableTypes` request to list the table types supported.

Example CommandTableTypes request

```
java -jar flight-sql-sample-client-application.jar -tls true -host data.dremio.cloud
-port 443 --pat '<personal-access-token>' -command GetTableTypes
```

Example output for CommandTableTypes request

```
table_type
TABLE
SYSTEM_TABLE
VIEW
```

Flight SQL Request: CommandExecute

This command submits a `CommandExecute` request to run a single SQL statement.

Example CommandExecute request

```
java -jar flight-sql-sample-client-application.jar -tls true -host data.dremio.cloud
-port 443 --pat '<personal-access-token>' -command Execute -query 'SELECT * FROM
Samples."samples.<Dremio-user-name>.com"."NYC-taxi-trips" limit 10'
```

Example output for CommandExecute request

```
pickup_datetimepassenger_counttrip_distance_mi fare_amount tip_amount total_amount
2013-05-27T19:15              1             1.26         7.5        0.0          8.0
2013-05-31T16:40              1             0.73         5.0        1.2          7.7
2013-05-27T19:03              2             9.23        27.5        5.0        38.33
2013-05-31T16:24              1             2.27        12.0        0.0         13.5
2013-05-27T19:17              1             0.71         5.0        0.0          5.5
2013-05-27T19:11              1             2.52        10.5       3.15        14.15
2013-05-31T16:41              5             1.01         6.0        1.1          8.6
```

```
2013-05-31T16:37              1             1.25         8.5         0.0         10.0
2013-05-31T16:39              1             2.04        10.0         1.5         13.0
2013-05-27T19:02              1            11.73        32.5        8.12        41.12
```

Code Samples

Create a FlightSqlClient

Refer to this code sample to create a `FlightClient`. Then, wrap your `FlightClient` in a `FlightSqlClient`:

Wrap FlightClient in FlightSqlClient

```
// Wraps a FlightClient in a FlightSqlClient
FlightSqlClient flightSqlClient = new FlightSqlClient(flightClient);

// Be sure to close the FlightSqlClient after using it
flightSqlClient.close();
```

Retrieve a List of Database Schemas

This code issues a CommandGetSchemas metadata request:

CommandGetSchemas metadata request

```
String catalog = null; // The catalog. (may be null)
String dbSchemaFilterPattern = null; // The schema filter pattern. (may be null)
FlightInfo flightInfo = flightSqlClient.getSchemas(catalog, dbSchemaFilterPattern);
```

Retrieve a List of Tables

This code issues a CommandGetTables metadata request:

CommandGetTables metadata request

```
String catalog = null;  // The catalog. (may be null)
String dbSchemaFilterPattern = "Samples\\ (1).samples.dremio.com";  // The schema filter
pattern. (may be null)
String tableFilterPattern = null;  // The table filter pattern. (may be null)
List<String> tableTypes = null;  // The table types to include. (may be null)
boolean includeSchema = false;  // True to include the schema upon return, false to not
include the schema.
FlightInfo   flightInfo   =   flightSqlClient.getTables(catalog,   dbSchemaFilterPattern,
tableFilterPattern, tableTypes, includeSchema);
```

Retrieve a List of Table Types That a Database Supports

This code issues a CommandGetTableTypes metadata request:

CommandGetTableTypes metadata request

```
FlightInfo flightInfo = flightSqlClient.getTableTypes();
```

Run a Query

This code issues a CommandExecute request:

CommandExecute request

```
FlightInfo flightInfo = flightSqlClient.execute("SELECT * FROM Samples.\"samples.myUserName.com\".\"NYC-taxi-trips\" limit 10");
```

Consume Data Returned for a Query

Consume data returned for query

```
FlightInfo flightInfo; // Use a FlightSqlClient method to get a FlightInfo

// 1. Fetch each partition sequentially (though this can be done in parallel)
for (FlightEndpoint endpoint : flightInfo.getEndpoints()) {

  // 2. Get a stream of results as Arrow vectors
  try (FlightStream stream = flightSqlClient.getStream(endpoint.getTicket())) {

    // 3. Iterate through the stream until the end
    while (stream.next()) {

      // 4. Get a chunk of results (VectorSchemaRoot) and print it to the console
      VectorSchemaRoot vectorSchemaRoot = stream.getRoot();
      System.out.println(vectorSchemaRoot.contentToTSVString());
    }
  }
}
```

Client Interactions with Dremio

This diagram shows an example of how an Arrow Flight SQL client initiates a Flight session and runs a query. It also shows what messages pass between the proxy at the Arrow Flight SQL endpoint, the control plane, and the execution plane.

The Flight client, having obtained a PAT from Dremio, calls the `execute()` method, which then sends a `getFlightInfo()` request. This request includes the query to run, the URI for the endpoint, and the bearer token (PAT). A single bearer token can be used for requests until it expires.

A `getFlightInfo()` request initiates a new Flight session, which has a duration of 120 minutes. A Flight session is identified by its ID. Session IDs are generated by the proxy at the Arrow Flight SQL endpoint. All requests that pass the same session ID are considered to be in the same Flight session.

The bearer token includes the user ID and the organization ID. From those two pieces of information, the proxy at the endpoint determines the project ID, and then passes the organization ID, project ID, and user ID in the `getFlightInfo()` request that it forwards to the control plane.

If the control plane is able to authenticate the Flight client by using the bearer token, it sends a response that includes FlightInfo to the proxy.

FlightInfo responses include the single endpoint for the control plane being used and the ticket for that endpoint. There is only one endpoint listed in FlightInfo responses.

The proxy at the endpoint adds the session ID and the project ID, and passes the response to the client.

The client sends a `getStream()` request that includes the ticket, a Cookie header for the session ID, the bearer token, and a Cookie header for the ID of the default project.

The proxy adds the organization ID and passes the `getStream()` request to the control plane.

The control plane devises the query plan and sends that to the execution plane.

The execution plane runs the query and sends the results to the control plane in one flight.

The control plane passes the results to the proxy.

The proxy passes the results to the client.

Was this page helpful?

Use the Sample Client

Command Syntax for the Sample Client

Examples

Code Samples

Create a FlightSqlClient

Retrieve a List of Database Schemas

Retrieve a List of Tables

Retrieve a List of Table Types That a Database Supports

Run a Query

Consume Data Returned for a Query

Client Interactions with Dremio

Apache Iceberg | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/developer/data-formats/iceberg

On this page

Apache Iceberg enables Dremio to provide powerful, SQL database-like functionality on data lakes using industry-standard SQL commands. Dremio currently supports Iceberg v2 tables, offering a solid foundation for building and managing data lakehouse tables. Certain features, such as Iceberg native branching and tagging, and the UUID data type, are not yet supported.

For a deeper dive into Apache Iceberg, see:

- Apache Iceberg: An Architectural Look Under the Covers
- What is Apache Iceberg?

Benefits of Iceberg Tables

Iceberg tables offer the following benefits over other formats traditionally used in the data lake, including:

- Schema evolution**:** Supports add, drop, update, or rename column commands with no side effects or inconsistency.
- Partition evolution**:** Facilitates the modification of partition layouts in a table, such as data volume or query pattern changes without needing to rewrite the entire table.
- **Transactional consistency:** Helps users avoid partial or uncommitted changes by tracking atomic transactions with atomicity, consistency, isolation, and durability (ACID) properties.
- **Increased performance:** Ensures data files are intelligently filtered for accelerated processing via advanced partition pruning and column-level statistics.
- **Time travel:** Allows users to query any previous versions of the table to examine and compare data or reproduce results using previous queries.
- Automatic optimization**:** Optimize query performance to maximize the speed and efficiency with which data is retrieved.
- **Version rollback:** Corrects any discovered problems quickly by resetting tables to a known good state.

Clustering

Clustered Iceberg tables in Dremio makes use of Z-Ordering to provide a more intuitive data layout with comparable or better performance characteristics to Iceberg partitioning.

Iceberg clustering sorts individual records in data files based on the clustered columns provided in the `CREATE TABLE` or `ALTER TABLE` statement. The data file level clustering of data allows Parquet metadata to be used in query planning and execution to reduce the amount of data scanned as part of the query. In addition, clustering

eliminates common problems with partitioned data, such as over-partitioned tables and partition skew.

Clustering provides a general-purpose file layout that enables both efficient reads and writes. However, you may not see immediate benefits from clustering if the tables are too small.

A common pattern is to choose clustered columns that are either primary keys of the table or commonly used for query filters. These column choices will effectively filter the working dataset, thereby improving query times. Clustered columns are ordered in precedence of filtering or cardinality with the most commonly queried columns of highest cardinality first.

Supported Data Types for Clustered Columns

Dremio Iceberg clustering supports clustered columns of the following data types:

˜DECIMAL`

˜INT`

˜BIGINT`

˜FLOAT`

˜DOUBLE`

˜VARCHAR`

˜VARBINARY`

˜DATE`

˜TIME`

˜TIMESTAMP`

Automated table maintenance eliminates the need to run optimizations for clustered Iceberg tables manually, although if using manual optimization, its behavior differs based on whether or not tables are clustered.

For clustered tables, `OPTIMIZE TABLE` incrementally reorders data to achieve the optimal data layout and manages file sizes. This mechanism may take longer to run on newly loaded or unsorted tables. Additionally, you may be required to run multiple `OPTIMIZE TABLE` SQL commands to converge on an optimal file layout.

For unclustered tables, `OPTIMIZE TABLE` combines small files or splits large files to achieve an optimal file size, reducing metadata overhead and runtime file open costs.

CTAS Behavior and Clustering

When running a `CREATE TABLE AS` statement with clustering, the data is written in an unordered way. For the best performance, you should run an `OPTIMIZE TABLE` SQL command after creating a table using a `CREATE TABLE AS` statement.

Iceberg Table Management

Learn how to manage Iceberg tables in Dremio with supported Iceberg features such as expiring snapshots and optimizing tables.

Vacuum

Each write to an Iceberg table creates a snapshot of that table, which is a timestamped version of the table. As snapshots accumulate, data files that are no longer referenced in recent snapshots take up more and more storage. Additionally, the more snapshots a table has, the larger its metadata becomes. You can expire older snapshots to delete the data files that are unique to them and to remove them from table metadata. It is recommended that you expire snapshots regularly. For the SQL command to expire snapshots, see `VACUUM TABLE`.

Sometimes failed SQL commands may leave orphan data files in the table location that are no longer referenced by any active snapshot of the table. You can remove orphan files in the table location by running `remove_orphan_files`. See `VACUUM TABLE` for details.

Optimization

Dremio provides automatic optimization, which automatically maintains Iceberg tables in the Open Catalog using a dedicated engine configured by Dremio. However, for immediate optimization, you can use the `OPTIMIZE TABLE` SQL command and route jobs to specific engines in your project by creating a routing rule with the `query_label()` condition and the `OPTIMIZATION` label. For more information, see Workload Management.

When optimizing tables manually, you can use:

`FOR PARTITIONS` to optimize selected partitions.

`MIN_INPUT_FILES` to consider the minimum number of qualified files needed for compaction. Delete files count towards determining whether the minimum threshold is reached.

Iceberg Catalogs in Dremio

The Apache Iceberg table format uses an Iceberg catalog service to track snapshots and ensure transactional consistency between tools. For more information about how Iceberg catalogs and tables work together, see Iceberg Catalog.

note

Currently, Dremio does not support the Amazon DynamoDB nor JDBC catalogs. For additional information on limitations of Apache Iceberg as implemented in Dremio, see Limitations.

The catalog is the source of truth for the current metadata pointer for a table. You can use Dremio's Open Catalog as a catalog for all your tables. You can also add external Iceberg catalogs as a source in Dremio, which allows you to work with Iceberg tables that are not cataloged in Dremio's Open Catalog.The list of Iceberg catalogs that can be

added as a source can be found here:

- AWS Glue Data Catalog
- Iceberg REST Catalog
- Snowflake Open Catalog
- Unity Catalog

Once a table is created with a specific catalog, you must continue using that same catalog to access the table. For example, if you create a table using AWS Glue as the catalog, you cannot later access that table by adding its S3 location as a source in Dremio. You must add the AWS Glue Data Catalog as a source and access the table through it.

Rollbacks

When you modify an Iceberg table using data definition language (DDL) or data manipulation language (DML), each change creates a new snapshot in the table's metadata. The Iceberg catalog tracks the current snapshot through a root pointer.
You can use the `ROLLBACK TABLE` SQL command to roll back a table by redirecting this pointer to an earlier snapshot—useful for undoing recent data errors. Rollbacks can target a specific timestamp or snapshot ID.
When you perform a rollback, Dremio creates a new snapshot identical to the selected one. For example, if a table has snapshots (1) `first_snapshot`, (2) `second_snapshot`, and (3) `third_snapshot`, rolling back to `first_snapshot` restores the table to that state while preserving all snapshots for time travel queries.

SQL Command Compatibility

Dremio supports running most combinations of concurrent SQL commands on Iceberg tables. To take a few examples, two `INSERT` commands can run concurrently on the same table, as can two `SELECT` commands, or an `UPDATE` and an `ALTER` command.

However, Apache Iceberg's Serializable Isolation level with non-locking table semantics can result in scenarios in which write collisions occur. In these circumstances, the SQL command that finishes second fails with an error. Such failures occur only for a subset of combinations of two SQL commands running concurrently on a single Iceberg table.

This table shows which types of SQL commands can and cannot run concurrently with other types on a single Iceberg table:

- Y: Running these two types of commands concurrently is supported.
- N: Running these two types of commands concurrently is not supported. The second command to complete fails with an error.
- D: Running two `OPTIMIZE` commands concurrently is supported if they run against different table partitions.

!SQL commands that cause concurrency conflicts

Table Properties

The following Apache Iceberg table properties are supported in Dremio. You can use these properties to configure aspects of Apache Iceberg tables:

Property	Description	Default
commit.manifest.target-size-bytes	The target size when merging manifest files.	`8 MB`
commit.status-check.num-retries	The number of times to check whether a commit succeeded after a connection is lost before failing due to an unknown commit state.	`3`
compatibility.snapshot-id-inheritance.enabled	Enables committing snapshots without explicit snapshot IDs.	`false` (always `true` if the format version is > 1)
format-version	The table's format version defined in the Spec. Options: `1` or `2`	`2`
history.expire.max-snapshot-age-ms	The maximum age (in milliseconds) of snapshots to keep as expiring snapshots.	`432000000` (5 days)
history.expire.min-snapshots-to-keep	The default minimum number of snapshots to keep as expiring snapshots.	`1`
write.delete.mode	The table's method for handling row-level deletes. See Row-Level Changes on the Lakehouse: Copy-On-Write vs. Merge-On-Read in Apache Iceberg for more information on which mode is best for your table's DML operations. Options: `copy-on-write` or `merge-on-read`	`copy-on-write`
write.merge.mode	The table's method for handling row-level merges. See Row-Level Changes on the Lakehouse: Copy-On-Write vs. Merge-On-Read in Apache Iceberg for more information on which mode is best for your table's DML operations. Options: `copy-on-write` or `merge-on-read`	`copy-on-write`
write.metadata.compression-codec	The Metadata compression codec. Options: `none` or `gzip`	`none`
write.metadata.delete-after-commit.enabled	Controls whether to delete the oldest tracked version metadata files after commit.	`false`
write.metadata.metrics.column.col1	Metrics mode for column `col1` to allow per-column tuning. Options: `none`, `counts`, `truncate(length)`, or `full`	(not set)
write.metadata.metrics.default	Default metrics mode for all columns in the table. Options: `none`, `counts`, `truncate(length)`, or `full`	`truncate(16)`
write.metadata.metrics.max-inferred-column-defaults	Defines the maximum number of top-level columns for which metrics are collected. The number of stored metrics can be higher than this limit for a table with nested fields.	`100`
write.metadata.previous-versions-max	The maximum number of previous version metadata files to keep before deleting after commit.	`100`
write.parquet.compression-codec	The Parquet compression codec. Options: `zstd`, `gzip`, `snappy`, or `uncompressed`	`zstd`
write.parquet.compression-level	The Parquet compression level. Supported for `gzip` and `zstd`.	`null`
write.parquet.dict-size-bytes	The Parquet dictionary page size (in bytes).	`2097152` (2 MB)
write.parquet.page-row-limit	The Parquet page row limit.	`20000`
write.parquet.page-size-bytes	The Parquet page size (in bytes).	`1048576` (1 MB)
write.parquet.row-group-size-bytes	Parquet row group size. Dremio uses this property as a target file size since it writes one row-group per Parquet file. Ignores the `store.parquet.block-size` and `dremio.iceberg.optimize.target_file_size_mb` support keys.	`134217728` (128 MB)
write.summary.partition-limit	Includes partition-level summary stats in snapshot

summaries if the changed partition count is less than this limit. | `0` |
| write.update.mode | The table's method for handling row-level updates. See Row-Level Changes on the Lakehouse: Copy-On-Write vs. Merge-On-Read in Apache Iceberg for more information on which mode is best for your table's DML operations. Options: `copy-on-write` or `merge-on-read` | `copy-on-write` |

You can configure these properties when you create or alter Iceberg tables.

Dremio uses the Iceberg default value for table properties that are not set. See Iceberg's documentation for the full list of table properties. To view the properties that are set for a table, use the SQL command `SHOW TBLPROPERTIES`.

In cases where Dremio has a support key for a feature covered by a table property, Dremio uses the table property instead of the support key.

Limitations

The following are limitations with Apache Iceberg as implemented in Dremio:

- Only Parquet file formats are currently supported. Other formats (such as ORC and Avro) are not supported at this time.

- Amazon DynamoDB and JDBC catalogs are currently not supported.

- Unable to use DynamoDB as a lock manager with the Hadoop catalog on Amazon S3.

- Dremio caches query plans for recently executed statements to improve query performance. However, running a rollback query using a snapshot ID invalidates all cached query plans that reference the affected table.

- If a table is running DML operations when a rollback query using a snapshot ID executes, the DML operations can fail to complete because the current snapshot ID has changed to a new value due to the rollback query. However, `SELECT` queries that are in the midst of executing can be completed.

- Clustering keys must be columns in the table. Transformations are not supported.

- You can run only one optimize query at a time on the selected Iceberg table partition.

- The optimize functionality does not support sort ordering.

Related Topics

- Automatic Optimization – Learn how Dremio optimizes Iceberg tables automatically.

- Load Data Into Tables - Load data from CSV, JSON, or Parquet files into existing Iceberg tables.

- SQL Commands – See the syntax of the SQL commands that Dremio supports for Iceberg tables.

Was this page helpful?

- Benefits of Iceberg Tables

- Clustering

- Iceberg Table Management
- Vacuum
- Optimization
- Iceberg Catalogs in Dremio
- Rollbacks
- SQL Command Compatibility
- Table Properties
- Limitations
- Related Topics

Parquet | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/developer/data-formats/parquet

On this page

This topic provides general information and recommendations for Parquet files.

Read Parquet Files

Dremio's vectorized Parquet file reader improves parallelism on columnar data, reduces latencies, and enables more efficient resource and memory usage.

Dremio supports off-heap memory buffers for reading Parquet files.

Dremio supports file compression with `snappy`, `gzip`, and `zstd` for reading Parquet files.

Parquet Limitations

Take into consideration the following limitations when generating and configuring Parquet files. Failure to adhere to these restrictions may cause errors to trigger when using Parquet files with Dremio.

Maximum nested levels are restricted to 16. Multiple structs may be defined up to a total nesting level of 16. Exceeding this results in a failed query.

Maximum allowable elements in an array are restricted to 128. The maximum allowable number of elements in an array may not exceed this quantity. Additional elements beyond the allowed 128 results in a query failure.

Maximum footer size is restricted to 16MB. The footer consists of metadata. This includes information about the version of the format, the schema, extra key-value pairs, and metadata for columns in the file. When the footer exceeds this size, a query failure occurs.

Recommended Configuration

When using other tools to generate Parquet files for consumption in Dremio, we recommend the following configuration:

Type	Implementation
Row Groups	Implement your row groups using the following: A single row group per file, and a target of 1MB-25MB column stripes for most datasets (ideally). By default, Dremio uses 256 MB row groups for the Parquet files that it generates.
Pages	Implement your pages using the following: Snappy compression, and a target of ~100K page size. Use a recent Parquet library to avoid bad statistics issues.
Statistics	Use a recent Parquet library to avoid bad statistics issues.

Was this page helpful?

- Read Parquet Files
- Parquet Limitations
- Recommended Configuration

Delta Lake | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/developer/data-formats/delta-lake

On this page

Delta Lake is an open-source table format that provides transactional consistency and increased scale for datasets by creating a consistent definition of datasets and including schema evolution changes and data mutations. With Delta Lake, updates to datasets are viewed in a consistent manner across all applications consuming the datasets, and users are kept from seeing inconsistent views of data during transformations. Consistent and reliable views of datasets in a data lake are maintained even as the datasets are updated and modified over time.

Data consistency for a dataset is enabled through the creation of a series of manifest files which define the schema and data for a given point in time, as well as a transaction log that defines an ordered record of every transaction on the dataset. By reading the transaction log and manifest files, applications are guaranteed to see a consistent view of data at any point in time, and users can ensure intermediate changes are invisible until a write operation is complete.

Delta Lake provides the following benefits:

- Large-scale support: Efficient metadata handling enables applications to readily process petabyte-sized datasets with millions of files
- Schema consistency: All applications processing a dataset operate on a consistent and shared definition of the dataset metadata such as columns, data types, partitions.

Supported Data Sources

The Delta Lake table format is supported with the following sources in the Parquet file

format:

- Amazon S3
- AWS Glue Data Catalog

Analyze Delta Lake Datasets

Dremio supports analyzing Delta Lake datasets on the sources listed above through a native and high-performance reader. It automatically identifies which datasets are saved in the Delta Lake format, and imports table information from the Delta Lake manifest files. Dataset promotion is seamless and operates the same as any other data format in Dremio, where users can promote file system directories containing a Delta Lake dataset to a table manually or automatically by querying the directory. When using Delta Lake format, Dremio supports datasets of any size including petabyte-sized datasets with billions of files.

Dremio reads Delta Lake tables created or updated by another engine, such as Spark and others, with transactional consistency. Dremio automatically identifies tables that are in the Delta Lake format and selects the appropriate format for the user.

Refresh Metadata

Metadata refresh is required to query the latest version of a Delta Lake table. You can wait for an automatic refresh of metadata or manually refresh it.

Example of Querying a Delta Lake Table

Perform the following steps to query a Delta Lake table:

- In Dremio, open the **Datasets** page.
- Go to the data source that contains the Delta Lake table.
- If the data source is not an AWS Glu Data Catalog, follow these steps:
- Hover over the row for the table and click !The Format Folder icon to the right. Dremio automatically identifies tables that are in the Delta Lake format and selects the appropriate format.
- Click **Save**.
- If the data source is an AWS Glue Data Catalog, hover over the row for the table and click !The Go To Table icon to the right.
- Run a query on the Delta Lake table to see the results.
- Update the table in the data source.
- Go back to the **Datasets** UI and wait for the table metadata to refresh or manually refresh it using the syntax below.

Syntax to manually refresh table metadata

```
ALTER TABLE `<path_of_the_dataset>`
```

```
REFRESH METADATA
```

The following statement shows refreshing metadata of a Delta Lake table.

Example command to manually refresh table metadata

```
ALTER TABLE s3."data.dremio.com".data.deltalake."tpcds10_delta"."call_center"
REFRESH METADATA
```

- Run the previous query on the Delta Lake table to retrieve the results from the updated Delta Lake table.

Limitations

- Creating Delta Lake tables is not supported.
- DML operations are not supported.
- Incremental Reflections are not supported.
- Metadata refresh is required to query the latest version of a Delta Lake table.
- Time travel or data versioning is not supported.
- Only Delta Lake tables with minReaderVersion 1 or 2 can be read. Column Mapping is supported with minReaderVersion 2.

Was this page helpful?

Source: dremio-cloud-explore-analyze.md

Explore and Analyze Your Data | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/

Dremio enables data engineers and data analysts to explore and analyze data regardless of the location of the data or the data format. You can explore and analyze using Dremio's console or use your tool of choice.

Discover Data: Dremio offers several ways of discovering your data. Semantic search, metadata, and lineage help you find the right data quickly. Enrich the semantics of your data with wikis and labels so that AI agents and users can quickly identify relevant data. See Discover Data.

- **Explore Using AI Agent**: Use natural language to discover datasets, analyze data, and generate visualizations without writing SQL. The AI Agent uses the AI Semantic Layer to provide relevant responses and ensures governed access to data. See Explore Using AI Agent.

- **Connect Client Applications**: Connect your preferred BI and data tools directly to Dremio to analyze data live without extracts or replication. See Connect Client Applications.

Was this page helpful?

Explore Using AI Agent | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/ai-agent

On this page

Understanding your data quickly using Dremio's AI Agent, an interface built into the Dremio console that allows users to converse with their data using natural language.

The AI Agent accesses data and entities that the logged in user has privileges on to address the prompt.

The AI Agent is currently optimized for the following tasks:

- **Discover and Explore**: Learn about the data that is available to you to answer your business question.

- **Analyze**: Ask questions using business terms using natural language and get insights instantly. The AI Agent goes beyond basic analysis to detect patterns in the data and return actionable insights.

- **Visualize**: Quickly visualize the patterns and trends in your data within the Dremio console.

- **Explain and Optimize SQL**: Ask the agent to review SQL queries, identify bottlenecks, and suggest optimizations. For more information on this, see **Explain SQL**.

- **Analyze and Improve Job Performance**: Ask the agent to review past jobs, identify performance issues, and suggest ways to improve them. For more information on this, see **Explain Job**.

As Dremio's AI Agent reasons through your questions and requirements, you're able to see the actions it is taking directly in the interface so you can review, audit, and understand how the response is generated.

Generative AI can make mistakes; therefore, you should verify all output.

Use Dremio's AI Agent

To use Dremio's AI Agent, you can access it by:

- Typing a question into the chat on the homepage in the Dremio console.

- Use the shortcut keys ⌘+Shift+G on a Mac or Ctrl+Shift+G on Windows to open the

agent.

To use the AI Agent, you need to be granted CALL MODEL on the default model provider.

Discover and Explore

Dremio's AI Agent will help you discover available data and provide a detailed breakdown of schema, as well as offer guidance on what tables and views you may want to use. The AI Agent will use wikis and labels as well as perform sampling or other simple analysis on the datasets to determine relevance and interesting patterns. The more detailed the question, the better the insight that the AI Agent can provide.

Okay Prompt	Great Prompt
What tables can I use?	Which tables or views have customer location data?
How can I analyze time series data?	Which tables or views can I use to do a time series analysis on customer activity?
What is the `customer_activity` table?	How is the `customer_activity` table structured, and what other tables does it relate to?

Analyze

Dremio's AI Agent will write and execute SQL on your behalf based on your natural language input and the information available from the semantic layer. From within the chat, you can further audit the SQL by expanding the tool calls in the chat window.

Okay Prompt	Great Prompt
I want to see analysis of customer activity	I want to see an analysis of customer purchase activity by region, by customer type for each month of the year.
Which customers are the most valuable?	Which customers have spent the most with us over the lifetime of the relationship?

Visualize

Dremio's AI Agent will visualize insights on your behalf based on your natural language input. The details you provide, including the chart type, axis requirements, grouping, or trendlines, will be considered by the LLM. The visualization will be accompanied by insights that serve as a narrative for the chart that the AI Agent generated. Once a visualization has been created, you can toggle between the visualization and a grid representation of the data that is used to back the visualization.

The AI Agent can return the following types of visualizations: Bar, Line, Area, Scatter, Pie, Heatmap

Okay Prompt	Great Prompt
Visualize the data	Visualize the data as a bar chart with month on the x asis and sum of purchase value as the y axis
Create a visual trendline showing me the activity	Create a visualization with a trendline showing customer activity by month?

Related Topics

Jobs – See the **Explain SQL** and **Explain Job** options on the Jobs page.

Data Privacy – Learn more about Dremio's data privacy practices.

Was this page helpful?

Use Dremio's AI Agent

Discover and Explore

Analyze

Visualize

Related Topics

Discover Data | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/discover

On this page

Dremio provides multiple ways to discover and explore your data. Whether you prefer browsing catalogs, using semantic search, or leveraging the AI Agent, you can find the data you need quickly and efficiently.

Data discovery capabilities in Dremio include the following:

Dremio's AI Agent for natural language data discovery and exploration

AI-enabled semantic search for objects and entities

Catalog exploration in the Dremio console

Metadata cards for dataset details

Discover using Dremio's AI Agent

Dremio's AI Agent makes it easy to find relevant data using natural language and your business terms. To learn more about using the AI Agent, see Explore Using AI Agent.

Search for Dremio Objects and Entities

You can quickly and easily find objects and entities with the AI-enabled search experience. Semantic search returns results of sources, folders, tables in Dremio's Open Catalog and external sources registered in Dremio, views in external sources registered in Dremio catalogs, user-defined functions (UDFs), Reflections, scripts, and jobs.

Semantic search takes object names and metadata such as wikis and labels into account to return results that are relevant to your search criteria.

To search for objects and entities:

- Click the search bar on the top right of the page and type in your search criteria. You can activate the search bar by using the shortcut keys `Command + K` (Mac) or `Ctrl + K` (Windows/Linux).

- Press the return or enter key to execute the search. Under **Recents**, you will see your recent searches.

- Select a search result to view details. For a table or view, you can click the icon to query the table or view in the SQL Runner.

Limitations and Considerations

- Semantic search has been optimized for English terms.

- After creating new objects, they can take a few hours to appear in search results.

- After deleting objects or entities, they can take a few hours to disappear from search results.

- Data discovery capabilities are optimized for data that is governed by Dremio. Files and folders in object storage that have not been formatted as a table in Dremio may not be easily discoverable.

Navigate Your Catalog

The Datasets page allows you to navigate through and explore objects that you have access to in Dremio. To learn more about the Datasets page, see Quick Tour of the Dremio Console. Once you have located the table or view that you are interested in, click it to open the Details panel and view comprehensive information about the dataset.

View Metadata of a Dataset

Wherever a dataset is referenced in Dremio, you can view a metadata card with details about the dataset.

To view the metadata, hover over a dataset to see a metadata card appear with details about the dataset. Key information displayed on dataset cards includes:

- **Dataset type and name**: Icon and title showing the dataset format and name

- **Quick actions**: Query, edit, or view the dataset

- **Path and labels**: Location and any applied labels

- **Usage metrics**: Jobs run and views created from the dataset

- **Ownership and dates**: Creator, creation date, and last modified

Related Topics

- Wikis and Labels – Learn more about using wikis and labels to enrich your data.

Data Privacy – Learn more about Dremio's data privacy practices.

Was this page helpful?

Discover using Dremio's AI Agent

Search for Dremio Objects and Entities

Limitations and Considerations

Navigate Your Catalog

View Metadata of a Dataset

Related Topics

Connect Client Applications | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/

You can connect to Dremio from several popular applications to query and visualize the data in Dremio:

Alteryx Designer

Apache Superset

Astrato

DBeaver

DbVisualizer

Deepnote

Domo

Hex

IBM Cognos Analytics

Looker

Microsoft Excel PowerPivot

Microsoft Power BI

Preset

SAP Business Objects

Tableau

ThoughtSpot

Dremio provides Arrow Flight SQL ODBC and JDBC drivers:

- Arrow Flight SQL JDBC
- Arrow Flight SQL ODBC

Dremio also supports the Dremio JDBC (Legacy).

Was this page helpful?

Hex | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/hex

On this page

Hex is a data workspace where you can analyze and share data from Dremio.

Supported Authentication Methods

Use a personal access token (PAT) obtained from Dremio. To create a PAT, follow the steps in the section Create a PAT.

Prerequisites

If you want to connect to a specific project in Dremio, copy the ID of the project. See Obtain the ID of a Project for the steps. After you obtain it, save it somewhere that you can retrieve it from during the procedure.

Create a Connection

1. Log into your Hex project.
2. Go to the **Data sources** tab on the side navigation bar.
3. Click **+ Add** > **Create project connection**.
4. Select **Dremio** from the project connections.
5. In the **Name** field, enter a name for the data connection.
6. (Optional) In the **Description** field, enter a brief description of the data connection.
7. In the **JDBC Connection** field, paste your JDBC connection string:

JDBC connection string

```
jdbc:dremio:direct=sql.dremio.cloud:443;ssl=true;
```

a. (Optional) Add ;PROJECT_ID={project-id} to the JDBC connection string and in the **Project ID** field, paste the ID of the project that you want to connect to. If the project ID isn't specified, then the default project will be used.

b. (Optional) Add ;engine={engine-name} to the JDBC connection string and in the

Engine Name field, specify the engine that you want the query routed to.

For the **Access Token**, paste your personal access token.

Click **Create Connection**.

Was this page helpful?

Supported Authentication Methods

Prerequisites

Create a Connection

Domo | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/domo

On this page

Domo is a cloud-based platform designed to provide direct, simplified, real-time access to business data for decision makers across the company with minimal IT involvement.

Supported Authentication Method

Authenticate to Dremio by using a personal access token (PAT). To create one, see Create a PAT.

Prerequisite

Obtain the ID of the project in Dremio that you want to connect to. For the steps, see Obtain the ID of a Project.

Create a Cloud Integration with Dremio

Click the **Data** tab to open the Datasets page.

Click the **Federated** tab to open the **Amplify existing cloud warehouses** dialog.

Next to **Native integration**, click **Dremio**.

In the **Cloud integrations** dialog, click **Add new integration**.

In step 1 of the **Connect a Dremio cloud integration** wizard, follow these sub-steps:

In the **Integration name** field, specify a unique name for the integration.

(Optional) In the **Integration description** field, briefly describe the integration.

Select **Dremio Cloud** as the connection type.

Click **Next**.

In step 2 of the wizard, follow these sub-steps:

In the **Dremio connection URL** field, specify the following connection URL, where `PROJECT_ID` is the ID of the project that you want to connect to:
Connection URL

```
jdbc:dremio:direct=sql.dremio.cloud:443;ssl=true;token_type=personal_access_token;PROJECT_ID=<project-ID>
```

Paste your PAT into the **Personal Access Token** field.

Click **Next**.

Select the tables that you want to use with Domo through this integration.

Click **Create Datasets**.

Datasets are created from the tables, though no data is moved or copied. Datasets in Domo are connections to data in data sources.

Was this page helpful?

Supported Authentication Method

Prerequisite

Create a Cloud Integration with Dremio

Looker | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/looker

On this page

You can use Looker to query and visualize data by means of Dremio.

Supported Authentication Method

Use a personal access token (PAT) obtained from Dremio. To create a PAT, follow the steps in Create a PAT.

Prerequisite

Copy the ID of the Dremio project that you want to connect to. See Obtain the ID of a Project for the steps. After you obtain it, save it somewhere that you can retrieve it from during the procedure.

Create a Connection

Log into Looker.

In the menu bar at the top of the page, select **Admin**, and then select **Connections** under **Database**.

- Click the **Add Connection** button in the top-right corner of the page to open the Connection Settings page for creating a connection.

- Specify a name for the connection.

- In the **Dialect** field, select **Dremio 11+**.

- In the **Remote Host:Port** fields, specify `sql.dremio.cloud` and `443`.

- In the **Database** field, specify any value. Though Looker requires a value in this field, Dremio does not use the value.

- In the **Username** and **Password** fields, specify your authentication credentials:

- In the **Username** field, type `$token`.

- In the **Password** field, paste your personal access token.

- Ensure that the **SSL** check box is selected.

- If there is more that one project in your Dremio organization and you are not connecting to the default project, specify this additional JDBC parameter in the **Additional Settings** section: `PROJECT_ID=<project id>`. To obtain the ID, see Obtain the ID of a Project.

- Click **Test These Settings** at the bottom of the page to check that the information that you specified is all valid.

- Click **Add Connection** if the test of the connection is successful.

The new connection is listed on the Connections page.

Was this page helpful?

- Supported Authentication Method

- Prerequisite

- Create a Connection

Preset | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/preset

On this page

You can use Preset, a cloud service for Superset, to query and visualize data.

Supported Authentication Method

Use a personal access token (PAT) obtained from a project.
To obtain one, follow the steps in Create a PAT.
After you obtain a PAT, it is recommended that you URL-encode it. To encode it locally on your system, you can follow these steps:

- In a browser window, right-click an empty area of the page and select **Inspect** or

Inspect Element, depending on your browser.

In the top bar of the inspection pane, click **Console**.

Type `encodeURIComponent("<PAT>")`, where `<PAT>` is the personal access token that you obtained from Dremio. The URL-encoded PAT appears in red on the next line. You can highlight it and copy it to your clipboard.

Prerequisite

Obtain the ID of the project that you want to connect to in Dremio. To obtain the ID, follow these steps inside the project:

In the Dremio console, hover over in the side navigation bar and select **Project settings**.

Select **General Information** in the project settings sidebar.

Copy the project ID to your system clipboard.

Create a Connection

Click **Settings** in the top-right corner, and select **Database Connections** under **Data**.

Click the **+Database** button in the top-right corner.

Select **Other** from the **Supported Databases** field of the Connect a Database dialog.

In the **Display Name** field, specify any name you prefer.

In the **Connect a Database** dialog, follow these steps:

Select **Other** from the **Supported Databases** field.

In the **Display Name** field, name the new connection.

In the **SQLALCHEMY URI** field, specify a URI that is in this format. Use an ampersand in front of each additional property that you add:SQLAlchemy URI format

```
dremio://$token:<PAT>@sql.dremio.cloud:443/<project-ID>;ssl=1[;<option>=<value>[;...]]
```

`<PAT>`: The URL-encoded personal access token that you obtained from Dremio. See Supported Authentication Method.

`<project-ID>`: The ID of the project that you want to connect to. See Prerequisite for the steps to obtain the ID.

`[;<option>=<value>[;...]]`: One or more optional encryption properties. Separate each property with a semicolon (`;`).Example SQLAlchemy URI

```
dremio://$token:hoYL2mqORpOv1Lq5WNOT-A-REAL-PATq5yeHEYon%2BOT0VHM0JYS%2BCMH7kpL%2BPQ%3D%
3D@sql.dremio.cloud:443/1df71752-NOT-A-PROJECT-ID-990e6b194aa4;ssl=1
```

Test the connection. If the test fails, check the syntax and values in the connection URI.

Click **Connect**.

Was this page helpful?

Supported Authentication Method

Prerequisite

Create a Connection

Astrato | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/astrato

On this page

Astrato is a no-code framework to build powerful data visualizations and custom data applications that actively drive better business decisions.

Supported Authentication Method

Authenticate to Dremio from Astrato by using a personal access token (PAT). To create one, see Create a PAT.

Create a Connection to Dremio

In Astrato, click **Data** in the left-hand navigation bar.

On the Data page, follow either of these sets of steps:

Create a connection from the **Data Views** section.

Click **Data Views**.

In the top-right corner, click **New Data View**.

Create a connection from the **Data Connections** section.

Click **Data Connection**.

In the top-right corner of the page, click **New connection**.

In the Create connection page, click the **Dremio** tile.

In the **Host** field, follow either of these steps:

If your Dremio organization is in the European control plane, leave the default value of `eu.dremio.cloud`.

If your Dremio organization is in the US control plane, specify `dremio.cloud`.

In the **Personal Access Token** field, paste your PAT.

Click **Test connection**.

-

In the Connect to Dremio page, follow these steps:

- Select the project that you want to connect to in your organization.
- Select the folder that you want to connect to in your project.
- Click **Connect**.

You can now define a data view from data that resides in the folder that you connected to.

Was this page helpful?

- Supported Authentication Method
- Create a Connection to Dremio

DBeaver | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/dbeaver

On this page

You can run SQL from DBeaver to explore your data in your data lakes and relational databases through Dremio.

Supported Versions

You can use any version of DBeaver for Linux, MacOS, or Windows, except for version 23.0.2.202304091457.

Supported Authentication Method

Authenticate to Dremio by using a personal access token. To create one, see Create a PAT.

Prerequisites

- Download the Arrow Flight SQL JDBC driver.
- For MacOS, ensure you have the latest version of Java Runtime Environment (JRE).

Connect to Dremio

Step 1: Add the JDBC Driver

You only need to add the Arrow Flight SQL JDBC driver once. You can skip this step if DBeaver already lists this driver in its Driver Manager dialog. To add the JDBC driver, follow these steps:

-

In the menubar, select **Database** > **Driver Manager**.

In the Driver Manager dialog, click **New**.

In the Settings section, follow these steps:

In the **Name** field, specify a name for the driver, such as "Arrow Flight SQL JDBC Driver".

In the **Driver Type** field, ensure that **Generic** is the selected driver type.

In the **Class Name** field, specify `org.apache.arrow.driver.jdbc.ArrowFlightJdbcDriver`.

In the **URL Template** field, specify `jdbc:arrow-flight-sql://data.dremio.cloud:443/` .

In the **Default Port** field, specify `443`.

In the Libraries section, click **Add File** and select the `.jar` file for the Arrow Flight SQL JDBC driver.

Click **OK**.

Step 2: Create a Connection

Once you've added the Arrow Flight SQL JDBC driver, follow these steps to create a connection to Dremio:

Select **Database** > **New Connection from JDBC URL**.

In the dialog that follows, enter `jdbc:arrow-flight-sql://data.dremio.cloud:443/`. At this point, DBeaver lists the driver in the **Drivers** field. If the driver is not immediately suggested, type and then delete a character in the input field to refresh suggestions.

Select the JDBC driver and click **Next**.

In the Connect to a Database dialog, follow these steps:

Select **URL** as the value for **Connect By**.

In the JDBC URL field, append `?token=<encoded_pat>` to the URL and replace `<encoded_pat>` with your URL-encoded personal access token.

note

If connecting to a non-default project, you must also append `&catalog=<project_id>` to the URL and replace `<project_id>` with your project ID.

(Optional) Click **Test Connection**. If the connection works, the **Connection Test** dialog opens and indicates that DBeaver is able to connect to Dremio. The connection is not held open. Click **OK**.

Click **Finish**.

Was this page helpful?

Supported Versions

Supported Authentication Method

- Prerequisites
- Connect to Dremio
- Step 1: Add the JDBC Driver
- Step 2: Create a Connection

Tableau | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/tableau

On this page

Connect Tableau to Dremio to derive powerful insights from your data and create real-time dashboards.

note

When using Tableau with Dremio, avoid using periods in space or dataset names. Due to differences in hierarchy support, periods in paths are treated as separators, resulting in errors when navigating or selecting spaces or datasets with periods in their names.

You can connect from your Tableau application to Dremio in either of two ways:

- Configure a reusable connection in Tableau Desktop, Tableau Server, or Tableau Cloud.
- Connect to a specific dataset by downloading the `.tds` file from Dremio and opening it in Tableau Desktop.

Supported Versions

Product	Supported Versions
Tableau Desktop	2022.1 and later
Tableau Server	2022.1 and later
Tableau Cloud	Latest version deployed by Tableau

Supported Authentication Methods

From Tableau, you can authenticate to Dremio with a username and password, or with a personal access token (PAT) that can be obtained from the Dremio console.

You can also configure single sign-on (SSO) through OAuth 2.0. For steps on how to configure SSO, see Enable SSO to Dremio from Tableau.

Tableau Desktop

Tableau Desktop includes a native connector that you can use to connect to Dremio.

Prerequisites for Using the Dremio JDBC Driver (Legacy)

To connect to Dremio, you'll also need to install the Dremio JDBC driver. Download the Dremio JDBC driver and copy it to the Tableau Desktop's `Drivers` folder.

Download driver for macOS by running this command in a Terminal window

```
curl -L https://download.dremio.com/jdbc-driver/dremio-jdbc-driver-LATEST.jar -o ~/Library/Tableau/Drivers/dremio-jdbc-driver-LATEST.jar
```

Download driver for Windows by running this command in a PowerShell window

```
Invoke-WebRequest -Uri "https://download.dremio.com/jdbc-driver/dremio-jdbc-driver-LATEST.jar" -OutFile "C:\Program Files\Tableau\Drivers\dremio-jdbc-driver-LATEST.jar"
```

Prerequisites for Using the Arrow Flight SQL JDBC Driver

The Tableau Desktop 2025.1+ connector for Dremio supports Arrow Flight SQL JDBC in place of the Dremio JDBC driver (Legacy). To change the driver, download the Arrow Flight SQL JDBC driver, copy it to Tableau Desktop's `Drivers` folder, and select the **Use Arrow Flight SQL Driver (preview)** option in the **Advanced** tab of the connection dialog.

Download driver for macOS by running this command in a Terminal window

```
curl -L https://repo1.maven.org/maven2/org/apache/arrow/flight-sql-jdbc-driver/18.3.0/flight-sql-jdbc-driver-18.3.0.jar -o ~/Library/Tableau/Drivers/flight-sql-jdbc-driver-18.3.0.jar
```

Download driver for Windows by running this command in a PowerShell window

```
Invoke-WebRequest -Uri "https://repo1.maven.org/maven2/org/apache/arrow/flight-sql-jdbc-driver/18.3.0/flight-sql-jdbc-driver-18.3.0.jar" -OutFile "C:\Program Files\Tableau\Drivers\flight-sql-jdbc-driver-18.3.0.jar"
```

Steps for Connecting

To create a Dremio source in Tableau Desktop:

- Open Tableau Desktop. If you already had Tableau Desktop open, restart the application. Under the **To a Server** section in the **Connect** panel, click **More**.
- Select **Dremio**. The **Dremio** connection dialog opens.
- In the **Product** field, select **Dremio Cloud**.
- In the **Region** field, select the Dremio control plane in which your Dremio organization is located: `US` or `Europe`. For Tableau 2024.3 or before, in the **Server** field, select `sql.dremio.cloud (US)` or `sql.eu.dremio.cloud (EU)`.
-

In the **Authentication** field, select **Personal Access Token** or **OAuth 2.0**.

If you selected **Personal Access Token**, in the **Password** field, specify your PAT.

If you selected **OAuth 2.0**, specify one of these URLs in the **Dremio Authentication Server** field:

If your Dremio organization is on the US control plane: `https://login.dremio.cloud`

If your Dremio organization is on the EU control plane: `https://login.eu.dremio.cloud`

(Optional) In the **Project** field, if your datasets are in a non-default project of your Dremio organization or you do not have access to the default project, paste the ID of the project that you want to connect to. To obtain the project ID, see Obtain the ID of a project.

(Optional for Tableau 2025.1+) If you are using the Arrow Flight SQL JDBC driver, in the **Advanced** tab, select the **Use Arrow Flight SQL Driver (preview)** option. Ensure that you have the Arrow Flight SQL JDBC driver downloaded.

(Optional) In the **Advanced** tab, specify an **Engine** to run queries on.

Click **Sign In**.

Create a Live Connection to a Dataset from Dremio

You can generate a Tableau Datasource (`.tds`) file that represents a live connection to a dataset that is in Dremio. No actual data is stored in this file, and you can think of it as a shortcut to a Tableau session with a preconfigured view of your data.

note

The `.tds` file download option must be enabled for users to have access to this feature. To enable this feature, see Enable the .tds File Download in the Dremio Console.

OAuth is the only supported authentication mechanism for `.tds` files.

To download a `.tds` file:

On the Datasets page of your Dremio project, find the dataset you want to work with and open the Details panel for the dataset.

Click the button that displays the Tableau logo. Dremio downloads a `.tds` file to your system.

Open the `.tds` file.

Authenticate to Dremio in the browser window that Tableau opens. The dataset will open in Tableau Desktop.

Tableau Server

Tableau Server includes a native connector that you can use to connect to Dremio.

Prerequisites for Using the Dremio JDBC (Legacy) Driver

To connect to Dremio, you'll need to install the Dremio JDBC driver. Download the Dremio JDBC driver and copy it to the `Drivers` folder.

Download driver for Windows by running this command in a PowerShell window

```
Invoke-WebRequest -Uri "https://download.dremio.com/jdbc-driver/dremio-jdbc-driver-LATEST.jar" -OutFile "C:\Program Files\Tableau\Drivers\dremio-jdbc-driver-LATEST.jar"
```

Download driver for Linux by running this command in a command-line window

```
curl -L https://download.dremio.com/jdbc-driver/dremio-jdbc-driver-LATEST.jar -o /opt/tableau/tableau_driver/jdbc/dremio-jdbc-driver-LATEST.jar
```

Prerequisites for Using the Arrow Flight SQL JDBC Driver

The Tableau Server 2025.1+ connector for Dremio supports Arrow Flight SQL JDBC in place of the Dremio JDBC driver (Legacy). To change the driver, download the Arrow Flight SQL JDBC driver, copy it to the `Drivers` folder, and select the **Use Arrow Flight SQL Driver (preview)** option in the **Advanced** tab of the connection dialog.

Download driver for Windows by running this command in a PowerShell window

```
Invoke-WebRequest -Uri "https://repo1.maven.org/maven2/org/apache/arrow/flight-sql-jdbc-driver/18.3.0/flight-sql-jdbc-driver-18.3.0.jar" -OutFile "C:\Program Files\Tableau\Drivers\flight-sql-jdbc-driver-18.3.0.jar"
```

Download driver for Linux by running this command in a command-line window

```
curl -L https://repo1.maven.org/maven2/org/apache/arrow/flight-sql-jdbc-driver/18.3.0/flight-sql-jdbc-driver-18.3.0.jar -o /opt/tableau/tableau_driver/jdbc/flight-sql-jdbc-driver-18.3.0.jar
```

Steps for Connecting

To create a Dremio source in Tableau Server:

In a web browser, navigate to your Tableau Server site.

In your workbook, click **Add a Data Source**. Alternatively, you can publish an existing data source to Tableau Server.

In the **Connect to Data** dialog, select **Dremio** under the **Connectors** tab.

In the **Dremio** connection dialog, for the **Product** field, select **Dremio Cloud**.

In the **Region** field, select the Dremio control plane in which your Dremio organization is located: `US` or `Europe`. For Tableau 2024.3 or before, in the **Server** field, select `sql.dremio.cloud (US)` or `sql.eu.dremio.cloud (EU)`.

In the **Authentication** field, select **Personal Access Token** or **OAuth 2.0**.

If you selected **Personal Access Token**, in the **Password** field, specify your PAT.

If you selected **OAuth 2.0**, specify one of these URLs in the **Dremio Authentication Server** field:

If your Dremio organization is on the US control plane: `https://login.dremio.cloud`

If your Dremio organization is on the EU control plane: `https://login.eu.dremio.cloud`

(Optional) In the **Project** field, if your datasets are in a non-default project of your Dremio organization or you do not have access to the default project, paste the ID of the project that you want to connect to. To obtain the project ID, see Obtain the ID of a project.

(Optional for Tableau 2025.1+) If you are using the Arrow Flight SQL JDBC driver, in the **Advanced** tab, select the **Use Arrow Flight SQL Driver (preview)** option. Ensure that you have the Arrow Flight SQL JDBC driver downloaded.

(Optional) In the **Advanced** tab, specify an **Engine** to run queries on.

Click **Sign In**.

Tableau Cloud

Tableau Cloud includes a native connector that you can use to connect to Dremio.

note

The Tableau Cloud 2025.1 connector for Dremio has an option to use the Arrow Flight SQL JDBC driver in place of the Dremio JDBC driver to power the connection to Dremio. In the **Advanced** tab, select the **Use Arrow Flight SQL Driver (preview)** option.

To create a Dremio source in Tableau Cloud:

In a web browser, navigate to your Tableau Cloud site.

Click **New** > **Published Data Source** to create a reusable data source or **Data** > **Add a Data Source** from within a workbook. Alternatively, you can publish an existing data source to Tableau Cloud.

In the **Connect to Data** dialog, select Dremio under the Connectors tab.

In the **Dremio** connection dialog, for the **Product** field, select **Dremio Cloud**.

In the **Region** field, select the Dremio control plane in which your Dremio organization is located: `US` or `Europe`.

In the **Authentication** field, select **Personal Access Token** or **OAuth 2.0**.

If you selected **Personal Access Token**, in the **Password** field, specify your PAT.

If you selected **OAuth 2.0**, specify one of these URLs in the **Dremio Authentication Server** field:

If your Dremio organization is on the US control plane: `https://login.dremio.cloud`

If your Dremio organization is on the EU control plane: `https://login.eu.dremio.cloud`

(Optional) In the **Project** field, if your datasets are in a non-default project of your Dremio organization or you do not have access to the default project, paste the ID of the project that you want to connect to. To obtain the project ID, see Obtain the ID of a Project.

(Optional for Tableau 2025.1+) If you are using the Arrow Flight SQL JDBC driver, in the **Advanced** tab, select the **Use Arrow Flight SQL Driver (preview)** option.

(Optional) In the **Advanced** tab, specify an **Engine** to run queries on.

Click **Sign In**.

Advanced Configuration

Enable the `.tds` File Download in the Dremio console

`ADMIN` privileges are required to make updates to this setting.

To enable users to download `.tds` files for datasets in Dremio, follow these steps:

In the Dremio console, click in the side navigation bar and select **Project settings**.

Select **BI Applications** in the project settings sidebar.

Select the **Tableau** tab.

Toggle the **Enable Tableau Desktop** setting on.

After the organization administrator completes these steps, refresh your browser window.

Enable SSO to Dremio from Tableau

SSO using OAuth 2.0 is supported by Tableau Desktop 2022.3 or later, Tableau Server, and Tableau Cloud.

Users of Tableau Desktop will use SSO authentication whether connecting directly to Dremio or connecting through a `.tds` file downloaded from Dremio. If you want to use SSO to authenticate when connecting to Dremio through a `.tds` file, ensure that SSO is enabled and configured for your Dremio cluster before the file is downloaded.

To enable SSO to Dremio from Tableau, ensure that your Dremio cluster has SSO configured with your identity provider and follow these steps:

For Tableau Server only, follow the configuration steps.

-

Follow the steps to enable SSO to Dremio from Tableau.

Configure SSO for Tableau Server

note

Use OAuth 2.0 with Tableau Server
If you are using Tableau Server and you want to use OAuth 2.0 to authenticate to Dremio, you must have TLS enabled for Tableau Server for OAuth 2.0 to work. See Example: SSL Certificate - Generate a Key and CSR in the Tableau's documentation for additional information.

To configure SSO using OAuth for Tableau Server, follow these steps:

Run the following command in the Tableau Services Manager (TSM) command line. The only variable that you need to set the value for is `<tableau-server-domain-name-or-ip>`, which is the domain name or IP of your Tableau Server deployment:

Configure OAuth for Tableau Server

```
tsm configuration set -k oauth.config.clients -v "[{\"oauth.config.id\":\"dremio\",
\"oauth.config.client_id\":\"https\:\/\/connectors.dremio.app\/tableau\",
\"oauth.config.client_secret\":\"test-client-secret\",
\"oauth.config.redirect_uri\":\"https://<tableau-server-domain-name-or-ip>/auth/add_oaut
h_token\"}]" --force-keys
```

To apply the changes to Tableau Server, run the command `tsm pending-changes apply`.

Configure Dremio

To enable SSO authentication to Dremio from Tableau:

In the Dremio console, click in the side navigation bar and select **Project settings**.

Select **BI Applications** in the project settings sidebar.

On the BI Applications page, click **Tableau**.

Ensure that **Enable single sign-on for Tableau** is toggled on.

For Tableau Server only: In the **Redirect URIs** field, paste in the redirect URI for your Tableau Server. If you have set up more than one Tableau Server, you can add multiple URIs, separating them with commas. Each URI uses this format, where `<tableau-server>` is the hostname or IP address of Tableau Server:

Redirect URI for Tableau Server

```
https://<tableau-server>/auth/add_oauth_token
```

Customize the Connection String

If you want to add JDBC parameters to the JDBC URL that Tableau generates for connections to Dremio, parameters other than those Tableau sets through the Dremio connection dialog, see Use a Properties file to customize a JDBC connection in the Tableau documentation.

Manually Install the Dremio Connector

If you are previewing a feature that hasn't been released or you have been provided a `.taco` file with a fix that hasn't been released, you can manually install this version of the Dremio connector for temporary use.

To manually install the connector:

Download the `dremio.taco` file.

Move the `dremio.taco` file:

Copy dremio.taco file on macOS

```
cp <download-location>/dremio.taco ~/Library/Tableau/Connectors/
```

Copy dremio.taco file on Windows

```
copy C:\<download-location>\dremio.taco "C:\Program Files\Tableau\Connectors"
```

Move dremio.taco file for Linux (Tableau Server only)

```
mv <download-location>/dremio.taco /opt/tableau/connectors/dremio.taco
```

(Optional) If a new version of the Dremio JDBC driver is required, download it and copy it to Tableau Desktop's `Drivers` folder by running the following command:

Download driver for macOS

```
curl https://download.dremio.com/jdbc-driver/dremio-jdbc-driver-LATEST.jar -o -l ~/Library/Tableau/Drivers/dremio-jdbc-driver-LATEST.jar
```

Download driver for Windows

```
Invoke-WebRequest -Uri "https://download.dremio.com/jdbc-driver/dremio-jdbc-driver-LATEST.jar" -OutFile "C:\Program Files\Tableau\Drivers\dremio-jdbc-driver-LATEST.jar"
```

For Linux, download the driver from the download site and move it by using this command:
Download driver for Linux (Tableau Server only)

```
mv <download-location>/dremio-jdbc-driver-LATEST.jar
```

```
/opt/tableau/tableau_driver/jdbc/dremio-jdbc-driver-LATEST.jar
```

Now you can connect to Dremio from Tableau Desktop. or Tableau Server.

Was this page helpful?

- Supported Versions
- Supported Authentication Methods
- Tableau Desktop
- Prerequisites for Using the Dremio JDBC Driver (Legacy)
- Prerequisites for Using the Arrow Flight SQL JDBC Driver
- Steps for Connecting
- Create a Live Connection to a Dataset from Dremio
- Tableau Server
- Prerequisites for Using the Dremio JDBC (Legacy) Driver
- Prerequisites for Using the Arrow Flight SQL JDBC Driver
- Steps for Connecting
- Tableau Cloud
- Advanced Configuration
- Enable the `.tds` File Download in the Dremio console
- Enable SSO to Dremio from Tableau
- Customize the Connection String
- Manually Install the Dremio Connector

Deepnote | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/deepnote

On this page

You can use Deepnote to explore data in Dremio with Python and SQL.

Supported Authentication Method

Use a personal access token (PAT) obtained from Dremio. To create a PAT, follow the steps in the section Create a PAT.

Create an Integration with Dremio

After logging into Deepnote, click **Integrations** on the left.

Scroll down to **Create new integration**.

Under **Data warehouses & lakes**, click **Dremio**.

Specify a name for the integration.

In the **Host name** field, specify either `data.dremio.cloud` (Dremio's US control plane) or `data.eu.dremio.cloud` (Dremio's European control plane).

In the **Port** field, specify the number 443.

(Optional) In the **Schema** field, specify the database schema to use by default when a schema is not given in a query.

In the **Token** field, paste the PAT that you obtained from Dremio Cloud.

Click **Create integration**.

note

Do not toggle on the **Use SSH** switch. Dremio integrations do not support SSH tunnels.

Deepnote gives you the option of associating the integration with a project immediately. If you click **Skip**, the integration is listed under **Workspace integrations** on the **Integrations** page.

Was this page helpful?

Supported Authentication Method

Create an Integration with Dremio

Drivers | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/drivers/

Dremio provides Arrow Flight SQL ODBC and JDBC drivers:

Arrow Flight SQL JDBC

Arrow Flight SQL ODBC

Dremio also supports the Dremio JDBC (Legacy).

note

To connect from supported client applications, follow the connection steps for that specific application rather than installing drivers separately. For example, to connect from Tableau, follow the steps in Tableau, which include driver installation instructions.

Was this page helpful?

ThoughtSpot | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/thoughtspot

On this page

You can use ThoughtSpot to search directly against your data in Dremio for live analytics and actionable insights.

Supported Versions

Dremio supports ThoughtSpot cloud 8.3 and ThoughtSpot software 7.2.1.

Supported Authentication Methods

Use a personal access token (PAT) obtained from Dremio. To create a PAT, follow the steps in the section Create a PAT.

If you want to use an OAuth application for authentication, you will first need to add the OAuth application and then copy the client ID.

Prerequisites

If you want to connect to a specific project in Dremio, copy the ID of the project. See Obtain the ID of a Project for the steps. After you obtain it, save it somewhere that you can retrieve it from during the procedure.

Create a Connection

note

While you're using the connection, the data fields that you create, modify, and delete in Dremio are reflected as table columns in ThoughtSpot. To account for new or outdated fields, you will need to go back into the data connection to check or uncheck the columns that you want added or removed on the Select Tables page.

Log into ThoughtSpot.

Navigate to **Data** > **Connections** > **Add Connection**.

On the Choose Your Data Warehouse page, specify your data connection details:

In the **Name your connection** field, enter a name.

(Optional) In the **Describe your connection** field, enter a brief description.

For the **Choose your data warehouse** field, select **Dremio**.

Click **Continue**.

On the Dremio Connection Details page, to provide your authentication credentials, follow either of these steps:

a. To use a personal access token that you obtained from Dremio, select **Use Service**

Account.

b. To use an OAuth application that you added in Dremio, select **Use OAuth**.

In the **Host** field, enter sql.dremio.cloud.

In the **Port** field, enter 443.

Follow either of these steps based on the authentication type that you chose:

a. For using a personal access token:

In the **User** field, type `$token`.

In the **Password** field, paste your personal access token.

(Optional) In the **Project ID** field, paste the project ID.

b. For using an OAuth application:

In the **Project ID** field, paste the project ID.

In the **OAuth Client ID** field, paste the client ID.

In the **OAuth Client Secret** field, enter your password.

In the **Auth URL** field, enter the authorization URL of the application.

In the **Access Token URL** field, enter the URL of the access token.

(Optional) In the **Schema** field, enter the project schema.

Click **Continue**.

On the Select Tables page, you can see all the data tables and views from Dremio. To select tables and columns from that list, select a table and check the boxes next to the columns for that table.

Click **Create Connection**.

In the **Create Connection** dialog, click **Confirm**.

Was this page helpful?

Supported Versions

Supported Authentication Methods

Prerequisites

Create a Connection

DbVisualizer | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/dbvisualizer

On this page

You can use DbVisualizer to query and visualize data by means of Dremio.

Supported Versions

You can use any version of DbVisualizer, as long as you use Dremio JDBC Driver 14.0.0 or later.

Supported Authentication Methods

There are two methods of authenticating that you can choose from when you connect from DbVisualizer to Dremio:

- Use Microsoft Entra ID as an enterprise identity provider

  To configure Microsoft Entra ID, see Microsoft Entra ID.

  note

  You can use Microsoft authentication only if the admin for your Dremio organization has enabled it.

- Use a personal access token (PAT) obtained from Dremio

  To create a PAT, follow the steps in Create a PAT.

Prerequisites

- Download the Dremio JDBC driver.
- If you do not want to connect to the default project in your Dremio organization, copy the ID of the Dremio Cloud project that you want to connect to. See Obtain the ID of a Project for the steps. After you obtain it, save it somewhere that you can retrieve it from during the procedure.

Add Dremio's JDBC Driver to DbVisualizer's Driver Manager

1. Launch DbVisualizer.
2. Select **Tools** > **Driver Manager**.
3. In the **Driver Name** list of the **Driver Manager** dialog, select **Dremio**.
4. Click the folder icon to find and select the downloaded Dremio JDBC driver.
5. Close the **Driver Manager** dialog.

Create a Connection

1. Launch DbVisualizer.
2. Select **Database** > **Create Database Connection**.

In the **Use Connection Wizard?** dialog, click **No Wizard**.

Name the connection.

Ensure that these default values are set:

Field	Value
Settings Format	Server Info
Database Type	Auto Detect (Dremio)
Driver	Dremio
Connection Type	Direct

In the **Database Server** field, specify `sql.dremio.cloud`.

In the **Database Port** field, specify `443`.

In the **Database Userid** and **Database Password** fields, specify your authentication credentials:

If you want to authenticate by using a Microsoft account and password, and Microsoft Entra ID is configured as an enterprise identity provider for Dremio Cloud, specify the username and password for the account.

If you want to authenticate by using a personal access token, specify these values:

In the **Username** field, type `$token`.

In the **Password** field, paste your personal access token.

Click **Properties**.

Click the plus sign to add a new parameter.

Name the parameter `ssl`.

Specify `true` for the value of this parameter.

If you do not want to connect to the default project in your organization, follow these steps:

a. Click the plus sign to add a new parameter.

b. Name the parameter `PROJECT_ID`.

c. In the `Value` field, paste the ID of the project that you want to connect to.

Click **Apply**.

Click **Connect**.

If the connection works, DbVisualizer displays a message as shown below (the reported version numbers might differ):

`Dremio Server 20.0.0-202112201840340507-df2e9b7c`

`Dremio JDBC Driver 19.1.0-202111160130570172-0ee00450`

You can now expand your Dremio connection to see a list of the data sources that are in the project.

Was this page helpful?

- Supported Versions
- Supported Authentication Methods
- Prerequisites
- Add Dremio's JDBC Driver to DbVisualizer's Driver Manager
- Create a Connection

Apache Superset | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/apache-superset

On this page

You can use Superset to query and visualize data.

Supported Versions

- Superset 1.5.3 and later
- Dremio SQLAlchemy connector 3.0.2 and later

Supported Authentication Method

Use a personal access token (PAT) created in a project. To create one, follow the steps Create a PAT. After you obtain a PAT, it is recommended that you URL-encode it. To encode it locally on your system, you can follow these steps:

1. In a browser window, right-click an empty area of the page and select **Inspect** or **Inspect Element**, depending on your browser.
2. In the top bar of the inspection pane, click **Console**.
3. Type `encodeURIComponent("<PAT>")`, where `<PAT>` is the personal access token that you obtained from Dremio. The URL-encoded PAT appears in red on the next line. You can highlight it and copy it to your clipboard.

Prerequisites

If you installed Superset according to the instructions for installing from scratch, install the Dremio SQLAlchemy Connector on the system or in the VM where Apache Superset is running. Instructions are in the sqlalchemy_dremio repository in GitHub.

Create a Connection

If you are using a version of Superset earlier than 2.1.0, follow these steps:

Select **Data** > **Databases** in the menu bar at the top of the screen.

Click the **Database** button in the top-right corner of the screen.

If you are using version 2.1.0 or later of Superset, follow these steps:

Click **Datasets** in the menu bar at the top of the screen.

Click the plus (+) icon in the top-right corner.

Select **Data** > **Connect database**.

In the **Connect a Database** dialog, follow these steps:

Select **Other** from the **Supported Databases** field.

In the **Display Name** field, name the new connection.

In the **SQLALCHEMY URI** field, specify a URI that is in this format. Use an ampersand in front of each additional property that you add:SQLAlchemy URI format

```
dremio+flight://data.dremio.cloud:443/[<schema>]?token=<PAT>&UseEncryption=true[&<option>=<value>[&...]]
```

`<schema>`: The name of the database schema to use by default when a schema is not given in a query. Providing a schema is optional. Specifying a schema does not prevent queries from being issued for other schemas; such queries must explicitly include the schema.

`<PAT>`: The URL-encoded personal access token that you obtained from Dremio. See Supported Authentication Method.

`[&<option>=<value>[&...]]` is one or more optional properties from the SSL connection properties and Advanced properties tables below. Separate each property with an ampersand (`&`).Example SQLAlchemy URI

```
dremio+flight://data.dremio.cloud:443/?token=dOOfxnJlTnebGu7Beta9NOT-A-REAL-PATyfOoNbJwEMep7UjkQu0JTsFXpYGm==&UseEncryption=true
```

4. Test the connection. If the test fails, check the syntax and values in the connection URI.
5. Click **Connect**.

SSL Connection Properties

Use the following properties to configure SSL encryption and verification methods:

Name	Type	Description	Default Value
UseEncryption	integer	Forces the client to use an SSL-encrypted connection to communicate with Dremio. Accepted value: `true`, the client communicates with Dremio only using SSL encryption. This is the only possible value.	true

| disableCertificateVerification | integer | Specifies whether to verify the host certificate against the trust store. Accepted values: `false`, verifies the certificate against the trust store; `true`, does not verify the certificate against the trust store. | false |
| trustedCerts | string | The full path of the .pem file containing certificates trusted by a CA, for the purpose of verifying the server. If this option is not set, defaults to using the trusted CA certificates .pem file. The TLS connection fails if you do not specify a value when UseEncryption is true and disableCertificateVerification is false. | N/A |

Advanced Properties

Name	Type	Description	Default Value	Required?
routing_engine	string	The engine to route queries to while a connection remains open.	N/A	No
routing_tag	string	The tag to be associated with all queries executed within a connection session.	N/A	No

Was this page helpful?

- Supported Versions
- Supported Authentication Method
- Prerequisites
- Create a Connection
- SSL Connection Properties
- Advanced Properties

Alteryx Designer | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/alteryx-designer

On this page

You can use Alteryx Designer to quickly prepare, blend, conform, and analyze data from datasets in Dremio.

Supported Versions

Alteryx Designer 10.6+

Prerequisite

Download, install, and configure the ODBC driver for Arrow Flight SQL.

note

- The driver supports 64-bit Windows 10 or later.

The personal access token (PAT) that you add to the DSN that you configure for the driver determines the Dremio project that the driver connects you to. Ensure that you create your PAT in the correct project.

Select Dremio as a Data Source

In Alteryx Designer, select **File** > **New Workflow**.

Drag the **Input Data** tool from the tool palette on to the workflow canvas.

In the configuration properties for Input Data, click the arrow on the right side of the **Connect a File or Database** field.

In the Data connections dialog, follow these steps:

a. Select **Recent** and click **Clear List** in the top-right corner if there are any entries on the page.

b. Select **Data Sources**.

c. Scroll down to the option **Generic connection**.

d. Click either **ODBC** or **OleDB**.

If you clicked **ODBC**, follow these steps in the ODBC Connection dialog:

a. In the **Data Source Name** field, select the data source name for the Arrow Flight SQL ODBC driver.

b. Leave the **User name** and **Password** field blank. The authentication credentials for connecting to Dremio are already present in the user DSN.

c. Click **OK**.

If you clicked **OleDB**, follow these steps in the Data Link Properties dialog:

a. On the **Provider** tab, select **Microsoft OLE DB Provider for ODBC Drivers**.

b. Click **Next>>**.

c. For step 1 on the **Connection** tab, select **Use data source name**, and then select the data source name for the Arrow Flight SQL ODBC driver.

d. For step 2 on the **Connection** tab, leave the **User name** and **Password** fields blank. The authentication credentials for connecting to Dremio are already present in the user DSN.

e. (Optional) Click **Test Connection** to find out whether the info you specified on this tab is correct.

f. Click **OK**.

You can now browse and query datasets that are in Dremio.

Was this page helpful?

Supported Versions

- Prerequisite

- Select Dremio as a Data Source

Microsoft Power BI | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/microsoft-power-bi

On this page

Connect Microsoft Power BI to visualize your data and create reports.

You can connect Power BI to Dremio in one of the following ways:

- Configure a reusable connection to use in Power BI Desktop, Power BI Gateway, or Power BI Service. Power BI Service can connect to Dremio through DirectQuery or through Power BI Gateway.

- Connect to a specific dataset by downloading the `.pbids` file from Dremio and opening it in Power BI Desktop.

Supported Authentication Methods

From Power BI, you can authenticate to Dremio with one of the following methods:

- **Personal access token (PAT)**: For details, see Create a PAT.

- **Single sign-on (SSO) through OAuth 2.0**: For steps on how to configure SSO, see Enable SSO to Dremio from Power BI.

Connect to Dremio from Power BI

The Power BI connector for Dremio now supports connectivity through the open-source Arrow Database Connectivity (ADBC) driver, which Dremio highly recommends using to connect to Dremio. To enable reports to use ADBC, see Enable Connectivity with ADBC.

Existing connections will continue to work, but we recommend using the embedded ADBC driver for all new reports and migrating existing reports to ADBC to benefit from improved performance and supportability.

To connect to Dremio from Power BI Desktop:

- Click **Get data**, search for `Dremio`, select **Dremio Cloud**, and click **Connect**.

- In the Dremio Cloud dialog, follow these steps:

a. Use the Flight SQL ADBC driver and in the **Server** field specify which of Dremio's control planes to connect to:

- If your Dremio organization is on the US control plane: `adbc://data.dremio.cloud`

- If your Dremio organization is on the EU control plane: `adbc://data.eu.dremio.cloud`

b. (Optional) In the **Project** field, if your datasets are in a non-default project of your Dremio organization or you do not have access to the default project, paste the ID of the project that you want to connect to. To obtain the project ID, see Obtain the ID of a Project.

c. (Optional) In the **Engine** field, specify the name of the query-execution engine for your project. For information about query-execution engines, see Manage Engines.

d. (Optional) In the **Native Query** field, specify a SQL query as the data input source.

e. Under **Data Connectivity mode**, select either **Import** or **DirectQuery**. Click **OK**.

f. For **Authentication Method**, select **Key** or **Microsoft Account**.

Key: Paste in the personal access token you obtained from Dremio. For details, see Create a PAT.

Microsoft Account: Click **Sign in**, and then specify your credentials.

note

Creating Dataflows through Power BI Service is also supported. To create a dataflow, click **New** > **Dataflow**. For the data source connection, follow the steps above.

Create a Live Connection to a Dataset from Dremio

You can generate a Microsoft Power BI Data Source (`.pbids`) file that represents a live connection to a dataset that is in Dremio. No actual data is stored in this file, and you can think of it as a shortcut to a Power BI Desktop session with a preconfigured view of your data.

note

The `.pbids` file download option must be enabled for users to have access to this feature. To enable this feature, see Enable the .pbids file download.

To download a `.pbids` file:

On the Datasets page in Dremio, find the dataset you want to work with and open the Details panel for the dataset.

Click the button that displays the Power BI logo. Dremio downloads a `.pbids` file to your system.

Open the `.pbids` file.

Authenticate using a personal access token or your organizational account and click **Load**.

Connect to Dremio via PrivateLink

Use these instructions to connect to Dremio if your organization uses PrivateLink for secure private connectivity.

Prerequisites

- A PrivateLink connection configured in your AWS VPC.
- The November 2025 version of Power BI Desktop.

Connect with an Organizational Account

- In Power BI Desktop, click Get data.
- In the Get Data dialog, choose Blank Query and click **Connect**.
- Select Advanced Editor and add the following:

```
let
                                        Source                                          =
DremioCloud.DatabasesByServerV370("adbc://<orgAlias>.data.privatelink.dremio.cloud",
null,     null,     null,     null,     "Enabled-PEM",     [AuthorizationServerPort=443,
AuthorizationServerDomain="<orgAlias>.login.privatelink.dremio.cloud"])
in
 Source
```

where `<orgAlias>` is the organization alias implemented in your PrivateLink configuration.

- Click **Done**.
- Select **Edit Credentials**.
- Choose Organizational Account and click **Sign in**.

Connect with a Personal Access Token

- In Power BI Desktop, click Get data, choose Dremio Cloud, and click **Connect**.
- In the Dremio Cloud connection box, enter the PrivateLink DNS name created in your PrivateLink configuration, in the form

```
adbc://<orgAlias>.data.privatelink.dremio.cloud
```

where `<orgAlias>` is your PrivateLink organization alias. Click **OK**.

Power BI Gateway

To enable Power BI users to connect to Dremio via Power BI Gateway:

- In Power BI Service, click **...** next to your profile picture at the top-right corner of the browser and navigate to **Settings > Manage gateways**.
- Under **GATEWAY CLUSTERS**, select the gateway you created previously.
- Select the checkbox **Allow user's cloud data sources to refresh through this gateway cluster**.

At the top of the page, click **Add data sources to use the gateway**. This launches the *Data Source Settings* page.

Enter a **Data Source Name**.

Select the **Data Source Type** drop-down menu and select **Dremio Cloud**.

In the **Server** field, specify which of Dremio's control planes to connect to:

If your Dremio organization is on the US control plane: `sql.dremio.cloud`.

If your Dremio organization is on the EU control plane: `sql.eu.dremio.cloud`.

(Optional) In the **Project** field, if your datasets are in a non-default project of your Dremio organization or you do not have access to the default project, paste the ID of the project that you want to connect to. To obtain the project ID, see Obtain the ID of a project.

For **Authentication Method**, select **Key** or **Microsoft Account**.

Key: Paste in the personal access token you obtained from Dremio. For details, see Create a PAT.

Microsoft Account: Click **Sign in**, and then specify your credentials.

Ignore the **Engine** field. It is not used.

Under **Advanced Settings**, set the **Connection Encryption setting for this data source** to **Encrypted**.

Click **Add**. A **Connection Successful** message is shown on top of the Data Source Settings page.

Advanced Configuration

Enable Connectivity with ADBC

Dremio supports connectivity through Arrow Database Connectivity (ADBC). To enable this for Power BI Service, see the following options.

Enable the ADBC Option for a New Connection

In Power BI Desktop, click **Get data**.

In the Get Data dialog, locate and select **Dremio Cloud**, and click **Connect**.

In the Dremio Cloud dialog, in the **Server** field, enter `adbc://data.dremio.cloud` or `adbc://data.eu.dremio.cloud`, depending on which control plane your Dremio account is on.

(Optional) Complete the other fields in the dialog as you normally would.

Click **OK**.

-

Authenticate using your preferred method, and click **Connect**.

Enable the ADBC Option for an Existing Connection

In Power BI Desktop, go to **Data source settings**, select your source, and click **Change source**.

In the Dremio Cloud dialog, update the **Server** field to `adbc://data.dremio.cloud` or `adbc://data.eu.dremio.cloud`, depending on which control plane your Dremio account is on. If you're unable to edit the source this way, click **Transform data**, then click **Advanced Editor** in the **Home** tab. In the dialog that appears, update the hostname/server with the `adbc://` prefix, and click **Done**.

Click **OK**.

Reauthenticate using your preferred method, and click **Connect**.

Enable the `.pbids` File Download in the Dremio Console

`ADMIN` privileges are required to make updates to this setting.

To enable users to download `.pbids` files for datasets in the Dremio console, follow these steps:

Click in the side navigation bar and select **Project Settings**.

Select **BI Applications** in the projects settings sidebar.

Toggle the **Microsoft Power BI Desktop** setting on.

After the organization administrator completes these steps, refresh your browser window.

Enable SSO to Dremio from Power BI

When Single Sign-On (SSO) is enabled, viewers of reports in Power BI Service run them under their own Power BI username instead of as the user who published the reports, or under the username of the user who set up Power BI Gateway. SSO is supported for DirectQuery mode.

To enable SSO to Dremio from Power BI, ensure that your Dremio organization is configured with Microsoft Entra ID and follow these steps:

In the Dremio console, click in the side navigation bar and select **Organization Settings**.

Select **BI Applications** from the organization settings sidebar.

On the BI Applications page, click **Power BI**.

Ensure that **Enable single sign-on for Power BI** is toggled on.

For **Microsoft Entra Tenant ID**, enter the tenant ID of your Microsoft Entra ID

account. The tenant ID of each Microsoft Entra ID account can only be assigned to a single Dremio organization.

- For **User Claim Mapping**, specify the key of the user claim that Dremio must look up in access tokens to find the username of the user attempting to log in. See User Claim Mapping for more information about this field.

- Click **Save**.

In the Power BI Admin portal, select **Tenant settings** and toggle on the **Enabled** switch under **Dremio SSO**.

Enable SSO for a DirectQuery Report

To enable SSO for a report that uses DirectQuery:

In Power BI Service, open the workspace to which you published the report.

- Find the dataset that is associated with the report, click the three dots next to its name, and select **Settings**.

- Expand the **Data source credentials** section and click **Edit credentials**.

In the configure dialog, follow these steps:

In the **Authentication method** field, select one of these options:

Key: Paste your personal access token into the **Account Key** field.

- **OAuth2**: Authenticate by using your Microsoft ID and password.

In the **Privacy level setting for this data source** field, ensure that **Private** is selected.

- Select the check box **Report viewers can only access this data source with their own Power BI identities using DirectQuery**.

- Click **Sign in**.

Enable SSO for Reports with Power BI Gateway

To enable SSO when you are using Power BI Gateway:

In Power BI Service, open the workspace to which you published the report.

- Find the dataset that is associated with the report, click the three dots next to its name, and select **Settings**.

In the settings for the dataset, expand **Gateway connection**.

- Recreate your data source by following these steps:

- Select the **Maps to** field.

- Select **Manually add to gateway**.

-

In the New data source dialog, create a data source that matches the one that you previously used for your dataset. However, give the new data source a different name.

In the **Authentication method** field, select one of these options:

Key: Paste your personal access token into the **Account Key** field.

OAuth2: Click **Edit credentials** and select the option **Use SSO via Microsoft Entra ID for DirectQuery queries**.

Click **Create**.

Under **Gateway connection**, verify that the new data source is selected in the **Maps to** field.

Arrow Database Connectivity (ADBC) Limitations

ADBC is not enabled by default. It must be enabled by the owner of the report.

NativeQuery is not supported.

Metadata calls are not cached.

SSO is not supported in environments that use different domain names for the UI and Flight services.

Power BI Desktop occasionally caches errors that might affect future connection attempts until the cache is cleared.

Complex data types such as `MAP` and `INTERVAL` are not supported.

When using DirectQuery, chaining functions is supported, but some complex scenarios may not work as expected. Complex optional parameters for functions are not supported.

Troubleshoot Power BI

Cached Data Issues

If you have previously installed older versions of Power BI Desktop, cached data may interfere with the newer versions of the Flight SQL drivers resulting in connection errors.

Problem

For example, when using Flight SQL ADBC, cached connection data in Power BI could cause the following errors:

`ADBC: IOError [] [FlightSQL] [FlightSQL] unresolved address (Unavailable; GetObjects(GetDBSchemas))`

`ADBC: IOError [] [FlightSQL] [FlightSQL] connection error: desc = "transport: authentication handshake failed: credentials: cannot check peer: missing selected ALPN

property. If you upgraded from a grpc-go version earlier than 1.67, your TLS connections may have stopped working due to ALPN enforcement. For more details, see: https://github.com/grpc/grpc-go/issues/434" (Unavailable; GetObjects(GetDBSchemas))`

Solution

Clear the Power BI Desktop cache and any cached data source permissions involving Dremio connections by following these steps:

Clear Power BI Desktop Caches.

In Power BI Desktop, go to **File** > **Options and Settings** > **Data Source Settings**.

Select **Global Permissions**.

Clear all cached connections by clicking **Clear All Permissions**, or select specific Dremio data sources and click **Clear Permissions**.

After completing these steps, try reconnecting to Dremio using the instructions above.

Large Result Sets

Problem

When fetching data from Dremio with ADBC you may see the following error:

`Unexpected error: [FlightSQL] grpc: received message larger than max (43747370 vs. 16777216) (ResourceExhausted; DoGet: endpoint 0: [])`

Solution

By default, the ADBC driver accepts only messages up to 16 MiB in size. This can be fixed by updating the Power BI M expression to customize the connection as follows:

```
let
     Source = DremioCloud.DatabasesByServerV370("adbc://data.dremio.cloud", null, null, null, null, "Enabled-PEM", [AdbcMaxMessageSize=33554432]) // 32 MiB
in
    Source
```

Was this page helpful?

Supported Authentication Methods

Connect to Dremio from Power BI

Create a Live Connection to a Dataset from Dremio

Connect to Dremio via PrivateLink

Power BI Gateway

- Advanced Configuration
- Enable Connectivity with ADBC
- Enable the `.pbids` File Download in the Dremio Console
- Enable SSO to Dremio from Power BI
- Arrow Database Connectivity (ADBC) Limitations
- Troubleshoot Power BI
- Cached Data Issues
- Large Result Sets

Dremio JDBC (Legacy) | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/drivers/jdbc-legacy

On this page

You can use the Dremio JDBC driver (Legacy) version 25+ to connect to Dremio from JDBC client applications. This driver is licensed under Apache-2.0.

Prerequisites

Java 11 is required as of Dremio JDBC (Legacy) version 25.

note

The Arrow Flight SQL JDBC driver is the recommended driver for connectivity to Java-based applications. The Dremio JDBC driver (Legacy) will not be updated or fixed moving forward.

Supported Authentication Methods

- Use personal access tokens. To generate one, see Personal Access Tokens.
- Use JSON Web Tokens (JWT) from an external token provider. To use a JWT, you must have OAuth enabled in Dremio. For more information about using JWTs, see External Token Providers.

Download and Install

You can download the JDBC driver from here. The driver does not require installation.

Connect to Dremio

note

If you are using the JDBC driver to connect to Dremio from a supported client application, refer to the documentation for creating connections from that application.

If you want to start with the base JDBC connection string for your Dremio project:

Click Project Settings in the side navigation bar.

Select **General Information** in the project settings sidebar.

Copy the connection string that is in the **JDBC Connection** field.

To construct a connection string:

Set the subprotocol to `jdbc:dremio:`.

Set the property `direct` equal to `sql.dremio.cloud:443`.

Add one of these types of authentication credentials for connecting from your JDBC client application to Dremio:

Use a personal access token (PAT) in either of the following ways:

Set `user` to `$token` and use the PAT as the password when the client application does not support OAuth:

Use PAT as password when client app does not support OAuth

```
jdbc:dremio:direct=sql.dremio.cloud:443;user=$token;password=<personal-access-token>;
```

Set `token_type` to `personal_access_token`, use the PAT as the password, and set `username` to null when the client application supports OAuth:

Use PAT as password

```
jdbc:dremio:direct=sql.dremio.cloud:443;token_type=personal_access_token;password=<perso
nal-access-token>;username=;
```

Use a JSON Web Token (JWT). You can use a JWT when the tool used with the JDBC driver supports OAuth:

Use a JWT

```
jdbc:dremio:direct=sql.dremio.cloud:443;token_type=jwt;password=<jwt>;username=;
```

Set the property `ssl` equal to `true`: `ssl=true;`

Add the ID of the project that you are connecting to: `project_id=<project-id>;`

(Optional) Route queries to a particular engine in your project, set the property `engine` to the name of an engine: `engine=<engine-name>;`

Connection Parameters

Encryption Parameters

To encrypt communication between your JDBC client applications and Dremio, use the SSL JDBC connection parameters and a fully qualified host name to configure the JDBC connection string and connect to Dremio.

SSL JDBC Connection Parameter	Type	Description	Default Value	Required
disableCertificateVerification	boolean	Controls whether the driver verifies the host certificate against the trust store. * The driver **will** verify the certificate against the trust store when it is set to `false`. * The driver **will not** verify the certificate against the trust store when the value is set to `true` .	`false`	No
disableHostVerification	boolean	Forces the driver to verify that the host in the certificate is the host being connected to. * The driver **will** verify the certificate against the host being connected to when it is set to `false`. * The driver **will not** verify the certificate against the host whe it is set to `true`.	`false`	No
ssl	boolean	Forces the client to use an SSL encrypted connection to communicate with the Dremio server. * SSL encryption is disabled with the client when it is set to `false`. * The client communicates with the Dremio server only using SSL encryption when it is set to `true`. **Note:** To connect to Dremio, SSL must be enabled.	`false`	Yes
trustStoreType	string	The trustStore type. Accepted value is: JKS PKCS12 The following property only applies to **Windows**. * If the `useSystemTrustStore` parameter is set to true, the accepted values are: `Windows-MY` and `Windows-ROOT`. * Import the certificate into the **Trusted Root Certificate Authorities** and set `trustStoreType=Windows-ROOT`. * Import the certificate into **Trusted Root Certificate Authorities** or **Personal** and set `trustStoreType=Windows-MY`.	`None`	No
trustStore	string	Path to the truststore. If this parameter is not specified, it defaults to Java truststore (`$JAVA_HOME/lib/security/cacerts`) and the trustStorePassword parameter is ignored.	`$JAVA_HOME/lib/security/cacerts`	No
useSystemTrustStore	boolean	Bypasses trustStoreType and automatically picks the correct truststore based on the operating system: * Keychain on MacOS * Local Machine and Current User Certificate Stores on Windows * Default truststore on other systems	`true`	No
trustStorePassword	string	Password to the truststore.	`None`	No

SOCKS Proxy Connection Parameters

If you want to connect to Dremio through a SOCKS proxy, use these connection parameters:

Parameter	Type	Description	Default Value	Required?
socksProxyHost	string	The IP address or hostname of the SOCKS proxy.	N/A	Yes
socksProxyPort	integer	The port to use on the SOCKS proxy.	1080	No
socksProxyUsername	string	The username to use for connections.	N/A	No
socksProxyPassword	string	The password to use for connections.	N/A	Only if a username is specified.

Advanced Parameters

Name	Type	Description	Default Value
quoting	string	Specifies which type of character to use to delimit values in queries. The value can be `BACK_TICK`, `BRACKET`, or `DOUBLE_QUOTE`.	`DOUBLE_QUOTE`
routing_tag	string	When this parameter is set, the specified tag is associated with all queries executed within a session. Rules can check for the presence of a tag with the function "tag()". For more information, see Workload Management.	N/A
stringColumnLength	string	The maximum length of data in columns of the STRING datatype and of complex datatypes. The range is 1 to 2147483647.	1024.

Parameterized Queries with Prepared Statements

Dremio supports using parameters in prepared statements for SELECT queries.

The parameter marker is `?` in prepared statements. To execute a prepared statement, you must set the parameter marker with one of the supported data types and set methods.

The example below uses the Date type parameter and the `setDate` set method. For set methods, the first argument is the index of the parameter marker in the SQL query, starting from 1. This example includes only one parameter marker, and the second argument is the value for the parameter marker. After you set the parameter, you can execute the prepared statement by calling the `executeQuery()` method on the prepared statement.

Example prepared statement with parameters

```
public class HelloWorld {
  public static void main(String[] args) {
       try (PreparedStatement stmt = getConnection().prepareStatement("SELECT * FROM
(values (DATE '2024-02-20'), (null)) AS a(id) WHERE id=?")) {
      Date date = Date.valueOf(LocalDate.of(2024, 02, 20));
      stmt.setDate(1, date);
      try (ResultSet rs = stmt.executeQuery()) {
        assertThat(rs.getMetaData().getColumnCount()).isEqualTo(1);
        assertThat(rs.next()).isTrue();
        assertThat(rs.getDate(1)).isEqualTo(date);
        assertThat(rs.next()).isFalse();
      }
    }
  }
}
```

The example below demonstrates how to reuse the same prepared statement by defining a different set method and parameter value.

Example prepared statement with different set method and parameters

```
public class HelloWorld {
  public static void main(String[] args) {
       try (PreparedStatement stmt = getConnection().prepareStatement("SELECT * FROM
(values (DATE '2024-02-20'), (null)) AS a(id) WHERE id=?")) {
      Date date = Date.valueOf(LocalDate.of(2024, 02, 20));
      stmt.setDate(1, date);
```

```
      try (ResultSet rs = stmt.executeQuery()) {
        assertThat(rs.getMetaData().getColumnCount()).isEqualTo(1);
        assertThat(rs.next()).isTrue();
        assertThat(rs.getDate(1)).isEqualTo(date);
        assertThat(rs.next()).isFalse();
      }
      stmt.setDate(1, Date.valueOf(LocalDate.of(2025, 02, 20)));
      try (ResultSet rs = stmt.executeQuery()) {
        assertThat(rs.next()).isFalse();
      }
    }
  }
}
```

The following example shows how to use more than one parameter in a prepared statement.

Example prepared statement with two parameters

```
public class HelloWorld {
  public static void main(String[] args) {
       try (PreparedStatement stmt = getConnection().prepareStatement("SELECT * FROM
(values (1), (2), (null)) AS a(id) WHERE id = ? OR id < ?")) {
      stmt.setInt(1, 1);
      stmt.setInt(2, 3);
      try (ResultSet rs = stmt.executeQuery()) {
        assertThat(rs.getMetaData().getColumnCount()).isEqualTo(1);
        assertThat(rs.next()).isTrue();
        assertThat(rs.getInt(1)).isEqualTo(1);
        assertThat(rs.next()).isFalse();
      }
    }
  }
}
```

Supported Data Types and Set Methods

To execute a prepared statement, you must set the parameter marker with one of the supported set methods listed in the table below.

Column Data Type	Supported Set Methods
Integer	setInt(), setShort(), setNull()
Numeric	setInt(), setShort(), setLong(), setBigDecimal(), setNull()
Decimal	setShort(), setInt(), setLong(), setBigDecimal(), setNull()
BigInt	setShort(), setInt(), setLong(), setBigDecimal(), setNull()
Double	setDouble(), setFloat(), setNull()
Float	setFloat(), setNull()
Char	setString(), setNull()
Varchar	setString(), setNull()
Boolean	setBoolean(), setNull()

Time	setTime(), setNull()
TimeStamp	setTimestamp(), setNull()
Date	setDate(), setNull()
VarBinary	setNull(), setBytes()

Was this page helpful?

Prerequisites

Supported Authentication Methods

Download and Install

Connect to Dremio

Connection Parameters

Encryption Parameters

SOCKS Proxy Connection Parameters

Advanced Parameters

Parameterized Queries with Prepared Statements

Supported Data Types and Set Methods

IBM Cognos Analytics | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/ibm-cognos-analytics

On this page

You can run SQL from Cognos Analytics to explore your data through Dremio. Cognos Analytics Dynamic Query supports connections to Dremio through the legacy JDBC driver.

Supported Versions

To learn which versions of Dremio are supported with IBM Cognos 11.2.x, see DQM testing of vendor-supported client driver versions for each Cognos Analytics 11.2.x release.

To learn which versions of Dremio are supported with IBM Cognos 12.0.x, see DQM testing of vendor-supported client driver versions for each Cognos Analytics 12.0.x release.

Supported Authentication Methods

Use a Dremio personal access token (PAT) for authentication.

Create a Connection

Launch Cognos Analytics.

Navigate to **Manage** > **Data Server Connections**.

Click **Add Data Server** and select **Dremio** as the type of connection.

In the **JDBC URL** field, enter the connection string in the JDBC Connection field in the project settings' General Information sidebar in Dremio. The EU and US have different control planes, so the connection strings differ slightly depending on the control plane you're using.

JDBC connection string example for US control plane

```
jdbc:dremio:direct=sql.eu.dremio.cloud:443[;ssl=true;PROJECT_ID=<YOUR_PROJECT_ID>;ENGINE
=<OPTIONAL_ENGINE_NAME>]
```

JDBC connection string example for EU control plane

```
jdbc:dremio:direct=sql.eu.dremio.cloud:443[;ssl=true;PROJECT_ID=<YOUR_PROJECT_ID>;ENGINE
=<OPTIONAL_ENGINE_NAME>]
```

In the **Username** field, enter `$token`, and in the **Password** field, enter your Dremio PAT to authenticate to Dremio.

Click **Save**.

Click **Test** to confirm the connection.

Was this page helpful?

Supported Versions

Supported Authentication Methods

Create a Connection

SAP Business Objects | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/sap-business-objects

On this page

You can use SAP Business Objects to query and visualize data by means of Dremio.

Supported Versions

SAP Business Objects 4.0+

Prerequisite

Download, install, and configure the Arrow Flight SQL ODBC driver.

Connect to Dremio

Open Information Design Tool.

Select **File** > **New** > **Project**.

Set the name of the project and click **Finish**.

Right-click the project and select **New** > **Relational Connection**.

Specify a name for the relational connection and click **Next**.

Select the Generic ODBC3 datasource driver and click **Next**.

Follow either of these steps:

Select **Use existing data source** and select the Arrow Flight SQL ODBC DSN.

Select Use connection string, select the Arrow Flight SQL ODBC driver, and specify this base connection string:

Base connection string

```
HOST=data.dremio.cloud;PORT=443;token=<personal-access-token>;
UseEncryption=true;DisableCertificateVerification=false;
```

See Connection Parameters for additional connection parameters that are available.

(Optional) Test the connection.

Click **Finish**.

Dremio schemas and tables are now available.

Was this page helpful?

Supported Versions

Prerequisite

Connect to Dremio

Driver Release Notes | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/drivers/release-notes/

On this page

This section contains the changes for the drivers that are supported for use with Dremio.

Arrow Flight SQL Drivers

- Arrow Flight SQL JDBC
- Arrow Flight SQL ODBC

Other Drivers

- Dremio JDBC Driver (Legacy)

note

The Arrow Flight SQL JDBC driver is the recommended driver for connectivity to Java-based applications. The Dremio JDBC driver (Legacy) will not be updated or fixed moving forward.

Was this page helpful?

- Arrow Flight SQL Drivers
- Other Drivers

Microsoft Excel PowerPivot | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/microsoft-excel-powerpivot

On this page

Prerequisites

- Ensure that your operating system is 64-bit Windows 10 or later.
- Download, install, and configure the Arrow Flight SQL ODBC driver.

Update the DSN Configuration

- Launch ODBC Data Sources on your Windows system.
- Select the **User DSN** tab.
- Select the DSN entry that you created when you configured the Arrow Flight SQL ODBC driver.
- Click **Configure**.
- In the **Advanced Properties** section, add the following key/value pair:

Key: quoting

Value: BRACKET

Connect to Dremio

Open Excel.

Click the **Power Pivot** tab and then click **Manage**.

Select **From Other Sources**.

In the Table Import Wizard, select **Others (OLEDB/ODBC)**.

Click **Next**.

Click **Build**.

In the Data Link Properties dialog, follow these steps:

a. On the **Provider** tab, select **Microsoft OLE DB Provider for ODBC Drivers**.

b. Click **Next>>**.

c. For step 1 on the **Connection** tab, select **Use data source name**, and then select the data source name for the Arrow Flight SQL ODBC driver.

d. For step 2 on the **Connection** tab, leave the **User name** and **Password** fields blank. The authentication credentials for connecting to Dremio Cloud are already present in the user DSN.

e. (Optional) Click **Test Connection** to find out whether the info you specified on this tab is correct.

f. Click **OK**.

Click **Next**.

Ensure that the option **Select from a list of tables and views to choose the data to import**.

Click **Next**.

Select the tables and views that you want to import data from.

Click **Finish**.

Was this page helpful?

Prerequisites

Update the DSN Configuration

Connect to Dremio

Arrow Flight SQL JDBC | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/drivers/arrow-flight-sql-jdbc

On this page

The Arrow Flight SQL JDBC driver is an open-source driver that is based on the specifications for the Java Database Connectivity (JDBC) API. The Flight SQL JDBC driver uses Apache Arrow, so it is able to move large amounts of data faster, in part because it does not need to serialize and then deserialize data.

This driver is licensed under Apache-2.0.

Prerequisites

You can use the Arrow Flight SQL JDBC 18.3.0 driver on systems that:

- Support Java versions: Java 11+
- Run the following 64-bit operating systems:
  - Linux: RedHat/CentOS
  - Windows 10 and later
  - macOS

Supported Authentication Method

You can use personal access tokens for authenticating to Dremio. To generate one, see Personal Access Tokens.

Download and Install

- Download the Driver: You can download the driver at Arrow Flight SQL JDBC Driver.
- Integrate the Driver: To integrate the driver into your development environment, you need to add the location of the driver to your classpath to inform the Java Virtual Machine (JVM) and the Java compiler where to locate the driver class files and resources during compilation and runtime.
- For name the of the driver class, specify `org.apache.arrow.driver.jdbc.ArrowFlightJdbcDriver`.

Connect to Dremio

Use this template to create a direct connection to Dremio Cloud:

Template for the JDBC URL

```
jdbc:arrow-flight-sql://data.dremio.cloud:443/?token=<encoded_pat>[&catalog=<project_id>][&schema=<schema>][&<properties>]
```

- `token`: The personal access token to use to authenticate to Dremio. See Personal Access Tokens for information about enabling and creating PATs. You must URL-encode PATs that you include in JDBC URLs. See URL-encode Values for suggested steps.
- `catalog`: Specifies the project ID of a project in Dremio Cloud. You can use this to

connect to non-default Dremio Cloud projects.

˜schema`: The name of the schema (data source or folder, including child paths, such as `mySource.folder1` and `folder1.folder2`) to use by default when a schema is not specified in a query.

˜<properties>`: A list of JDBC properties for encrypting connections and routing queries to particular engines. Values must be URL-encoded. See URL-encode Values for suggested steps.

To authenticate to Dremio Cloud, pass in a personal access token (PAT) with the `token` property. Use the PAT as the value. See Personal Access Tokens for information about enabling the use of PATs in Dremio and about creating PATs. You must URL-encode PATs that you include in JDBC URLs. To encode a PAT locally on your system, you can follow the steps in URL-encode Values.

Connection Parameters

Encryption Parameters

If you are setting up encrypted communication between your JDBC client applications and the Dremio server, use the SSL JDBC connection parameters and fully qualified hostname to
configure the JDBC connection string and connect to Dremio.

note

This driver does not yet support these features:

˜Disabling host verification

˜Impersonation

Properties	Value	Required	Description
`disableCertificateVerification`	`true` or `false`	[Optional]	If `true`, Dremio does not verify the host certificate against the truststore. The default value is `false`.
`trustStoreType`	string	[Optional]	Default: JKS The trustStore type. Allowed values are : `JKS`, `PKCS12` If the useSystemTrustStore option is set to true (on Windows only), the allowed values are: `Windows-MY`, `Windows-ROOT` Import the certificate into the **Trusted Root Certificate Authorities** and set `trustStoreType=Windows-ROOT`. Also import the certificate into **Trusted Root Certificate Authorities** or **Personal** and set `trustStoreType=Windows-MY`.
`trustStore`	string	[Optional]	Path to the truststore. If not provided, the default Java truststore is used (usually `$JAVA_HOME/lib/security/cacerts`) and the trustStorePassword parameter is ignored.
`useSystemTrustStore`	`true` or `false`	[Optional]	By default, the value is `true`. Bypasses trustStoreType and automatically picks the correct truststore based on the operating system: Keychain on MacOS, Local Machine and Current User Certificate Stores on Windows, and default truststore on other operating systems. If you are using an operating system other than MacOS or Windows, you must use the `trustStorePassword` property to pass the password of the truststore. Here is an example of a connection string for Linux: `jdbc:arrow-flight-sql://data.dremio.cloud:443/?useEncryption=true&token=1234&trust

StorePassword=901234` |
| `trustStorePassword` | string | [Optional] | Password to the truststore. |

Advanced Parameters

Name	Type	Description	Default Value
quoting	string	Specifies which type of character to use to delimit values in queries. The value can be `BACK_TICK`, `BRACKET`, or `DOUBLE_QUOTE`.	`DOUBLE_QUOTE`
routing_tag	string	When this parameter is set, the specified tag is associated with all queries executed within a session. Rules can check for the presence of a tag with the function "tag()". For more information, see Workload Management.	N/A
stringColumnLength	string	The maximum length of data in columns of the STRING datatype and of complex datatypes. The range is 1 to 2147483647.	1024

URL-encode Values

To encode a personal access token (PAT) or property value locally on your system, you can follow these steps:

In a browser window, right-click an empty area of the page and select **Inspect**.

Click **Console**.

Type `encodeURIComponent("<PAT-or-value>")`, where `<PAT-or-value>` is the personal access token that you obtained from Dremio or the value of a supported JDBC property. The URL-encoded PAT or value appears on the next line. You can highlight it and copy it to your clipboard.

Parameterized Queries with Prepared Statements

Prepared statements allow you to dynamically pass parameters to SQL queries using placeholders, ensuring safer query execution by separating the query structure from the values in parameters.

With a prepared statement, parameters (`?`) can be set at runtime using set methods to reuse queries with different values.

note

This feature requires Apache Arrow 18.3.0 or later. It supports `SELECT` and `DML` statements.

To use parameterized queries with prepared statements, follow these steps:

Use the `prepareStatement()` method to define a query with parameters, which act as placeholders for dynamic values.

Set the values by replacing each parameter with a value using the appropriate set methods.

Ensure all parameters are set before running the query, with indexing starting at 1. If parameters are not set before running the query, JBDC throws an exception.

- Call `executeQuery()` to run the `SELECT` query and retrieve results, or `executeUpdate()` to run the `DML` query and retrieve the count of modified records.

Java examples for SELECT and DML queries

```
PreparedStatement preparedStatement = connection.prepareStatement(
  "SELECT * FROM employees WHERE department = ? AND salary > ?");
preparedStatement.setString(1, "Engineering");
preparedStatement.setDouble(2, 75000);
ResultSet resultSet = preparedStatement.executeQuery();

PreparedStatement preparedStatement = connection.prepareStatement(
  "DELETE FROM employees WHERE department = ? AND salary > ?");
preparedStatement.setString(1, "Engineering");
preparedStatement.setDouble(2, 75000);
int rowsUpdated = preparedStatement.executeUpdate();
```

Supported Data Types and Set Methods

Column Data Type	**Supported Set Methods**
Integer	`setInt()`, `setShort()`, `setNull()`
Numeric	`setInt()`, `setShort()`, `setLong()`, `setBigDecimal()`, `setNull()`
Decimal	`setShort()`, `setInt()`, `setLong()`, `setBigDecimal()`, `setNull()`
BigInt	`setShort()`, `setInt()`, `setLong()`, `setBigDecimal()`, `setNull()`
Double	`setDouble()`, `setFloat()`, `setNull()`
Float	`setFloat()`, `setNull()`
Char	`setString()`, `setNull()`
Varchar	`setString()`, `setNull()`
Boolean	`setBoolean()`, `setNull()`
Time	`setTime()`, `setNull()`
Timestamp	`setTimestamp()`, `setNull()`
Date	`setNull()`
VarBinary	`setBytes()`, `setNull()`

Limitations

The JDBC client does not support the `setDate()` method due to mismatched date encoding formats between the Arrow Flight JDBC client and Dremio.

Differences between the Arrow Flight SQL JDBC and the Dremio JDBC (Legacy) Driver

The Arrow Flight SQL JDBC driver differs from the Dremio JDBC (Legacy) driver in the following:

- Requires Java 11+.

- Supports `ResultSet.getBoolean()` on `varchar` columns in which boolean values are represented as these strings: "0", "1", "true", "false".

- Supports `null` Calendar in calls to `ResultSet.getDate()`, `ResultSet.getTime()`, and `ResultSet.getTimestamp()`
When a call to one of these methods has no `Calendar` parameter, or the `Calendar` parameter is `null`, the Flight JDBC driver uses the default timezone when it constructs the returned object.

- Supports `ResultSet.getDate()`, `ResultSet.getTime()`, and `ResultSet.getTimestamp()` on `varchar` columns in which dates, times, or timestamps are represented as strings.

- Supports varchar values that represent numeric values in calls to `ResultSet.getInteger()`, `ResultSet.getFloat()`, `ResultSet.getDouble()`, `ResultSet.getShort()`, `ResultSet.getLong()`, and `ResultSet.getBigDecimal()`

- Supports integer values in calls to `getFloat()`
Integers returned gain one decimal place.

- Supports the native SQL complex types `List`, `Map`, and `Struct`
Dremio's legacy JDBC driver uses String representations of these types.

- Supports using the Interval data type in SQL functions.

- Removes support for calling `ResultSet.getBinaryStream()` on non-binary data types. Though such support exists in traditional JDBC drivers, it is not in the specification for the JDBC API.

note

Calling `DatabaseMetadata.getCatalog()` when connected to Dremio returns empty. Other `DatabaseMetadata` methods return null values in the `TABLE_CAT` column. This is expected behavior because Dremio does not have a catalog.

Supported Conversions from Dremio Datatypes to JDBC Datatypes

DREMIO TYPE	**JDBCARROW TYPE**
BIGINT	Int
BIT	Bool
DATE	Date
DECIMAL	Decimal
DOUBLE	FloatingPoint(DOUBLE)
FIXEDSIZEBINARY	FixedSizeBinary
FLOAT	FloatingPoint(SINGLE)
INT	Int
INTERVAL_DAY_SECONDS	Interval(DAY_TIME)
INTERVAL_YEAR_MONTHS	Interval(YEAR_MONTH)
LIST	List
MAP	Map
NULL	Null
OBJECT	Not Supported
STRUCT	Struct
TIME	Time(MILLISECOND)
TIMESTAMP	Timestamp(MILLISECOND)
VARBINARY	Binary

| VARCHAR | Utf8 |

Add the Root CA Certificate to Your System Truststore

- At a command-line prompt, run this command:

```
openssl s_client -showcerts -connect data.dremio.cloud:443 </dev/null
```

- Copy the last certificate, including the lines `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----`, to your clipboard.

- Create a text file and paste the certificate into it.

- Save the text file as `cert.pem`.

- If you are using MacOS, follow these steps:

a. In Finder, double-click the `cert.pem` file.

b. In the dialog that opens, select the option to add the root certificate to the system truststore.

- If you are using Windows, follow these steps:

a. At a command-line prompt, enter one of these commands:

- `certlm` if you want to add the certificate for all user accounts on your Windows system.

- `certmgr` if you want to add the certificate only for the current user account.

b. Right-click the folder **Trusted Root Certification Authorities**.

c. Select **Import**.

d. Browse for the `cert.pem` file and import it.

- If you are using a version of Linux, follow the instructions for your version.

- If you are developing your own client application to use the driver to connect to Dremio, add the certificate to the Java truststore. You must know the path to the `cacerts` file from `$JAVA_HOME`.

- If you are using Java 11, run this command:

```
keytool -import -trustcacerts -file cert.pem -alias gtsrootr1ca -keystore $JAVA_HOME/lib/security/cacerts
```

Limitations

- Impersonation is not supported.

- Disabling host verification is not supported.

Was this page helpful?

- Prerequisites
- Supported Authentication Method
- Download and Install
- Connect to Dremio
- Connection Parameters
- Encryption Parameters
- Advanced Parameters
- URL-encode Values
- Parameterized Queries with Prepared Statements
- Supported Data Types and Set Methods
- Limitations
- Differences between the Arrow Flight SQL JDBC and the Dremio JDBC (Legacy) Driver
- Supported Conversions from Dremio Datatypes to JDBC Datatypes
- Add the Root CA Certificate to Your System Truststore
- Limitations

Arrow Flight SQL ODBC | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/drivers/arrow-flight-sql-odbc

On this page

You can use the Arrow Flight SQL ODBC 0.9.7 driver to connect to Dremio from ODBC client applications.

This driver is licensed under GNU Library General Public License, Version 2.

Prerequisites

You can use the Arrow Flight SQL ODBC 0.9.7 driver on systems that:

- Run the following 64-bit operating systems:
  - Linux: RedHat/CentOS
  - Windows 10 and later
  - macOS

Supported Authentication Method

You can use personal access tokens for authenticating to Dremio. To generate one, see Personal Access Tokens.

Windows

Download and Install

To download and install the Arrow Flight SQL ODBC driver:

1. Download the Windows 64-bit version of the driver from the ODBC driver download page.
2. Run the installer.
3. (Optional) In the **User Account Control** page, click **Yes**. This page appears only if there is user account control configured on your Windows machine.
4. In the **Welcome to Dremio** page, click **Next**.
5. Click **Install**.
6. In the **Installation Complete** page, click **Next**.
7. In the **Completing Arrow Flight SQL ODBC Driver Setup Wizard** page, click **Finish**.

Connect to Dremio

caution

Do not follow these steps if you are using Microsoft Power BI Desktop to connect to Dremio. For the steps for configuring Power BI, see Connecting from Microsoft Power BI.

note

Before following these steps, generate a personal access token in Dremio. See Personal Access Tokens.

To configure a user DSN:

1. Go to **Start Menu** > **Window Administrative Tools**. Click **ODBC Data Sources (64-bit)**.
2. In the **ODBC Data Source Administrator (64-bit)** dialog, click **System DSN**.
3. Select **Arrow Flight SQL ODBC DSN** and click **Configure**.
4. (Optional) Change the data source name.
5. In the **Host name** field, `data.dremio.cloud` for the US control plane, or `data.eu.dremio.cloud` for the European control plane.
6. -

In the **Port** field, specify `443`.

In the **Authentication Type** field, select **Token Authentication**.

In the **Authentication Token** field, paste a personal access token.

Click the **Advanced** tab.

Ensure that the **Use Encryption** option is selected.

For additional parameters, see Connection Parameters.

If you ever need to enable tracing for troubleshooting problems with the driver, click the **Tracing** tab in the **ODBC Data Source Administrator (64-bit)** dialog, set the log-file path, and then click **Start Tracing Now**.

Linux

Download and Install

To download and install the Arrow Flight SQL ODBC driver:

Download the Linux version of the driver from the ODBC driver download page.

Run the following command to install the driver and automatically create the data source name (DSN) `Arrow Flight SQL ODBC DSN`:

Install Dremio ODBC driver

```
sudo yum localinstall <dremio-odbc-rpm-path>
```

Connect to Dremio

note

Before configuring, ensure that unixODBC is installed.

In Dremio, generate a personal access token. See Personal Access Tokens.

If you want to base your configuration on examples, copy the content of the `odbc.ini` and `odbcinst.ini` files in the `/opt/arrow-flight-sql/odbc-driver/conf` directory and paste the content into your system `/etc/odbc.ini` and `/etc/odbcinst.ini` files.

To configure the properties in the odbc.ini file:

For `HOST`, specify `data.dremio.cloud` for the US control plane, or `data.eu.dremio.cloud` for the European control plane.

For `PORT`, specify `443`.

For `TOKEN`, specify a personal access token.

Ensure that the value of `SSL` is `1`.

For additional parameters, see Connection Parameters.

note

To find out unixODBC has created your `odbc.ini` and `odbcinst.ini` files, run this command: `odbcinst -j`

If you ever need to enable tracing for troubleshooting problems with the driver, see the help for unixODBC.

macOS

Download and Install

To download and install the Arrow Flight SQL ODBC driver:

Intel Macs Only

This driver only supports Intel-based Macs. It is not compatible with Apple Silicon M1, M2, and M3 processors.

- Download the macOS driver version from the ODBC driver download page.
- Go to the download location and double-click the downloaded `.dmg` file.
- Double-click the `.pkg` file.
- In the **Welcome to the Arrow Flight SQL ODBC Driver Installer** page, click **Continue**.
- In the **Standard Install on "Macintosh HD"** page, Click **Install**. Optionally, if you want to change the install location, click **Change Install Location** and navigate to the new location.
- In the **Installer is trying to install new software** dialog, specify your macOS password. Then, click **Install Software**.
- After the installation is complete, click **Close**.

Connect to Dremio

note

Before configuring, follow these steps:

- Ensure that ODBC Manager is installed.
- In Dremio, generate a personal access token. See Personal Access Tokens.

To configure a system DSN:

- Launch ODBC Manager.
- On the User DSN page, select **Arrow Flight SQL ODBC DSN** and click **Configure**.
-

(Optional) Change the DSN.

In the **Host** field, specify `data.dremio.cloud` for the US control plane, or `data.eu.dremio.cloud` for the European control plane.

In the **Port** field, specify `443`.

Select the **UID** field and click **Remove**.

Select the **PWD** field and click **Remove**.

In the **UseEncryption** field, specify `true`.

Click **Add** to add a line for a new parameter. Change `keyword` to `TOKEN`, Paste a personal access token as the value.

For additional parameters, see Connection Parameters.

If you ever need to enable tracing for troubleshooting problems with the driver, see the help for your driver manager.

Connection Parameters

Primary Connection Parameters

Use these parameters to configure basic connection details such as what data source to connect with.

note

The Arrow Flight SQL ODBC driver does not support password-protected `.pem`/`.crt` files or multiple `.crt` certificates in a single `.pem`/`.crt` file.

Name	Type	Description	Default Value
Host	string	`data.dremio.cloud` for the US control plane, `data.eu.dremio.cloud` for the European control plane.	None
Port	integer	Sets the TCP port number that Dremio uses to listen to connections from ODBC clients.	443
Schema	string	Provides the name of the database schema to use by default when a schema is not specified in a query. However, this does not prevent queries from being issued for other schemas. Such queries must explicitly include the schema.	None
Token	string	Sets the personal access token to use when authenticating to Dremio. See Creating a Token for the steps to generate a personal access token.	None

Encryption Parameters

Use the following parameters to configure SSL encryption and verification methods for regular connections.

Name	Type	Description	Default Value
useEncryption	integer	Forces the client to use an SSL-encrypted connection to communicate with Dremio. Accepted values include: `true`: The client communicates

with Dremio only using SSL encryption. This is the only possible value. `false`: The value cannot be false. | true |
disableCertificateVerification	integer	Specifies whether the driver should verify the host certificate against the trust store. Accepted values are: `false`: The driver verifies the certificate against the trust store. `true`: The driver does not verify the certificate against the trust store.	false
useSystemTrustStore	integer	Controls whether to use a CA certificate from the system's trust store, or from a specified .pem file. `true`: The driver verifies the connection using a certificate in the system trust store. `false`: The driver verifies the connection using the .pem file specified by the trustedCerts parameter.	true on Windows and macOS, false on Linux (which does not have a system truststore)
trustedCerts	string	The full path of the .pem file containing certificates trusted by a CA, for the purpose of verifying the server. If this option is not set, then the driver defaults to using the trusted CA certificates .pem file installed by the driver. The exact file path varies according to the operating system on which the driver is installed. The path for the Windows driver is different from the path set for the macOS driver. The TLS connection fails if you do not specify a value when useEncryption is true and disableCertificateVerification is false.	N/A

Advanced Parameters

Name	Type	Description	Default Value
quoting	string	Specifies which type of character to use to delimit values in queries. The value can be `BACK_TICK`, `BRACKET`, or `DOUBLE_QUOTE`.	`DOUBLE_QUOTE`
routing_tag	string	When this parameter is set, the specified tag is associated with all queries executed within a session. Rules can check for the presence of a tag with the function "tag()". For more information, see Workload Management.	N/A
stringColumnLength	string	The maximum length of data in columns of the STRING datatype and of complex datatypes. The range is 1 to 2147483647.	1024

Example of a Basic Connection String

Some BI client applications, such as Microsoft Excel, let you specify a connection string, rather than select a DSN, for connecting to Dremio. If you want to connect by using a connection string, you can use this example, basic connection string as a basis for your own:

Example connection string

```
host=data.dremio.cloud;port=443;useEncryption=1;disableCertificateVerification=1;token=<personal-access-token>
```

Supported Conversions from Dremio Datatypes to ODBC Datatypes

| Dremio Data Types | SQL_C_BINARY | SQL_C_BIT | SQL_C_CHAR | SQL_C_WCHAR | SQL_C_STINYINT | SQL_C_UTINYINT | SQL_C_SSHORT | SQL_C_USHORT | SQL_C_SLONG | SQL_C_ULONG | SQL_C_SBIGINT | SQL_C_UBIGINT | SQL_C_FLOAT | SQL_C_DOUBLE | SQL_C_NUMERIC | SQL_C_DATE | SQL_C_TIME | SQL_C_TIMESTAMP | SQL_C_GUID | SQL_C_INTERVAL_* |

| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
BOOLEAN	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Invalid	Invalid	Invalid	Invalid	N
VARBINARY	Y	Invalid	N	N	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid
DATE	N	Invalid	Y	Y	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Y	Invalid	Y	Invalid	Invalid
FLOAT	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Invalid	Invalid	Invalid	Invalid	N
DECIMAL	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Invalid	Invalid	Invalid	Invalid	N
DOUBLE	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Invalid	Invalid	Invalid	Invalid	N
INTERVAL (day to seconds)	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Invalid	Invalid	Invalid	N
INTERVAL (years to months)	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Invalid	Invalid	Invalid	N
INT	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Invalid	Invalid	Invalid	Invalid	N
BIGINT	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Invalid	Invalid	Invalid	Invalid	N
TIME	N	N	Y	Y	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Y	Y	Invalid	Invalid
TIMESTAMP	N	N	Y	Y	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Y	Y	Invalid	Invalid
VARCHAR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Invalid
STRUCT	N	N	Y	Y	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid
LIST	N	Invalid	Y	Y	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid	Invalid

Was this page helpful?

Prerequisites

Supported Authentication Method

Windows

Download and Install

Connect to Dremio

Linux

Download and Install

Connect to Dremio

macOS

Download and Install

Connect to Dremio

Connection Parameters

- Primary Connection Parameters
- Encryption Parameters
- Advanced Parameters
- Example of a Basic Connection String
- Supported Conversions from Dremio Datatypes to ODBC Datatypes

Dremio JDBC (Legacy) Release Notes | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/drivers/release-notes/jdbc-legacy

On this page

This article contains the release notes for the Dremio JDBC driver (Legacy). See Dremio JDBC Driver (Legacy) for documentation.

note

The Arrow Flight SQL JDBC driver is the recommended driver for connectivity to Java-based applications. The Dremio JDBC driver (Legacy) will not be updated or fixed moving forward.

Legacy JDBC Driver 17.0.0 (June 2021)

Connection.getCatalog() would always return `null`.
Connection.getCatalog() now returns the current catalog for the connection.

Legacy JDBC Driver 15.2.0 (March 2021)

- Dremio uses the local timezone rather than UTC for datetime values.

Legacy JDBC Driver 15.0.0 (March 2021)

- Provides a `useSystemTrustStore` property that bypasses `trustStoreType` and automatically selects the correct Truststore based on the operating system. See JDBC Parameters for Dremio Wire Encryption for more information.
- Dremio no longer maps empty usernames to anonymous. Rather, Dremio treats empty usernames as empty.

Legacy JDBC Driver 14.0.0 (February 2021)

- Provides a new class loader from a previously-loaded class when no class loader is available for a thread.

Legacy JDBC Driver 11.0.0 (November 2020)

- Support for TLS SNI when connecting to a TLS-enabled Dremio deployment. Dremio implicitly sets the TLS SNI property to the hostname used in the connection string.

Was this page helpful?

- Legacy JDBC Driver 17.0.0 (June 2021)
- Legacy JDBC Driver 15.2.0 (March 2021)
- Legacy JDBC Driver 15.0.0 (March 2021)
- Legacy JDBC Driver 14.0.0 (February 2021)
- Legacy JDBC Driver 11.0.0 (November 2020)

Arrow Flight SQL JDBC Release Notes | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/drivers/release-notes/arrow-flight-sql-jdbc

On this page

You can connect to Dremio through the Arrow Flight SQL JDBC driver. The driver is open-source and you are free to use it with Dremio's data lakehouse platform or any other data platform that has an Arrow Flight SQL endpoint.
These release notes summarize Dremio-specific updates, compatibility notes, and limitations.

For more information about this driver, see Arrow Flight SQL JDBC.

Version 10.0.0 (November 2022)

Security

Transport Layer Security (TLS) communication is enabled by default for Arrow Flight client applications.

Limitations

- Time offsets are not being reported in query results.
- User impersonation is not yet supported.

Recommendation

It is recommended to use the Arrow Flight SQL JDBC driver instead of the Dremio JDBC (Legacy) driver. The Dremio JDBC (Legacy) driver will not be updated or fixed moving

forward.

Was this page helpful?

Version 10.0.0 (November 2022)

Security

Limitations

Recommendation

Arrow Flight SQL ODBC Release Notes | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/explore-analyze/client-apps/drivers/release-notes/arrow-flight-sql-odbc

On this page

This article contains the release notes for the Arrow Flight SQL ODBC driver.

See Arrow Flight SQL ODBC for documentation.

0.9.7 (August 2025)

Issues Fixed

General Updates

Fixed an issue on macOS where the error was sometimes not displayed in Microsoft Excel.

DX-90575

General Updates

Fixed an issue with an inconsistent searchable attribute returned by the `SQLColAttribute` function.

DX-102851

General Updates

Fixed an issue where Microsoft Excel was not showing small decimals correctly.

DX-104574

0.9.6 (June 2025)

Issues Fixed

General Updates

Resolved an issue where the Arrow Flight SQL ODBC driver failed to connect to TLS-secured Flight endpoints.

General Updates

Fixed an issue where calling metadata functions like `SQLPrimaryKeysW` or `SQLForeignKeysW` caused an error.
The driver now handles these calls more gracefully.

0.9.5 (May 2025)

What's New

General Updates

A new driver configuration flag is available for macOS (Intel and Apple Silicon): `hideSQLTablesListing`.

DX-101630

0.9.4 (April 2025)

What's New

General Updates

The Arrow Flight SQL ODBC driver now supports Apple Silicon. Download the driver.

General Updates

Upgraded to Arrow Flight v9 for enhanced compatibility and performance.

Issues Fixed

General Updates

Fixed date handling for pre-1970 dates in Microsoft tools.

General Updates

Fixed segmentation fault in Arrow Flight SQL ODBC Driver.

Was this page helpful?

0.9.7 (August 2025)

0.9.6 (June 2025)

0.9.5 (May 2025)

0.9.4 (April 2025)

Source: dremio-cloud-get-started.md

Get Started with Dremio Cloud | Dremio

Documentation

Original URL: https://docs.dremio.com/dremio-cloud/get-started/

On this page

To get started, sign up for an account at dremio.com/get-started and follow the guided setup to create your first project.

This guide shows you how to analyze transportation data and find usage patterns using natural language queries with Dremio's AI Agent. You'll work with the `dremio_samples.nyc_citibikes.citibikes` table that contains over 115 million bike-sharing records from New York City—real patterns from one of the world's busiest transportation networks.

Step 1: Discover and Explore Your Data

Understanding your dataset structure is crucial before diving into analysis. Start by getting an overview of the bike-sharing data to understand what insights are possible.

On the homepage, enter the following prompt into Dremio's AI Agent chat box: `Give me an overview of the nyc_citibikes.citibikes dataset`.

The AI Agent uses capabilities like semantic search to find relevant datasets. Review the response to understand the schema, field definitions, and what types of analysis are possible with bike-sharing data.

Step 2: Compare User Behavior

Ask a question in natural language to compare how different user types interact with the bike-sharing service. This analysis reveals usage patterns that inform operational decisions.

Ask the AI Agent to compare how subscribers and casual riders use the service: `Give me the total number of rides and the average trip duration grouped by user type across the dataset.`

The AI Agent generates and runs SQL using Dremio's query engine to answer this question. The results show that subscribers typically show higher ride frequency but shorter durations, indicating commuter behavior. Casual riders often have longer trips but lower frequency, suggesting leisure or tourist usage patterns. This analysis reveals how different user types require different operational strategies for bike availability, pricing models, and infrastructure investment.

Step 3: Analyze Peak Demand Patterns

Dive deeper to understand when different user groups are most active. This temporal analysis provides insights for operational planning and resource allocation.

Ask the AI Agent to reveal hourly demand patterns: `Analyze hourly ride patterns to find when demand peaks. Show a line chart of total rides per hour of the day, separated by user type, and include a short report highlighting the busiest hours for each group and recommendations for bike availability`.

The AI Agent will create a clear chart showing distinct patterns. Subscribers peak during rush hours (8-9 AM, 5-6 PM) indicating commuter usage, while casual riders peak during midday and weekends showing leisure patterns. This temporal analysis reveals opportunities for dynamic pricing during peak hours, optimal timing for maintenance during low-demand periods, and capacity planning insights for fleet optimization.

Step 4: Run Comparative Analysis

Use the AI Agent to identify the most influential factors affecting rider behavior. This analysis compares multiple variables to determine primary drivers of ridership patterns.

Ask the AI Agent to run a comprehensive comparative analysis: `Run comparative analysis on seasonal, daily, hourly, and bike type patterns to identify which factor has the most significant impact on ride behavior. Then create a detailed visualization of the most influential factor.`

The AI Agent will compare multiple variables and automatically identify which factor has the biggest impact on rider behavior, complete with actionable recommendations. This analysis reveals primary drivers of ridership, correlation insights between variables, and predictive indicators for forecasting usage patterns.

Step 5: Try Your Own Analysis

Now that you understand how to analyze transportation data with the AI Agent, try exploring other bike-sharing questions.

You can also analyze your own data using Dremio's AI Agent.

Summary

You have completed a transportation data analysis using natural language. You explored the dataset structure, compared user behavior patterns, analyzed peak demand times, and identified influential factors affecting ridership. Discovery, exploration, and analysis that could previously take hours can now be done in minutes with Dremio's AI Agent.

Troubleshoot

- If a prompt doesn't work as expected, try simplifying the request or verifying that you're referencing the correct dataset (`nyc_citibikes.citibikes`).
- Contact your administrator if sample data isn't available in your environment.

Related Topics

- Bring Your Data – Load, connect, and prepare your data.
- Quick Tour of the Dremio Console – Learn how to navigate Dremio.
- Add a User – Invite team members to your organization.

Was this page helpful?

- Step 1: Discover and Explore Your Data
- Step 2: Compare User Behavior
- Step 3: Analyze Peak Demand Patterns
- Step 4: Run Comparative Analysis
- Step 5: Try Your Own Analysis
- Summary
- Troubleshoot
- Related Topics

Build Your First Agentic Lakehouse Use Case | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/get-started/use-case

On this page

This guide will help you turn a business question into a working, governed data product using your own data, all within your 30-day Dremio Cloud trial.

In the Getting Started guide, you saw how Dremio's AI Agent can take you from question to insight within minutes using sample data. With this guide, you'll connect your data, prepare and transform it, build reusable views with semantics, and deliver insights using Dremio's AI Agent or your preferred tool of choice. The end goal is flexible: you might produce an aggregated view that analysts and AI agents query regularly, or a dashboard. Either way, you'll experience the full value of Dremio Cloud as an agentic lakehouse — open, governed, and self-optimizing.

Prerequisites

Before you begin, ensure that you have the following:

- A Dremio Cloud account: You'll need an active Dremio Cloud trial account. If you haven't already, sign up at dremio.com/get-started for a 30-day trial with $400 in free credits.
- Access to data: Identify at least one data source you can connect to, such as object storage or databases. If you don't have these, then you will need a set of local files that you can upload to Dremio Cloud.
- (Optional) Access to your BI tool: If your use case requires a dashboard, you will need access to your BI tool. This is optional, as you can use Dremio's AI Agent to generate basic charts to visualize trends directly within the Dremio console.

Step 1: Identify a Business Use Case

Begin by identifying a business use case that you will be implementing using this guide. The use case you choose should have clear value and measurable results, not a massive data project but a business question that matters.

How to Do It

Pick a concrete business question, such as:

How are customer support metrics trending this quarter?

Which product lines are driving margin growth?

What are our top churn risks by region?

We recommend that you select a business question that can be answered using a few datasets, going across a maximum of two sources.

Step 2: Add Your Data

To implement the data model that answers the business question you identified in the Identify a Business Use Case section, you will first add your data to the project in your Dremio Cloud account.

How to Do It

You can add data to your project in one of three ways:

Load Data into the Open Catalog: Dremio provides a default Open Catalog, powered by Apache Polaris. You can load data directly into this catalog as an Iceberg table in your silver layer using your tool of choice, such as Fivetran, dbt, and Airbyte. From there, Dremio manages all Iceberg table metadata and governance while keeping your data in an open format. For instructions on how to load data into your Open Catalog, see Load Data into Tables.

Connect an Existing Source: Connect your object store, catalogs, databases so Dremio can query data in place. This is your bronze layer of data. For a list of supported sources and step-by-step connection instructions, see Connect to Your Data.

Upload Local Files: Upload local files (CSV, JSON, or Parquet) for quick exploration if you don't have direct access to your data sources. Dremio will write the uploaded data into an Iceberg table in your project's Open Catalog. For step-by-step instructions on how to upload files, see Upload Local Files.

Whichever method you choose, Dremio provides live, federated access to all of your data. This flexibility allows you to move from data connection to analysis in minutes.

Step 3: Clean and Transform Data

Dremio lets you prepare data from across different sources without having to move it. You're able to use natural language to generate the SQL using Dremio's AI Agent or write SQL yourself. Your data preparation steps can be represented as views; no additional pipelines are required.

How to Do It

Use SQL and AI Functions: Prepare and transform data using SQL Functions. You can also turn unstructured data, such as images or PDFs, into a structured, governed Iceberg table using AI Functions.

Use Dremio's AI Agent: Ask the built-in AI Agent to identify issues with the data and generate SQL to prepare and transform it. For example, you can ask the AI Agent to:

Generate SQL to remove null values in the revenue column.

Generate SQL to join orders and customers on customerID.

Add a column for gross margin = revenue - cost.

Each transformation can be saved as a view in your silver layer, giving you reusable building blocks. This way, your transformations are continuously updated as more data comes in with no additional changes required from you. This approach replaces complex ETL pipelines with a simple workflow that keeps your data fresh, governed, and easy to iterate on. For instructions on how to create views, see Create a View.

Step 4: Build Views for Aggregations and Metrics

Once you've created your silver layer by cleansing and transforming your data, you can create your gold layer of views. These views will capture aggregations and metrics and will be ready for exploration, ad-hoc analysis, or dashboards.

How to Do It

Use SQL Functions: Aggregate and build out metrics using SQL Functions.

Use Dremio's AI Agent: Ask the built-in AI Agent to generate the SQL for your view. For example, you can ask the agent to *Give me the SQL for views that summarize the average response time by call center employees and the customer sentiment by region.*

Aggregations and metrics are saved as governed views in your Open Catalog. For instructions on how to create views, see Create a View.

Step 5: Add Semantics to Views

Data only becomes valuable when everyone can interpret it in the same way. The AI Semantic Layer gives your datasets shared meaning, so when an analyst or AI Agent is looking at "fiscal Q2" or "positive sentiment", they're applying the same business logic every time.

How to Do It

Enrich Your Data with Semantics: Generate wikis and labels on your views to reduce the amount of time being spent on manual tasks. For more information on generating semantics, see Generate Wikis and Labels.

You can add additional context, such as usage notes, definitions specific to your industry, and common queries.
These definitions and classifications are stored with the data, guiding both natural language queries, SQL generation, and manual exploration.

Step 6: Deliver Insights

Now that you have connected, curated, aggregated, and enriched your data, you can deliver on the outcome for the business question you defined in Step 1. The outcome may be the aggregated view you created in the previous step that teams and agents will use directly, or it may be a dashboard that tracks metrics over time. With Dremio Cloud, you're able to deliver on either one.

How to Do It

Use Dremio's AI Agent for Actionable Insights: Dremio's AI Agent can analyze patterns and trends directly from views. You and your users can ask the business question you identified in Step 1, along with other questions. The AI Agent will use the semantics and samples of the data to generate the appropriate SQL queries that provide you with insights and visualizations of the data. For example, on sales data, you can ask the AI Agent to *Create a chart to show the trends in sales across regions over the last year and provide an analysis on the changes.*

Create a Dashboard Using Your Tool of Choice: If you already have a dashboard or report that you would like to update or you want to create a new one to represent the insights on your data, you can connect to Dremio from tools like Tableau, Microsoft Power BI, and others using Flight SQL JDBC/ODBC connections. For a list of supported tools and step-by-step instructions on connecting, see Connect Client Applications.

Step 7: Operationalize the Use Case

Each use case is operationalized when it's governed, monitored, and shareable.

How to Do It

Access Control Policies: Create and implement access control policies from role-based access to more granular row and column-level policies. For more information, see Privileges and Row-Access and Column-Masking Policies.

Monitor Query Volumes and Performance: Track performance and usage of the data. Dremio's Autonomous Management capability automatically handles data management and ensures reliable and fast query performance. In Dremio, you're able to monitor queries and their performance.

Cost Management: Review consumption and spend of this use case within the Dremio console. These dashboards show how much compute and storage each workload consumes, helping you plan budgets, optimize workloads, and estimate spend before moving to production. For more information, see Usage.

Operationalizing your first use case ensures it remains reliable, governed, and cost-effective. You gain insight into both performance and consumption trends, enabling you to scale confidently while maintaining control of your budget.

Wrap Up and Next Steps

You've now implemented your first use case on Dremio Cloud by:

- Defining a valuable business use case
- Adding your own data to your Open Catalog or by connecting existing data sources
- Cleaning and transforming the data
- Creating reusable views with semantics
- Delivering insights via AI or dashboards
- Operationalizing the data through governance and monitoring

Next, extend your use case with additional business questions or another business domain.

Related Topics

- Dremio MCP Server - Use Dremio's hosted MCP server to customize your agentic workflow.
- Visual Studio Code - Use the Visual Studio (VS) Code extension for Dremio for development and analysis.
- Optimize Performance - Learn about how Dremio autonomously optimizes performance.

Was this page helpful?

How to Do It

Wrap Up and Next Steps

Related Topics

Quick Tour of the Dremio Console | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/get-started/quick-tour

On this page

This quick tour introduces you to the main areas of the Dremio console, including the homepage, Datasets, SQL Runner, and Jobs pages. You'll learn how to navigate the interface and access key features of your agentic lakehouse.

Console Navigation

The side navigation bar provides links to key areas of the Dremio console.

!Dremio console navigation.

Location	Description
1	**Homepage**: Central landing page when you log in.
2	**Datasets**: Interface for exploring tables and views across the default Open Catalog, other catalogs, object storage, and database sources.
3	**SQL Runner**: Editor for constructing and querying data.
4	**Jobs**: History of executed SQL and job details.
5	**Project and Organization Settings**: Configuration for your catalog, engines, and routing rules in your project and management of authentication, users, billing, and projects in your organization.
6	**Documentation and Support**: Access point for documentation, the Community Forum, or the Support Portal.
7	**Account Settings**: Section for managing general information, personal access tokens, appearance preferences, and logout options.

Datasets Page

The Datasets page provides navigation and management for data in your Open Catalog, other catalogs, object stores, and databases.

!Datasets page navigation and management interface.

Location	Description
1	**Project Name**: Name of the current project being explored.
2	**Namespaces**: Logical containers that organize data objects within Dremio's Open Catalog, providing hierarchical organization and access control for tables, views, and folders.
3	**Sources**: Self-hosted catalogs, object stores, or databases.
4	**Path**: Dot-separated identifier indicating the location of the object, starting with

the source or catalog name, followed by any folders, and ending with the name of table or view. |

SQL Runner

The SQL Runner provides a query editor for running SQL. Access via in the side navigation bar.

!SQL Runner interface

Location	Description
1	**Data Panel**: Area for exploring data across your Open Catalog, other catalogs, object stores, and databases, with drag-and-drop support for adding objects into the SQL editor.
2	**Scripts Panel**: Panel for saved SQL scripts that can be reused and shared with other users in your organization. Each script includes creation/modification timestamps and editor context and requires VIEW privileges.
3	**SQL Editor**: Workspace for creating and editing SQL with autocomplete, syntax highlighting, and function lookup. See SQL Reference for supported SQL. You may also highlight SQL, right-click, and select **Explain SQL** to start a chat with the AI Agent, which features a summary of the query's overview, datasets, and architecture. For more details, see Explain SQL.
4	**Run**: Execution of the SQL, which returns the complete result set.
5	**Preview**: Option for previewing the result set, which returns a subset of rows in less time than running the SQL.
6	**Engine**: Dropdown menu for selecting an engine for SQL execution. By default, Automatic is selected, which routes the query to the appropriate engine based on engine routing rules. For more details, see Manage Engines.
7	**Results Panel**: Table displaying the results of your query with options to download, copy, or edit values.
8	**Job Summary**: Tab showing the job status, query type, start time, duration, and job ID.
9	**Transformations**: Tools for applying transformations such as Add Column, Group By, Join, Filter, Convert Data Type, and Add Calculated Field that automatically update SQL.
10	**Execution State**: Indicator displaying the job status, record count, and execution time, with a link to view full job details. Includes options to download results as JSON, CSV, or Parquet files, or copy data to the clipboard.
11	**Details Panel**: Right-side panel for viewing and managing dataset metadata, including columns, ownership, searchable labels, and wiki content.

Limitations and Considerations

Row Limit: `COUNT(*)` and `SELECT` query results are limited to one million rows and may be truncated based on thread distribution. When truncated, a warning appears. To obtain complete results, use JDBC or ODBC drivers.

CSV Download: CSV download is unavailable for result sets with complex data types (union, map, array). The download and copy results options can be enabled or disabled for a specific project by navigating to **Project Settings** > **Preferences**.

Was this page helpful?

Console Navigation

Datasets Page

SQL Runner

Limitations and Considerations

Source: dremio-cloud-help-support.md

Help and Support | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/help-support/

The section contains additional details including:

Limits

Well-Architected Framework

Keyboard Shortcuts

Was this page helpful?

Limits | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/help-support/limits

On this page

For each organization, Dremio imposes limits on the use of its resources. The following limits are grouped according to Dremio components.

Please contact Dremio to discuss if extra capacity is required.

Organization

Item	Enterprise Trial	Enterprise Paid
Number of projects	1	200
Enterprise identity providers	1	1
Number of users	5	15,000
User invitations at once	5	10
Pending user invitations	5	100
Daily user invitations	5	100
Number of custom roles	10	5,000
Layers of nested roles	10	10
Direct custom role members	5	1,000

Projects

| Item | Enterprise Trial | Enterprise Paid |

| --- | --- | --- |
Number of engines	1	50
Number of sources	100	100
Folder nesting depth	8	8
Number of tables	Unlimited	Unlimited
Number of views	Unlimited	Unlimited
Number of scripts per user	1,000	1,000
ACLs update rate per minute	600	600

Engines

Item	Enterprise Trial	Enterprise Paid
Replica sizes	XS	2XS, XS, S, M, L, XL, 2XL, 3XL
Number of replicas	3	100
Query concurrency	The query concurrency limits are determined by the replica sizes as described in Engines.	The query concurrency limits are determined by the replica sizes as described in Engines.
Query runtime max limit	Min 30 seconds	Min 30 seconds

Datasets

Item	Enterprise Trial	Enterprise Paid
Metadata refresh time - data lake	15 minutes	15 minutes
Metadata refresh time - RDBMS	1 hour	1 hour
Reflection refresh frequency	1 hour	1 hour
Wiki character limit	100k characters	100k characters
Number of JSON files	300,000	300,000
Row width	16 MB	16 MB

Reflections

Item	Enterprise Trial	Enterprise Paid
Maximum number of Reflections (including enabled and disabled Reflections)	500	500
Autonomous Reflections	100	100

Arrow Flight SQL (ADBC, ODBC, and JDBC)

Item	Enterprise Trial	Enterprise Paid
Max returned data volume	10GB	10GB
Flight Service Data Pipeline Drain Timeout	50 seconds	50 seconds

Rate Limits

Rate limits are enforced on a single IP address and apply across all organizations and projects.

Item	Enterprise Trial	Enterprise Paid
Login rate per user per second	100	100
SCIM reads per minute - user	180	180
SCIM writes per minute - user	300	300
API calls per minute	1,200	1,200
API: `/job/{id}/results` calls per minute	1,000	1,000
API: `/job/{id}/cancel` calls per minute	100	100
API: `/job/{id}` calls per minute	100	100
API: `/login/userpass` calls per second	45	45
Access control list update rate per minute	60	60

Was this page helpful?

- Organization
- Projects
- Engines
- Datasets
- Reflections
- Arrow Flight SQL (ADBC, ODBC, and JDBC)
- Rate Limits

Keyboard Shortcuts | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/help-support/keyboard-shortcuts

On this page

Keyboard shortcuts for functions supported by Dremio are available for macOS, Windows, and Linux.

SQL Editor

While using the SQL editor on the SQL Runner page or Datasets page, you can use shortcuts for commonly used actions, as shown in the following table:

Function	macOS Shortcut	Windows/Linux Shortcut
Preview	Cmd + Enter	Ctrl + Enter
Run	Cmd + Shift + Enter	Ctrl + Shift + Enter
Search	Cmd + K	Ctrl + K
Comment Out/In	Cmd + /	Ctrl + /
Find	Cmd + f	Ctrl + f
Trigger Autocomplete	Ctrl + Space	Ctrl + Space
Format Query	Cmd + Shift + f	Ctrl + Shift + f
Delete Line	Cmd + Shift + k	Ctrl + Shift + k
Toggle AI Agent	Cmd + Shift + g	Ctrl + Shift + g

Was this page helpful?

SQL Editor

Well-Architected Framework | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/help-support/well-architected-framework/

On this page

Dremio’s well-architected framework is a resource for anyone who is designing or operating solutions with Dremio. It provides insight from lessons learned through helping hundreds of customers be successful. The framework is composed of pillars that describe design principles as well as best practices based on those principles.

The Well-Architected Framework is considered complementary to the Dremio Shared Responsibility Model. The Shared Responsibility Model lays out Dremio's responsibilities and your responsibilities for maintaining and operating an optimal Dremio environment, while the Well-Architected Framework provides details for carrying out your responsibilities.

Key Pillars of Dremio’s Well-Architected Framework

Dremio’s well-architected framework follows five common pillars from cloud providers AWS, Microsoft, and Google and a sixth Dremio-specific pillar:

- Security
- Performance Efficiency
- Cost Optimization
- Reliability
- Operational Excellence
- Self-Serve Semantic Layer

Each pillar includes principles, best practices, and how-to articles on the pillar's theme.

Dremio's well-architected framework covers best practices related to configuration and operation of Dremio. Read Architecture for more information about the Dremio architecture.

Was this page helpful?

- Key Pillars of Dremio’s Well-Architected Framework

Security | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/help-support/well-architected-framework/security

On this page

The security pillar is essential to ensuring that your data is secured properly when using Dremio to query your data lakehouse. The security components are especially important to architect and design your data platform. After your workloads are in production, you must continue to review your security components to ensure compliance and eliminate threats.

Principles

Leverage Industry-Standard Identity Providers and Authorization Systems

Dremio integrates with leading social and enterprise identity providers and data authorization systems. For robust enterprise integration with corporate policies, it is essential to leverage those third-party systems. We recommend systems that use multi-factor authentication methods and are connected to single sign-on (SSO) platforms.

Design for Least-Privilege Access to Objects

When providing self-service access to your data lakehouse via Dremio's AI semantic layer, access should only be granted to the data that is required for the role accessing the data.

Best Practices

Protect Access Credentials

Where possible, leverage identity providers such as Microsoft Entra ID and Okta in conjunction with System for Cross-domain Identity Management (SCIM) where applicable to ensure that you never need to share passwords with Dremio. SSO with Microsoft Entra ID or Okta is also recommended where possible.

Leverage Role Based Access Controls

Access to each catalog, folder, view, and table can be managed and regulated by roles. Roles are used to organize privileges at scale rather than managing privileges for each individual user. You can create roles to manage privileges for users with different job functions in your organization, such as "Analyst" and "Security_Admin" roles. Users who are members of a role gain all of the privileges granted to the role. Roles can also be nested. For example, the users in the "UK" role can automatically be members of the "EMEA" role.

Access control protects the integrity of your data and simplifies the data architecture available to users based on their roles and responsibilities within your organization. Effective controls allow users to access data that is central to their work without regard for the complexities of where and how the data is physically stored and organized.

Was this page helpful?

-

Principles

Leverage Industry-Standard Identity Providers and Authorization Systems

Design for Least-Privilege Access to Objects

Best Practices

Protect Access Credentials

Leverage Role Based Access Controls

Reliability | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/help-support/well-architected-framework/reliability

On this page

The reliability pillar focuses on ensuring your system is up and running and can be quickly and efficiently restored in case of unexpected downtime.

Principles

Set Engine Routing Rules and Engine Settings

Dremio’s engine routing rules and engine settings are powerful and protect the system from being overloaded by queries that exceed currently available resources.

Monitor and Measure Platform Activity

To ensure the reliability of your Dremio project, you must regularly monitor and measure its activity.

Best Practices

Initialize Engine Routing and Engine Settings

It is important to set up engine routing rules and engines with sensible concurrency, replica, and time limits. It's better to spin replicas at sensible concurrency limits rather than risk a large number of rogue queries bringing down the engine.

Use the Monitor Page in the Dremio Console

As an administrator using the Dremio console, you can effectively monitor catalog usage and jobs within your projects. The Monitor page provides detailed visualizations and metrics that allow you to track usage patterns, resource consumption, and user impact.

In the Catalog Usage tab, you can view the 10 most-queried datasets and source folders, along with relevant statistics such as linked jobs and acceleration usage. The Catalog Usage tab excludes system tables and INFORMATION_SCHEMA datasets and focuses solely on user queries.

In the Jobs tab, you can access comprehensive metrics on job performance, including daily job counts, failure rates, and user activity. Visualizations include graphs of completed and failed jobs, job states, and the 10 longest-running jobs, providing an overview of job execution and performance trends.

We recommend that administrators frequently review the Monitor page, including daily consumption patterns and the weekly and monthly aggregate. Monitoring insights like the most queried datasets over time can help administrators optimize performance, adapt a Reflection strategy, and leverage the jobs-per-engine distribution to improve workload management and resource allocation.

Perform Impact Analysis if Security Rules Change

Dremio’s control plane interacts with your own virtual private clouds for query execution. If you make changes to your security rules after they are initially set and working correctly with Dremio, perform impact analysis to make sure that your connectivity with Dremio remains unaffected.

Was this page helpful?

- Principles
- Set Engine Routing Rules and Engine Settings
- Monitor and Measure Platform Activity
- Best Practices
- Initialize Engine Routing and Engine Settings
- Use the Monitor Page in the Dremio Console
- Perform Impact Analysis if Security Rules Change

Cost Optimization | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/help-support/well-architected-framework/cost-optimization

On this page

Although it's important to get the best performance possible with Dremio, it's also important to optimize costs associated with managing the Dremio platform.

Principles

Minimize Running Executor Nodes

Dremio can scale to many hundreds of nodes, but any given engine should have only as many nodes as are required to satisfy the current load and meet service-level agreements.

Dynamically Scale Executor Nodes Up and Down

When running Dremio engines, designers can leverage concurrency per replica and minimum and maximum number of replicas to dynamically expand and contract capacity based on load.

Eliminate Unnecessary Data Processing

As described in the best practices for Pillar 2: Performance Efficiency, creating too many Reflections, especially those that perform similar work to other Reflections or provide little added benefit in terms of query performance, can incur unnecessary costs because Reflections need system resources to rebuild. For this reason, consider removing any unnecessary Reflections.

To avoid the need to process data that is not required for a query to succeed, use filters that can be pushed down to the source wherever possible. Enabling partitioning on source data that are in line with the filters also helps speed up data retrieval.

Also, optimize source data files by merging smaller files or splitting larger files whenever possible.

Best Practices

Size Engines to the Minimum Replicas Required

To avoid accruing unnecessary cost, reduce the number of active replicas in your engines to the minimum (typically 1, but 0 when the engine is not in use on weekends or non-business hours). A minimum replica count of 0 delays the first query of the day due to engine startup, which you can mitigate with an external script that executes a dummy SQL statement prior to normal daily use.

Remove Unused Reflections

Analyze the results in Dremio `sys.project.jobs_recent` system table along with the results for the system tables `sys.project.reflections` and `sys.project.materializations` to get information about the frequency at which each Reflection present in Dremio is leveraged. You can further analyze Reflections that are not being leveraged to determine if any are still being refreshed, and if they are, how many times they have been refreshed in the reporting period and how many hours of cluster execution time they have been consuming.

Checking for and removing unused Reflections is good practice because it can reduce clutter in the Reflection configuration and often free up many hours of cluster execution cycles that can be used for more critical workloads.

Optimize Metadata Refresh Frequency

Ensure metadata-refresh frequencies are set appropriately based on what you know about the frequency that metadata is changing in the data source.

The default metadata refresh frequency set against data sources is once per hour, which is too frequent for many data sources. For example, if data in the sources are only updated once every 6 hours, it is not necessary to refresh the data sets every hour. Instead, change the refresh schedule to every 6 hours in the data source settings.

Furthermore, because metadata refreshes can be scheduled at the data source level, overridden at each individual table level, and performed programmatically, it makes sense to review each new data source to determine the most appropriate setting for it. For example, for data lake sources, you might set a long metadata refresh schedule such as 3000 weeks so that the scheduled refresh is very unlikely to fire, and then perform the refresh programmatically as part of the extract, transform, and load (ETL) process, where you know when the data generation has completed. You might set relational data sources to refresh every few days, but then override the source-level setting for tables that change more frequently.

When datasets are updated as part of overnight ETL runs, it doesn’t make sense to refresh the dataset metadata until you know the ETL process is finished. In this case, you can create a script that triggers the manual refresh of each dataset in the ETL process after you know the dataset ETL is complete.

For data sources that contain a large number of datasets but few datasets that change their structure or have new files added, it makes little sense to refresh at the source level on a fixed schedule. Instead, set the metadata to a long source-level refresh timeframe like 52 weeks and use scripts to trigger a manual refresh against a specific dataset.

If you set the metadata refresh schedule for a long timeframe and you do not have any scripting mechanism to refresh your metadata, when a query runs and the planner notices that the metadata is stale or invalid, Dremio performs an inline metadata refresh during the query planning phase. This can have a negative impact on the duration of query execution because it also incorporates that metadata refresh duration.

Was this page helpful?

- Principles
- Minimize Running Executor Nodes
- Dynamically Scale Executor Nodes Up and Down
- Eliminate Unnecessary Data Processing
- Best Practices
- Size Engines to the Minimum Replicas Required
- Remove Unused Reflections
- Optimize Metadata Refresh Frequency

Operational Excellence | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/help-support/well-architected-framework/operational-excellence

On this page

Following a regular schedule of maintenance tasks is key to keeping your Dremio project operating at peak performance and efficiency. The operational excellence pillar describes the tasks required to maintain an operationally healthy Dremio project.

Principles

Regularly Evaluate Engine Resources

As workloads expand and grow on your Dremio project, it is important to evaluate engine usage to ensure that you have correctly sized engines and the right number of replicas.

Regularly Evaluate Query Performance

Regular query performance reviews help you identify challenges and mitigate them before they become a problem. For example, if you find an unacceptably large number of queries waiting on engine or replica starts, you can adjust the minimum, maximum, and last replica auto-stop settings. If you see an unacceptable number of query execution failures, you can adjust concurrency limits per replica more appropriately or revisit the semantic layer and introduce Reflections to improve performance.

Clean Up Tables with Vacuum

Open Catalog automates Iceberg maintenance operations like compaction and vacuum, which maximizes query performance, minimizes storage costs, and eliminates the need to run manual data maintenance.

Optimize Tables

When operating on Iceberg tables and using Open Catalog, you can schedule optimization jobs to help you manage the accumulation of data files that occurs through data manipulation language (DML) operations. Regular maintenance ensures optimal query performance on these tables.

Regularly Monitor Live Metrics for Dremio

To ensure smooth operations in Dremio, collect metrics and take action when appropriate. Read Monitor for more details.

Best Practices

Optimize Workload Management Rules

Because workloads and volumes of queries change over time, you should periodically reevaluate workload management engine routing rules and engines and adjust for optimal size, concurrency, and replica limits.

Configure Engines

When possible, leverage engines to segregate workloads. Configuring engine and usage offers the following benefits:

- Platform stability: if one engine goes down, it won’t affect other engines.
- Flexibility to start and stop engines on demand at certain times of day.
- Engines can be sized differently based on workload patterns.
- It's possible to separate queries from different tenants into their own engine to enable a chargeback model.

We recommend separate engines for the following types of workloads:

- Reflection refreshes.
- Metadata refreshes.
- API queries.
- Queries from BI tools.
- Extract, transform, and load (ETL)-type workloads like CREATE TABLE AS (CTAS) and Iceberg DML.
- Ad hoc data science queries with long execution times.

In multi-tenant environments like multiple departments or geographic locations where chargeback models can be implemented for resource usage, we recommend having a separate set of engines per tenant.

Optimize Query Performance

When developing the semantic layer, it is best to create the views in each of the three layers according to best practices without using Reflections, then test queries of the application layer views to gauge baseline performance.

For queries that appear to be running sub-optimally, we recommend analyzing the query profile to determine whether any bottlenecks can be removed to improve performance. If performance issues persist, place Reflections where they will have the most benefit. A well-architected semantic layer allows you to place Reflections at strategic locations in the semantic layer such that large volumes of queries benefit from the fewest number of Reflections, such as in the business layer where a view is constructed by joining several other views.

Design Reflections for Expensive Query Patterns

- Review query history (jobs) to determine the most expensive and most-frequent

queries being submitted.

- Look in the job profiles for these queries. Tables and views referenced by multiple queries that perform expensive scans, joins, and aggregations are good candidates for Reflections.

- Examine the SQL for the selected queries that reference the same table or view to find patterns that can help you define a Reflection on that table or view that satisfies as many of those queries as possible.

Avoid the “More Is Always Better” Approach

Creating more Reflections than are necessary to support your data consumers can lead to the use of more resources than might be optimal for your environment, both in terms of system resources and the time and attention devoted to working with them.

Establish Criteria for When to create Reflections

Create them only when data consumers are experiencing slow query responses, or when reports are not meeting established SLAs.

Create Reflections Without Duplicating the Work of Other Reflections

Dremio recommends that, when you create tables and views, you create them in layers:

- The bottom or first layer consists of your tables.

- In the second layer are views, one for each table, that do lightweight preparation of data for views in the next layers. Here, administrators might create views that do limited casting, type conversion, and field renaming, and redacting sensitive information, among other prepping operations. Administrators can also add security by subsetting both rows and fields that users in other layers are not allowed to access. The data has been lightly scrubbed and restricted to the group of people who have the business knowledge that lets them use these views to build higher-order views that data consumers can use. Then, admins grant access to these views to users who create views in the next layer, without being able to see the raw data in the tables.

- In the third layer, users create views that perform joins and other expensive operations. This layer is where the intensive work on data is performed. These users then create Reflections (raw, aggregation, or both) from their views.

- In the fourth layer, users can create lightweight views for dashboards, reports, and visualization tools. They can also create aggregation Reflections, as needed.

Establish a Routine for Checking How often Reflections Are Used

At regular intervals, check for Reflections that are no longer being used by the query planner and evaluate whether they should be removed. Query patterns can change over time, and frequently-used Reflections can gradually become less relevant.

Use Supporting Anchors

Anchors for Reflections are views that data consumers have access to from their business-intelligence tools. As you develop a better understanding of query patterns, you might want to support those patterns by creating Reflections from views that perform expensive joins, transformations, filters, calculations, or a combination of those operations. You would probably not want data consumers to be able to access those views directly in situations where the query optimizer did not use any of the Reflections created from those views. Repeated and concurrent queries on such views could put severe strain on system resources.

You can prevent queries run by data consumers from accessing those views directly. Anchors that perform expensive operations and to which access is restricted are called supporting anchors.

For example, suppose that you find these three, very large tables are used in many queries:

- Customer
- Order
- Lineitem

You determine that there are a few common patterns in the user queries on these tables:

- The queries frequently join the three tables together.
- Queries always filter by `commit_date < ship_date`
- There is a calculated field in most of the queries: `extended_price * (1-discount) AS revenue`

You can create a view that applies these common patterns, and then create a raw Reflection to accelerate queries that follow these patterns.

First, you create a folder in the Dremio space that your data consumers have access to. Then, you configure this folder to be invisible and inaccessible to the data consumers.

Next, you write the query to create the view, you follow these guidelines:

- Use `SELECT *` to include all fields, making it possible for the query optimizer to accelerate the broadest set of queries. Alternatively, if you know exactly which subset of fields are used in the three tables, you can include just that subset in the view.
- Add any calculated fields, which in this case is the revenue field.
- Apply the appropriate join on the three tables.
- Apply any filters that are used by all queries, which in this case is only `commit_date < ship_date`.
- Always use the most generic predicate possible to maximize the number of queries that will match.

Next, you run the following query to create a new view:

Create a new view

```
SELECT *, extendedprice * (1 - discount) AS revenue FROM customer AS c, orders AS o, lineitem AS l WHERE c.c_custkey = o.o_custkey AND l.l_orderkey = o.o_orderkey AND o.commit_date < o.ship_date
```

Then, you save the view in the folder that you created earlier.

Finally, you create one or more raw Reflections on this new supporting anchor. If most of the queries against the view were aggregation queries, you could create an aggregation Reflection. In both cases, you can select fields, as needed, to sort on or partition on.

The result is that, even though the data consumers do not have access to the supporting anchor, Dremio can accelerate their queries by using the new Reflections as long as they have access to the tables that the Reflections are ultimately derived from: Customer, Order, and Lineitem.

If the query optimizer should determine that a query cannot be satisfied by any of the Reflections, it is possible, if no other views can satisfy it, for the query to run directly against the tables, as is always the case with any query.

Horizontally Partition Reflections that Have Many Rows

If you select a field for partitioning in a data Reflection, Dremio physically groups records together into a common directory on the file system. For example, if you partition by the field Country, in which the values are two-letter abbreviations for the names of countries, such as US, UK, DE, and CA, Dremio stores the data for each country in a separate directory named US, UK, DE, CA, and so on. This optimization allows Dremio to scan a subset of the directories based on the query, which is an optimization called partition pruning.

If a user queries on records for which the value of Country is US or UK, then Dremio can apply partition pruning to scan only the US and UK directories, significantly reducing the amount of data that is scanned for the query.

When you are selecting a partitioning field for a data Reflection, ask yourself these questions:

- Is the field used in many queries?
- Are there relatively few unique values in the field (low cardinality)?

To partition the data, Dremio must first sort all records, which consumes resources. Accordingly, partition data only on fields that can be used to optimize queries. In addition, the number of unique values for a field should be relatively small, so that Dremio creates only a relatively small number of partitions. If all values in a field are unique, the cost to partition outweighs the benefit.

In general, Dremio recommends the total number of partitions for a Reflection to be less than 10,000.

Because Reflections are created as Apache Iceberg tables, you can use partition transforms to specify transformations to apply to partition columns to produce partition

values. For example, if you choose to partition on a column of timestamps, you can set partition transforms that produce partition values that are the years, months, days, or hours in those timestamps. The following table lists the partition transforms that you can choose from.

note

If a column is listed as a partition column, it cannot also be listed as a sort column for the same Reflection.

In aggregation Reflections, each column specified as a partition column or used in transform must also be listed as a dimension column.

In raw Reflections, each column specified as a partition column or used in transform must also be listed as a display column.

Value | Type of Partition Transform | Description || IDENTITY | identity(<column_name>) | Creates one partition per value. This is the default transform. If no transform is specified for a column named by the `name` property, an IDENTITY transform is performed. The column can use any supported data type. |
YEAR	year(<column_name>)	Partitions by year. The column must use the DATE or TIMESTAMP data type.
MONTH	month(<column_name>)	Partitions by month. The column must use the DATE or TIMESTAMP data type.
DAY	day(<column_name>)	Partitions on the equivalent of dateint. The column must use the DATE or TIMESTAMP data type.
HOUR	hour(<column_name>)	Partitions on the equivalent of dateint and hour. The column must use the TIMESTAMP data type.
BUCKET	bucket(<integer>, <column_name>)	Partitions data into the number of partitions specified by an integer. For example, if the integer value N is specified, the data is partitioned into N, or (0 to (N-1)), partitions. The partition in which an individual row is stored is determined by hashing the column value and then calculating `<hash_value> mod N`. If the result is 0, the row is placed in partition 0; if the result is 1, the row is placed in partition 1; and so on. The column can use the DECIMAL, INT, BIGINT, VARCHAR, VARBINARY, DATE, or TIMESTAMP data type.
TRUNCATE	truncate(<integer>, <column_name>)	If the specified column uses the string data type, truncates strings to a maximum of the number of characters specified by an integer. For example, suppose the specified transform is truncate(1, stateUS). A value of `CA` is truncated to `C`, and the row is placed in partition C. A value of `CO` is also truncated to `C`, and the row is also placed in partition C. If the specified column uses the integer or long data type, truncates column values in the following way: For any `truncate(L, col)`, truncates the column value to the biggest multiple of L that is smaller than the column value. For example, suppose the specified transform is `truncate(10, intColumn)`. A value of 1 is truncated to 0 and the row is placed in the partition 0. A value of 247 is truncated to 240 and the row is placed in partition 240. If the transform is `truncate(3, intColumn)`, a value of 13 is truncated to 12 and the row is placed in partition 12. A value of 255 is not truncated, because it is divisble by 3, and the row is placed in partition 255. The column can use the DECIMAL, INT, BIGINT, VARCHAR, or VARBINARY data type. **Note:** The truncate transform does not change column values. It uses column values to calculate the correct partitions in which to place rows.

Partition Reflections to Allow for Partition-Based Incremental Refreshes

Incremental refreshes of data in Reflections are much faster than full refreshes. Partition-based incremental refreshes are based on Iceberg metadata that is used to identify modified partitions and to restrict the scope of the refresh to only those partitions. For more information about partition-based incremental refreshes, see Types of Refresh for Reflections on Apache Iceberg Tables, Filesystem Sources, Glue Sources, and Hive Sources in Refresh Reflections.

For partition-based incremental refreshes, both the base table and its Reflections must be partitioned, and the partition transforms that they use must be compatible. The following table lists which partition transforms on the base table and which partition transforms on Reflections are compatible:

Partition Transform on the Base Table	Compatible Partition Transforms on Reflections
Identity	Identity, Hour, Day, Month, Year, Truncate
Hour	Hour, Day, Month, Year
Day	Day, Month, Year
Month	Month, Year
Year	Year
Truncate	Truncate

note

If both a base table and a Reflection use the Truncate partition transform, follow these rules concerning truncation lengths:

If the partition column uses the String data type, the truncation length used for the Reflection must be less than or equal to the truncation length used for the base table.

If the partition column uses the Integer data type, the remainder from the truncation length on the Reflection (A) divided by the truncation length on the base table (B) must be equal to 0: `A MOD B = 0`

If the partition column uses any other data type, the truncation lengths must be identical.

If a base table uses the Bucket partition transform, partition-based incremental refreshes are not possible.

Partition Aggregation Reflections on Timestamp Data in Very Large Base Tables

Suppose you want to define an aggregation Reflection on a base table that has billions of rows. The base table includes a column that either uses the TIMESTAMP data type or includes a timestamp as a string, and the base table is partitioned on that column.

In your aggregation Reflection, you plan to aggregate on timestamp data that is in the base table. However, to get the benefits of partition-based incremental refresh, you need to partition the Reflection in a way that is compatible with the partitioning on the base table. You can make the partitioning compatible in either of two ways:

By defining a view on the base table, and then defining the aggregation Reflection on that view

-

By using the advanced Reflection editor to define the aggregation Reflection on the base table

Define an Aggregation Reflection on a View

If the timestamp column in the base table uses the TIMESTAMP data type, use one of the functions in this table to define the corresponding column in the view. You can partition the aggregation Reflection on the view column and use the partition transform that corresponds to the function.

Function in View Definition	Corresponding Partition Transform
DATE_TRUNC('HOUR', <base_table_column>)	HOUR(<view_col>)
DATE_TRUNC('DAY', <base_table_column>)	DAY(<view_col>)
DATE_TRUNC('MONTH', <base_table_column>)	MONTH(<view_col>)
DATE_TRUNC('YEAR', <base_table_column>)	YEAR(<view_col>)
CAST <base_table_column> as DATE	DAY(<view_col>)
TO_DATE(<base_table_column>)	DAY(<view_col>)

If the timestamp column in the base table uses the STRING data type, use one of the functions in this table to define the corresponding column in the view. You can partition the aggregation Reflection on the view column and use the partition transform that corresponds to the function.

Function in View Definition	Corresponding Partition Transform
LEFT(<base_table_column>, X)	TRUNCATE(<view_col>, X)
SUBSTR(<base_table_column>, 0, X)	TRUNCATE(<view_col>, X)
SUBSTRING(<base_table_column>, 0, X)	TRUNCATE(<view_col>, X)

Define an Aggregation Reflection on a Base Table

When creating or editing the aggregation Reflection in the Advanced View, as described in Manual Reflections, follow these steps:

Set the base table's timestamp column as a dimension.

!Setting the column as a dimension.

Click the down-arrow next to the green circle.

Select **Date** for the date granularity.

!Selecting the granularity.

Use Dimmensions with Low Cardinality

Use dimensions that have relatively low cardinality in a table or view. The higher the cardinality of a dimension, the less benefit an aggregation Reflection has on query performance. Lower cardinality aggregation Reflections require less time to scan.

Create One Aggregation Reflection for Each Important Subset of Dimensions

- For a single table or view, create one aggregation Reflection for each important subset of dimensions in your queries, rather than one aggregation Reflection that includes all dimensions. Multiple small aggregation Reflections (versus one large one) are good for isolated pockets of query patterns on the same table or view that do not overlap. If your query patterns overlap, use fewer larger aggregation Reflections.

There are two cautions that accompany this advice, however:

- Be careful of creating aggregation Reflections that have too few dimensions for your queries.

If a query uses more dimensions than are included in an aggregation Reflection, the Reflection cannot satisfy the query and the query optimizer does not run the query against it.

- Be careful of creating more aggregation Reflections than are necessary to satisfy queries against a table or view.

The more Reflections you create, the more time the query optimizer requires to plan the execution of queries. Therefore, creating more aggregation Reflections than you need can slow down query performance, even if your aggregation Reflections are low-cardinality.

Sort Reflections on High-Cardinality Fields

The sort option is useful for optimizing queries that use filters or range predicates, especially on fields with high cardinality. If sorting is enabled, during query execution, Dremio skips over large blocks of records based on filters on sorted fields.

Dremio sorts data during the execution of a query if a Reflection spans multiple nodes and is composed of multiple partitions.

Sorting on more than one field in a single data Reflection typically does not improve read performance significantly and increases the costs of maintenance tasks.

For workloads that need sorting on more than one field, consider creating multiple Reflections, each being sorted on a single field.

Create Reflections from Joins that are Based on Joins from Multiple Queries

Joins between tables, views, or both tend to be expensive. You can reduce the costs of joins by performing them only when building and refreshing Reflections.

As an administrator, you can identify a group of queries that use similar joins. Then, you can create a general query that uses a join that is based on the similar joins, but does not include any additional predicates from the queries in the group. This generic query can serve as the basis of a raw Reflection, an aggregation Reflection, or both.

For example, consider the following three queries which use similar joins on views A, B and C:

Three queries with joins on views A, B, and C

```
SELECT a.col1, b.col1, c.col1 FROM a join b on (a.col4 = b.col4) join c on
```

```
(c.col5=a.col5)
  WHERE a.size = 'M' AND a.col3 > '2001-01-01' AND b.col3 IN ('red','blue','green')
SELECT a.col1, a.col2, c.col1, COUNT(b.col1) FROM a join b on (a.col4 = b.col4) join c
on (c.col5=a.col5)
  WHERE a.size = 'M' AND b.col2 < 10 AND c.col2 > 2 GROUP BY a.col1, a.col2, c.col1
SELECT a.col1, b.col2 FROM a join b on (a.col4 = b.col4) join c on (c.col5=a.col5)
  WHERE c.col1 = 123
```

You can write and run this generic query to create a raw Reflection to accelerate all three original queries:

Create a Reflection to accelerate three queries

```
SELECT a.col1 , a.col2, a.col3, b.col1, b.col2, b.col3, c.col1, c.col2 FROM a join b on
(a.col4 = b.col4) join c on (c.col5=a.col5)
```

Time Reflection Refreshes to Occur After Metadata Refreshes of Tables

Time your refresh Reflections to occur only after the metadata for their underlying tables is refreshed. Otherwise, Reflection refreshes do not include data from any files that were added to a table since the last metadata refresh, if any files were added.

For example, suppose a data source that is promoted to a table consists of 10,000 files, and that the metadata refresh for the table is set to happen every three hours. Subsequently, Reflections are created from views on that table, and the refresh of Reflections on the table is set to occur every hour.

Now, one thousand files are added to the table. Before the next metadata refresh, the Reflections are refreshed twice, yet the refreshes do not add data from those one thousand files. Only on the third refresh of the Reflections does data from those files get added to the Reflections.

Rotation Personal Access Tokens

When Dremio personal access tokens (PATs) are used in custom applications, consider scripting an automated periodic refresh to avoid job failures when the PATs expire.

Monitor Dremio Projects

It's important to set up a good monitoring solution to maximize your investment in Dremio and identify and resolve issues related to Dremio projects before they have a broader impact on workload. Your monitoring solution should ensure overall cluster health and performance.

Was this page helpful?

Principles

Regularly Evaluate Engine Resources

- Regularly Evaluate Query Performance
- Clean Up Tables with Vacuum
- Optimize Tables
- Regularly Monitor Live Metrics for Dremio
- Best Practices
- Optimize Workload Management Rules
- Configure Engines
- Optimize Query Performance
- Design Reflections for Expensive Query Patterns
- Avoid the “More Is Always Better” Approach
- Establish Criteria for When to create Reflections
- Create Reflections Without Duplicating the Work of Other Reflections
- Establish a Routine for Checking How often Reflections Are Used
- Use Supporting Anchors
- Horizontally Partition Reflections that Have Many Rows
- Partition Reflections to Allow for Partition-Based Incremental Refreshes
- Use Dimmensions with Low Cardinality
- Create One Aggregation Reflection for Each Important Subset of Dimensions
- Sort Reflections on High-Cardinality Fields
- Create Reflections from Joins that are Based on Joins from Multiple Queries
- Time Reflection Refreshes to Occur After Metadata Refreshes of Tables
- Rotation Personal Access Tokens
- Monitor Dremio Projects

Performance Efficiency | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/help-support/well-architected-framework/performance-efficiency

On this page

Dremio is a powerful massively parallel processing (MPP) platform that can process terabyte-scale datasets. To get the best performance from your Dremio environment, follow these design principles and best practices for implementation.

Dimensions of Performance Optimization

When optimizing Dremio, several factors can affect workload and project performance. Queries submitted to Dremio must be planned on the control plane before being routed for execution. The resource requirements and degree of optimization of individual queries can vary widely. Those queries can be rewritten and optimized on their own without regard to a larger engine.

Beyond individual queries, executor nodes have individual constraints of memory and CPU. Executors in Dremio are also part of an engine that groups executors together to process queries in parallel across multiple machines. The size of the engine that a query runs on can affect its performance. To enhance the ability to handle additional queries beyond certain concurrency thresholds, configure replicas for the engine.

These dimensions of performance optimization can be simplified in the following decision tree, which addresses the most common scenarios. In the decision tree, `engine_start_epoch_millis > 0` implies that the engine is down.

!Decision tree diagram that shows common performance optimization scenarios for Dremio.

Principles

Perform Regular Maintenance

Conduct regular maintenance to ensure that your project is set up for optimal performance and can handle more data, queries, and workloads. Regular maintenance will establish a solid baseline from which you can design and optimize. Dremio can be set up to automatically optimize and vacuum tables in your Open catalog.

Optimize Queries for Efficiency

Before worrying about scaling out your engines, it is important to optimize your semantic layer and queries to be as efficient as possible. For example, if there are partition columns, you should use them. Create sorted or partitioned Reflections. Follow standard SQL writing best practices such as applying functions to values rather than columns in where clauses.

Optimize Engines

Dremio provides several facilities to allow workload isolation and ensure your queries do not overload the engines. Multiple engines are used to keep some queries from affecting others and concurrency rules are used to buffer queries to prevent overloading any one particular engine.

Best Practices

Design Semantic Layer for Workload Performance

Dremio’s enterprise-scale semantic layer clearly defines the boundary between your physically stored tables and your logical, governed, and self-service views. The semantic layer seamlessly allows data engineers and semantic data modelers to create views based on tables without having to make copies of the physical data.

Since interactive performance for business users is a key capability of the semantic layer, when appropriate, Dremio can leverage physically optimized representations of source data known as Reflections. When queries are made against views that have Reflections enabled, the query optimizer can accelerate a query by using one or more Reflections to partially or entirely satisfy that query rather than processing the raw data in the underlying data source. Queries do not have to be rewritten to take advantage of Reflections. Instead, Dremio's optimizer automatically considers Reflection suitability while planning the query.

With Dremio, you can create layers of views that allow you to present data to business consumers in a format they need while satisfying the security requirements of the organization. Business consumers do not need to worry about which physical locations the data comes from or how the data is physically organized. A layered approach allows you to create sets of views that can be reused many times across multiple projects.

Leveraging Dremio’s layering best practices promotes a more-performant, low-maintenance solution that can provide agility to development teams and business users as well as better control over data.

Improve the Performance of Poor-Performing Queries

Run `SELECT * FROM sys.project.jobs_recent` to analyze the query history and determine which queries are performing sub-optimally. This allows you to consider a number of factors, including the overall execution time of a query. Identify the 10 longest-running queries to understand why they are taking so long. For example, is it the time taken to read data from the source, lacking CPU cycles, query spilling to disk, query queued at the start, or another issue? Did the query take a long time to plan?

note

Read Query Performance Analysis and Improvement for details about query performance analysis techniques. This white paper was developed based on Dremio Software, but the content applies equally to Dremio.

The query history also allows you to focus on planning times. You should also investigate queries to pinpoint high planning time, which could be due to the complexity of the query (which you can address by rewriting the query) or due to many Reflections being considered (which indiciates that too many Reflections are defined in the environment). Read Remove Unused Reflections for more information about identifying redundant Reflections in your Dremio project.

The query history also allows you to focus on metadata refresh times, which could be due to inline metadata refresh. Read Optimize Metadata Refresh Frequency for more information about checking metadata refresh schedules.

Sometimes, query performance is inconsistent. A query may complete execution in less than 10 seconds in one instance but require 1 minute of execution time in another instance. This is a sign of resource contention in the engine, which can happen in high-volume environments or when too many jobs (including user queries, metadata, and reflections) are running at the same time. We recommend having separate,

dedicated engines for metadata refreshes, Reflection refreshes, and user queries to reduce the contention for resources when user queries need run concurrently with refreshes.

For Reflection jobs that require excessive memory, we recommend two Reflection refresh engines of different sizes, routing the Reflections that require excessive memory to the larger engine. This is typically needed for Reflections on views that depend on the largest datasets and can be done with the ALTER TABLE ROUTE REFLECTIONS command.

Read Dremio Profiles to Pinpoint Bottlenecks

Dremio job profiles contain a lot of fine-grained information about how a query was planned, how the phases of execution were constructed, how the query was actually executed, and the decisions made about whether to use Reflections to accelerate the query.

For each phase of the query that is documented in the job profile, check the start and end times of the phase for an initial indication of the phase in which any bottlenecks are located. After you identify the phase, check the operators of that phase to identify which operator or thread within the operator may have been the specific bottleneck. This information usually helps determine why a query performs sub-optimally so that you can plan improvements. Reasons for bottlenecks and potential improvement options include:

- High metadata retrieval times from inline metadata refresh indicate that you should revisit metadata refresh settings.

- High planning time can be caused by too many Reflections or may mean that a query is too complex and should be rewritten.

- High engine start times indicate that the engine is down. The enqueued time may include replica start time. You may be able to mitigate these issues with minimum replica and last replica auto-stop settings.

- High setup times in table functions indicate overhead due to opening and closing too many small files. High wait times indicate that there is a network or i/o delay in reading the data. High sleep times in certain phases could indicate CPU contention.

note

Read Reading Dremio Job Profiles for details about job profile analysis techniques. This white paper was developed based on Dremio Software, but the content applies equally to Dremio.

Engine Routing and Workload Management

Since the workloads and volumes of queries change over time, reevaluate engine routing settings, engine sizes, engine replicas, and concurrency per replica and adjust as needed to rebalance the proportion of queries that execute concurrently on a replica of an engine.

Right-Size Engines and Executors

Analyze the query history to determine whether a change in the number of executors in your engines is necessary.

When the volume of queries being simultaneously executed by the current set of executor nodes in an engine starts to reach a saturation point, Dremio exhibits several symptoms. Saturation point is typically manifested as increased sleep time during query execution. Sleep time is incurred when a running query needs to wait for available CPU cycles due to all available CPUs being in operation. Another symptom is an increased number of queries spilling to disk or out-of-memory exceptions.

You can identify these symptoms by analyzing the system table by running `SELECT * FROM sys.project.jobs_recent`. The resulting table lists query execution times, planning time, engine start times, enqueued times, and job failure.

Failure to address these symptoms can result in increasing query failures, increasing query times, and queries spilling to disk, which in turn lead to a bad end-user experience and poor satisfaction. Spilling to disk ensures that a query succeeds because some of its high-memory-consuming operations are processed via local disks. This reduces the memory footprint of the query significantly, but the trade-off is that the query inevitably runs more slowly.

You can alleviate these issues by adding replicas to the engine and reducing concurrency per replica and adding a larger engine, then altering the engine routing rules to route some of the workload to the new engine. Remember that a query executes on the nodes of a single replica or an engine, not across multiple replicas or multiple engines.

A good reason to create a new engine is when a new workload is introduced to your Dremio project, perhaps by a new department within an organization, and the queries cause the existing engine setup to degrade in performance. Creating a new engine to isolate the new workload, most likely by creating rules to route queries from users in that organization to the new engine, is a useful way of segregating workloads.

Leverage Reflections to Improve Performance

When developing use cases in Dremio’s semantic layer, it’s often best to build out the use case iteratively without any Reflections to begin with. Then, as you complete iterations, run the queries and analyze the data in the query history to deduce which queries take the longest to execute and whether any common factors among a set of slow queries are contributing to the slowness.

For example, if a set of five slow queries are each derived from a view that contains a join between two relatively large tables, you might find that adding a raw Reflection on the view that is performing the join helps to speed up all five queries because doing so creates an Apache Iceberg materialization of the join results, which is automatically used to accelerate views derived from the join. This provides the query planning and performance benefits of Apache Iceberg and allows you to partition the Reflection to accelerate queries for which the underlying data weren't initially optimized. This is an important pattern because it means you can leverage a small number of Reflections to speed up many workloads.

Raw Reflections can be useful when you have large volumes of JSON or CSV data. Querying such data requires processing the entire data set, which can be inefficient. Adding a raw Reflection over the JSON or CSV data again allows for an Apache Iceberg representation of that data to be created and opens up all of the planning and

performance benefits that come along with it.

Another use of raw Reflections is to offload heavy queries from an operational data store. Often, database administrators do not want their operational data stores (for example, online transaction processing databases) overloaded with analytical queries while they are busy processing billions of transactions. In this situation, you can leverage Dremio raw Reflections again to create an Apache Iceberg representation of the operational table. When a query comes in that needs the data, Dremio reads the Reflection data instead of going back to the operational source.

Another very important use case that often requires raw Reflections is when you join on-premises data to cloud data. In this situation, retrieving the on-premises data often becomes a bottleneck for queries due to the latency in retrieving data from the source system. Leveraging a default raw Reflection on the view where the data is joined together often yields significant performance gains.

If you have connected Dremio to client tools that issue different sets of GROUP BY queries against a view, and the GROUP BY statements take too long to process compared to the desired service level agreement, consider adding an aggregation Reflection to the view to satisfy the combinations of dimensions and measures that are submitted from the client tool.

Read Best Practices for Creating Raw and Aggregation Reflections when you are considering how and where to apply Reflections.

Failing to make use of Dremio Reflections means you could be missing out on significant performance enhancements for some of your poorest-performing queries. However, creating too many Reflections can also have a negative impact on the system as a whole. The misconception is often that more Reflections must be better, but when you consider the overhead in maintaining and refreshing Reflections at intervals, Reflection refreshes can end up stealing valuable resources from your everyday workloads, especially if you have not created a dedicated Reflection refresh engine.

Where possible, organize your queries by pattern. The idea is to create as few Reflections as possible to service as many queries as possible, so finding points in the semantic tree through which many queries go can help you accelerate a larger number of queries. The more Reflections you have that may be able to accelerate the same query patterns, the longer the planner takes to evaluate which Reflection is best suited for accelerating the query being planned.

Optimize Metadata Refresh Performance

Add a dedicated metadata refresh engine to your Dremio project. This ensures that all metadata refresh activities for Parquet, Optimized Row Columnar (ORC), and Avro datasets that are serviced by executors are completed in isolation from any other workloads and prevents problems with metadata refresh workloads taking CPU cycles and memory away from business-critical workloads. This gives the refreshes have the best chance of finishing in a timely manner.

Was this page helpful?

Dimensions of Performance Optimization

Principles

-

Perform Regular Maintenance

- Optimize Queries for Efficiency
- Optimize Engines
- Best Practices
- Design Semantic Layer for Workload Performance
- Improve the Performance of Poor-Performing Queries
- Read Dremio Profiles to Pinpoint Bottlenecks
- Engine Routing and Workload Management
- Right-Size Engines and Executors
- Leverage Reflections to Improve Performance
- Optimize Metadata Refresh Performance

AI Semantic Layer | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/help-support/well-architected-framework/self-serve-semantic-layer

On this page

Dremio has a unique capability in its AI Semantic Layer, which is where the magic happens in mapping the physical structure of the underlying data storage to how the data is consumed via SQL queries. When you optimally design and maintain the semantic layer, data is more discoverable, writing queries is more straightforward, and performance is optimized.

Principles

Layer Views

Layering your views allows you to balance security, performance, and usability. Layered views help you expose the data in your physical tables to external consumption tools in the format the tools require, with proper security and performance. A well-architected semantic layer consists of three layers to organize your views: preparation, business, and application. Each layer serves a purpose in transforming data for consumption by external tools.

Annotate Datasets to Enhance Discovery and Understanding

You can label and document datasets within Dremio to make data more discoverable and verifiable and allow you to apply governance.

Best Practices

Use the Preparation Layer to Map 1-1 to Tables

The preparation layer is closest to the data source. This layer is used to organize and expose only the required datasets from the source rather than all datasets the source contains. In the preparation layer, each view is mapped to the table that it is derived from in the data source, and there are no joins to other views.

Typically, a data engineer is responsible for preparing the data in the preparation layer. The data engineer should apply column aliasing so that all downstream views can use the normalized column names. Casting column data types should also be done in the preparation layer so that all higher-level views can leverage the correct type and conversion is done only once. Data should be cleansed in the preparation layer for central management and to ensure that all downstream views use clean data. Derived columns based on existing columns should be configured in the preparation layer so that all future layers can use the new columns.

Use the Business Layer to Logically Join Datasets

The business layer provides a holistic view of all data across your catalog or folder. It is the first layer where joins among and between sources should occur. All views in the business layer must be built by either querying resources in the preparation layer or querying other resources in the same business layer.

- Querying resources in the preparation layer: views in the business layer should start with selecting all columns from the preparation layer of that view. This is typically a 1-1 mapping between the preparation and business layer view.

- Querying other resources in the same business layer: when joining two views together, they should be joined from the business layer representation of the view, not the preparation layer. This allows all changes made in the business layer to propagate to all joins.

Use your list of common terms to describe the key business entities in your organization, such as customer, product, and order. Typically, a data modeler works with business experts and data providers to define the views that represent the business entities.

You can create many sub-layers inside the business layer, each consisting of views for different subject areas or verticals. These views are reusable components that can and should be shared across business lines. Typically, views do not filter rows or columns in the business layer; this is deferred to the application layer.

Use the business layer to improve productivity for analytics initiatives and minimize the risk of duplicative efforts in your organization by reducing the cost of service delivery to lines of business, providing a self-service model for data engineers to quickly provision datasets, and enabling data consumers to quickly use and share datasets.

Use the Application Layer to Arrange Datasets for Consumption

Application layer views are arranged for the needs of data consumers and organizational departments. Typically, data consumers like analysts and data scientists use the views from the business layer and work directly in the application layer to create and modify views in their own dashboards.

If the application layer provides self-service access to Dremio's AI Semantic Layer, you should expose all business layer views in the application layer at minimum. Even if the view is created by running `SELECT * from BUSINESS_VIEW`, it provides logical separation for security and performance improvements.

If the application layer is not for self-service but for particular applications, the views in the application layer should be built on top of those self-service views in the application layer, adding any application-specific logic. Application logic should be row filters as needed by the application. Columns can be left as-is, and the list of columns the application selects are reduced in the SQL query.

Leverage Labels to Enhance Searchability

Use Dremio's label functionality to create and assign labels to tables and views to group related objects and enhance the discoverability of data across your organization. You can search for sets of tables and views based on a label or click on a label in the Dremio console to start a search based on it. Objects can have multiple labels so that they can belong to different logical groups.

Create Wiki Content to Describe Datasets

Use Dremio's wiki functionality to add descriptions for catalogs, sources, folders, tables, and views. Wikis enhance understanding of data inside your organization. Wikis allow you to provide context for datasets, such as descriptions for each column, and content that helps users get started with the data, such as usage examples, notes, and points of contact for questions or issues.

Dremio wikis use GitHub-Flavored Markdown and are supported by a rich text editor.

To help eliminate the need for labor-intensive manual classification and cataloging, you can use generative AI to generate labels and wikis for your datasets. Enabling the generative AI feature in Dremio allows you to generate a detailed description of each dataset's purpose and schema. Dremio's generative AI bases its understanding on your schema and data to produce descriptions of datasets because it can determine how the columns within the dataset relate to each other and to the dataset as a whole.

Was this page helpful?

- Use the Application Layer to Arrange Datasets for Consumption
- Leverage Labels to Enhance Searchability
- Create Wiki Content to Describe Datasets

Source: dremio-cloud-manage-govern.md

Manage and Govern Your Data | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/manage-govern/

On this page

Data management focuses on the operational efficiency, performance, and reliability of your data at scale. With Dremio's autonomous management capabilities, many of these processes are intelligently automated; reducing manual effort and ensuring consistent optimization. Dremio automates table optimization by merging small files into optimally sized ones (typically around 512 MB), reducing metadata overhead, and reclaiming storage by physically removing deleted rows. It also reorganizes data to align with clustering specifications, ensuring consistent, high-performance queries across large datasets. Together, these autonomous management features help keep your lakehouse fast, efficient, and cost-effective.

Data governance is the foundation of a secure, reliable, and compliant lakehouse. It ensures that data across your environment is accurate, consistent, and properly controlled throughout its lifecycle. With Dremio, you can implement robust governance practices by maintaining complete data lineage for transparency and auditability, defining role-based and fine-grained (row-access and column-masking) access controls on data, and using documentation and tags to improve data discoverability. Together, these capabilities enable trustworthy, well-governed data that fuels analytics and AI with confidence.

Autonomous Management

Optimization

Managing Apache Iceberg tables is critical to maintaining fast and predictable query performance, especially for agentic AI workloads that demand low latency. As new data is ingested and tables are updated, metadata and small data files accumulate, leading to performance degradation over time. Dremio automates table optimization by merging small files into optimally sized ones (typically ~512 MB), reducing metadata overhead, organizing data to align with clustering specification and reclaiming storage by physically removing deleted rows.

Clustering

Dremio also reorganizes data to align with clustering specifications, ensuring consistent, high-performance queries at scale.

Materialize and Query Rewrite

Dremio can autonomously materialize datasets using Reflections, a precomputed and optimized copy of source data or a query result, designed to speed up query performance. Dremio's query optimizer can accelerate a query against tables or views by using one or more Reflections to partially or entirely satisfy that query, rather than processing the raw data in the underlying data source. Queries do not need to reference Reflections directly. Instead, Dremio rewrites queries on the fly to use the Reflections that satisfy them. For more information, see Reflections.

Governance

Lineage

Track and visualize how data flows through your lakehouse, from source to consumption. Lineage helps you understand data origins, track transformations, identify dependencies, and perform impact analysis.

Wikis

Enrich data understanding by documenting datasets with wikis. Use Generative AI to automatically generate wikis, reducing manual documentation effort. Wikis are used by Dremio's AI Agent to understand the semantics of your environment and adhere to these definitions in response to user prompts.

Labels

Enhance data discoverability and searchability by categorizing datasets with labels. Use Generative AI to automatically generate labels, reducing manual cataloging effort.

Role-Based Access Control Policies

Manage access to datasets through roles rather than individual user grants for easier administration. Assign privileges to roles, simplifying management and ensuring users only have access to what they need to perform their job.

Row-Access and Column-Masking Policies

Apply fine-grained access controls to protect sensitive data using row-access and column-masking policies. Control access to specific rows and columns based on rules and conditions to maintain compliance and adhere to regulatory requirements. For more information, see Row-Access & Column-Masking Policies.

Related Topics

Roles – Manage role-based access control.

Explore and Analyze Your Data - Explore and analyze your governed data.

Catalog API - Lineage - Retrieve lineage information about datasets.

Was this page helpful?

Autonomous Management

Optimization

Clustering

Materialize and Query Rewrite

Governance

Lineage

Wikis

Labels

Role-Based Access Control Policies

Row-Access and Column-Masking Policies

Related Topics

Lineage | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/manage-govern/lineage

On this page

Lineage provides a graph of a dataset's relationships (its source, parent datasets, and child datasets) to illustrate how datasets are connected, where the data originates, while tracking its movement and transformations.

By default, the lineage graph focuses on the initially selected dataset and its relationships with other datasets, represented as nodes that display the dataset name and path. To view additional metadata, use the **Show/hide layers** options.

If you wish to track lineage for a different dataset node, the lineage graph needs to be refocused. To refocus the lineage graph on a different dataset, you can either click !This is the Focus icon. or on the right of the dataset name, and then select **Focus on this dataset**.

!This is a screenshot showing the option to refocus the lineage graph on a different dataset.

Privileges Required for Lineage

If you have the `SELECT` privilege on the parent datasets and the child datasets, you can see the parent datasets and data sources on the left. The child datasets appear on the right.

If you have only the `READ METADATA` privilege on the parent and child datasets, then you can only see limited metadata for these datasets.

If you do not have the `SELECT` or the `READ METADATA` privilege on the parent and child datasets, they are not visible.

Lineage Refresh with Dataset Schema Changes

For datasets in Iceberg REST catalogs, the lineage graphs are stored in Dremio's metadata cache, which is automatically refreshed at fixed time intervals. For more information, see Metadata Refresh. It is possible that the lineage graph might show an outdated schema for the dataset if the dataset schema has been recently updated and Dremio's metadata cache has not yet been refreshed.

Was this page helpful?

Automatic Optimization | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/manage-govern/optimization

On this page

As Apache Iceberg tables are written to and updated, data and metadata files accumulate, which can affect query performance. For example, small files produced by data ingestion jobs slow queries because the query engine must read more files.

To optimize performance, Dremio automates table maintenance in the Open Catalog. This process compacts small files into larger ones, partitions data based on the values of a table's columns, rewrites manifest files, removes position delete files, and clusters tables—improving query speed while reducing storage costs.

Automatic optimization runs on a dedicated engine configured by Dremio, ensuring peak performance without impacting project query workloads.

When Dremio optimizes a table, it evaluates file sizes, partition layout, and metadata organization to reduce I/O and metadata overhead. Optimization consists of five main operations: clustering, data file compaction, partition evolution, manifest file rewriting, and position delete files.

Clustering

Iceberg clustering sorts individual records in data files based on the clustered columns provided in the `CREATE TABLE` or `ALTER TABLE` statement.

To cluster a table, you must first define the clustering keys. Then, automatic optimization uses the clustering keys to optimize tables. For details, see Clustering.

Data File Compaction

Iceberg tables that are constantly being updated can have data files of various sizes. As

a result, query performance can be negatively affected by sub-optimal file sizes. The optimal file size in Dremio is 256 MB.

Dremio logically combines smaller files and splits larger ones to 256 MB (see the following graphic), helping to reduce metadata overhead and costs related to opening and reading files.

!Optimizing file sizes in Dremio.

Partition Evolution

To improve read or write performance, data is partitioned based on the values of a table's columns. If the columns used in a partition evolve over time, query performance can be impacted when the queries are not aligned with the current segregations of the partition. Dremio detects and rewrites these files to align with the current partition specification. This operation is used:

When select partitions are queried more often or are of more importance (than others), and it's not necessary to optimize the entire table.

When select partitions are more active and are constantly being updated. Optimization should only occur when activity is low or paused.

Manifest File Rewriting

Iceberg uses metadata files (or manifests) to track point-in-time snapshots by maintaining all deltas as a table. This metadata layer functions as an index over a table’s data and the manifest files contained in this layer speed up query planning and prune unnecessary data files. For Iceberg tables that are constantly being updated (such as the ingestion of streaming data or users performing frequent DML operations), the number of manifest files that are suboptimal in size can grow over time. Additionally, the clustering of metadata entries in these files may not be optimal. As a result, suboptimal manifests can impact the time it takes to plan and execute a query.

Dremio rewrites these manifest files quickly based on size criteria. The target size for a manifest file is based on the Iceberg table's property. If a default size is not set, Dremio defaults to 8 MB. For the target size, Dremio considers the range from 0.75x to 1.8x, inclusive, to be optimal. Manifest files exceeding the 1.8x size will be split while files smaller than the 0.75x size will be compacted.

This operation results in the optimization of the metadata, helping to reduce query planning time.

Position Delete Files

Iceberg v2 added the ability for delete files to be encoded to rows that have been deleted in existing data files. This enables you to delete or replace individual rows in immutable data files without the need to rewrite those files. Position delete files identify deleted rows by file and position in one or more data files, as shown in the following example.

`file_path`	`pos`

| `file:/Users/test.user/Downloads/gen_tables/orders_with_deletes/data/2021/2021-00.parquet` | `6` |
| `file:/Users/test.user/Downloads/gen_tables/orders_with_deletes/data/2021/2021-00.parquet` | `16` |

Dremio can optimize Iceberg tables containing position delete files. This is beneficial to do because when data files are read, the associated delete files are stored in memory. Also, one data file can be linked to several delete files, which can impact read time.

When tables are optimized in Dremio, the position delete files are removed and the data files that are linked to them are rewritten. Data files are rewritten if any of the following conditions are met:

- The file size is not within the optimum range.
- The partition's specification is not current.
- The data file has an attached delete file.

Related Topics

Apache Iceberg – Learn more about the Apache Iceberg table format.

Load Data Into Tables – Load data from CSV, JSON, or Parquet files into existing Iceberg tables.

Was this page helpful?

- Clustering
- Data File Compaction
- Partition Evolution
- Manifest File Rewriting
- Position Delete Files
- Related Topics

Wikis and Labels | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/manage-govern/wikis-labels

On this page

Wikis and labels help users document, organize, and discover datasets within the Open Catalog. This page explains how to manage wikis and labels, as well as how Dremio’s Generative AI features can assist in generating wikis and labels for you.

Wikis

Wikis for datasets provide an efficient way to document and describe datasets within

the Open Catalog. These wikis enable users to add comprehensive information, context, and relevant details about the datasets they manage.
With a user-friendly, rich text editor, the wikis support Github-flavored markdown, allowing users to format content easily and enhance readability.
Wikis ensure that dataset documentation is both accessible and structured, making it simpler for teams to understand the datasets and how to work with them effectively.

!This image shows an example of the Wiki editor in Dremio.

Manage Wikis

note

Ensure you have sufficient Role-Based Access Control (RBAC) privileges to view or edit wikis.

To view or edit the wiki for a dataset in the Dremio console:

- On the Datasets page, navigate to the folder where your dataset is stored.

- Hover over your dataset, and on the right-hand side, click the !This is the icon that represents more actions. icon.

- Click **Open Details Panel**.

- You can edit the dataset wiki by clicking **Edit Wiki**, writing your wiki content, and clicking **Save**.

Labels

Labels for datasets offer a powerful way to organize and retrieve datasets within a data catalog. By creating and assigning labels to datasets, users can easily search and filter through large collections related datasets.
Labels also enhance the search experience, allowing users to quickly locate datasets associated with a specific label. By clicking on a label, users can initiate a search that brings up all datasets linked to that label, streamlining the process of finding relevant data and improving overall data management.

The following image shows a dataset in the catalog with several label and a brief wiki. In this example, the label "pii-data" was used in the search field to narrow down on a customer dataset that contains Personally Identifiable Information (PII).

!This image shows an example of creating labels.

Manage Labels

note

Ensure you have sufficient Role-Based Access Control (RBAC) privileges to view or edit labels.

To view or edit the labels for a dataset in the Dremio console:

- On the Datasets page, navigate to the folder where your dataset is stored.

Hover over your dataset, and on the right-hand side, click the !This is the icon that represents more actions. icon.

Click **Open Details Panel**.

You can add a label by clicking on the icon, typing a label name (e.g. `PII`), and clicking **Enter**.

Generate Labels and Wikis Preview

To help eliminate the need for manual profiling and cataloging, you can use Generative AI to generate labels and wikis for your datasets.

note

If you haven't opted into the Generative AI features, see Dremio Preferences for the steps on how to enable.

Generate Labels

In order to generate a label, Generative AI bases its understanding on your schema by considering other labels that have been previously generated and labels that have been created by other users.

To generate labels:

Navigate to either the Details page or Details Panel of a dataset.

In the Dataset Overview on the right, click !This is the icon that represents Generative AI. to generate labels.

In the Generating labels dialog, review the labels generated for the dataset and decide which to save. If multiple labels have been generated, you can save some, all, or none of them. To remove, simply click the **x** on the label.

!This screenshot is showing how to generate a label.

Complete one of the following actions:

If these are the only labels for your dataset, click **Save**.

If you already have labels for the dataset and want to add these generated labels, click **Append**.

If you already have labels for the dataset and want to replace them with these generated labels, click **Overwrite**.

The labels for the dataset will appear in the Dataset Overview.

Generate Wikis

In order to generate a wiki, Generative AI bases its understanding on your schema and data to produce descriptions of datasets, because it can determine how the columns within the dataset relate to each other and to the dataset as a whole.

You can generate wikis only if you are the dataset owner or have `ALTER` privileges on the dataset.

To generate a wiki:

Navigate to either the Details page or Details Panel of a dataset.

In the Wiki section, click **Generate wiki**. A dialog will open and a preview of the wiki content will generate on the right of the dialog. If you would like to regenerate, click .

!This screenshot is showing how to generate wikis.

Click to copy the generated wiki content on the right of the dialog.

Click within the text box on the left and paste the wiki content.

(Optional) Use the toolbar to make edits to the wiki content. If you would like to regenerate, click !This is the icon that represents Generative AI. in the toolbar to regenerate wiki content in the preview.

Click **Save**.

The wiki for the dataset will appear in the Wiki section.

Related Topics

Search for Dremio Objects and Entities - Explore Dremio's semantic search capabilities.

Data Privacy - Learn more about Dremio's data privacy practices.

Was this page helpful?

Wikis

Manage Wikis

Labels

Manage Labels

Generate Labels and Wikis Preview

Related Topics

Row-Access and Column-Masking Policies | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/manage-govern/row-column-policies

On this page

Row-access and column-masking policies may be applied to tables, views, and columns via user-defined functions (UDFs). Using these policies, you can control access to

sensitive data based upon the rules and conditions you need to maintain compliance or adhere to regulatory requirements, while also removing the need to produce a secondary set of data with protected information manually removed.

The following restrictions apply to policies and UDFs:

- Only users with the ADMIN role can create UDFs.
- UDFs can only have one owner, which is the user that created the UDF, by default.
- You can transfer ownership of a UDF using the `GRANT OWNERSHIP` command (see Privileges).
- Users or roles must have the EXECUTE privilege in order to apply filtering and masking policies.

Column-Masking Policies

Column-masking is a way to mask—or scramble—private data at the column-level dynamically prior to query execution. For example, the owner of a table or view may apply a policy to a column to only display the year of a date or the last four digits of a credit card.

Column-masking policies may be any UDF with a scalar return type that is identical to the data type of the column on which it is applied. However, only one column-masking policy may be applied to each column.

In the following example of a user-defined function, only users within in the Accounting department in the state of California (CA) may see an entry's social security number (ssn) if the record lists an income above $10,000, otherwise the SSN value is masked with XXX-XX-.

Column-masking policy example

```
CREATE FUNCTION protect_ssn (ssn VARCHAR(11))
    RETURNS VARCHAR(11)
                        RETURN   SELECT   CASE   WHEN   query_user()='jdoe@dremio.com'   OR
is_member('Accounting') THEN ssn
             ELSE CONCAT('XXX-XXX-', SUBSTR(ssn,9,3))
         END;
```

Row-Access Policies

Row-access policies are a way to control which records in a table or view are returned for specific users and roles. For example, the owner of a table or view may apply a policy that filters out customers from a specific country unless the user running the query has a specific role.

Row-access policy example

```
CREATE FUNCTION country_filter (country VARCHAR)
    RETURNS BOOLEAN
           RETURN SELECT query_user()='jdoe@dremio.com' OR (is_member('Accounting') AND
country='CA');
```

Row-access policies may be any boolean UDF applied to the table or view. The return value of the UDF is treated logically in a query as an `AND` operator included in a `WHERE` clause. The return type of the UDF must be `BOOLEAN`, otherwise Dremio will give an error at execution time.

User-Defined Functions

A user-defined function, or UDF, is a callable routine that accepts input parameters, executes the function body, and returns a single value or a set of rows.

The UDFs which serve as the basis for filtering and masking policies must be defined independently of your sources. Not only does this allow organizations to use a single policy for multiple tables and views, but this also restricts user access to policies and prevents unauthorized tampering. Modifying a single UDF automatically updates the policy in the context of any tables or views using that access or mask policy.

The following process describes how policies are enforced with Dremio:

- A user with the ADMIN role creates a UDF to serve as a security policy.
- The administrator then sets the security policy to one or more tables, views, and/or columns.
- Dremio enforces the policy at runtime when an end-user performs a query.

Creating UDFs and attaching security policies is done through SQL commands. Policies are applied prior to execution during the query planning phase. At this point, Dremio checks first the table/view for a row-access policy and then each column accessed for a column-masking policy. If any policies are found, they are automatically applied to the policy's scope using the associated UDF in the query plan.

Query Substitutions

Row-access and column-masking function act as an "implicit view," replacing a table/view reference in an SQL statement prior to processing the query. This implicit view is created through an examination of each policy applied to a table, view, or column.

For example, jdoe@dremio.com has SELECT access to table_1. However, the column-masking policy protect_ssn is set for the column_1 column with a UDF to replace all but the last four digits of a social security number with X for anyone that is not a member of the Accounting department, or this user. When they run a query in Dremio that includes this column-masking policy, the following occurs:

- During the SQL Planning phase, Dremio identifies which tables, views, and columns are being accessed (table_1) and whether security policies must be enforced.
- The engine searches for any security policies set to the associated objects, such as protect_ssn (see Examples of UDFs below).
- When the protect_ssn policy is found for the object affected by the query, the query planner immediately modifies the execution path to incorporate the masking function.
- Query execution proceeds as normal with the associated UDF included within the execution path.

List Existing UDFs

To view all existing UDFs created in Dremio, use the `SHOW FUNCTIONS` SQL command.

List Existing Policies

To view row-access and column-masking policies, use a `SELECT` statement with the target table/view, system table, and policies specified.

List existing column-masking and row-access policies

```
SELECT view_name, masking_policies, row_access_policies FROM sys.project.views;
SELECT table_name, masking_policies, row_access_policies FROM sys.project."tables";
```

To view all column-masking policies set for a given table, use the `DESCRIBE TABLE` command.

Set a Policy

To create a row-access or column-masking policy, you must perform the following steps using the associated SQL commands:

Create a new UDF or replace an existing one using the `CREATE \[OR REPLACE\]` function command.

Create or replace UDF

```
CREATE FUNCTION country_filter (country VARCHAR)
RETURNS BOOLEAN
RETURN    SELECT    query_user()='jdoe@dremio.com'    OR    (is_member('Accounting')    AND
country='CA');

CREATE FUNCTION id_filter (id INT)
RETURNS BOOLEAN
RETURN SELECT id = 1;
```

Grant the EXECUTE privilege to the role/users to apply the policy.

Grant EXECUTE privilege

```
GRANT EXECUTE ON FUNCTION country_filter TO role Policy_Role;
```

Create a policy to apply the function use `ADD ROW ACCESS POLICY` for row-level access or `SET MASKING POLICY` for column-masking. These may be used with the `CREATE TABLE`, `CREATE VIEW`, `ALTER TABLE`, and `ALTER VIEW` commands.

Create policy to apply function

```
-- Add row-access policy
ALTER TABLE e.employee
```

```
ADD ROW ACCESS POLICY country_filter(country);

-- Add column-masking policy
ALTER VIEW e.employee_view
SET MASKING POLICY protect_ssn (ssn_col, region);

-- Create table with row policy
CREATE TABLE e.employee(
id INTEGER,
ssn VARCHAR(11),
country VARCHAR,
ROW ACCESS POLICY country_filter(country)
);

-- Create table with masking policy
CREATE VIEW e.employee_view(
ssn_col VARCHAR MASKING POLICY protect_ssn (ssn_col, region),
region VARCHAR,
state_col VARCHAR)
);
```

note

Both row-access and column-masking UDFs may be applied in a single security policy, or set individually.

Drop a Policy

To remove a security policy from a table, view, or row, use `UNSET MASKING POLICY` or `DROP ROW ACCESS POLICY` with `ALTER TABLE` or `ALTER VIEW`.

Remove security policy

```
ALTER TABLE w.employee DROP ROW ACCESS POLICY country_filter(country);
ALTER VIEW e.employees_view MODIFY COLUMN ssn_col UNSET MASKING POLICY protect_ssn;
```

Examples of UDFs

The following are examples of user-defined functions that you may create with Dremio.

Column-Masking Policies

Redact SSN

```
CREATE FUNCTION
    protect_ssn (val VARCHAR)
    RETURNS VARCHAR
    RETURN
        SELECT
            CASE
```

```
                WHEN query_user() IN ('jdoe@dremio.com','janders@dremio.com')
                    OR is_member('Accounting') THEN val
                ELSE CONCAT('XXX-XX-',SUBSTR(value,8,4))
            END;
```

Use column-masking and row-access policies

```
CREATE FUNCTION lower_country(country VARCHAR)
    RETURNS VARCHAR
    RETURN SELECT lower(country);

CREATE FUNCTION country_filter (country VARCHAR)
    RETURNS BOOLEAN
    RETURN SELECT query_user()='dremio'
                      OR (is_member('Accounting')
                              AND country='CA');

CREATE FUNCTION protect_ssn (ssn VARCHAR(11))
    RETURNS VARCHAR(11)
    RETURN SELECT CASE WHEN query_user()='dremio' OR is_member('Accounting') THEN ssn
            ELSE CONCAT('XXX-XXX-', SUBSTR(ssn,9,3))
        END;

CREATE FUNCTION salary_range (salary FLOAT, id INTEGER)
    RETURNS BOOLEAN
        RETURN SELECT CASE WHEN id > 1 AND salary > 10000 THEN true
            ELSE false
        END;
```

Use STRUCT

```
--
CREATE TABLE struct_demo (emp_info struct <name : VARCHAR>);
INSERT INTO nas.struct_demo VALUES(SELECT convert_from('{"name":"a"}', 'json'));
CREATE FUNCTION hello(nameCol struct<name:VARCHAR>) RETURNS struct<name:VARCHAR> RETURN
SELECT nameCol;
ALTER TABLE nas.struct_demo MODIFY COLUMN emp_info SET MASKING POLICY hello(emp_info);
```

Use LIST

```
CREATE FUNCTION hello_country(countryList LIST<VARCHAR>) RETURNS VARCHAR RETURN SELECT
'Hello World';
ALTER TABLE "test.json" MODIFY COLUMN country SET MASKING POLICY hello_country(country);
```

Row-Access Policies

Use simple filter expressions

```
CREATE FUNCTION country_filter (country VARCHAR)
    RETURNS BOOLEAN
```

```
        RETURN SELECT state='CA';
```

Match users

```
CREATE FUNCTION query_1(my_value varchar)
    RETURNS BOOLEAN
        RETURN SELECT CASE
            WHEN current_user = 'jdoe@dremio.com' THEN true
            ELSE false
        END;
```

Table-Driven Policy with a Subquery

Use a subquery as a table-driven policy

```
DROP TABLE <catalog-name>.salesmanagerregions;
CREATE TABLE <catalog-name>.salesmanagerregions (
    sales_manager varchar,
    sales_region varchar
);

INSERT INTO <catalog-name>.salesmanagerregions
VALUES ('john.smith@example.com', 'WW'),
('jane.doe@example.com', 'NA'),
('viktor.jones@example.com', 'EU');

CREATE TABLE  <catalog-name>.revenue (
    company varchar,
    region varchar,
    revenue decimal(18,2)
);

INSERT INTO <catalog-name>.revenue
VALUES ('Acme', 'EU', 2.5),
('Acme', 'NA', 1.5);

CREATE  OR  REPLACE  FUNCTION  security.sales_policy  (sales_region_in  varchar)  RETURNS
BOOLEAN
  RETURN SELECT is_member('sales_executive_role')
    OR EXISTS (
        SELECT 1 FROM <catalog-name>.salesmanagerregions
            WHERE user() = sales_manager
            AND sales_region = sales_region_in
        );

ALTER TABLE <catalog-name>.revenue
ADD ROW ACCESS POLICY security.sales_policy(region);

SELECT * FROM <catalog-name>.revenue;
-- company, region, revenue
-- Acme, NA, 1.50
```

Use Reflections on Datasets with Policies

Dremio supports Reflection creation on views and tables with row-access and column-masking policies defined on any of the underlying anchor datasets. See the following examples.

Example of a view with a row-access policy and a raw Reflection

```
-- Create nested views
CREATE OR REPLACE VIEW myView AS
  SELECT city, state, pop FROM Samples."samples.dremio.com"."zips.json"
  WHERE pop > 10000;
CREATE OR REPLACE VIEW myView2 AS
  SELECT city, state FROM myView
  WHERE STARTS_WITH(city, 'A');

-- Create a raw Reflection on the inner view
ALTER TABLE myView
  CREATE RAW REFLECTION myReflection
  USING DISPLAY(city, state);

-- Query the view after the Reflection is created
SELECT * FROM myView2;

-- Create a UDF
CREATE OR REPLACE FUNCTION isMA(state VARCHAR)
  RETURNS BOOLEAN
  RETURN SELECT CASE WHEN IS_MEMBER('hr') THEN state='MA'
      ELSE NULL
    END;

-- Add a row-access policy and query the view
ALTER TABLE myView
  ADD ROW ACCESS POLICY isMA("state");
SELECT * FROM myView2;
```

After running the last query, the Reflection is used to accelerate the query as shown in the results below:

The `Query1` results show that the row-access policy has been applied successfully:

The `Query2` results do not appear to those who are not members of HR:

The `Query2` results appear to those who are members of HR:

![](/assets/images/rcac_reflection_accelerated_member-9c4fd2be39fe181162620c678e

99766c.png)

Example of a table with a row-access policy and an aggregation Reflection

```
ALTER TABLE NAS.rcac.employee
  ADD ROW ACCESS POLICY is_recent_employee(hire_date);
ALTER TABLE NAS.rcac.employee
  CREATE AGGREGATE REFLECTION ar_tvrf_1 USING DIMENSIONS(hire_date);
SELECT MIN(SALARY) FROM NAS.rcac.employee
  GROUP BY hire_date;
```

Limitations

See the following limitations where datasets with row-access and/or column-masking policies cannot support Reflections:

- Policies with Multiple Arguments
- Aggregates on Masked Columns
- SET Operations
- NULL Generating JOINs
- Trimming Projects

Policies with Multiple Arguments

If a policy on an anchor dataset contains multiple columns, the Reflection created on the view containing the policy fails. See the following example:

Example of the limitation

```
-- Create tables
CREATE TABLE employees (
 id INT,
 hire_date DATE,
 ssn VARCHAR(11),
 name VARCHAR,
 country VARCHAR,
 salary FLOAT,
 job_id INT);
CREATE TABLE jobs (
 id INT,
 title VARCHAR,
 is_good BOOLEAN);

-- Create a view
CREATE VIEW job_salary_in_the_usa AS
 SELECT job_id, salary
 FROM employees
 WHERE country = 'USA';
```

```
-- Create a UDF
CREATE OR REPLACE FUNCTION hide_salary_on_bad_job(salary FLOAT, job_id_in INT)
  RETURNS BOOLEAN
  RETURN SELECT CASE WHEN IS_MEMBER('public') AND (
     SELECT is_good FROM jobs j WHERE job_id_in = j.id)
      THEN NULL
    ELSE salary
    END;

-- Add a column-masking policy
ALTER TABLE employees
 MODIFY COLUMN salary
 SET MASKING POLICY hide_salary_on_bad_job(salary, job_id);

-- Create a raw Reflection on the view
ALTER DATASET job_salary_in_the_usa
 CREATE RAW REFLECTION job_salary_drr USING DISPLAY(job_id, salary);
```

In the above example, the `job_salary_drr` Reflection fails to materialize due to the multi-argument policy on `test.tables.employees::salary`.

Aggregates on Masked Columns

You cannot create a raw Reflection on the view if there is a policy defined on the masked column.

Example of the limitation

```
CREATE OR REPLACE VIEW myView AS
  SELECT MIN(salary)
  FROM employees
```

In the above example, there is a policy defined on `salary`, so you cannot create a Reflection on this view.

NULL Generating JOINs

You can only apply the policy if it's on the “join side” of the join, such as:

- Left side of LEFT JOIN
- Right side of RIGHT JOIN
- Either side of INNER JOIN
- Neither side of FULL OUTER JOIN

If the policy is not on the "join side", the join generates NULL values for all the entries that didn’t match the join condition.

Example of the limitation

```
CREATE OR REPLACE VIEW myView AS
  SELECT emp.department_id, dept.department_name, emp.name
  FROM employees as emp
   RIGHT JOIN department as dept
   ON emp.department_id = dept.department_id
```

In the above example, there is a policy defined on the `employees` table, which is on the left side of the RIGHT JOIN, so you cannot create a Reflection on this view.

SET Operations

The policy must be defined on all UNION datasets and on the same field.

Example of the limitation

```
CREATE OR REPLACE VIEW myView AS
  SELECT * FROM a
  UNION SELECT * FROM employees
  UNION SELECT * FROM c
```

In the above example, there is a policy defined on the `employees` table, so you cannot create a Reflection on this view.

Trim Projects

In order to create a Reflection on a view, the view should reference all the fields that are part of the row-access and column-masking policies.

Example of the limitation

```
-- Create a UDF
CREATE OR REPLACE FUNCTION isMA(state VARCHAR)
  RETURNS BOOLEAN
  RETURN SELECT CASE WHEN IS_MEMBER('public') THEN state='MA'
      ELSE NULL
    END;

-- Create views
CREATE OR REPLACE VIEW myView1 AS
  SELECT city, state, pop FROM Samples."samples.dremio.com"."zips.json"
  WHERE pop > 10000;

-- Add a row-access policy
ALTER TABLE myView1
  ADD ROW ACCESS POLICY isMA("state");

-- Create views
CREATE OR REPLACE VIEW myView2 AS
  SELECT * FROM myView1;
CREATE OR REPLACE VIEW myView3 AS
  SELECT city, pop FROM myView1;
```

Trimming Projects

In the above example, you can create a Reflection on `myView2` but not on `myView3` since it trims the `state` column from the view which has a policy defined on it.

Was this page helpful?

- Column-Masking Policies
- Row-Access Policies
- User-Defined Functions
- Query Substitutions
- List Existing UDFs
- List Existing Policies
- Set a Policy
- Drop a Policy
- Examples of UDFs
- Column-Masking Policies
- Row-Access Policies
- Table-Driven Policy with a Subquery
- Use Reflections on Datasets with Policies
- Limitations

Source: dremio-cloud-overview.md

Overview | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/

Get Started

New to Dremio? Start here to create your organization, analyze data using natural language with Dremio's AI Agent, and take a tour of the Dremio console.

Start here

!Get Started

Quick Actions

[### Bring Your Data

Load data into Dremio's Open Catalog using your tool of choice. Connect catalogs, object storage, or databases for a unified view of your data.](/dremio-cloud/bring-data/)

[### Explore and Analyze Your Data

Discover, explore, and analyze your data using Dremio's AI Agent, by running SQL queries, or by using your BI tool of choice.](/dremio-cloud/explore-analyze/)

[### Manage and Govern Your Data

Organize Iceberg tables, track lineage, and add wikis and labels to build a shared semantic layer to provide AI with business context.](/dremio-cloud/manage-govern/)

What's New

Check out the latest features, enhancements, and fixes in Dremio's agentic lakehouse platform.

See all updates

Was this page helpful?

Source: dremio-cloud-security.md

Security and Compliance | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/security/

On this page

Dremio offers extensive security measures to help protect the integrity of your data, including access control and the ability to use external identity providers (IdPs). Dremio provides flexible native security features and integration with a wide range of third-party tools so that your organization can adhere to compliance and regulatory standards, enforce fine-grained permissions for your users, and retain your existing tools for authentication and authorization.

Authentication and Identity Management

Dremio supports industry-standard authentication and single sign-on (SSO) services, including OAuth 2.0/OpenID Connect. Organizations can configure integrated authentication (Active Directory or OpenID Connect) to centrally manage user accounts with strong password policies and SSO/multi-factor authentication (MFA).

Access Control

Dremio provides a comprehensive hierarchical privilege system for fine-grained access control across your organization.

Privileges – Complete privilege system with hierarchical inheritance from organization to individual objects.

Role-Based Access Control (RBAC) – Manage access through roles rather than individual user grants for easier administration.

Hierarchical Inheritance – Privileges granted at higher levels (Organization → Projects → Sources → Folders → Tables) automatically apply to nested objects.

Object Ownership – Automatic ownership assignment when creating objects, with transferable ownership capabilities.

Open Catalog Security – Structured access control for managed catalog systems.

Data Protection

Encryption in Transit – Your content is transmitted using TLS 1.2 or higher between client and control plane, and between control plane and data plane.

Encryption at Rest – Your data is encrypted at rest within the control plane using AES-256 or higher.

Customer-Managed Encryption – Deploy and manage your encryption keys for enhanced security.

Compliance and Certifications

Compliance – Review current compliance measures and audits Dremio has completed.

Current Certifications:

ISO 27001 – Information security management systems.

SOC 2 Type II – Security, availability, and confidentiality controls.

HIPAA – Healthcare data protection compliance.

Privacy Regulations:

GDPR – General Data Protection Regulation compliance.

CCPA – California Consumer Privacy Act compliance.

Was this page helpful?

Authentication and Identity Management

Access Control

Data Protection

Compliance and Certifications

Roles | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/security/roles

On this page

Roles are a set of privileges that can be assigned to users as needed. Roles can also be assigned to other roles to create a child-role hierarchy, where child roles inherit all privileges from their parent roles. This hierarchical system allows you to organize privileges at scale rather than managing privileges for each individual user (also called members).

You can define roles based on the types of users in your organization. For example, *Data_Analyst* and *Security_Admin* roles can be created to manage privileges for users with different job functions within an organization.

See the following role design guidelines:

- Keep the number of ADMIN role members to 1-2 administrators for security.
- Begin with 2-3 custom roles based on primary job functions.
- Create parent roles for common privilege sets, then add specific child roles as needed.
- Choose clear names that reflect the role's purpose (e.g., Sales_Analyst, Data_Engineer).
- Use prefixes such as DEPT_, PROJ_, or TEAM_ for consistency.
- Use the description field to explain each role's intent.

How Role Inheritance Works

Child roles automatically inherit all privileges from their parent roles, creating a cascading effect that simplifies privilege management.

Example Role Hierarchy

```
Data_Viewer (SELECT on public datasets only)
  └── Data_Analyst (inherits Data_Viewer + SELECT on specific datasets)
      └── Data_Engineer (inherits Data_Analyst + CREATE, ALTER privileges)
          └── Data_Admin (inherits Data_Engineer + admin privileges on data sources)
```

In this example, a Data_Engineer automatically gets all the privileges of Data_Analyst and Data_Viewer, plus their own additional CREATE and ALTER privileges.

System Roles

Dremio has two predefined system roles: ADMIN and PUBLIC. These roles can be used to manage privileges.

ADMIN

The ADMIN role is designed for administrative users who require superuser/global access. Users who are assigned this role are granted every privilege across all objects and resources in an organization. The privileges for the ADMIN role are immutable by users.

The first user in an organization is automatically assigned the ADMIN role.

Be cautious when assigning the ADMIN role. Users with ADMIN privileges can modify any data, delete objects, and manage other users' access.

PUBLIC

The PUBLIC role is assigned by default to all new users added to the organization and cannot be revoked from any user. Think of PUBLIC as the baseline access level that every user in your organization receives.

This role grants the following privileges to its members:

- USAGE on all engines
- USAGE on any predefined OAuth apps and External Token Providers.

SELECT and ALTER privileges are not granted for any sources and must be assigned by a user with the ADMIN role or through additional custom roles.

Additional privileges can be granted to the PUBLIC role to provide organization-wide baseline access.

Custom Roles

Custom roles can be created by any user or role that has the CREATE ROLE organization privilege, or by members of the ADMIN role.

You can assign a custom role to users or other roles (to create a child role). The custom role can then be assigned a set of privileges.

View All Roles

Use the Dremio Console

- Click !Settings in the side navigation bar and choose **Organization settings**.
- Select **Roles** in the organization settings sidebar.

Use SQL

ADMIN users can also list all roles using the `sys.organization.roles` system table:

Review all roles and their owners

```
SELECT r.role_name,
       r.role_type,
       r.owner_type,
       u.user_name as owner_name
FROM sys.organization.roles r
LEFT JOIN sys.organization.users u ON r.owner_id = u.user_id
ORDER BY r.role_name;
```

Create a Custom Role

Use the Dremio Console

Click !Settings in the side navigation bar and choose **Organization settings**.

Select **Roles** in the organization settings sidebar.

Click **Add Role** at the top-right corner of the screen.

In the Add Role dialog, for **Name**, enter the name to associate with the role, such as the position title or employee type that will be associated with the role.

(Optional) For **Description**, provide any details regarding the purpose of the role or its associated privileges.

Click **Add**.

Use SQL

You can also create custom roles using the `CREATE ROLE` command.

Edit a Custom Role

Use the Dremio Console

Click !Settings in the side navigation bar and choose **Organization settings**.

Select **Roles** in the organization settings sidebar.

On the Roles page, select the role.

On the Edit Role page, make any desired changes, such as adding or removing a child role and adding or removing a member.

Click **Save**.

Use SQL

You can also add or remove child roles and members using the GRANT ROLE and REVOKE ROLE SQL commands.

Remove a Custom Role

Removing a role will immediately revoke all associated privileges from its members. Ensure users have alternative access before deleting roles.

Use the Dremio Console

-

Click !Settings in the side navigation bar and choose **Organization settings**.

Select **Roles** in the organization settings sidebar.

On the Roles page, hover over the row of the role and click !Delete that appears next to the role.

Confirm that you want to delete the role.

Once confirmed, the role is deleted and cannot be retrieved.

Use SQL

You can also remove custom roles using the `DROP ROLE` command.

Add a Child Role

Child roles inherit all privileges from their parent roles. This creates a hierarchy where more specific roles build upon broader ones.

Use the Dremio Console

Click !Settings in the side navigation bar and choose **Organization settings**.

Select **Roles** in the organization settings sidebar.

On the Roles page, select the parent role, then select the **Roles** tab.

Click the dropdown multi-select field and either select the desired role or enter a role name to search for it.

Click *Add** when you have selected the desired entry or entries. When a child role is added, it will display below the dropdown in a list.

Click **Save**.

The child role appears in the table along the left side of the screen.

Use SQL

You can also add child roles to parent roles using the `GRANT ROLE` SQL command:

Example Association of a Child Role

```
-- Make Data_Analyst a child role of Analytics_Team
GRANT ROLE Data_Analyst TO ROLE Analytics_Team;
```

Remove a Child Role

Use the Dremio Console

Click !Settings in the side navigation bar and choose **Organization settings**.

Select **Roles** in the organization settings sidebar.

On the Roles page, select the parent role, then select the **Roles** tab.

Hover over the row of the role and click !Delete that appears next to the role.

Click **Save**.

Use SQL

You can also remove child roles from parent roles using the `REVOKE ROLE` SQL command.

Add a Member

Use the Dremio Console

Click !Settings in the side navigation bar and choose **Organization settings**.

Select **Roles** in the organization settings sidebar.

On the Roles page, select the role, then select the **Members** tab.

Click the dropdown multi-select field and either select the desired user (listed by email address) or enter an email address to search for it.

Click **Add** when you have selected the desired entry or entries. When a member is added, it will display below the dropdown in a list.

Click **Save**.

Use SQL

You can also add members to roles using the `GRANT ROLE` SQL command:

Example creating a role member

```
-- Assign Data_Analyst role to a user
GRANT ROLE Data_Analyst TO USER 'jane.doe@company.com';
```

Remove a Member

Users cannot remove themselves from the ADMIN role. If you are a member of the ADMIN role and wish to be removed from it, another user who has the necessary privileges must remove you.

Use the Dremio Console

Click !Settings in the side navigation bar and choose **Organization settings**.

Select **Roles** in the organization settings sidebar.

On the Roles page, select the role, then select the **Members** tab.

Hover over the row of the member and click !Delete that appears next to the member.

Click **Save**.

This removes them as a member of this role, and they will no longer possess the privileges associated with that role. However, the user still retains privileges associated with any other roles where they are members.

Use SQL

You can also remove members from roles using the `REVOKE ROLE` SQL command.

Limits and Considerations

There is a limit of 10 nested roles in a hierarchy. For more information, see Limits.

Was this page helpful?

Compliance | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/security/compliance

On this page

Dremio meets the IT control requirements for several compliance frameworks and certifications, as described below.

SOC 2 Type II Report

Dremio maintains compliance with the American Institute of Certified Public Accountants (AICPA) System and Organization Controls - Trust Services Criteria, commonly known as SOC 2.

Key Benefits

SOC 2 Type II reports provide an in-depth analysis of cloud service providers regarding the safeguards used to protect data and how controls are performed. These reports are issued by independent, third-party auditors and cover the key areas of security, availability, confidentiality, and privacy.

This independent assessment of Dremio provides a detailed report regarding the environments used to provide security and data privacy. The report includes descriptions of these controls, the tests performed to assess their effectiveness, the results of those tests, and an overall opinion regarding the design and operational effectiveness of the environments.

ISO 27001 Certification

ISO 27001 is an internationally recognized specification for an Information Security Management System (ISMS). ISO 27001 is the only auditable standard that addresses the overall management of information security rather than just which technical controls to implement.

Key Benefits

Obtaining ISO 27001:2022 certification demonstrates that Dremio employs a comprehensive framework of legal, physical, and technical controls for information risk management.

GDPR Compliance

Dremio is compliant with the storage and security of its data according to Article 27 of the General Data Protection Regulation (GDPR). Please see Dremio's Privacy Policy for additional information regarding our appointed European Data Protection Officer (EDPO) in the EU.

Key Benefits

As part of the European Union, specific regulations exist that require companies to maintain compliance with GDPR. This regulation governs the way user data is stored, processed, and utilized on Dremio. Specifically, it prevents the exploitation of user data and standardizes the data protection laws that services must follow throughout Europe.

CCPA Compliance

Dremio maintains compliance with the California Consumer Privacy Act (CCPA), which regulates the handling of personal data and prevents any unauthorized use or sale. Please see Dremio's Privacy Notice for California Residents for additional information.

Key Benefits

Adherence to CCPA by an organization ensures that California residents have the right to opt out of having their data sold to third parties, request disclosure of data collected, and request deletion of that data.

HIPAA Compliance

Dremio is compliant with the Health Insurance Portability and Accountability Act (HIPAA), a series of federal regulatory standards that outline the lawful use and disclosure of protected health information in the United States. HIPAA compliance is regulated by the Department of Health and Human Services (HHS) and enforced by the Office for Civil Rights (OCR).

Key Benefits

Adherence to HIPAA ensures that healthcare providers, health plans, healthcare clearinghouses, and business associates of HIPAA-covered entities must implement multiple safeguards to protect sensitive personal and health information.

Was this page helpful?

Privileges | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/security/privileges

On this page

Dremio provides a range of privileges for each type of securable object. These privileges work together to control access across your organization.

Key Concepts

Grants

Dremio privileges are granted to users and roles. Users possess all the privileges granted to their user identity and their roles. See `GRANT TO USER` and `GRANT TO ROLE` for example grants.

Privilege Inheritance

Dremio uses a hierarchical privilege system where most higher-level privileges apply to all objects within their scope:

Organization → Projects → Sources → Folders → Tables and Views

When you grant a privilege at a higher level, it applies to all relevant objects at lower levels. For example, granting SELECT at the project level gives SELECT access to all datasets in that project across all sources.

Ownership and Object Creation

The OWNERSHIP privilege is unique—it applies only to the specific object where it's assigned and is never inherited by nested objects. When you create any object, you automatically become its owner. This design maintains clear ownership boundaries, so a project owner doesn't automatically own every table in that project. OWNERSHIP grants full control of the specific object. Ownership can be transferred using the `GRANT OWNERSHIP` command.

Sharing Data Through Views

When you create a view based on a table, you become the owner of that view. Your privileges as the view owner determine whether the view can access the underlying table—creating a privilege chain. You can then grant other users access to your view, allowing them to see the table's data even though they don't have direct permission to access that table themselves. However, if you (or whoever last modified the view) lose access to the underlying table, the entire privilege chain breaks and the view stops working for everyone.

Organization Privileges

Organization privileges are the highest level in the hierarchy and control organization-wide operations and resources.

Privilege Type	Description
ALL	Shorthand to grant all supported privileges except OWNERSHIP.
CALL MODEL	Use the AI models available across all model providers. The PUBLIC role

has this privilege on the organization by default, but it can be revoked. |
CONFIGURE BILLING	Create and manage billing accounts for usage invoices.
CONFIGURE SECURITY	Configure organization security features including identity providers, external token providers, and custom OAuth applications.
CREATE MODEL PROVIDER	Create model providers for the organization.
CREATE PROJECT	Create new projects, each including an Open Catalog.
CREATE ROLE	Create and edit roles. See Custom Roles for details.
CREATE USER	Create and edit users. See Add a User for details.
MANAGE GRANTS	Grant or revoke privileges on the organization and all objects it contains.
OWNERSHIP	Full control of the organization; not inherited by nested objects.

Project Privileges

Project privileges control access to projects and apply to different categories of objects within the project. These privileges provide broad control across all sources, catalogs, and engines in the project.

Privilege Type	Applies To	Description
OWNERSHIP	Project	Full control of the project.
USAGE	Project	Access the project and its engines. Required for any other project operations.
VIEW JOB HISTORY	Project	View the job history page for all users across the entire project.
CREATE SOURCE	Sources	Create new data sources and modify source configurations throughout the project.
EXTERNAL QUERY	Sources	Run external queries on compatible sources.
ALTER	Datasets	Edit definitions, settings, wikis, and manage metadata. Create or remove folders and datasets where supported.
CREATE TABLE	Datasets	Create tables using `CREATE TABLE` and `CREATE TABLE AS` on sources that support table creation.
DELETE INSERT TRUNCATE UPDATE	Datasets	Execute DML operations on Apache Iceberg tables in compatible object storage.
DROP	Datasets	Remove tables and folders from all sources that support deletion operations.
SELECT	Datasets	Query contained datasets and view schema definitions, lineages, wikis, and labels.
ALTER REFLECTION	Reflections	Create, edit, and view all Reflections across the project. Includes access to Reflection pages, API endpoints, and job history.
VIEW REFLECTION	Reflections	View all Reflections across the project, including pages, API endpoints, and job history.
MODIFY	Engines	Complete engine management including workload settings, routing, and queues. Includes MONITOR and OPERATE.
MONITOR	Engines	View all engine settings including replicas, auto-stop settings, time limits, and tags across all engines.
OPERATE	Engines	Start, stop, enable, and disable all engines in the project.
MANAGE GRANTS	All Objects	Grant and revoke privileges on the project and all objects it contains.

Open Catalog Privileges

Open Catalog is a specialized source whose privileges control access to folders and

datasets within the catalog.

You can grant each of these privileges at the indicated scopes:

Catalog scope: Privileges are granted on the catalog and apply to all the catalog folders and datasets.

Folder scope: Privileges are granted to a specific folder and apply to all contained folders and datasets.

Dataset scope: Privileges are granted to a single table or view and apply only to that dataset.

Privilege Type	Catalog Scope	Folder Scope	Dataset Scope	Description
ALL	✔	✔	✔	Shorthand to grant all supported privileges except OWNERSHIP.
ALTER	✔	✔	✔	Edit contained table definitions, settings, wikis, and manage metadata operations. Add or remove folders.
ALTER REFLECTION	✔	✔	✔	Create, edit, and view Reflections on contained datasets, including pages, APIs, and job history.
DROP	✔	✔		Remove contained datasets and folders.
MANAGE GRANTS	✔	✔	✔	Grant and revoke privileges on contained objects.
OWNERSHIP	✔	✔	✔	Full control; not inherited by nested objects.
READ METADATA	✔	✔	✔	View metadata including column information and job history, limited to jobs you have permission to see.
SELECT	✔	✔	✔	Query contained datasets and view schema definitions, lineages, wikis, and labels.
USAGE	✔	✔		Use the immediate namespace or folder. Must be granted on every folder in the hierarchy path.
VIEW REFLECTION	✔	✔	✔	View Reflections on contained datasets, including pages, APIs, and job history.
WRITE	✔	✔	✔	Execute write operations `INSERT`, `UPDATE`, `TRUNCATE`, `DELETE` on contained Apache Iceberg tables.

Source Privileges

Source privileges control access to external data sources and datasets. All sources and other catalogs utilize these privileges in Dremio.

You can grant each of these privileges at the indicated scopes:

Source scope: Privileges are granted on the source and apply to all the source folders and datasets.

Folder scope: Privileges are granted to a specific folder and apply to all contained folders and datasets.

Dataset scope: Privileges are granted to a specific table or view and apply only to that dataset.

Privilege Type	Source Scope	Folder Scope	Dataset Scope	Description
ALL	✔	✔	✔	Shorthand to grant all supported privileges except OWNERSHIP.
ALTER	✔	✔	✔	Edit contained dataset definitions, settings, wikis, and manage metadata. Add or remove folders, promote or demote tables.

ALTER REFLECTION	✔	✔	✔	Create, edit, and view all Reflections on contained datasets, including pages, APIs, and job history.
CREATE TABLE	✔	✔		Create new tables using `CREATE TABLE` and `CREATE TABLE AS` (requires source to support table creation).
DELETE INSERT TRUNCATE UPDATE	✔	✔	✔	Execute associated DML operations `DELETE`, `INSERT`, `TRUNCATE`, `UPDATE` on all contained Apache Iceberg tables (requires compatible object storage).
DROP	✔	✔		Remove contained datasets and folders (requires source deletion support).
EXTERNAL QUERY	✔			Run external queries on compatible sources.
MANAGE GRANTS	✔	✔	✔	Grant and revoke privileges on contained objects.
MODIFY	✔			Access and modify configuration settings, connection parameters, and source-level properties.
OWNERSHIP	✔	✔	✔	Full control; not inherited by nested objects.
READ METADATA	✔	✔	✔	View metadata including column information and job history, limited to jobs you have permission to see.
SELECT	✔	✔	✔	Query contained datasets and view schema definitions, lineages, wikis, and labels.
VIEW REFLECTION	✔	✔	✔	View Reflections on contained datasets, including pages, APIs, and job history.

User-Defined Function Privileges

User-defined functions (UDFs) allow you to create reusable custom functions using SQL expressions.

Privilege Type	Description
ALL	Shorthand to grant all supported privileges except OWNERSHIP.
ALTER	Edit the function's wiki, definitions, and settings.
EXECUTE	Ability to run the UDF. Use the function as row-access and column-masking policies for tables and views.
MANAGE GRANTS	Grant and revoke privileges on the UDF.
OWNERSHIP	Full control of the UDF; not inherited by nested objects.

Engine Privileges

Engine privileges control access to specific named engines. Use engine privileges at the project level to manage all engines collectively.

Privilege Type	Description
ALL	Shorthand to grant all supported privileges except OWNERSHIP.
MANAGE GRANTS	Grant and revoke privileges on the specific engine.
MODIFY	Access and modify all engine settings including replicas, auto-stop configuration, time limits, and tags.
MONITOR	View all engine settings and configuration details without modification rights.
OPERATE	Start, stop, enable, and disable the engine.
OWNERSHIP	Full control of the engine; not inherited by nested objects.
USAGE	Execute queries using the engine. The PUBLIC role has this privilege on all engines by default, but it can be revoked.

Model Provider Privileges

Model provider privileges control access to AI model providers configured at the organization level. These privileges determine who can use, manage, and configure model providers for your organization.

Privilege Type	Description
CALL MODEL	Use the AI models available.
MODIFY	Access and modify all model provider settings.
MANAGE GRANTS	Grant and revoke privileges on the model provider.
OWNERSHIP	Full control of the model provider.

Script Privileges

Script privileges enable sharing of individual saved scripts with other users and roles.

Privilege Type	Description
ALL	Shorthand to grant all supported privileges except OWNERSHIP.
DELETE	Remove the script permanently.
MANAGE GRANTS	Grant and revoke privileges on the script.
MODIFY	Edit the script content and settings.
OWNERSHIP	Full control of the script; not inherited by nested objects.
VIEW	Access, view, and execute the script.

Identity Provider Privileges

Identity provider privileges control access to organization-level authentication and identity management settings.

Privilege Type	Description
ALL	Shorthand to grant all supported privileges except OWNERSHIP.
MODIFY	Access and modify identity provider settings, including configuration changes and updates.
MONITOR	View all identity provider settings and configuration details without modification rights.
OWNERSHIP	Full control of the identity provider; not inherited by nested objects.

Related Topics

Security Pillar – See the security design principles and best practices of the Dremio Well-Architected Framework.

Was this page helpful?

Key Concepts

Grants

Privilege Inheritance

- Ownership and Object Creation
- Sharing Data Through Views
- Organization Privileges
- Project Privileges
- Open Catalog Privileges
- Source Privileges
- User-Defined Function Privileges
- Engine Privileges
- Model Provider Privileges
- Script Privileges
- Identity Provider Privileges
- Related Topics

PrivateLink | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/security/privatelink

On this page

Dremio PrivateLink enables secure, private connectivity between your AWS VPC and Dremio services without exposing traffic to the public internet. This service allows you to access all Dremio control plane services, including the UI, REST APIs, and query execution endpoints.

When you enable PrivateLink for your Dremio organization, all Dremio services are accessible only through your VPC endpoint. However, the following Dremio services remain publicly accessible:

- `login.dremio.cloud` – OAuth server for programmatic authentication (API clients, JDBC/ODBC)
- `scim.dremio.cloud` – SCIM provisioning endpoint for identity provider integration (Microsoft Entra ID, Okta, etc.)
- `sql.dremio.cloud` – Dremio JDBC driver (Legacy) endpoint.

If your organization restricts outbound internet access, ensure the `accounts.dremio.cloud` (or `accounts.eu.dremio.cloud` for EU regions) domain is allowed in your firewall rules for authentication to function properly. This authentication service is used during single sign-on (SSO) login flows.

Upon activation of PrivateLink, Dremio console sessions terminate immediately, JDBC/ODBC/API sessions terminate within one hour, and running queries may be interrupted.

Before activating PrivateLink in your Dremio organization:

- Verify your VPC endpoint is available.

- Confirm DNS resolution is working and connections through the endpoint are functioning.

- Schedule a maintenance window and notify users.

PrivateLink uses a **service-based routing** approach with the following domain structure:

`<orgAlias>.<resource>.privatelink.dremio.cloud`

Domain Components

- **orgAlias** – Your organization's unique identifier that routes connections to your Dremio organization. Requirements:

  - Starts with a letter (a-z, A-Z)

  - Contains only letters, digits, and hyphens

  - Ends with a letter or digit (not a hyphen)

  - Length: 3-63 characters

  - Case-insensitive (stored as lowercase)

  - Follows RFC 1035 DNS naming conventions

- **resource** – The Dremio services in the connection. The following interfaces are not supported by PrivateLink:

  - `sql.dremio.cloud` for the Dremio JDBC driver (Legacy). Dremio recommends the Arrow Flight SQL JDBC driver using the `data.dremio.cloud` service endpoint when using PrivateLink.

  - `mcp.dremio.cloud` for AI agent integration. Once PrivateLink is activated, this endpoint will not be available.

- **privatelink.dremio.cloud** – The PrivateLink domain suffix for all private connections

Examples:

- `acme-corp.app.privatelink.dremio.cloud` – Routes to the Dremio console at `app.dremio.cloud`

- `acme-corp.api.privatelink.dremio.cloud` – Routes to the REST API at `api.dremio.cloud`

Network Components

PrivateLink uses a VPC endpoint in your AWS VPC to provide secure, private connectivity to Dremio services. Users and applications within the VPC connect through the VPC endpoint using your privately hosted DNS name resolution. Remote users connect via VPN to access the VPC and its resources.

Certificate Management

Dremio uses wildcard certificates for `*.privatelink.dremio.cloud`. No additional certificate management is required. Server certificates are managed by Dremio, and standard TLS verification applies to client vertification. All certificates are publicly logged.

Prerequisites

Before setting up PrivateLink, ensure you have:

AWS Requirements

VPC: Your VPC in the same region as your Dremio service, where you want to enable PrivateLink connectivity.

Subnets: At least one subnet in your VPC. When you create a VPC endpoint, you select one or more subnets, and AWS creates an Elastic Network Interface (ENI) in each selected subnet. All ENIs belong to the same VPC endpoint. Select subnets in multiple availability zones for high availability—if one availability zone fails, traffic continues to flow through ENIs in other zones.

VPC Endpoints: Permission to create and manage VPC endpoints.

Security Groups: Ability to create or modify security groups.

Network Requirements

DNS Resolution: Ability to configure private DNS (such as Route 53 Private Hosted Zones) or CNAME records in your VPC. You will need to create CNAME records that map PrivateLink URLs like `acme-corp.app.privatelink.dremio.cloud` to your VPC endpoint DNS name. While you could technically connect using the VPC endpoint DNS name directly, DNS configuration is required for proper TLS certificate validation and to enable host-based routing to different Dremio services, including `app`, `api`, `data`, and `login`.

TLS/SSL: Your environment must support TLS 1.2 or higher.

Client Requirements

Arrow Flight Drivers: All SQL clients and BI tools must use Arrow Flight-based drivers. Some clients and tools provide their own embedded drivers, but you must use the Dremio Arrow Flight JDBC and ODBC drivers in place of those embedded drivers.

Configuration Steps

To create a PrivateLink connection:

Create a VPC Endpoint – In the Amazon Management Console, create a VPC endpoint for connecting to an endpoint service as the service consumer, using the steps defined by AWS.

For **Type**, choose **Endpoint services that use NLBs and GWLBs**.

For **Service Name**, enter the Dremio service name for your Dremio region:

- us-east-1: `com.amazonaws.vpce.us-east-1.vpce-svc-0c795b359782ac685`
- us-west-2: `com.amazonaws.vpce.us-west-2.vpce-svc-0b42aeb4681d6f4a4`

- Select your VPC, subnets, and additional configurations.
- Optionally define a DNS name for your VPC endpoint and enter that name in your privately hosted DNS.
- Click **Create endpoint**.

- **Configure a Security Group** – Attach a security group with the following rules:

- Inbound Rules:

Type	Protocol	Port Range	Source	Description
HTTPS	TCP	443	Your VPC CIDR or specific security groups	Allow HTTPS traffic from your resources

- Outbound Rules:

Type	Protocol	Port Range	Destination	Description
HTTPS	TCP	443	0.0.0.0/0	Allow outbound HTTPS (required for SSO authentication)

- **Configure Private DNS** – Create CNAME records in your private DNS (Route 53 Private Hosted Zone or equivalent) to map Dremio service domains to your VPC endpoint DNS name. See AWS documentation for creating a private hosted zone. Create one CNAME record for each PrivateLink URL associated with a Dremio service. Replace `<orgAlias>` with your organization alias and `<vpc-endpoint-dns-name>` with the DNS name of your VPC endpoint (found in the AWS Console under VPC > Endpoints).

- `<orgAlias>.app.privatelink.dremio.cloud` → `<vpc-endpoint-dns-name>`
- `<orgAlias>.api.privatelink.dremio.cloud` → `<vpc-endpoint-dns-name>`
- `<orgAlias>.data.privatelink.dremio.cloud` → `<vpc-endpoint-dns-name>`
- `<orgAlias>.login.privatelink.dremio.cloud` → `<vpc-endpoint-dns-name>`

- **Configure Client Tools** – Configure client applications to use the PrivateLink endpoints:

- **Power BI Desktop** - See Connect to Dremio via PrivateLink.

- **JDBC/ODBC Drivers** - Update connection strings to use `<orgAlias>.data.privatelink.dremio.cloud`.

- **REST API Clients** - Update base URL to `https://<orgAlias>.api.privatelink.dremio.cloud`.

- **Verify Connectivity** – Test connectivity to Dremio using the VPC endpoint and private DNS:

- Test DNS resolution using `nslookup <orgAlias>.app.privatelink.dremio.cloud`. This should resolve to private IP addresses in your VPC.

From a system within your VPC, test access to the Dremio console by navigating to `https://<orgAlias>.app.privatelink.dremio.cloud`. You should see the Dremio login page.

From a system within your VPC, test API access by calling an API endpoint with a base URL of `curl https://<orgAlias>.api.privatelink.dremio.cloud/api/v0/`.

Enable PrivateLink – Enable PrivateLink by filing a support ticket with Dremio Support at the Dremio Support Portal. In the support ticket, provide:

Your **orgAlias**

Your Dremio **Organization ID** by clicking !Settings in the side navigation bar, choosing **Organization Settings**, and then copying the **Organization ID**.

Your VPC endpoint ID from the AWS Console.

Confirmation that connectivity works using your new VPC endpoint.

Resume Operation – Resume operation utilizing your PrivateLink connections.

Was this page helpful?

Domain Components

Network Components

Certificate Management

Prerequisites

Configuration Steps

Authentication | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/security/authentication/

On this page

Dremio supports multiple authentication methods for different connection types and user scenarios.

Use Case	Connection Type	Recommended Method
Interactive web access	Dremio console	Single Sign-On or Username/Password
SQL clients	JDBC/ODBC clients	Personal Access Tokens (PAT) or Username/Password
Development & testing	Client applications, REST API	Personal Access Tokens (PAT)
Production scripts & automation	Client applications, REST API	OAuth access tokens via PAT Exchange
Custom apps with existing IdP	Client applications, REST API	OAuth access tokens via External JWT Exchange

Username/Password

Username and password authentication allows users to sign in directly to Dremio using their email address and a password managed within Dremio. This method is suitable for users who don't have access to an enterprise identity provider or need standalone accounts. Users can reset their passwords through the Dremio console or via email reset links.

Single Sign-On

Users authenticate through configured identity providers using OIDC protocols. Dremio supports all OIDC-compliant enterprise identity providers, such as Microsoft Entra ID and Okta, as well as social identity providers like Google and GitHub. Users experience automatic login if already signed in to their identity provider.

Personal Access Tokens (PAT)

Personal access tokens are long-lived authentication credentials that allow programmatic access to Dremio without using passwords. PATs function like API keys and can be used in scripts, applications, and automated processes to authenticate requests.

Token lifespan: PATs can be configured with custom expiration periods up to 180 days or set to never expire. You control the lifespan when creating the token.

Security considerations:

- PATs can have lifespans up to 180 days, making them convenient but potentially risky if compromised.
- Store PATs securely using environment variables or secret management systems.
- Never include PATs in code repositories or logs.
- Regularly rotate PATs and revoke unused tokens.
- Consider using PAT Exchange for enhanced security in production environments.

Users can create and manage PATs through their Account Settings in the Dremio console.

OAuth Access Tokens

OAuth access tokens are short-lived credentials obtained by exchanging other authentication methods (such as PATs or external JWTs). These tokens provide several security advantages:

Limited lifespan: Tokens expire after 1 hour, reducing risk if compromised.

- **Reduced credential exposure:** Your primary credentials (PAT or password) are only used to obtain the token.
- **Standardized format:** Compatible with OAuth 2.0 standards and tooling.
- **Automatic refresh:** Can be programmatically renewed without re-entering credentials.

Token lifespan: OAuth access tokens expire after 1 hour. Applications should implement refresh logic to obtain new tokens before expiration. When a token expires, API requests will return an authentication error, requiring your application to exchange credentials again for a new token.

OAuth access tokens are the recommended authentication method for production applications accessing Dremio's REST API and client drivers. You can obtain OAuth access tokens through PAT Exchange or External JWT Exchange.

PAT Exchange

Converting PATs to short-lived OAuth access tokens improves security by reducing exposure windows for compromised tokens. This is the recommended method for obtaining OAuth access tokens for REST API access.

The process:

- Create a PAT in your Dremio account settings.
- Exchange the PAT for an OAuth access token via the `/oauth/token` REST API.
- Use the OAuth access token for all subsequent API requests.
- Refresh the token before it expires (within 1 hour).

External JWT Exchange

Applications can exchange JSON Web Tokens (JWTs) from external token providers for Dremio OAuth access tokens, enabling authentication without exposing user credentials. This method is useful for custom applications that need to authenticate users through their existing identity provider (such as Microsoft Entra ID or Okta) and then access Dremio on their behalf.

The process:

- User authenticates with the external identity provider.
- Application receives a JWT from the identity provider.
- Application exchanges the JWT for a Dremio OAuth access token via the `/oauth/token` REST API.
- Application uses the Dremio OAuth access token to make authenticated requests.
- Application refreshes the token before it expires.

This approach allows applications to maintain a seamless authentication experience while securing access to Dremio resources.

Was this page helpful?

- Username/Password
- Single Sign-On
- Personal Access Tokens (PAT)

- OAuth Access Tokens

Identity Providers | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/security/authentication/idp/

On this page

Identity providers (IdPs) are services that store and manage digital identities. An IdP authenticates users via username-password combinations and other credentials, as typically used for cloud computing and managing user identities. The following IdPs are supported with Dremio:

- Enterprise identity providers, including Microsoft Entra ID, Okta, and other OpenID Connect (OIDC) providers.
- Social identity providers, including GitHub, Microsoft, and Google.

View an IdP

To view an IdP configured for Dremio:

- In the Dremio console, click !Settings in the side navigation bar and then select **Organization settings**.
- Select **Authentication** from the organization settings sidebar.

Remove an IdP

You can only remove enterprise IdPs. Social IdPs cannot be removed as they are preconfigured with Dremio.

To remove an enterprise IdP:

- Click !Settings in the side navigation bar and then select **Organization settings**.
- Select **Authentication** from the organization settings sidebar.
- Click !Delete on the row of the IdP to remove. Removing an activated IdP removes it as a login option for all users within your organization. You must manually reconfigure the IdP if you want to use it again as a login option.
- Confirm that you want to remove the IdP. The IdP is then deleted along with any associated settings.

SCIM

System for Cross-domain Identity Management (SCIM) automates the synchronization of user accounts between your identity provider (IdP) and Dremio, eliminating the need for manual user management. When configured, IdPs send the credentials of assigned users securely via SCIM to your Dremio organization, automatically creating new user accounts if needed. These new users, also referred to as external users, can then log in to Dremio according to the policies set by your credential manager.

You cannot reset or change an external user's email address or password from Dremio because these tasks are governed by your organization's credential manager. If you delete an external user from Dremio, the IdP automatically re-adds the user's account the next time that user attempts to log in. To properly revoke access to Dremio, follow the steps for Microsoft Entra ID or Okta.

Configure Microsoft Entra ID with SCIM

You can use Microsoft Entra ID to securely provision external users in Dremio with SCIM. See SCIM Provisioning with Microsoft Entra ID for more information and instructions.

Configure Okta with SCIM

Dremio supports the Okta SCIM provisioning feature, which allows you to automatically create Dremio user accounts if they do not already exist, update user attributes in Dremio, and deactivate user accounts, all from Okta.

Before you can configure Okta SCIM provisioning, you must configure Okta as an IdP in Dremio. Follow the instructions in Okta as an Identity Provider to integrate the Dremio application in your Okta organization and add Okta as an OpenID Connect (OIDC) IdP in Dremio.

After you configure Okta as an IdP, you can configure Okta to use SCIM for secure user provisioning.

Limits and Considerations

To provide a consistent experience, Dremio uses rate limits for SCIM provisioning requests. For more information, see Limits.

Dremio allows one update to a user or group at a time. While the update is in progress, Dremio locks the user or group and rejects concurrent requests to update the same user or group.

Was this page helpful?

View an IdP

Remove an IdP

SCIM

Configure Microsoft Entra ID with SCIM

Configure Okta with SCIM

Limits and Considerations

Okta | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/security/authentication/idp/okta

On this page

Dremio supports Okta as an enterprise identity provider. Okta administrators can enable single sign-on (SSO) authentication using Okta as the trusted third party.

Prerequisites

Configuring OIDC SSO in Okta requires:

- Super Administrator access in Okta
- The CONFIGURE SECURITY organization-level privilege or membership in the ADMIN role.

Supported Features

Dremio supports the following Okta SSO features:

- **Service provider-initiated (SP-initiated) SSO**: Dremio uses the OpenID Connect (OIDC) protocol for SP-initiated SSO. When users provide their email address to log in to Dremio, Dremio sends an authentication request to Okta. Okta then authenticates the user's identity, and the user is logged in to Dremio.
- **SCIM**: Dremio also allows you to take advantage of Okta's System for Cross-domain Identity Management (SCIM) provisioning feature and manage Dremio user access from Okta. After you configure Okta for OIDC SSO in this guide, see SCIM with Okta to configure SCIM provisioning.

Configure OIDC SSO

To configure Okta OIDC SSO for Dremio users:

- In Okta, navigate to **Applications** > **Applications** and click **Browse App Catalog**.
- Type `Dremio` in the search field and select **Dremio** from the list of search results.
- Click **Add Integration**.
- (Optional) Type a custom label in the *Application label* field.
- Select your Dremio control plane region from the *Region* dropdown menu: US or EU.
- Click **Done**. Okta creates the Dremio application and displays the application's *Assignments* tab.
- Click the **Sign On** tab.
- Copy and save the client ID and client secret listed under *OpenID Connect*. The client ID and client secret are sensitive information and should be kept secure. You will use them to configure authentication in Dremio later in this procedure.
- Click the **OpenID Provider Metadata** link to open the OpenID configuration for the application.
- Copy and save the URL value for the `issuer` key at the top of the OpenID configuration. You will use it to configure authentication in Dremio later in this

procedure.

In the Dremio console, click !Settings in the side navigation bar and select **Organization settings**.

Select **Authentication** in the organization settings sidebar.

In the Enterprise section, click **Add Provider** to open the Add Provider dialog.

In Step 1, select **Okta** from the dropdown menu.

In Step 3, enter the issuer URL, client ID, and client secret information that you copied from Okta in the corresponding fields.

Click **Add**. After the page loads, you should see Okta as an authentication provider in the *Enterprise* section.

Click the **Enabled** toggle to activate the Okta authentication provider.

Okta is now configured as an enterprise authentication provider. **Log in with Okta** appears in the list of login options for your Dremio users.

Assign People and Groups to the Dremio Application

Follow the instructions in the Okta documentation to assign people or assign groups to the Dremio application to ensure that users can use Okta for SSO login. The users you assign, whether individually or through their membership in an assigned group, can use **Log in with Okta** immediately.

Use privileges and roles to manage user access to objects in Dremio.

Use Okta SSO to Log In to Dremio

Any Okta user who is assigned to the Dremio application can log in with Okta immediately. To use Okta SSO to log in to Dremio:

Open the Dremio login page.

Type your email address in the *Email* field and click **Continue**.

Click **Log in with Okta**.

When you are redirected to the Okta website for authentication, enter your Okta username and password and click **Sign In**.

Okta authenticates your identity and redirects you to Dremio, which then logs you in.

To configure Okta's SCIM provisioning feature and use Okta to manage access for Dremio users, see SCIM with Okta.

Revoke Okta SSO Login for a User or Group

To revoke users' access to Okta SSO login for Dremio:

In Okta, open your Dremio application and select the **Assignments** tab.

In the left menu, under *Filters*, select **People** to deactivate a user or **Groups** to deactivate a group of users.

Find the row for the user or group you want to deactivate and click the **X** on the right side of the row.

In the confirmation dialog that appears, click **OK**.

Starting immediately, the deactivated users cannot use Okta OIDC SSO to log in to Dremio. To completely delete Dremio users, you must also manually remove their user accounts in Dremio.

Troubleshoot

This section describes some things to keep in mind about OIDC SSO in Okta.

To add the Dremio application in Okta and configure OIDC SSO, you must be a super administrator in the Okta organization.

If you revoke a user's access to use Okta SSO login in Okta, the user can still log in to Dremio with their Dremio username and password. To completely delete the user so that they cannot log in to Dremio at all, you must manually remove their user accounts in Dremio.

Configure Okta with SCIM

System for Cross-domain Identity Management (SCIM) automates the synchronization of user accounts between your identity provider (IdP) and Dremio, eliminating the need for manual user management. When configured, your IdP securely sends user credentials to Dremio via SCIM, automatically creating accounts for new users as needed. These users can then log in to Dremio according to your organization's authentication policies.

Before you can configure SCIM provisioning, you must configure Okta as an identity provider (IdP) in Dremio. See Okta as an Identity Provider to integrate the Dremio application in your Okta organization and add Okta as an OpenID Connect (OIDC) single sign-on (SSO) IdP in Dremio. When that is complete, follow this guide to configure Okta to use SCIM for secure user provisioning.

Prerequisites

Configuring SCIM provisioning in Okta requires:

Super Administrator access in Okta

The CONFIGURE SECURITY organization-level privilege or membership in the ADMIN role.

A Dremio personal access token (PAT)

You must configure Okta as an identity provider using the Dremio application **before** you proceed with SCIM provisioning.

Supported Features

Dremio supports the following Okta SCIM provisioning features:

- **Create Users**: Automatically create a new user account in Dremio for Okta users who are assigned to the Dremio application, whether they are assigned individually or as members of a group that is assigned to the application.

Update User Attributes: Automatically update user information in Dremio when a user's profile information is updated in Okta.

- **Deactivate Users**: Prevent users from logging in to Dremio when they are deactivated in Okta.

- **Group Push**: Push Okta groups and their members to Dremio to automatically create Dremio roles and members.

Configure SCIM Provisioning

To configure and enable SCIM provisioning in Okta:

- Confirm that you have configured Okta as an identity provider using the Dremio application.

- In Okta, navigate to **Applications** > **Applications**.

- Find the Dremio application in the list of applications and click to open it.

- Click the **Provisioning** tab.

- Click **Configure API Integration**.

- Select **Enable API integration**.

- Enter the Dremio PAT in the *API Token* field.

- Click **Test API Credentials**. You should see a confirmation message that the connection was verified successfully.

- Click **Save**. Okta displays the *Provisioning to App* page.

- Click **Edit**.

- Select **Enable** for the *Create Users*, *Update User Attributes*, and *Deactivate Users* options.

- Click **Save**.

SCIM provisioning is now configured and enabled. You can create new users, update user attributes, and deactivate users in Dremio, all from Okta.

Create Users

After you configure Okta's SCIM provisioning and enable the *Create Users* option, Dremio automatically creates a new Dremio user account for anyone you assign to

Dremio who does not already have an account. New Dremio users can log in to Dremio with Okta SSO immediately, and administrators can view their user accounts in Dremio.

New users are automatically members of the PUBLIC role in Dremio.

User email addresses are controlled by Okta rather than Dremio. If a user's email address changes, you must create a new user in Okta and assign them to the Dremio application. Then, the user can use the new email address to log in to Dremio as a new user.

Update User Attributes

With SCIM provisioning configured, updates to user attributes in Okta are propagated to the user account in Dremio. Follow the instructions in the Okta documentation to edit user attributes.

The *First name* and *Last name* attributes are mapped to user accounts in Dremio. After you configure Okta's SCIM provisioning and enable the *Update User Attributes* option, you can change these user attributes in Okta to update the corresponding user information in Dremio.

Deactivate Users

When you revoke a user or group in Okta, the affected users cannot use Okta OIDC SSO to log in to Dremio. After you configure Okta's SCIM provisioning and enable the *Deactivate Users* option, deactivated users become inactive in Dremio and cannot log in to Dremio at all, whether with Okta OIDC SSO or username and password.

To completely delete Dremio users, you must also manually remove their user accounts in Dremio.

Group Push

If you enable the group push feature, Okta pushes your designated groups to Dremio as roles and populates the roles with the Okta group's members. Follow the instructions in the Okta documentation to enable group push.

Before you enable group push, make sure to follow Okta's instructions to assign the group to the Dremio application.

Use Okta to manage any roles you create with group push. Any changes you make to a role or its membership in Dremio are immediately overwritten by the next push from Okta. Making changes in Dremio can result in synchronization errors.

To remove a Dremio role created by group push, unlink the pushed group in the Dremio application. Unlinking the pushed group deletes the corresponding role in Dremio but does not delete the group members' Dremio user accounts.

Troubleshoot

This section describes some things to keep in mind about SCIM provisioning in Okta with the Dremio application.

- Group push is not supported for groups that do not have any members. Pushing a group that does not have any members will result in an error.

- In Okta, it is possible to change a user's username. Dremio does not allow username updates. If you change a user's Okta username after the user is assigned to the Dremio application, Okta sends a request to update the username in Dremio. Dremio denies the request because Dremio username changes are not allowed.

- Changing an existing user's primary email address in Okta has no effect on the user's account in Dremio. To permit a user to authenticate to Dremio with the new email address, add the user to Okta as a new person using the new email address. Then, assign the new Okta user to the Dremio application, either individually or by adding them to an assigned group. Okta creates a new Dremio user who can use Okta SSO to log in to Dremio with the new email address.

- If you remove a user from an assigned group and the user is still listed as ACTIVE in Dremio, check the *Assignments* tab in the Dremio application to make sure the user isn't separately assigned as a person. Okta only sends deactivate requests for users who are both unassigned as a person and removed from assigned groups.

Was this page helpful?

- Prerequisites

- Supported Features

- Configure OIDC SSO

- Assign People and Groups to the Dremio Application

- Use Okta SSO to Log In to Dremio

- Revoke Okta SSO Login for a User or Group

- Troubleshoot

- Configure Okta with SCIM

- Prerequisites

- Supported Features

- Configure SCIM Provisioning

- Create Users

- Update User Attributes

- Deactivate Users

- Group Push

- Troubleshoot

Social Identity Providers | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/security/authentication/idp/social-idp

On this page

A social identity provider (IdP) enables users to log in to Dremio using their existing accounts from these services. You can use the following providers:

- GitHub
- Google
- Microsoft

By default, these options are preconfigured and active, which means they are immediately available as login options for users unless deactivated by an admin.

Log In with a Social IdP

Follow these steps to log in to your organization with an enabled social IdP:

1. Navigate to Dremio's login screen, enter your email address, and proceed to the next screen.
2. Click the icon of the desired social IdP (GitHub, Google, or Microsoft) that you want to use. You will be redirected to the corresponding provider's login page.

!Social login interface

3. Enter your credentials. If successful, you will be redirected to the Dremio homepage.

Activate and Deactivate Social IdPs

You must be an admin to activate or deactivate a social IdP. Follow these steps to deactivate or activate social providers:

1. In the Dremio console, click !Settings in the side navigation bar and then select **Organization settings**.
2. Select **Authentication** from the organization settings sidebar.
3. To deactivate a provider, toggle **Enabled** to off. Deactivating a social IdP removes this IdP as a login option for all users in your organization.
4. To activate a deactivated, toggle **Enabled** to on.

Was this page helpful?

- Log In with a Social IdP
- Activate and Deactivate Social IdPs

Application Authentication | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/security/authentication/app-authentication/

On this page

Application authentication enables programmatic access to Dremio for automated workflows, integrations, and service-to-service communication. Unlike user authentication, which is designed for interactive sessions, application authentication provides secure, token-based access for applications, scripts, and third-party tools.

Application authentication is essential for:

- **API Integrations**: Connecting BI tools, ETL pipelines, and custom applications.
- **Automated Workflows**: Scheduled data processing and reporting tasks.
- **Service-to-Service Communication**: Microservices accessing Dremio resources.
- **CI/CD Pipelines**: Automated testing and deployment processes.

Dremio supports two primary application authentication methods that differ fundamentally in their authentication flow and token issuance:

Method	Authentication Flow	Token Issuer	Best For
OAuth Applications	Redirect to Dremio login, user authenticates, redirect back with token	OAuth access token from Dremio	Third-party applications, custom applications requiring standard OAuth
External Token Providers	User authenticates with enterprise IdP, JWT used directly with Dremio	JWT from your identity provider, OAuth access token from Dremio	Enterprise SSO environments, existing JWT infrastructure

OAuth Applications

OAuth 2.0 provides secure, standardized authorization for third-party applications. This method is ideal when you need user consent or want to integrate with applications that already support OAuth flows.

Key Features:

- Supports industry-standard OAuth 2.0 flows
- Manages granular permissions through Dremio role-based access control and access policies
- Logs user activity

External Token Providers

External token providers allow you to use JSON Web Tokens (JWTs) issued by your existing OAuth server or identity provider. This approach is ideal for enterprises with established identity infrastructure.

Key Features:

-

Leverages existing identity systems

Supports custom claims and token validation

Integrates with enterprise SSO

Manages centralized tokens

Was this page helpful?

OAuth Applications

External Token Providers

Personal Access Tokens | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/security/authentication/personal-access-token

On this page

Personal access tokens (PATs) are randomly generated tokens associated with a user that are used in place of a password to authenticate with Dremio. PATs can last up to 180 days before they expire and provide a secure way to enable programmatic access, automation, and CI/CD workflows.

When using a PAT, you have the same privileges and roles as the user who created the token. This means a PAT can only access what the user can access.

When to Use PATs

Dremio recommends using OAuth access tokens for most use cases, as they provide enhanced security through shorter lifespans and centralized management. PATs should primarily be used in scenarios where OAuth tokens are not supported or practical.

PATs may be appropriate for:

Legacy systems: Applications that cannot support OAuth authentication flows.

Simple scripts: Quick automation tasks where OAuth setup overhead is not justified.

Development and testing: Temporary access for development workflows.

ODBC/JDBC connections: When OAuth is not supported by the client application.

Create a PAT

To create a PAT:

Click the User icon (user initials) on the side navigation bar and select **Account Settings**.

Select **Personal Access Tokens** in the account settings sidebar.

On the Personal Access Tokens page, click **Generate Token** in the top-right corner of the screen.

In the Generate Token dialog, for **Label**, add a descriptive identifier explaining what the PAT is for (e.g., "CI Pipeline - Data Tests" or "Tableau Integration").

For **Lifetime**, enter the number of days the PAT will be valid. The default PAT lifetime is 30 days, and the maximum lifetime is 180 days.

Click **Generate**.

Important: Copy the generated PAT immediately and save it to a secure location. The token is shown only once and cannot be retrieved later.

Manage PATs

View PAT Metadata

A PAT is shown only once during creation. However, you can view the token ID, label, creation date, and expiration status for all PATs in your account.

To view the metadata for all the PATs you have created:

Click the User icon (user initials) on the side navigation bar and select **Account Settings**.

Select **Personal Access Tokens** from the settings sidebar.

The Personal Access Tokens page displays all the metadata for PATs, both active and expired, for your account.

Delete a PAT

Each user can delete PATs in their own account.

To delete an existing PAT:

Click the User icon (user initials) on the side navigation bar and select **Account Settings**.

Select **Personal Access Tokens** in the account settings sidebar.

On the Personal Access Tokens page, click !Delete for the PAT that you want to delete.

In the Delete Token dialog, click **Delete** to confirm. The PAT is deleted and cannot be retrieved.

Delete All PATs

Any user can delete all PATs from their own account. ADMIN users cannot delete PATs on behalf of other users.

To delete all PATs for your account:

Click the User icon (user initials) on the side navigation bar and select **Account Settings**.

Select **Personal Access Tokens** in the account settings sidebar.

On the Personal Access Tokens page, click **Delete All** in the top-right corner of the screen.

In the Delete All Tokens dialog, click **Delete** to confirm that you want to delete all PATs in the list. After a PAT has been deleted, it cannot be retrieved.

Use PATs

PATs can be used to authenticate with various Dremio interfaces:

REST API**:** Use PATs for programmatic access and automation.

JDBC**:** Connect applications using JDBC drivers.

ODBC**:** Connect applications using ODBC drivers.

Dremio web application: Use your PAT as a password to log in.

For specific connection details and examples, see the documentation for each connection method.

Limits and Considerations

Self-service only: Users can only create and manage PATs for themselves—even ADMIN users cannot create or manage PATs on behalf of other users.

User permissions: PATs are tied to user accounts—if a user is deactivated, their PATs stop working.

No privilege restriction: PATs cannot be scoped to fewer privileges than the user has.

Token management: Use descriptive labels and set appropriate expiration times for each token.

Was this page helpful?

Limits and Considerations

Microsoft Entra ID | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/security/authentication/idp/microsoft-entra-id

On this page

Dremio supports Microsoft Entra ID as an enterprise identity provider. Microsoft Entra ID administrators can follow these instructions to enable single sign-on (SSO) authentication and allow users to log in to Dremio using Microsoft Entra ID as the trusted third party.

Prerequisites

Configuring SSO in Microsoft Entra ID requires:

- Privileges in Microsoft Entra ID that permit you to add, configure, and register applications.
- The CONFIGURE SECURITY organization-level privilege or membership in the ADMIN role.

Configure an Application for SSO

To configure SSO in Microsoft Entra ID for Dremio users:

1. In the Azure portal under **Azure services**, click the **Microsoft Entra ID** tile.
2. In the left navigation menu under **Manage**, click **App registrations**.
3. Click **New registration**.
4. Type a name for the application in the **Name** field.
5. Select your desired account type in the **Supported account types** list. The default selection is `Accounts in this organizational directory only (<your org> only - Single tenant)`.
6. Under **Redirect URI**, in the **Select a platform** dropdown list, select **Web** and enter the following URI in the provided field:
   - US region: <https://accounts.dremio.cloud/login/callback>
   - EMEA region: <https://accounts.eu.dremio.cloud/login/callback>
7. Click **Register**.
8. Copy and save the value for the `Application (client) ID`. You will use it to configure authentication in Dremio later in this procedure.
9. In the left navigation menu under **Manage**, click **Certificates & secrets**.
10. -

Click **New client secret**.

In the **Add a client secret** panel, type a description for the secret in the **Description** field and select your desired lifespan for the secret in the **Expires** dropdown list.

Click **Add**.

Copy and save the value for the secret. The secret value is sensitive information and should be kept private. You will use it to configure authentication in Dremio later in this procedure.

In the left navigation menu under **Manage**, click **API permissions**.

Confirm that the following permission is listed under **API / Permissions name*:

User.Read: Permits users to log in to the application and permits the application to read the profiles and basic company information for logged-in users.

Click **Add a permission**.

In the **Request API permissions** panel, click the **Microsoft Graph** tile.

Click the **Delegated permissions** tile.

Under **OpenId permissions**, click the checkboxes next to the following options:

email: Permits the application to read users' primary email addresses.

openid: Permits users to sign in to the application with their work or school accounts and permits the application to view basic user profile information.

profile: Permits the application to view basic user profile information (name, avatar, and email address).

Click **Add permissions**. The list of configured permissions should now include the following permissions:

email

openid

profile

In the left navigation menu under **Manage**, click **Branding & properties**.

Copy and save the **Publisher domain** (`<domain_name>.onmicrosoft.com`). You will use it to configure authentication in Dremio later in this procedure.

In the Dremio console, click !Settings on the left navigation bar and then select **Organization settings**.

Click the **Authentication** tab in the left sidebar.

In the **Enterprise** section, click **Add Provider** to open the Add Provider dialog.

In Step 1, select **Microsoft Entra ID** in the dropdown list.

In Step 3, enter the domain, client ID, and secret information that you copied from Microsoft Entra ID in the corresponding fields.

Click **Add**. After the page loads, you should see Microsoft Entra ID listed as an authentication provider in the **Enterprise** section.

Click the **Enabled** toggle to activate the Microsoft Entra ID authentication provider.

Microsoft Entra ID is now configured as an enterprise authentication provider. **Log in with Microsoft Entra ID** appears in the list of login options for your Dremio users. Any Microsoft Entra ID user in your organization can use **Log in with Microsoft Entra ID** for SSO login.

Assign People and Groups to the Microsoft Entra ID Application

The Microsoft Entra ID application is configured to allow SSO login for any Microsoft Entra ID user in your organization. To adjust the application settings so that only users who are assigned to the app can use Microsoft Entra ID SSO to log in to Dremio:

In the Azure portal under **Azure services**, click the **Microsoft Entra ID** tile.

In the left navigation menu under **Manage**, click **Enterprise applications**.

Click the name of the SSO application.

In the left navigation menu under **Manage**, click **Properties**.

Find the **Assignment required?** toggle and click **Yes**.

Click **Save**.

With user assignment required, users who are not assigned to the application receive an error message from Microsoft when they try to use Microsoft Entra ID SSO for Dremio.

Follow the instructions in the Microsoft Entra ID documentation to assign users and groups to your application.

Before the user can click **Log in with Microsoft Entra ID** in the list of login options for Dremio, one of the following conditions must be met:

The user has been invited by an admin and has activated their account through an email link.

An admin has set up SCIM provisioning and synced the user via SCIM.

Use privileges and roles to manage user access to objects in Dremio.

Use Microsoft Entra ID SSO to Log in to Dremio

To use Microsoft Entra ID SSO to log in to Dremio:

Open the Dremio console login page:

US region: <https://app.dremio.cloud/>

EMEA region: <https://app.eu.dremio.cloud/>

Type your email address in the **Email** field and click **Continue**.

Click **Log in with Microsoft Entra ID**.

You will be redirected to the Microsoft website for authentication.

Microsoft Entra ID authenticates your identity and redirects you to Dremio, which then logs you in.

You can use the Microsoft Entra ID SCIM provisioning feature to sync groups and memberships from Microsoft Entra ID to Dremio and manage access for Dremio users and groups. To configure, see Configure Microsoft Entra ID with SCIM.

Revoke Microsoft Entra ID SSO Login for a User or Group

To revoke users' access to Microsoft Entra ID SSO login for Dremio:

In Microsoft Entra ID, navigate to your application.

Find the row for the user or group you want to deactivate and click to select the checkbox for the user or group.

Click **Remove**.

In the confirmation dialog, click **Yes**.

Starting immediately, the users cannot use Microsoft Entra ID SSO to log in to Dremio.

If you revoke a user's access to use Microsoft Entra ID SSO login in Microsoft Entra ID and the user has created a Dremio password for login, they can still log in to Dremio with their Dremio username and password. To completely delete Dremio users so that they cannot log in to Dremio at all, you must also delete or deactivate the user through SCIM provisioning or manually remove their user accounts in Dremio.

Configure Microsoft Entra ID with SCIM

System for Cross-domain Identity Management (SCIM) automates the synchronization of user accounts between your identity provider (IdP) and Dremio, eliminating the need for manual user management. When configured, your IdP securely sends user credentials to Dremio via SCIM, automatically creating accounts for new users as needed. These users can then log in to Dremio according to your organization's authentication policies.

Prerequisites

Configuring SCIM provisioning in Microsoft Entra ID requires:

Privileges in Microsoft Entra ID that permit you to register and configure applications.

A Dremio personal access token (PAT) for a Dremio user who is a member of the ADMIN role.

Configure an Application for SCIM Provisioning

To create an application for SCIM provisioning in Microsoft Entra ID:

In the Azure portal under **Azure services**, click the **Microsoft Entra ID** tile.

In the left navigation menu under **Manage**, click **Enterprise applications**.

Click **New application**.

Click **Create your own application**.

In the **Create your own application** panel, type a name for the application in the provided field.

Under **What are you looking to do with your application?** select the **Integrate any other application you don't find in the gallery (Non-gallery)** option.

Click **Create**.

In the left navigation menu under **Manage**, click **Provisioning**.

Click **Get started**.

In the **Provisioning Mode** dropdown list, select **Automatic**.

Under **Admin Credentials**, enter the correct **Tenant URL** for your control plane:

US control plane: `https://scim.dremio.cloud/scim/v2/?aadOptscim062020`

EU control plane: `https://scim.eu.dremio.cloud/scim/v2/?aadOptscim062020`

note

The Tenant URL must include the `aadOptscim062020` query parameter due to a Microsoft Entra ID issue with SCIM 2.0 compliance.

If you previously configured a SCIM app with Microsoft Entra ID, SCIM syncing may fail for requests to deactivate users, add and update user attributes, and remove group members. If you observe these failures, follow the Microsoft documentation to upgrade from the older customappsso job to the SCIM job.

Enter your Dremio PAT in the **Secret Token** field.

(Optional) Click **Test Connection** to confirm that Microsoft Entra ID can connect to the tenant URL.

Click **Save**.

(Optional) Click the down arrow next to **Settings** and adjust the settings as desired. Click **Save** when you are finished.

Return to the **Provisioning Overview** page for the application.

In the left navigation menu under **Manage**, click **Provisioning**.

Under **Provisioning Status**, toggle the setting to **On**.

Click **Save**.

SCIM provisioning is now configured and enabled. You can create users, update user

attributes, and deactivate users in Dremio, all from Microsoft Entra ID.

Read Microsoft's documentation about how long it takes to provision users for details about Microsoft Entra ID's initial and incremental provisioning cycles.

If desired, you can use Microsoft Entra ID's scoping filters to apply attribute-based rules for user provisioning. Read Scoping users or groups to be provisioned with scoping filters in the Microsoft documentation for more information.

Create Users

After you configure a Microsoft Entra ID application for SCIM provisioning, you must assign users and groups to the application. Dremio automatically creates a new Dremio user account for anyone you assign to the SCIM application who does not already have an account. Follow the instructions in the Microsoft documentation to assign users and groups to an application.

Create Roles

If you add a group to your SCIM application in Microsoft Entra ID, your designated group becomes a role in Dremio populated with the group's members. Follow the instructions in the Microsoft documentation to assign users and groups to an application.

Use Microsoft Entra ID to manage any roles you create with groups. Any changes you make to a role or its membership in Dremio are immediately overwritten by the next provisioning cycle from Microsoft Entra ID. Making changes in Dremio can result in synchronization errors.

Update User Attributes

With SCIM provisioning configured, updates to user attributes in Microsoft Entra ID are propagated to the user account in Dremio. Follow the instructions in the Microsoft documentation to edit user profile information.

First name and **Last name** attributes in Microsoft Entra ID are mapped to user accounts in Dremio. After you configure an application for SCIM provisioning in Microsoft Entra ID and assign users to it, you can change these user attributes in Microsoft Entra ID to update the corresponding user information in Dremio.

Microsoft Entra ID controls user **email addresses**. If a user's email address changes, you must create a new user in Microsoft Entra ID and assign them to the application for SCIM provisioning. Then, assign the new Microsoft Entra ID user to the SCIM application (either individually as a user or by adding them to an assigned group). Microsoft Entra ID creates a new Dremio user who can log in to Dremio with the new email address as a new user.

Deactivate Users

When you delete a user or group from the application for SCIM provisioning in Microsoft Entra ID, the affected users become inactive in Dremio and cannot log in to Dremio at all, whether with Microsoft Entra ID SSO or username and password.

To delete a user or group from your SCIM application in Microsoft Entra ID:

In the Azure portal under **Azure services**, click the **Microsoft Entra ID** tile.

In the left navigation menu under **Manage**, click **Enterprise applications**.

Find your SCIM application in the list and click the application's name.

In the left navigation menu under **Manage**, click **Users and groups**.

Click to select the checkbox for the user or group you want to remove.

Click **Remove**.

In the confirmation dialog, click **Yes**.

The users you deleted, whether individually or by their group membership, become inactive in Dremio. If you delete a group, Microsoft Entra ID automatically removes the group's corresponding role in Dremio.

If you delete a group in Microsoft Entra ID, the group's corresponding role is automatically removed in Dremio and the group members' Dremio user accounts are set to inactive. Deleting a Microsoft Entra ID group does not delete the group members' Dremio user accounts.

To completely delete Dremio users, you must manually remove their user accounts in Dremio in addition to deleting the users and any groups they belong to from the SCIM application in Microsoft Entra ID.

Was this page helpful?

Prerequisites

Configure an Application for SSO

Assign People and Groups to the Microsoft Entra ID Application

Use Microsoft Entra ID SSO to Log in to Dremio

Revoke Microsoft Entra ID SSO Login for a User or Group

Configure Microsoft Entra ID with SCIM

Prerequisites

Configure an Application for SCIM Provisioning

Create Users

Create Roles

Update User Attributes

Deactivate Users

Generic OIDC | Dremio Documentation

Original URL:

https://docs.dremio.com/dremio-cloud/security/authentication/idp/generic-oidc-provider

On this page

Dremio supports the generic OpenID Connect (OIDC) authentication protocol as an enterprise identity provider. OIDC provider administrators can register a Dremio application and use it to enable single sign-on (SSO) and allow users to log in using an OIDC provider as the trusted third party.

note

To configure Microsoft Entra ID or Okta as an identity provider, see:

Microsoft Entra ID as an Identity Provider

Okta as an Identity Provider

Dremio also allows you to use System for Cross-domain Identity Management (SCIM) provisioning to manage Dremio user access from your OIDC provider. After you configure your provider for OIDC SSO, refer to your OIDC provider's documentation to configure SCIM. See SCIM with a Generic OpenID Connect Provider to use SCIM provisioning in Dremio.

Prerequisites

Configuring SSO in a generic OIDC provider requires:

Privileges in the OIDC provider that permit you to add, configure, and register applications.

The CONFIGURE SECURITY organization-level privilege or membership in the ADMIN role.

Configure OIDC SSO

To configure OIDC SSO for Dremio users:

In Dremio, on the organization page, click !Settings on the left navigation bar, then select **Organization settings**.

Select **Authentication** in the organization settings sidebar.

Click **Add Provider** to open the Add Provider dialog.

In Step 1, select **OpenID Connect (OIDC)** from the dropdown menu.

Copy and save the **Redirect URL** listed in Step 2. The redirect URL is sensitive information and should be kept secure. You will need it to register the `Dremio` application in your OIDC provider portal in the next step.

In your OIDC provider portal, register `Dremio` as an application.

Copy and save the client ID and client secret for your OIDC provider. The client ID and client secret are sensitive information and should be kept secure. You will use them to configure authentication in Dremio later in this procedure.

Copy and save the `issuer` value from the OIDC configuration. You will use it to configure authentication in Dremio later in this procedure.

In Dremio, in Step 3 of the **Add Provider** dialog, enter the issuer URL, client ID, and client secret that you copied from your OIDC provider portal in the corresponding fields.

Click **Add**. After the page loads, you should see your OIDC provider in the **Enterprise** section.

Click the **Enabled** toggle to activate your OIDC provider.

OIDC as an enterprise identity provider is now configured. **Log in with SSO** appears in the list of login options for your Dremio users.

Use SSO to Log In to Dremio

Any user who is assigned to the `Dremio` application in your OIDC provider can log in with SSO immediately. To use SSO to log in to Dremio:

Open the Dremio login page.

Type your email address in the **Email** field and click **Continue**.

If you belong to more than one Dremio organization, select the organization to log in to.

Click **Log in with SSO**.

When you are redirected to your OIDC provider for authentication, enter your username and password.

The OIDC provider authenticates your identity and redirects you to Dremio, which then logs you in.

To configure SCIM provisioning to manage access for Dremio users, see SCIM with a Generic OpenID Connect Provider.

Revoke SSO Login for a User or Group

To revoke users' access to SSO login for Dremio:

In your OIDC provider's portal, navigate to the `Dremio` application.

Open the assignment settings for the `Dremio` application.

Find the user or group whose access you want to revoke and follow your OIDC provider's procedures to revoke access.

Starting immediately, the deactivated users cannot use OIDC SSO to log in to Dremio.

To completely delete Dremio users, you must also manually remove their user accounts in Dremio.

Configure a Generic OpenID Connect Provider with SCIM

System for Cross-domain Identity Management (SCIM) automates the synchronization of user accounts between your identity provider (IdP) and Dremio, eliminating the need for manual user management. When configured, your IdP securely sends user credentials to Dremio via SCIM, automatically creating accounts for new users as needed. These users can then log in to Dremio according to your organization's authentication policies.

Before you can configure SCIM provisioning, you must configure a generic OIDC provider as an enterprise identity provider in Dremio. Follow the instructions in Generic OpenID Connect Identity Provider to integrate a `Dremio` application in a generic OIDC provider for single sign-on (SSO) in Dremio. When that is done, follow this guide to configure SCIM for secure user provisioning.

Prerequisites

Configuring SCIM provisioning requires:

- Privileges in your OIDC provider that permit you to register and configure applications.
- The CONFIGURE SECURITY organization-level privilege or membership in the ADMIN role.
- A Dremio personal access token (PAT) for a Dremio user who is a member of the ADMIN role.

Configure SCIM Provisioning

The steps required to configure and enable SCIM provisioning vary for different OIDC providers. Follow the instructions in your OIDC provider's documentation.

Use a Dremio PAT as the **API Token** or **Secret Token** value when you configure authentication for SCIM requests in your OIDC provider's portal.

US Control Plane

```
https://scim.dremio.cloud/scim/v2
```

EU Control Plane

```
https://scim.eu.dremio.cloud/scim/v2
```

After SCIM provisioning is configured and enabled, you can create users, update user attributes, and deactivate users in Dremio from your OIDC provider's portal.

Create Users

After you configure SCIM provisioning, Dremio automatically creates a new Dremio user account for anyone you assign to the `Dremio` application in your OIDC provider who does not already have an account. New Dremio users can log in to Dremio with SSO immediately, and administrators can view their user accounts in Dremio.

-

New users are automatically members of the PUBLIC role in Dremio.

User email addresses are controlled by your OIDC provider rather than Dremio. If a user's email address changes, you must create a new user in your OIDC provider and assign them to the `Dremio` application. Then, the user can use the new email address to log in to Dremio as a new user.

Update User Attributes

With SCIM provisioning configured, updates to user attributes in your OIDC provider are propagated to the user account in Dremio.

The first name and last name attributes are mapped to user accounts in Dremio. After you configure SCIM provisioning and allow user attributes to be updated, you can change these user attributes in your OIDC provider to update the corresponding user information in Dremio.

Deactivate Users

When you revoke a user or group in your OIDC provider, the affected users cannot use OIDC SSO to log in to Dremio. After you configure SCIM provisioning and deactivate users, they become inactive in Dremio and cannot log in to Dremio at all with SSO.

To completely delete Dremio users, you must also manually remove their user accounts in Dremio.

Troubleshoot

This section describes some considerations about OIDC SSO and SCIM provisioning with the `Dremio` application in your OIDC provider.

SCIM provisioning

Dremio does not allow username updates. If you change a user's username in your OIDC provider after the user is assigned to the `Dremio` application, the OIDC provider sends a request to update the username in Dremio. Dremio denies the request because Dremio username changes are not allowed.

Changing an existing user's primary email address in the OIDC provider has no effect on the user's account in Dremio. To permit a user to authenticate to Dremio with the new email address, add the user to your OIDC provider as a new person using the new email address. Then, assign the new user to the `Dremio` application (either individually as a person or by adding them to an assigned group). The OIDC provider creates a new Dremio user who can use SSO to log in to Dremio with the new email address.

OIDC SSO

Refer to your OIDC provider's documentation to ensure that you have privileges that permit you to add the `Dremio` application in your OIDC provider and configure OIDC SSO.

If you revoke a user's access to SSO login, the user can still log in to Dremio with their

Dremio username and password. To completely delete the user so that they cannot log in to Dremio at all, you must manually remove their user accounts in Dremio.

Was this page helpful?

Prerequisites

Configure OIDC SSO

Use SSO to Log In to Dremio

Revoke SSO Login for a User or Group

Configure a Generic OpenID Connect Provider with SCIM

Prerequisites

Configure SCIM Provisioning

Create Users

Update User Attributes

Deactivate Users

Troubleshoot

Prerequisites

Original URL: https://docs.dremio.com/dremio-cloud/security/authentication/app-authentication/oauth-apps

On this page

This topic describes how to configure OAuth applications to integrate third-party applications with Dremio. This allows Dremio and third-party applications to interact without exposing user login credentials. For example, an organization might use GitLab accounts to access Dremio. In the unlikely event of a Dremio data breach, the organization's login credentials would remain unaffected and protected.

Additional authentication and security measures are available in Authentication.

Native – Mobile, desktop, CLI, and smart device apps that run natively on their respective operating systems, such as iOS and Chrome OS.

Single-Page Application (SPA) – JavaScript-enabled, front-end applications that use an API, such as Angular, React, and Vue.

Web – Traditional web applications that utilize redirects, such as Java, PHP, and ASP.NET.

Prerequisites

Before setting up OAuth applications, ensure you have:

-

Dremio admin privileges or the CONFIGURE SECURITY privilege.

An OIDC-compliant Identity Provider (IDP) configured in Dremio if OAuth applications will rely on external authentication.

Add an OAuth App

To add a new OAuth application in Dremio:

Click !Settings on the left navigation bar and select **Organization settings**, then select **OAuth Applications**.

Click **Add Application** in the top-right corner of the screen.

Enter a value for **OAuth Application Name** to identify the associated service.

Enter a value for **Redirect URI**. This value is used as the destination for return responses (tokens) after successfully authenticating a user. If there is an issue with the provided URI's format, red text will display below the field to indicate the required format.

Select the desired **OAuth Application Type** from the dropdown menu. The type of application selected determines which authentication flow Dremio will follow. This cannot be changed after the application is added.

Click **Add** to create the application service. A success message will appear at the top of the screen.

Upon creating the application, the dialog will refresh with a new field: **Client ID**. Copy this value, as it is needed to link with the third-party OAuth application. Include this string where the **Client ID** is required by your respective OAuth application.

Edit an OAuth App

To edit an existing OAuth application in Dremio:

Click !Settings on the left navigation bar and select **Organization settings**, then select **OAuth Applications**.

Click the edit icon for the desired application.

Make any necessary changes to the application name or redirect URI. You cannot change the **Client ID** or **OAuth Application Type**.

Click **Save**.

Was this page helpful?

Add an OAuth App

Edit an OAuth App

External Token Providers | Dremio Documentation

Original URL: https://docs.dremio.com/dremio-cloud/security/authentication/app-authentication/external-token

On this page

External token providers are OIDC identity providers that issue JSON Web Tokens (JWTs) when a user authenticates using an application client. After receiving a JWT from the external token provider, the client application uses Dremio token exchange to obtain an OAuth access token used to create connections to Dremio.

The OIDC external token provider does not need to be the same identity provider used by the Dremio console for single sign-on (SSO). The provider requires an application registration specifying the OAuth authorization flow to be used between the external token provider and the client to obtain the JWT that will be sent to Dremio.

This page outlines the steps for configuring an external token provider so Dremio can interpret and validate the JWTs issued by your provider.

The OIDC specification describes the content of the JWT and the authorization process. Claims in a JWT contain information asserted about a subject. They are key/value pairs in which the key is a string, and the value can be any JSON type (a string, a number, a boolean, an array, or a JSON object).

Example: External JWT Claims from Microsoft Entra ID

```
{
  "aud": "0853fce0-c748-4c54-aa58-f5b9af279840",
  "iss": "https://login.microsoftonline.com/3e334762-b0c6-4c36-9faf-93800f0d6c71/v2.0",

  "upn": "gnarly@dremio.com"
}
```

Prerequisites

Before setting up External Token Providers, ensure you have:

- Dremio admin privileges or the CONFIGURE SECURITY privilege.
- An OIDC-compliant Identity Provider configured with an application registration for your client.
- Access to the following information from your IDP:
  - **Audience** – Application ID or resource URI
  - **User claim mapping** – The claim containing the Dremio username
  - **Issuer URL** – Identity provider identification
  - **JWKS URL** – Optional location of public keys

Define an External Token Provider

Dremio requires the following configuration values from your OIDC identity provider.

tip

The examples below are specific to Microsoft Entra ID. Your identity provider may require additional configuration of a client application registration that depends on the OAuth authorization flow used between your client and your provider. To configure your application registration, consult your identity provider documentation.

Audience

The audience value identifies the intended recipients of the external JWT. It can generally be an array of case-sensitive strings or URI values. The audience is contained in the `aud` claim in the external JWT.

When using Microsoft Entra ID, the audience can be the Application ID assigned to your app in the Microsoft Entra ID portal or the resource URI. In v2.0 tokens, this value is always the Application ID. In v1.0 tokens, it can be the Application ID or the resource URI used in the request, depending on how the client requested the token. Dremio supports v1.0 and v2.0 JWTs from Microsoft Entra ID.

Example Audience Claim with Microsoft Entra ID Application ID

```
"aud": "0853fce0-c748-4c54-aa58-f5b9af279840"
```

User Claim Mapping

The user claim mapping identifies the claim in the external JWT that contains the Dremio username.

When using Microsoft Entra ID authentication, Dremio usernames must align with the User Principal Name (UPN) claim for correct linking of user group memberships via the Azure Graph Client.

When a user is added to a Power BI workspace, the user's identity is also represented by the User Principal Name (UPN), which has the format of an email address.

The JWT contains the UPN claim, named `upn`, and its value.

Example: UPN Claim from Microsoft Entra ID

```
"upn": "gnarly@dremio.com"
```

The `user claim mapping` field of the external token provider requires the name of the claim used in the JWT, which in this case is `upn`.

Issuer URL

The issuer URL identifies the identity provider that issued the JWT. It is contained in the external JWT's `iss` claim. When using Microsoft Entra ID, the issuer claim includes the Microsoft Entra ID tenant identifier. Only one external token provider in the system

should use the combination of a given audience and issuer.

Example Issuer Claim with Microsoft Entra ID

```
"iss": "https://login.microsoftonline.com/3e334762-b0c6-4c36-9faf-93800f0d6c71/v2.0"
```

JWKS URL

The JWKS URL is an endpoint that hosts the JWK Set (JWKS), a set of public keys used for verifying the JWT signature. This value is optional; if you do not provide a JWKS URL value when configuring the external token provider, Dremio retrieves the JWKS URL from `{issuer URL}/.well-known/openid-configuration`.

For Microsoft Entra ID, the JWKS URL is typically of the form `https://login.microsoftonline.com/{tenant_id}/discovery/v2.0/keys`.

Example: JWKS URL from Microsoft Entra ID

```
https://login.microsoftonline.com/58a43618-7933-4e0d-906e-1c1a2a867ad6/discovery/v2.0/keys
```

Manage External Token Providers

The Dremio administrator or a user with the CONFIGURE SECURITY privilege can view and manage external token providers in Dremio.

View External Token Providers

To view external token providers:

In the Dremio console, click !Settings on the left navigation bar, and select **Organization settings**.

Click **External Token Providers**. The External Token Providers page lists the external token providers configured for Dremio.

Add an External Token Provider

To add an external token provider:

In the Dremio console, click !Settings in the side navigation bar, and select **Organization settings**.

Click **External Token Providers**.

Click **Add Provider** at the top-right corner of the External Token Providers page.

In the Add Provider dialog, complete the configuration using the fields described in Define an External Token Provider.

Click **Add**.

When you add an external token provider, Dremio automatically enables it. To deactivate it, toggle the Enabled switch on the External Token Providers page.

Each external token provider must use a different combination of issuer and audience. If multiple external token providers share the same issuer and audience, authentication will fail regardless of whether the token providers are enabled.

Edit an External Token Provider

To edit an external token provider:

In the Dremio console, click !Settings in the side navigation bar, and select **Organization settings**.

Click **External Token Providers**.

On the External Token Providers page, find the row for the external token provider you want to edit and click !Edit at the right side of the row.

In the Edit Provider dialog, update the values using the fields described in Define an External Token Provider.

Click **Save**.

Delete an External Token Provider

To delete an external token provider:

In the Dremio console, click !Settings in the side navigation bar, and select **Organization settings**.

Click **External Token Providers**.

On the External Token Providers page, find the row for the external token provider you want to delete and click !Delete at the right side of the row.

In the Delete External Provider dialog, click **Delete**.

Use the External Token Provider

Retrieve an External JWT

This sample application uses the Microsoft Authentication Library to authenticate a user with the OAuth authorization code flow.

`client_id` is the Application (Client) ID assigned to your app by Microsoft Entra ID when the app was registered.

`app_redirect_url` or reply URL is the location of the client app where Microsoft Entra ID sends an external JWT after the user has successfully logged in, such as `https://myapp.com/auth/callback` or `http://localhost:3000/auth/callback`. The redirect URI is defined in the Microsoft Entra ID application registration for the client.

-

`dremio_scope_name` is the API scope you defined for the client in the Microsoft Entra ID application profile. Dremio requires a scope of `dremio.all` in token exchange, regardless of the scope configured in the application registration.

˜tenant_id` is your Microsoft Entra ID tenant identifier.

Example: Retrieving a Microsoft JWT

```
import msal

client_id = "TODO"
app_redirect_url = "TODO"
dremio_scope_name = "TODO"
tenant_id = "TODO"

authority_url = "https://login.microsoftonline.com/" + tenant_id
app = msal.PublicClientApplication(client_id, authority=authority_url)
auth_code_flow = app.initiate_auth_code_flow(
    scopes=[dremio_scope_name],
    redirect_uri=app_redirect_url
)  # PKCE is included in the MSAL Python library

state = auth_code_flow['state']

authorization_code = "TODO: retrieved from the browser"

external_access_token = ""

if authorization_code:
    auth_result = app.acquire_token_by_auth_code_flow(
        auth_code_flow=auth_code_flow,
        auth_response={"code": authorization_code, "state": state}
    )
    if "access_token" in auth_result:
        external_access_token = auth_result["access_token"]
    else:
        print("Error: no access token")
    if "refresh_token" in auth_result:
        refresh_token = auth_result["refresh_token"]
    else:
        print("Error: no refresh token")
else:
    print("Error: no auth code")
```

Exchange a JWT

The client must use the Dremio `/oauth/token` REST API to exchange the JWT for an OAuth access token.

Was this page helpful?

Prerequisites

-

- Define an External Token Provider
- Audience
- User Claim Mapping
- Issuer URL
- JWKS URL
- Manage External Token Providers
- View External Token Providers
- Add an External Token Provider
- Edit an External Token Provider
- Delete an External Token Provider
- Use the External Token Provider
- Retrieve an External JWT
- Exchange a JWT

Source: sql-docs.md

Dremio Docs in One PDF

This repo is for having a markdown file of all Dremio docs that can be easily used with AI tools to better generate SQL queries.

Reserved Words

Dremio reserves ANSI keywords and additional keywords to perform SQL queries on datalake sources and relational databases. These reserved keywords are part of the grammar of the SQL language that is used by Dremio to parse and understand SQL statements.

However, you can use these reserved keywords as an object name by enclosing a keyword in double quotes (for example, "boolean").

ABS, ACCESS, ACOS, AES_DECRYPT, AGGREGATE, ALL, ALLOCATE, ALLOW, ALTER, ANALYZE, AND, ANY, APPROX_COUNT_DISTINCT, APPROX_PERCENTILE, ARE, ARRAY_AVG, ARRAY_CAT, ARRAY_COMPACT, ARRAY_CONTAINS, ARRAY_GENERATE_RANGE, ARRAY_MAX_CARDINALITY, ARRAY_MAX, ARRAY_MIN, ARRAY_POSITION, ARRAY_REMOVE_AT, ARRAY_REMOVE, ARRAY_SIZE, ARRAY_SUM, ARRAY_TO_STRING, ARRAY, ARROW, AS, ASCII, ASENSITIVE, ASIN, ASSIGN, ASYMMETRIC, AT, ATAN, ATAN2, ATOMIC, AUTHORIZATION, AUTO, AVG, AVOID, BASE64, BATCH, BEGIN_FRAME, BEGIN_PARTITION, BEGIN, BETWEEN, BIGINT, BIN_PACK, BIN, BINARY_STRING, BINARY, BIT_AND, BIT_LENGTH, BIT_OR, BIT, BITWISE_AND, BITWISE_NOT, BITWISE_OR, BITWISE_XOR, BLOB, BOOL_AND, BOOL_OR, BOOLEAN, BOTH, BRANCH, BROUND, BTRIM, BY, CACHE, CALL, CALLED, CARDINALITY, CASCADED,

CASE, CAST, CATALOG, CBRT, CEIL, CEILING, CHANGE, CHAR_LENGTH, CHAR, CHARACTER_LENGTH, CHARACTER, CHECK, CHR, CLASSIFIER, CLOB, CLOSE, CLOUD, COALESCE, COL_LIKE, COLLATE, COLLECT, COLUMN, COLUMNS, COMMIT, COMMITS_OLDER_THAN, COMPUTE, CONCAT_WS, CONCAT, CONDITION, CONNECT, CONSTRAINT, CONTAINS, CONVERT_FROM, CONVERT_REPLACEUTF8, CONVERT_TIMEZONE, CONVERT_TO, CONVERT, COPY, CORR, CORRESPONDING, COS, COSH, COT, COUNT, COVAR_POP, COVAR_SAMP, CRC32, CREATE, CROSS, CUBE, CUME_DIST, CURRENT_CATALOG, CURRENT_DATE_UTC, CURRENT_DATE, CURRENT_DEFAULT_TRANSFORM_GROUP, CURRENT_PATH, CURRENT_ROLE, CURRENT_ROW, CURRENT_SCHEMA, CURRENT_TIME, CURRENT_TIMESTAMP, CURRENT_TRANSFORM_GROUP_FOR_TYPE, CURRENT_USER, CURRENT, CURSOR, CYCLE, DATA, DATABASES, DATASETS, DATE_ADD, DATE_DIFF, DATE_FORMAT, DATE_PART, DATE_SUB, DATE_TRUNC, DATE, DATEDIFF, DATETYPE, DAY, DAYOFMONTH, DAYOFWEEK, DAYOFYEAR, DEALLOCATE, DEC, DECIMAL, DECLARE, DEDUPE_LOOKBACK_PERIOD, DEFAULT, DEFINE, DEGREES, DELETE, DENSE_RANK, DEREF, DESCRIBE, DETERMINISTIC, DIMENSIONS, DISALLOW, DISCONNECT, DISPLAY, DISTINCT, DOUBLE, DROP, DYNAMIC, E, EACH, ELEMENT, ELSE, EMPTY_AS_NULL, EMPTY, ENCODE, END_FRAME, END_PARTITION, END-EXEC, END, ENDS_WITH, ENGINE, EQUALS, ESCAPE_CHAR, ESCAPE, EVERY, EXCEPT, EXEC, EXECUTE, EXISTS, EXP, EXP, EXPIRE, EXPLAIN, EXTEND, EXTERNAL, EXTRACT, FACTORIAL, FALSE, FETCH, FIELD_DELIMITER, FIELD, FILE_FORMAT, FILES, FILTER, FIRST_VALUE, FLATTEN, FLOAT, FLOOR, FOLDER, FOR, FOREIGN, FRAME_ROW, FREE, FROM_HEX, FROM, FULL, FUNCTION, FUSION, GEO_BEYOND, GEO_DISTANCE, GEO_NEARBY, GET, GLOBAL, GRANT, GRANTS, GREATEST, GROUP, GROUPING, GROUPS, HASH, HASH64, HAVING, HEX, HISTORY, HOLD, HOUR, IDENTITY, IF, ILIKE, IMINDIR, IMPORT, IN, INCLUDE, INDICATOR, INITCAP, INITIAL, INNER, INOUT, INSENSITIVE, INSERT, INSTR, INT, INTEGER, INTERSECT, INTERSECTION, INTERVAL, INTO, IS [NOT] DISTINCT FROM, IS [NOT] FALSE, IS [NOT] NULL, IS [NOT] TRUE, IS_BIGINT, IS_INT, IS_MEMBER, IS_SUBSTR, IS_UTF8, IS_VARCHAR, IS, ISDATE, ISNUMERIC, JOB, JOIN, JSON_ARRAY, JSON_ARRAYAGG, JSON_EXISTS, JSON_OBJECT, JSON_OBJECTAGG, JSON_QUERY, JSON_VALUE, LAG, LANGUAGE, LARGE, LAST_DAY, LAST_QUERY_ID, LAST_VALUE, LATERAL, LAZY, LCASE, LEAD, LEADING, LEAST, LEFT, LENGTH, LEVENSHTEIN, LIKE_REGEX, LIKE, LIMIT, LISTAGG, LN, LOCAL, LOCALSORT, LOCALTIME, LOCALTIMESTAMP, LOCATE, LOG, LOG10, LOGS, LOWER, LPAD, LSHIFT, LTRIM, MANIFESTS, MAP_KEYS, MAP_VALUES, MASK_FIRST_N, MASK_HASH, MASK_LAST_N, MASK_SHOW_FIRST_N, MASK_SHOW_LAST_N, MASK, MASKING, MATCH_NUMBER, MATCH_RECOGNIZE, MATCH, MATCHES, MAX_FILE_SIZE_MB, MAX, MAXDIR, MD5, MEASURES, MEDIAN, MEMBER, MERGE, METADATA, METHOD, MIN_FILE_SIZE_MB, MIN_INPUT_FILES, MIN, MINDIR, MINUS, MINUTE, MISSING, MOD, MODIFIES, MODULE, MONITOR, MONTH, MONTH, MONTHS_BETWEEN, MORE, MULTISET, NATIONAL, NATURAL, NCHAR, NCLOB, NDV, NEW, NEXT_DAY, NEXT, NO, NONE, NORMALIZE_STRING, NORMALIZE, NOT, NOTIFICATION_PROVIDER, NOTIFICATION_QUEUE_REFERENCE, NOW, NTH_VALUE, NTILE, NULL_IF, NULL, NULLIF, NUMERIC, NVL, OCCURRENCES_REGEX, OCTET_LENGTH, OF, OFFSET, OLD, OLDER_THAN, OMIT, ON, ONE, ONLY, OPEN, OPERATE, OPTIMIZE, OR, ORDER, ORPHAN, OUT, OUTER, OVER, OVERLAPS, OVERLAY, OWNERSHIP, PARAMETER, PARSE_URL, PARTITION, PARTITIONS, PATTERN, PER, PERCENT_RANK, PERCENT, PERCENTILE_CONT, PERCENTILE_DISC, PERIOD, PERMUTE, PI, PIPE, PIPES, PIVOT, PMOD, POLICY, PORTION, POSITION_REGEX, POSITION, POW, POWER, PRECEDES, PRECISION, PREPARE, PREV, PRIMARY, PROCEDURE, PROJECT, PROMOTION, QUALIFY, QUARTER, QUERY_USER, QUERY, QUOTE_CHAR, QUOTE, RADIANS, RANDOM, RANGE, RANK, RAW, READS, REAL, RECORD_DELIMITER, RECURSIVE, REF, REFERENCE, REFERENCES, REFERENCING, REFLECTION, REFLECTIONS, REFRESH, REGEX, REGEXP_COL_LIKE, REGEXP_EXTRACT, REGEXP_LIKE, REGEXP_MATCHES, REGEXP_REPLACE, REGEXP_SPLIT, REGR_AVGX, REGR_AVGY, REGR_COUNT, REGR_INTERCEPT, REGR_R2, REGR_SLOPE,

REGR_SXY, REGR_SYY, RELEASE, REMOVE, RENAME, REPEAT, REPEATSTR, REPLACE, RESET, RESULT, RETAIN_LAST, RETAIN_LAST_COMMITS, RETAIN_LAST_SNAPSHOTS, RETURN, RETURNS, REVERSE, REVOKE, REWRITE, RIGHT, ROLE, ROLLBACK, ROLLUP, ROUND, ROUTE, ROW_NUMBER, ROW, ROWS, RPAD, RSHIFT, RTRIM, RUNNING, SAVEPOINT, SCHEMAS, SCOPE, SCROLL, SEARCH, SECOND, SEEK, SELECT, SENSITIVE, SESSION_USER, SET, SHA, SHA1, SHA256, SHA512, SHOW, SIGN, SIMILAR_TO, SIMILAR, SIN, SINH, SIZE, SKIP, SMALLINT, SNAPSHOT, SNAPSHOTS_OLDER_THAN, SNAPSHOTS, SOME, SOUNDEX, SPECIFIC, SPECIFICTYPE, SPLIT_PART, SQL, SQLEXCEPTION, SQLSTATE, SQLWARNING, SQRT, SQRT, ST_FROMGEOHASH, ST_GEOHASH, START, STARTS_WITH, STATIC, STATISTICS, STDDEV_POP, STDDEV_SAMP, STDDEV, STREAM, STRING_BINARY, STRPOS, SUBMULTISET, SUBSET, SUBSTR, SUBSTRING_INDEX, SUBSTRING_REGEX, SUBSTRING, SUCCEEDS, SUM, SYMMETRIC, SYSTEM_TIME, SYSTEM_USER, SYSTEM, TABLE, TABLES, TABLESAMPLE, TAG, TAN, TANH, TARGET_FILE_SIZE_MB, TBLPROPERTIES, THEN, TIME_FORMAT, TIME, TIMESTAMP_FORMAT, TIMESTAMP, TIMESTAMPADD, TIMESTAMPDIFF, TIMESTAMPTYPE, TIMEZONE_HOUR, TIMEZONE_MINUTE, TINYINT, TO_CHAR, TO_DATE, TO_HEX, TO_NUMBER, TO_TIME, TO_TIMESTAMP, TO, TOASCII, TRAILING, TRANSACTION_TIMESTAMP, TRANSLATE_REGEX, TRANSLATE, TRANSLATION, TREAT, TRIGGER, TRIM_ARRAY, TRIM_SPACE, TRIM, TRUE, TRUNCATE, TYPEOF, UCASE, UESCAPE, UNBASE64, UNHEX, UNION, UNIQUE, UNIX_TIMESTAMP, UNKNOWN, UNNEST, UNPIVOT, UNSET, UPDATE, UPPER, UPSERT, USAGE, USE, USER, USING, VACUUM, VALUE_OF, VALUE, VALUES, VAR_POP, VAR_SAMP, VARBINARY, VARCHAR, VARYING, VERSIONING, VIEW, VIEWS, WEEK, WEEKOFYEAR, WHEN, WHENEVER, WHERE, WIDTH_BUCKET, WINDOW, WITH, WITHIN, WITHOUT, WRITE, XOR, YEAR

Data Types

`Data Types`
A data type classifies data and determines the operations allowed on it. Dremio supports numeric, string and binary, boolean, date and time, and semi-structured types.

The following topics are covered:
`Coercions Support`: how types are coerced when source and table differ.
`Summary of Supported Data Types in Dremio`.

`Numeric Data Types`

DECIMAL

INT

BIGINT

FLOAT

DOUBLE

`String & Binary Data Types`

VARCHAR

VARBINARY

`Boolean Data Type`

BOOLEAN

`Date & Time Data Types`

`note:` Dremio retrieves TIME and TIMESTAMP values as UTC without conversion.

DATE

TIME

TIMESTAMP

INTERVAL

`Semi-structured Data Types`

STRUCT

LIST

MAP

SQL Commands

`SELECT`

Dremio supports querying using standard `SELECT` statements. You can query tables and views in connected sources and catalogs.

When working with Apache Iceberg tables, you can:

- Query table metadata
- Run queries by snapshot ID

note
Dremio supports reading positional and equality deletes for Iceberg v2 tables. Dremio writes using copy-on-write by default and supports merge-on-read when enabled in

table properties.

Syntax

```
[ WITH ... ]
SELECT [ ALL | DISTINCT ]
{ *
| <column_name1>, <column_name2>, ... }
FROM { <table_name> | <view_name>
| TABLE ( <iceberg_metadata> ( <table_name> ) )
| UNNEST ( <list_expression> ) [ WITH ORDINALITY ] }
[ { PIVOT | UNPIVOT } ( <expression> ) ]
[ WHERE <condition> ]
[ GROUP BY <expression> ]
[ QUALIFY <expression> ]
[ ORDER BY <column_name1>, <column_name2>, ... [ DESC ] ]
[ LIMIT <count> ]
[ AT { { REF[ERENCE] | BRANCH | TAG | COMMIT } <reference_name>
[ AS OF <timestamp> ]
| { SNAPSHOT <snapshot_id> | <timestamp> } } ]
| <function_name>
[ AT { REF[ERENCE] | BRANCH | TAG | COMMIT } <reference_name> ]
[ AS OF <timestamp> ]
| TABLE(<source_name>.EXTERNAL_QUERY ('<select_statement>'))
```

Parameters

`WITH`
Defines a common table expression (CTE).

`ALL` / `DISTINCT`
`ALL` returns all rows. `DISTINCT` removes duplicates.

`*`
Selects all columns.

`<column_name>`
One or more columns to query.

`FROM <table_name> | <view_name>`
Source of the data.

`FROM TABLE(<iceberg_metadata>(<table_name>))`
Queries Iceberg system metadata. Metadata includes:

Data files

History

Manifest files

Partition statistics

-

Snapshots

Supported Iceberg Metadata Tables

`table_files(<table_name>)`

Returns metadata for each data file including:

- file_path
- file_format
- partition
- record_count
- file_size_in_bytes
- column_sizes
- value_counts
- null_value_counts
- nan_value_counts
- lower_bounds
- upper_bounds
- key_metadata
- split_offsets

`table_history(<table_name>)`

Returns:

- made_current_at
- snapshot_id
- parent_id
- is_current_ancestor

`table_manifests(<table_name>)`

Returns:

- path
- length
- partition_spec_id

- added_snapshot_id
- added_data_files_count
- existing_data_files_count
- deleted_data_files_count
- partition_summaries

`table_partitions('<table_name>')`

Includes:

- partition
- record_count
- file_count
- spec_id

`table_snapshot(<table_name>)`

Includes:

- committed_at
- snapshot_id
- parent_id
- operation
- manifest_list
- summary

`clustering_information('<table_name>')`

Includes:

- table_name
- clustering_keys
- clustering_depth
- last_clustering_timestamp

`UNNEST(<list_expression>) [WITH ORDINALITY]`

Expands a LIST into rows.

-

`WITH ORDINALITY` adds index values.

`PIVOT` / `UNPIVOT`

˜`PIVOT`: rows → columns

˜`UNPIVOT`: columns → rows

note: Aliases between table/subquery and pivot clauses are not supported.

`WHERE <condition>`

Filters records using comparison or logical operators (`=`, `>=`, `<`, `AND`, `OR`, `IN`, etc.).

`GROUP BY <expression>`

Groups rows to compute aggregations like `COUNT()`, `SUM()`, `AVG()`.

`QUALIFY <expression>`

Filters results *after* window functions are evaluated.

`ORDER BY <column_name> [DESC]`

Sorts results.

`LIMIT <count>`

Restricts number of returned rows.

`AT REF | BRANCH | TAG | COMMIT <reference_name>`

Time-travel and versioned queries.

˜`REF` for any reference

˜`BRANCH`, `TAG`, `COMMIT`

˜Commit hashes must be in double quotes

`AS OF <timestamp>`

Reads the reference as of a timestamp.

`AT SNAPSHOT <snapshot_id>`

Reads a specific Iceberg or Delta snapshot.

`<function_name>`

Execute a UDF.

`TABLE(<source>.EXTERNAL_QUERY('...'))`

Runs a native query directly on external systems.

Limitations:

- Only SELECT statements allowed
- No batched/multi-statement returns
- Views created from EXTERNAL_QUERY cannot move before first refresh

Examples

Query an existing table

```
SELECT *
FROM Samples."samples.dremio.com"."zips.json";
```

Query a specific column

```
SELECT city
FROM Samples."samples.dremio.com"."zips.json";
```

DISTINCT example

```
SELECT DISTINCT city
FROM Samples."samples.dremio.com"."zips.json";
```

WHERE clause

```
SELECT *
FROM Samples."samples.dremio.com"."zips.json"
WHERE state = 'MA' AND city = 'AGAWAM';
```

QUALIFY example (in SELECT)

```
SELECT passenger_count, trip_distance_mi, fare_amount,
RANK() OVER (PARTITION BY passenger_count ORDER BY trip_distance_mi) AS pc_rank
FROM "NYC-taxi-trips"
```

```
QUALIFY pc_rank = 1;
```

QUALIFY example (in QUALIFY)

```
SELECT passenger_count, trip_distance_mi, fare_amount
FROM "NYC-taxi-trips"
QUALIFY RANK() OVER (PARTITION BY passenger_count ORDER BY trip_distance_mi) = 1;
```

GROUP BY and ORDER BY

```
SELECT COUNT(city), city, state
FROM Samples."samples.dremio.com"."zips.json"
GROUP BY state, CITY
ORDER BY COUNT(city) DESC;
```

CTE example

```
WITH cte_quantity (Total) AS (
SELECT SUM(passenger_count) AS Total
FROM Samples."samples.dremio.com"."NYC-taxi-trips"
WHERE passenger_count > 2
GROUP BY pickup_datetime
)
SELECT AVG(Total) AS average_pass
FROM cte_quantity;
```

PIVOT / UNPIVOT example

```
ALTER DATASET Samples."samples.dremio.com"."SF weather 2018-2019.csv"
REFRESH METADATA auto promotion FORCE UPDATE;

SELECT * FROM (
SELECT EXTRACT(YEAR FROM CAST(F AS DATE)) AS "YEAR",
EXTRACT(MONTH FROM CAST(F AS DATE)) AS "MONTH",
K AS MAX_TEMP
FROM Samples."samples.dremio.com"."SF weather 2018-2019.csv"
WHERE F <> 'DATE'
)
PIVOT (
max(MAX_TEMP) FOR "MONTH" IN (1 AS JAN, 2 AS FEB, 3 AS MAR, 4 AS APR, 5 AS MAY, 6 AS
JUN,
7 AS JUL, 8 AS AUG, 9 AS SEP, 10 AS OCT, 11 AS NOV, 12 AS "DEC")
)
UNPIVOT (
GLOBAL_MAX_TEMP FOR "MONTH" IN (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV,
"DEC")
```

```
)
ORDER BY "YEAR","MONTH";
```

UNNEST example

```
SELECT index, UPPER(array_item)
FROM UNNEST (ARRAY['a','b','c']) WITH ORDINALITY AS my_table (array_item, index)
ORDER BY index;
```

Query using a branch reference

```
SELECT *
FROM myCatalog.demo_table AT REF main_branch;
```

Query using a commit

```
SELECT *
FROM myCatalog.demo_view AT COMMIT "7f643f2b9cf250ce1f5d6ff4397237b705d866fbf34d714";
```

Time-travel by timestamp

```
SELECT *
FROM myTable AT TIMESTAMP '2022-01-01 17:30:50.000';
```

Time-travel by snapshot

```
SELECT *
FROM myTable AT SNAPSHOT '5393090506354317772';
```

Query Iceberg history

```
SELECT *
FROM TABLE(table_history('myTable'))
WHERE snapshot_id = 4593468819579153853;
```

Count snapshots

```
SELECT COUNT(*)
FROM TABLE(table_snapshot('myTable'))
```

Time-travel by timestamp

```
SELECT *
FROM myTable AT TIMESTAMP '2022-01-01 17:30:50.000';
```

Time-travel by snapshot

```
SELECT *
FROM myTable AT SNAPSHOT '5393090506354317772';
```

Query Iceberg history

```
SELECT *
FROM TABLE(table_history('myTable'))
WHERE snapshot_id = 4593468819579153853;
```

Count snapshots

```
SELECT COUNT(*)
FROM TABLE(table_snapshot('myTable'))
GROUP BY snapshot_id;
```

External query

```
SELECT *
FROM TABLE(<source_name>.EXTERNAL_QUERY('SELECT * FROM Actor'));
```

External query with string literal

```
SELECT *
FROM TABLE(<source_name>.EXTERNAL_QUERY('SELECT string_col FROM tbl WHERE string_col =
''test'' '));
```

JOIN external query

```
SELECT B.customer_id, A.product_id, A.price
FROM TABLE(<source_name>.EXTERNAL_QUERY('SELECT product_id, price FROM products')) AS A,
source_b.sales AS B
WHERE B.product_id = A.product_id;
```

Time-travel by snapshot

```
SELECT *
FROM myTable AT SNAPSHOT '5393090506354317772';
```

Query Iceberg history

```
SELECT *
FROM TABLE(table_history('myTable'))
WHERE snapshot_id = 4593468819579153853;
```

Count snapshots

```
SELECT COUNT(*)
FROM TABLE(table_snapshot('myTable'))
GROUP BY snapshot_id;
```

Escaping quotes

```
SELECT *
FROM TABLE(<source_name>.EXTERNAL_QUERY('SELECT string_col FROM tbl WHERE string_col =
''john '''' s car '''));
```

External query

```
SELECT *
FROM TABLE(<source_name>.EXTERNAL_QUERY('SELECT * FROM Actor'));
```

External query with string literal

```
SELECT *
FROM TABLE(<source_name>.EXTERNAL_QUERY('SELECT string_col FROM tbl WHERE string_col =
''test'' '));
```

Escaping quotes

```
SELECT *
FROM TABLE(<source_name>.EXTERNAL_QUERY('SELECT string_col FROM tbl WHERE string_col =
''john '''' s car '''));
```

JOIN external query

```
SELECT B.customer_id, A.product_id, A.price
FROM TABLE(<source_name>.EXTERNAL_QUERY('SELECT product_id, price FROM products')) AS A,
source_b.sales AS B
WHERE B.product_id = A.product_id;
```

Column Aliasing

If you specify an alias, you can reference it elsewhere in the query.

Example 1

```
SELECT c_custkey AS c, lower(c)
FROM "customer.parquet";
```

Example 2

```
SELECT c_custkey AS c, lower(c)
FROM (
SELECT c_custkey, c_mktsegment AS c
FROM "customer.parquet"
);
```

Example 3

```
SELECT c_name AS n, n
FROM (
SELECT c_mktsegment AS n, c_name
FROM "customer.parquet"
) AS MY_TABLE
WHERE n = 'BUILDING';
```

Example 4

```
SELECT c_custkey
FROM (
SELECT c_custkey, c_name AS c
FROM "customer.parquet"
)
WHERE c = 'aa';
```

Example 5

```
SELECT *
FROM (
SELECT c_custkey AS c, c_name
FROM "customer.parquet"
)
JOIN "orders.parquet" ON c = o_orderkey;
```

Example 6

```
SELECT c_custkey AS c
FROM "customer.parquet"
JOIN "orders.parquet" ON c = o_orderkey;
```

Distributing Data Evenly Across Engines (BROADCAST Hint)

Use a BROADCAST hint if a join is skewed.

note: Not supported on views; ignored for nested-loop joins.

Syntax (inline): `/*+ BROADCAST */`

Example 1

```
SELECT *
FROM T1 /*+ BROADCAST */
INNER JOIN t2 ON t1.key = t2.key
INNER JOIN t3 ON t2.key = t3.key;
```

Example 2

```
SELECT *
FROM T1
INNER JOIN (
SELECT key, max(cost) cost
FROM t2 /*+ BROADCAST */
) T2 ON t1.key = t2.key
INNER JOIN t3 ON t2.key = t3.key;
```

`copy_errors()` — Inspect Rejected COPY INTO Records

Syntax

```
SELECT *
FROM TABLE(copy_errors('<table_name>', ['<query_id>']));
```

Fields Returned

˜job_id`: ID of the COPY job

˜file_name`: file path

˜line_number`: physical line number

˜row_number`: record position

˜column_name`: column with error

˜error`: error description

`ALTER PIPE` Preview

Changes an existing autoingest pipe.

Syntax

```
ALTER PIPE <pipe_name>
{ SET PIPE_EXECUTION_RUNNING = { TRUE | FALSE }
| [ DEDUPE_LOOKBACK_PERIOD <number_of_days> ]
AS COPY INTO <table_name>
FROM '@<storage_location_name>[ /<folder_name> ]'
[ FILE_FORMAT 'csv' | 'json' | 'parquet']
[ ( [csv_format_options] | [json_format_options] | [parquet_format_options]) ]
}
```

CSV Format Options

```
[ DATE_FORMAT '<string>' ]
[ EMPTY_AS_NULL [ '<boolean>' ] [, ...] ]
[ ESCAPE_CHAR '<escape_character>' ]
[ EXTRACT_HEADER '<boolean>' ]
[ FIELD_DELIMITER '<character>' ]
[ NULL_IF ('<string>' [, ...]) ]
[ ON_ERROR 'skip_file' ]
[ QUOTE_CHAR '<character>' ]
[ RECORD_DELIMITER '<character>' ]
[ SKIP_LINES <n> ]
[ TIME_FORMAT '<string>' ]
[ TIMESTAMP_FORMAT '<string>' ]
[ TRIM_SPACE [ '<boolean>' ] ]
```

JSON Format Options

```
[ DATE_FORMAT '<string>' ]
[ EMPTY_AS_NULL [ '<boolean>' ] [, ...] ]
```

```
[ NULL_IF ('<string>' [, ...]) [, ...] ]
[ ON_ERROR 'skip_file' ]
[ TIME_FORMAT '<string>' ]
[ TIMESTAMP_FORMAT '<string>' ]
[ TRIM_SPACE [ '<boolean>' ] ]
```

Parquet Format Options

```
[ ON_ERROR 'skip_file' ]
```

Parameters

`<pipe_name>`
Unique name of the autoingest pipe. Cannot be modified.

`SET PIPE_EXECUTION_RUNNING = { TRUE | FALSE }`
Controls whether the pipe triggers a `COPY INTO` on notifications.

`TRUE`: pipe active

`FALSE`: pipe paused (default is TRUE)

`DED UPE_LOOKBACK_PERIOD <days>`

Days to look back for deduplication (0–90).

Default: 14 days.

`AS COPY INTO <table_name>`
Target Iceberg table. Use full qualifier if not in current context.

`@<storage_location_name>`
Source location for files. Must exist as a configured Dremio source.

note: Autoingest pipes ingest only from Amazon S3.

`/<folder_name>`
Optional subfolder under the storage location.

`FILE_FORMAT 'csv' | 'json' | 'parquet'`
Required. All files must match format.

CSV/JSON may be compressed (`.gz`, `.bz2`)

Parquet supports only `ON_ERROR 'skip_file'`

CSV Format Option Details

`DATE_FORMAT '<string>'`
Defaults to `YYYY-MM-DD`.

`EMPTY_AS_NULL '<boolean>'`

Default: `TRUE`.

`ESCAPE_CHAR '<escape_character>'`
Default: `"`.

`EXTRACT_HEADER '<boolean>'`
Default: `TRUE`.

`FIELD_DELIMITER '<character>'`
Default: `,`.

`NULL_IF ('<string>' ...)`
Strings to convert to NULL.

`ON_ERROR 'skip_file'`
Stops at first error and logs error to `sys.project.copy_errors_history`.

`QUOTE_CHAR '<character>'`
Default: `"`.

`RECORD_DELIMITER '<character>'`
Default: `\r\n`.

`SKIP_LINES <n>`
Skips initial lines.

`TIME_FORMAT '<string>'`
Default: `HH24:MI:SS.FFF`.

`TIMESTAMP_FORMAT '<string>'`
Default: `YYYY-MM-DD HH24:MI:SS.FFF`.

`TRIM_SPACE '<boolean>'`
Default: `FALSE`.

JSON Format Option Details

`DATE_FORMAT '<string>'` – default: `YYYY-MM-DD`

`EMPTY_AS_NULL '<boolean>'` – default: `TRUE`

`NULL_IF ('<string>'...)` – replace values with NULL

`ON_ERROR 'skip_file'` – default, only supported mode

`TIME_FORMAT '<string>'` – default: `HH24:MI:SS.FFF`

`TIMESTAMP_FORMAT '<string>'` – default: `YYYY-MM-DD HH24:MI:SS.FFF`

`TRIM_SPACE '<boolean>'` – default: `FALSE`

Parquet Format Option Details

`ON_ERROR 'skip_file'` only

Logs first error to history

- Requires extra file processing
- Skips entire file on any error

Examples

Pause an autoingest pipe

```
ALTER PIPE test_pipe
SET PIPE_EXECUTION_RUNNING = FALSE
```

Change the pipe storage location

```
ALTER PIPE test_pipe
AS COPY INTO Table_one
FROM '@s3_source/folder'
FILE_FORMAT 'json'
```

`ALTER SOURCE`

Change the configuration or status of an existing source.

Syntax

```
ALTER SOURCE <source_name>
{ CLEAR PERMISSION CACHE | REFRESH STATUS }
```

To run `ALTER SOURCE`, you need the **MODIFY** privilege on the source.

Parameters

`<source_name>`
Name of the source to alter.

`CLEAR PERMISSION CACHE`

- Clears the AWS Lake Formation permission cache.
- Applies only to AWS Glue Data Catalog sources.
- Dremio caches Lake Formation permissions for one hour.
- Use this command after changing permissions in AWS Lake Formation to invalidate the cache immediately.

note:
Any change to AWS Glue Data Catalog settings also clears the permission cache.

`REFRESH STATUS`
Refreshes the status of the source.

Examples

Clear the Lake Formation permission cache

```
ALTER SOURCE glue1
CLEAR PERMISSION CACHE
```

Refresh status for an Amazon S3 source

```
ALTER SOURCE S3
REFRESH STATUS
```

`ALTER TABLE`

Update a table's definition or schema.

Syntax

```
ALTER TABLE <table_name>
{ ADD PRIMARY KEY ( <column_name> [ , ... ] )
| DROP PRIMARY KEY
| ADD COLUMNS ( <column_name> <data_type> [ NULL ] [ , ... ] )
| DROP COLUMN <column_name>
| { ALTER | MODIFY | CHANGE } COLUMN <old_name> <new_name> <data_type> [ { NULL | NOT
NULL | DROP NOT NULL } ]
| MODIFY COLUMN <column_name> { SET MASKING POLICY <function_name> ( <column_name> [,
... ] ) | UNSET MASKING POLICY <function_name> }
| { ADD | DROP } ROW ACCESS POLICY <function_name> ( <column_name> [, ... ] )
| CLUSTER BY ( <column_name> [ , ... ] )
| DROP CLUSTERING KEY
| LOCALSORT BY ( <column_name> [ , ... ] )
| REFRESH METADATA [ IN <catalog_name> ] [ FOR PARTITIONS ( <partition_name> =
'<value>') ] [ { AVOID | AUTO } PROMOTION ] [ { FORCE | LAZY } UPDATE ] [ { MAINTAIN |
DELETE } WHEN MISSING ]
| FORGET METADATA
| SET TBLPROPERTIES ( '<property_name>' = '<property_value>' [ , ... ] )
| UNSET TBLPROPERTIES ( '<property_name>' [ , ... ] )
| CREATE AGGREGATE REFLECTION <reflection_name> USING { DIMENSIONS ( <column_name> [ ,
... ] ) | MEASURES ( <column_name> ( <aggregation_type> ) [ , ... ] ) ) | DIMENSIONS (
<column_name> [ , ... ] ) MEASURES ( <column_name> ( <aggregation_type> ) [ , ... ] ) }
[ PARTITION BY ( { <column_name> | <partition_transform> } [ , ... ] ) ] [ LOCALSORT BY
( <column_name> [ , ... ] ) ]
| CREATE EXTERNAL REFLECTION <reflection_name> USING <table_name>
| CREATE RAW REFLECTION <reflection_name> USING DISPLAY ( <column_name> [ , ... ] ) [
```

```
PARTITION BY ( { <column_name> | <partition_transform> } [ , ... ] ) ] [ LOCALSORT BY (
<column_name> [ , ... ] ) ]
| DROP REFLECTION <reflection_name>
| REFRESH REFLECTIONS
| ROUTE REFLECTIONS TO { DEFAULT ENGINE | ENGINE { <engine_name> | <engine_uuid> } }
| { ADD | DROP } PARTITION FIELD { <column_name> | <partition_transform> }
}
```

Parameters

`<table_name>`
The name of the table that you want to alter.

`ADD PRIMARY KEY ( <column_name> [ , ... ] )`
Specifies to use one or more existing columns as the primary key of a table. Primary keys provide hints to the query planning during join planning. They can be added to Apache Iceberg tables only. Uniqueness of the values in a primary key is not enforced.

`DROP PRIMARY KEY`
Removes a table's primary key. The columns that make up the primary key remain in the table.

`ADD COLUMNS ( <column_name> <data_type> [ NULL ] [ , ... ] )`
Creates one or more columns that have the specified names, data types, character limits, and nullability properties.

Supported primitive types: `BOOLEAN`, `VARBINARY`, `DATE`, `FLOAT`, `DECIMAL`, `DOUBLE`, `INTERVAL`, `INT`, `BIGINT`, `TIME`, `TIMESTAMP`, `VARCHAR`.

Complex types syntax:

`ROW( name primitive_or_complex_type, .. )`

`ARRAY(primitive_or_complex_type)`

`STRUCT <name : primitive_or_complex_type, ... >`

`{ LIST | ARRAY } < primitive_or_complex_type >`

Use `NULL` to allow NULL values (default). You cannot add a `NOT NULL` column to an existing table.

`DROP COLUMN <column_name>`
Drops the specified column. This action cannot be undone.

`{ ALTER | MODIFY | CHANGE } COLUMN <old_column_name> <new_column_name> <data_type>`
Changes the data type for a column, and gives you the option to rename the column. Renaming is supported except for Parquet, JSON, or BSON.

Allowed primitive type changes:

`INT` to `BIGINT`

`FLOAT` to `DOUBLE`

˜DECIMAL(p, s)` to `DECIMAL(p', s)` (widening precision)

˜[{ NULL | NOT NULL | DROP NOT NULL }]`

˜NULL`: Allow NULL values.

˜NOT NULL`: Prevent NULL values.

˜DROP NOT NULL`: Change from preventing NULLs to allowing them.

˜MODIFY COLUMN <column_name> { SET | UNSET } MASKING POLICY`
Sets or unsets a masking policy on a column. (Enterprise only)

˜{ ADD | DROP } ROW ACCESS POLICY`
Adds or removes a row-access policy. (Enterprise only)

˜CLUSTER BY (<column_name> [, ...])`
Columns to cluster the table by. Future `OPTIMIZE TABLE` commands will follow this scheme. (Enterprise only)

˜DROP CLUSTERING KEY`
Removes clustering keys.

˜LOCALSORT BY (<column_name> [, ...])`
Columns to sort new data by.

˜REFRESH METADATA`
Refreshes table metadata.

˜FOR PARTITIONS (<partition_name> = '<value>')`: Partial refresh.

˜{ AVOID | AUTO } PROMOTION`: Handle file promotion.

˜{ FORCE | LAZY } UPDATE`: Force full update or lazy update.

˜{ MAINTAIN | DELETE } WHEN MISSING`: Handle missing metadata.

˜FORGET METADATA`
Deletes metadata until next refresh.

˜SET TBLPROPERTIES`
Sets table properties (e.g., for Iceberg).

˜UNSET TBLPROPERTIES`
Removes table properties.

˜CREATE AGGREGATE REFLECTION`
Creates an aggregation reflection.

˜DIMENSIONS`: Columns for dimensions.

˜MEASURES`: Columns for measures and aggregation types (`COUNT`, `MIN`, `MAX`, `SUM`, `APPROXIMATE COUNT DISTINCT`).

˜PARTITION BY`: Horizontal partitioning.

˜LOCALSORT BY`: Sort order.

-

`CREATE EXTERNAL REFLECTION`
Creates an external reflection using a derived table.

`CREATE RAW REFLECTION`
Creates a raw reflection.

`USING DISPLAY`: Columns to include.

`DROP REFLECTION`
Drops a reflection.

`REFRESH REFLECTIONS`
Triggers reflection refresh.

`ROUTE REFLECTIONS TO`
Specifies engine for reflection jobs.

`{ ADD | DROP } PARTITION FIELD`
Adds or drops partition fields (Iceberg only).
Transforms: `identity`, `year`, `month`, `day`, `hour`, `bucket`, `truncate`.

Examples

Add a primary key

```
ALTER TABLE services ADD PRIMARY KEY (Country_ID);
```

Add columns

```
ALTER TABLE services ADD COLUMNS (county varchar);
```

Modify column type

```
ALTER TABLE services MODIFY COLUMN tip_amount tip_amount DECIMAL;
```

Modify struct column

```
ALTER TABLE struct_type MODIFY COLUMN a a struct<x: varchar, y: bigint>;
```

Rename and modify column

```
ALTER TABLE services MODIFY COLUMN tip_amount gratuity_amount DECIMAL;
```

Change column nullability

```
ALTER TABLE age_table CHANGE age age INT DROP NOT NULL;
```

Add multiple columns

```
ALTER TABLE my_table ADD COLUMNS ( email VARCHAR NULL, date_of_birth DATE );
```

Refresh metadata

```
ALTER TABLE services REFRESH METADATA;
```

Refresh metadata with options

```
ALTER TABLE services REFRESH METADATA AUTO PROMOTION LAZY UPDATE MAINTAIN WHEN MISSING;
```

Partial metadata refresh

```
ALTER TABLE Samples."samples.dremio.com"."zips.json" REFRESH METADATA FOR PARTITIONS (state = 'TX');
```

Forget metadata

```
ALTER TABLE Samples."samples.dremio.com"."zips.json" FORGET METADATA;
```

Create raw reflection

```
ALTER TABLE Sales."customers" CREATE RAW REFLECTION customers_by_country USING DISPLAY (id,lastName,firstName,address,country) PARTITION BY (country) LOCALSORT BY (lastName);
```

Create aggregate reflection

```
ALTER TABLE Samples."samples.dremio.com"."zips.json" CREATE AGGREGATE REFLECTION per_state USING DIMENSIONS (state) MEASURES (city (COUNT)) LOCALSORT BY (state);
```

Route reflections

```
ALTER TABLE "Table 1" ROUTE REFLECTIONS TO ENGINE "Engine 1";
```

Cluster by

```
ALTER TABLE clustered_table CLUSTER BY (Col_one, Col_two, Col_three);
```

Drop clustering key

```
ALTER TABLE clustered_table DROP CLUSTERING KEY;
```

`ALTER VIEW`

Change an existing view.

Syntax

```
ALTER VIEW <view_name>
{ REFRESH METADATA [ FOR PARTITIONS ( <partition_name> = '<value>') ] [ { AVOID | AUTO } PROMOTION ] [ { FORCE | LAZY } UPDATE ] [ { MAINTAIN | DELETE } WHEN MISSING ]
| CREATE EXTERNAL REFLECTION <reflection_name> USING <view_name>
| CREATE AGGREGATE REFLECTION <reflection_name> USING { DIMENSIONS ( <column_name1>, <column_name2>, ... ) | MEASURES ( <column_name1> ( <aggregation_type>, <column_name2> <aggregation_type> , ... ) ) | DIMENSIONS ( <column_name1>, <column_name2>, ... ) MEASURES ( <column_name1> ( <aggregation_type>, <column_name2> <aggregation_type> , ... ) ) } [ PARTITION BY ( <column_name1>, <column_name2>, ... ) ] [ LOCALSORT BY ( <column_name1>, <column_name2>, ... ) ]
| CREATE RAW REFLECTION <reflection_name> USING DISPLAY ( <column_name1>, <column_name2>, ...) [ PARTITION BY ( <column_name1>, <column_name2>, ... ) ] [ LOCALSORT BY ( <column_name1>, <column_name2>, ... ) ]
| DROP REFLECTION <reflection_name>
| REFRESH REFLECTIONS
| ROUTE REFLECTIONS TO { DEFAULT ENGINE | ENGINE { <engine_name> | <engine_uuid> } }
}
```

Parameters

`<view_name>`
The name of the view that you want to alter.

`REFRESH METADATA`
Refreshes metadata (Iceberg REST Catalog sources only).

`FOR PARTITIONS ( <partition_name> = '<value>' )`: Partial refresh.

`{ AVOID | AUTO } PROMOTION`: Handle file promotion.

`{ FORCE | LAZY } UPDATE`: Force full update or lazy update.

`{ MAINTAIN | DELETE } WHEN MISSING`: Handle missing metadata.

`CREATE EXTERNAL REFLECTION`
Creates an external reflection.

`CREATE AGGREGATE REFLECTION`
Creates an aggregate reflection.

`CREATE RAW REFLECTION`
Creates a raw reflection.

`DROP REFLECTION`
Drops a reflection.

`REFRESH REFLECTIONS`
Triggers reflection refresh.

`ROUTE REFLECTIONS TO`
Specifies engine for reflection jobs.

Examples

Create raw reflection on view

```
ALTER VIEW Sales."customers" CREATE RAW REFLECTION customers_by_country USING DISPLAY (id,lastName,firstName,address,country) PARTITION BY (country) LOCALSORT BY (lastName);
```

Create aggregate reflection on view

```
ALTER VIEW Samples."samples.dremio.com"."zips.json" CREATE AGGREGATE REFLECTION per_state USING DIMENSIONS (state) MEASURES (city (COUNT)) LOCALSORT BY (state);
```

Route reflections

```
ALTER VIEW "View 1" ROUTE REFLECTIONS TO ENGINE "Engine 1";
```

`ANALYZE TABLE`

Compute and delete statistics for tables, including estimated number of distinct values, number of rows, and number of null values.

Syntax

```
ANALYZE TABLE <table_name> FOR { ALL COLUMNS | COLUMNS ( <column_name1>, <column_name2>, ... ) } { COMPUTE | DELETE } STATISTICS
```

Parameters

`<table_name>`
The path to the table.

`FOR { ALL COLUMNS | COLUMNS (...) }`
Specify columns to analyze.

`{ COMPUTE | DELETE } STATISTICS`
Compute or delete statistics.

Examples

Compute statistics for all columns

```
ANALYZE TABLE Samples."samples.dremio.com"."NYC-taxi-trips" FOR ALL COLUMNS COMPUTE STATISTICS
```

Compute statistics for specific columns

```
ANALYZE TABLE Samples."samples.dremio.com"."NYC-taxi-trips" FOR COLUMNS (fare_amount, tip_amount) COMPUTE STATISTICS
```

`COPY INTO`

Load data from CSV, JSON, or Parquet files into an existing Apache Iceberg table.

Syntax

```
COPY INTO <table_name>
FROM '@<storage_location_name>[/<path>[/<file_name>] ]'
[ FILES ( '<file_name>' [ , ... ] ) | REGEX '<regex_pattern>' ]
[ FILE_FORMAT 'csv' | 'json' | 'parquet' ]
[ ( [csv_format_options] | [json_format_options] | [parquet_format_options] ) ]
```

Parameters

`<table_name>`
The target table.

`FROM '@<storage_location_name>...'`
Source location. Can be a directory or specific file.

`FILES ( '<file_name>' ... )`
List of specific files to load.

`REGEX '<regex_pattern>'`
Regex pattern to match files.

`FILE_FORMAT`
`csv`, `json`, or `parquet`.

Format Options

CSV Options:

`DATE_FORMAT`: Date format string.

`EMPTY_AS_NULL`: Treat empty strings as NULL (default TRUE).

`ESCAPE_CHAR`: Escape character (default `"`).

`EXTRACT_HEADER`: First line is header (default TRUE).

`FIELD_DELIMITER`: Field separator (default `,`).

`NULL_IF`: Strings to replace with NULL.

`ON_ERROR`: `abort` (default), `continue`, or `skip_file`.

`QUOTE_CHAR`: Quote character (default `"`).

`RECORD_DELIMITER`: Record separator (default `\r\n`).

`SKIP_LINES`: Number of lines to skip.

`TIME_FORMAT`: Time format string.

`TIMESTAMP_FORMAT`: Timestamp format string.

`TRIM_SPACE`: Trim whitespace (default FALSE).

JSON Options:

`DATE_FORMAT`

`EMPTY_AS_NULL`

`NULL_IF`

`ON_ERROR`

`TIME_FORMAT`

`TIMESTAMP_FORMAT`

`TRIM_SPACE`

Parquet Options:

`ON_ERROR`: `abort` or `skip_file`.

Examples

Copy from specific file

```
COPY INTO context.myTable FROM '@SOURCE/bucket/path/folder' FILES ('fileName.csv') (ON_ERROR 'continue')
```

Copy JSON files

```
COPY INTO context.MyTable FROM '@SOURCE/bucket/path/folder/' FILE_FORMAT 'json'
```

Copy using Regex

```
COPY INTO context.myTable FROM '@SOURCE/bucket/path/folder' REGEX '.*.csv'
```

Copy with CSV options

```
COPY INTO context.myTable FROM '@SOURCE/bucket/path/folder' FILE_FORMAT 'csv' (RECORD_DELIMITER '\n', FIELD_DELIMITER '\t')
```

`CREATE FOLDER`

Create a new folder in your catalog.

Syntax

```
CREATE FOLDER [ IF NOT EXISTS ] <folder_name>
```

Parameters

`IF NOT EXISTS`
Prevent error if folder exists.

`<folder_name>`
Name of the folder. Cannot include `/`, `:`, `[`, `]`.

Examples

Create folder

```
CREATE FOLDER myFolder
```

Create if not exists

```
CREATE FOLDER IF NOT EXISTS myFolder
```

`CREATE FUNCTION`

Creates user-defined functions (UDFs) in the catalog.

Syntax

```
CREATE [ OR REPLACE ] FUNCTION [ IF NOT EXISTS ] <function_name> ( [
<function_parameter> [, ...] ] )
RETURNS { <data_type> | TABLE ( <column_name> [, ...] ) }
RETURN { <expression> | <query> }
```

Parameters

`OR REPLACE`
Replaces existing UDF. Cannot use with `IF NOT EXISTS`.

`IF NOT EXISTS`
Creates only if it doesn't exist. Cannot use with `OR REPLACE`.

`<function_name>`
Name of the UDF.

`<function_parameter>`
`parameter_name` and `data_type`.

`RETURNS <data_type>`
Return type for scalar function.

`RETURNS TABLE ( <column_name> [, ...] )`
Return signature for tabular function.

`RETURN { <expression> | <query> }`
Body of the UDF. Expression for scalar, query for tabular.

Examples

Scalar function

```
CREATE FUNCTION area (x DOUBLE, y DOUBLE) RETURNS DOUBLE RETURN x * y;
```

Tabular function

```
CREATE FUNCTION all_fruits() RETURNS TABLE (name VARCHAR, hue VARCHAR) RETURN SELECT *
```

```
FROM <catalog-name>.t2;
```

`CREATE PIPE`

Create a pipe object that automatically ingests files from a cloud storage location.

Syntax

```
CREATE PIPE [ IF NOT EXISTS ] <pipe_name>
[ DEDUPE_LOOKBACK_PERIOD <number_of_days> ]
AS COPY INTO <table_name>
FROM '@<storage_location_name>[ /<folder_name> ]'
[ FILE_FORMAT 'csv' | 'json' | 'parquet']
[ ( [csv_format_options] | [json_format_options] | [parquet_format_options] ) ]
```

Parameters

`IF NOT EXISTS`
Prevent error if pipe exists.

`<pipe_name>`
Unique name of the pipe.

`DEDUPE_LOOKBACK_PERIOD <days>`
Days to check for duplicates (0-90, default 14).

`AS COPY INTO <table_name>`
Target Iceberg table.

`FROM '@<storage_location_name>...'`
Source location (Amazon S3 only).

`FILE_FORMAT`
`csv`, `json`, or `parquet`.

Examples

Create pipe

```
CREATE PIPE Example_pipe AS COPY INTO Pipe_sink FROM '@<storage_location_name>/folder' FILE_FORMAT 'csv'
```

Create pipe with dedupe lookback

```
CREATE PIPE Example_pipe DEDUPE_LOOKBACK_PERIOD 5 AS COPY INTO Table_one FROM '@<storage_location_name>/files' FILE_FORMAT 'csv'
```

`CREATE ROLE`

Create a new role.

Syntax

```
CREATE ROLE <role_name>
```

Parameters

`<role_name>`
Name of the new role.

Example

```
CREATE ROLE role1
```

`CREATE TABLE`

Create a new table.

Syntax

```
CREATE TABLE [ IF NOT EXISTS ] <table_name>
( <column_name> <data_type> [ { NULL | NOT NULL } ] [ , ... ] )
[ MASKING POLICY <function_name> ( <column_name> [ , ... ] ) ]
[ ROW ACCESS POLICY <function_name> ( <column_name> [ , ... ] ) ]
[ PARTITION BY ( { <column_name> | <partition_transform> } [ , ... ] ) ]
[ CLUSTER BY ( <column_name> [ , ... ] ) ]
[ LOCALSORT BY ( <column_name> [ , ... ] ) ]
[ TBLPROPERTIES ( '<property_name>' = '<property_value>' [ , ... ] ) ]
```

Parameters

`IF NOT EXISTS`
Prevent error if table exists.

`<table_name>`
Name of the table.

`( <column_name> <data_type> ... )`
Column definitions.

Primitive types: `BOOLEAN`, `VARBINARY`, `DATE`, `FLOAT`, `DECIMAL`, `DOUBLE`, `INTERVAL`, `INT`, `BIGINT`, `TIME`, `TIMESTAMP`, `VARCHAR`.

Complex types: `ROW`, `ARRAY`, `STRUCT`, `LIST`.

Nullability: `NULL` (default) or `NOT NULL`.

`MASKING POLICY`
Set masking policy (Enterprise).

`ROW ACCESS POLICY`
Set row access policy (Enterprise).

`PARTITION BY`
Partition columns or transforms (`identity`, `year`, `month`, `day`, `hour`, `bucket`, `truncate`).

`CLUSTER BY`
Cluster columns (Enterprise).

`LOCALSORT BY`
Sort columns within files.

`TBLPROPERTIES`
Table properties (Iceberg).

Examples

Create table with columns

```
CREATE TABLE employees (PersonID int, LastName varchar, FirstName varchar, Address varchar, City varchar)
```

Create table with partitions

```
CREATE TABLE myTable (col1 int, col2 date) PARTITION BY (month(col2))
```

Create table with complex types

```
CREATE TABLE my_table (name VARCHAR NOT NULL, age INT NULL, address STRUCT<street VARCHAR, zip INT NOT NULL, city VARCHAR NOT NULL>);
```

`CREATE TABLE AS`

Create a new table as a select statement from another table.

Syntax

```
CREATE TABLE [ IF NOT EXISTS ] <table_name>
[ ( <column_name> <data_type> [ { NULL | NOT NULL } ] [ , ... ] ) ]
```

```
[ PARTITION BY ( { <column_name> | <partition_transform> } [ , ... ] )
[ CLUSTER BY ( <column_name> [ , ... ] ) ]
[ LOCALSORT BY ( <column_name> [ , ... ] ) ]
[ TBLPROPERTIES ( '<property_name>' = '<property_value>' [ , ... ] ) ]
AS <select_statement>
```

Parameters

`IF NOT EXISTS`
Prevent error if table exists.

`<table_name>`
Name of the table.

`( <column_name> <data_type> ... )`
Optional column definitions to override source.

`PARTITION BY`
Partition columns or transforms.

`CLUSTER BY`
Cluster columns (Enterprise).

`LOCALSORT BY`
Sort columns.

`TBLPROPERTIES`
Table properties.

`AS <select_statement>`
Query to populate the table.

Examples

Create table from select

```
CREATE TABLE demo_table AS SELECT * FROM Samples."samples.dremio.com"."zips.json"
```

Create table with partition and sort

```
CREATE TABLE demo_table2 PARTITION BY (state) LOCALSORT BY (city) AS SELECT * FROM Samples."samples.dremio.com"."zips.json"
```

Create table from time travel query

```
CREATE TABLE demo.example_table AS SELECT * FROM "oracle_tpch".DREMIO.JOBS AT TAG Jan2020
```

`CREATE USER`

Create a new user.

Syntax

```
CREATE USER <username>
```

Parameters

`<username>`
Email of the user (in double quotes).

Example

```
CREATE USER "user@dremio.com"
```

`CREATE VIEW`

Create or replace a view.

Syntax

```
CREATE [ OR REPLACE ] VIEW <view_name> AS <select_statement>
```

Parameters

`OR REPLACE`
Replaces existing view.

`<view_name>`
Name of the view.

`AS <select_statement>`
Query to populate the view.

Examples

Create view

```
CREATE VIEW demo.example_view AS SELECT * FROM "oracle_tpch".DREMIO.JOBS
```

Replace view

```
CREATE OR REPLACE VIEW demo.example_view AS SELECT * FROM "oracle_tpch".DREMIO.INVENTORY
```

`DESCRIBE FUNCTION`

Returns the metadata about an existing user-defined function (UDF).

Syntax

```
{ DESC | DESCRIBE } FUNCTION <function_name>
```

Parameters

`<function_name>`
Name of the UDF.

Examples

```
DESCRIBE FUNCTION hello
```

`DESCRIBE PIPE`

Get high-level information about the settings and configuration of a specific autoingest pipe.

Syntax

```
DESCRIBE PIPE <pipe_name>
```

Parameters

`<pipe_name>`
Name of the pipe.

Examples

```
DESCRIBE PIPE Example_pipe
```

`DESCRIBE TABLE`

Provide high-level information regarding the overall column properties of an existing dataset.

Syntax

```
DESCRIBE TABLE <table_name>
```

Parameters

`<table_name>`
Name of the table.

Example

```
DESCRIBE TABLE taxistats
```

`DROP FOLDER`

Remove a folder from a catalog.

Syntax

```
DROP FOLDER [ IF EXISTS ] <folder_name> [ .<child_folder_name> ]
```

Parameters

`IF EXISTS`
Prevent error if folder doesn't exist.

`<folder_name>`
Name of the folder.

Examples

Drop folder

```
DROP FOLDER myFolder
```

Drop child folder

```
DROP FOLDER myFolder.resources
```

`DROP FUNCTION`

Drops a user-defined function (UDF) in the catalog.

Syntax

```
DROP FUNCTION [ IF EXISTS ] <function_name> [ AS OF <timestamp> ]
```

Parameters

`IF EXISTS`
Prevent error if function doesn't exist.

`<function_name>`
Name of the UDF.

Examples

Drop function

```
DROP FUNCTION hello
```

Drop if exists

```
DROP FUNCTION IF EXISTS hello
```

`DROP PIPE`

Removes the specified pipe from a source.

Syntax

```
DROP PIPE <pipe_name>
```

Parameters

`<pipe_name>`
Name of the pipe.

Examples

```
DROP PIPE Example_pipe
```

`DROP ROLE`

Removes a role.

Syntax

```
DROP ROLE <role_name>
```

Parameters

`<role_name>`
Name of the role.

Examples

```
DROP ROLE role1
```

`DROP TABLE`

Removes a table from your data source.

Syntax

```
DROP TABLE [ IF EXISTS ] <table_name>
```

Parameters

`IF EXISTS`
Prevent error if table doesn't exist.

`<table_name>`
Name of the table.

Example

```
DROP TABLE demo.example_table
```

`DROP USER`

Removes a user.

Syntax

```
DROP USER <username>
```

Parameters

`<username>`
Email of the user (in double quotes).

Example

```
DROP USER "user@dremio.com"
```

`DROP VIEW`

Removes a view.

Syntax

```
DROP VIEW [ IF EXISTS ] <view_name>
```

Parameters

`IF EXISTS`
Prevent error if view doesn't exist.

`<view_name>`
Name of the view.

Examples

```
DROP VIEW demo.example_view
```

`GRANT ROLE`

Grant a role to a user or a role.

Syntax

```
GRANT ROLE <role_name> TO { ROLE | USER } <role_or_user_name>
```

Parameters

`<role_name>`
Name of the role to grant.

`TO { ROLE | USER } <role_or_user_name>`
Recipient role or user.

Examples

Grant role to user

```
GRANT ROLE role1 TO USER "user@dremio.com"
```

Grant role to role

```
GRANT ROLE subrole TO ROLE role1
```

`GRANT TO ROLE`

Grant privileges to a role.

Syntax

```
GRANT { <objectPrivilege> | ALL } ON { <object_type> <object_name> } TO ROLE <role_name>
```

Parameters

`<objectPrivilege>`
Privilege to grant (e.g., `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `ALTER`, `DROP`, `OWNERSHIP`).

`<object_type>`
Type of object (e.g., `PROJECT`, `SOURCE`, `TABLE`, `VIEW`, `FOLDER`).

`<object_name>`
Name of the object.

`<role_name>`
Name of the role.

Examples

Grant create project

```
GRANT CREATE PROJECT, CREATE CLOUD ON ORG TO ROLE "DATA_ENGINEER"
```

Grant select on table

```
GRANT SELECT ON TABLE myTable TO ROLE "DATA_ENGINEER"
```

`GRANT TO USER`

Grant privileges to a user.

Syntax

```
GRANT { <objectPrivilege> | ALL } ON { <object_type> <object_name> } TO USER <username>
```

Parameters

`<objectPrivilege>`
Privilege to grant.

`<object_type>`
Type of object.

`<object_name>`
Name of the object.

`<username>`
Email of the user.

Examples

Grant select on project

```
GRANT SELECT ON PROJECT TO USER "user@dremio.com"
```

Grant ownership on catalog

```
GRANT OWNERSHIP ON CATALOG prodCatalog TO USER "user@dremio.com"
```

`RESET ENGINE`

Clears any session-specific execution engine set using the SET ENGINE command.

Syntax

```
RESET ENGINE
```

Examples

```
RESET ENGINE;
```

`REVOKE FROM ROLE`

Revoke privileges from a role.

Syntax

```
REVOKE { <objectPrivilege> | ALL } ON { <object_type> <object_name> } FROM ROLE
<role_name>
```

Parameters

`<objectPrivilege>`
Privilege to revoke.

`<object_type>`
Type of object.

`<object_name>`
Name of the object.

`<role_name>`
Name of the role.

Examples

Revoke modify from role

```
REVOKE MODIFY, MONITOR ON CLOUD "Default Cloud" FROM ROLE "DATA_ENGINEER"
```

Revoke ownership from role

```
REVOKE OWNERSHIP ON CATALOG prodCatalog FROM ROLE data_engineer
```

`REVOKE FROM USER`

Revoke privileges from a user.

Syntax

```
REVOKE { <objectPrivilege> | ALL } ON { <object_type> <object_name> } FROM USER
<username>
```

Parameters

`<objectPrivilege>`
Privilege to revoke.

`<object_type>`
Type of object.

`<object_name>`
Name of the object.

`<username>`
Email of the user.

Examples

Revoke select from user

```
REVOKE SELECT ON PROJECT FROM USER "user@dremio.com"
```

Revoke ownership from user

```
REVOKE OWNERSHIP ON CATALOG prodCatalog FROM USER "user@dremio.com"
```

`REVOKE ROLE`

Revoke a role from the role or user.

Syntax

```
REVOKE ROLE <role_name> FROM { ROLE | USER } <role_or_user_name>
```

Parameters

`<role_name>`
Name of the role to revoke.

`FROM { ROLE | USER } <role_or_user_name>`
Role or user to revoke from.

Example

```
REVOKE ROLE role1 FROM USER "user@dremio.com"
```

`SET ENGINE`

Specify the engine that will be used to execute subsequent queries in the current session.

Syntax

```
SET ENGINE <engine_name>
```

Parameters

`<engine_name>`
Name of the engine.

Examples

```
SET ENGINE first_engine;
```

`SET TAG`

Specify the routing tag that will be used to route subsequent queries in the current session.

Syntax

```
SET TAG <tag_name>
```

Parameters

`<tag_name>`
Name of the routing tag.

Examples

```
SET TAG Dashboard;
```

`SHOW CREATE TABLE`

Show the definition that creates the specified table.

Syntax

```
SHOW CREATE TABLE <table_name>
```

Parameters

`<table_name>`
Name of the table.

Examples

```
SHOW CREATE TABLE "company_data".employees
```

`SHOW CREATE VIEW`

Show the definition for a view.

Syntax

```
SHOW CREATE VIEW <view_name>
```

Parameters

`<view_name>`
Name of the view.

Examples

```
SHOW CREATE VIEW "company_data".Locations."offices_by_region"
```

`SHOW FUNCTIONS`

Returns the list of user-defined functions (UDFs).

Syntax

```
SHOW FUNCTIONS [ AS OF <timestamp> ] [ LIKE { <pattern> } ]
```

Parameters

`AS OF <timestamp>`
Optional timestamp.

`LIKE { <pattern> }`
Optional pattern to filter results.

Examples

Show all functions

```
SHOW FUNCTIONS;
```

Show functions matching pattern

```
SHOW FUNCTIONS LIKE 'hello';
```

`SHOW TABLES`

Show all the tables that are available in a catalog.

Syntax

```
SHOW TABLES [ IN <catalog_name> ]
```

Parameters

`IN <catalog_name>`
Optional catalog name.

Examples

Show tables

```
SHOW TABLES
```

Show tables in catalog

```
SHOW TABLES IN myCatalog
```

`SHOW VIEWS`

Show all the views that are available in a catalog.

Syntax

```
SHOW VIEWS [ IN <catalog_name> ]
```

Parameters

`IN <catalog_name>`
Optional catalog name.

Examples

Show views

```
SHOW VIEWS
```

Show views in catalog

```
SHOW VIEWS IN myCatalog
```

`RESET TAG`

Clears any session-specific routing tag set using the SET TAG command.

Syntax

```
RESET TAG
```

Examples

```
RESET TAG;
```

`VACUUM TABLE`

Remove older table snapshots and orphan files from Iceberg tables.

Syntax

```
VACUUM TABLE <table_name>
{ EXPIRE SNAPSHOTS [ older_than = <timestamp> ] [ retain_last = <count> ]
| REMOVE ORPHAN FILES [ older_than = <timestamp> ] [ location = <path> ] }
```

Parameters

`<table_name>`
Name of the table.

`EXPIRE SNAPSHOTS`
Remove old snapshots.

`older_than`: Timestamp limit (default: 5 days ago).

`retain_last`: Minimum snapshots to keep.

`REMOVE ORPHAN FILES`
Remove files not in metadata.

`older_than`: Creation timestamp limit (default: 3 days ago).

`location`: Directory to search.

Examples

Expire snapshots

```
VACUUM TABLE my_table EXPIRE SNAPSHOTS older_than = '2023-01-01 00:00:00.000'
retain_last = 5
```

Remove orphan files

```
VACUUM TABLE my_table REMOVE ORPHAN FILES older_than = '2023-01-01 00:00:00.000'
```

`WITH`

Defines a common table expression (CTE), which is a temporary named result set.

Syntax

```
WITH <cte_name> [ ( <cte_column1>, <cte_column2>, ... ) ] AS ( <query> )
SELECT ...
```

Parameters

`<cte_name>`
Name of the CTE.

`<cte_column>`
Optional column names.

`AS ( <query> )`
Query defining the CTE.

Examples

```
WITH cte_quantity (Total) AS ( SELECT SUM(passenger_count) as Total FROM Samples."samples.dremio.com"."NYC-taxi-trips" where passenger_count > 2 GROUP BY pickup_datetime )
SELECT AVG(Total) average_pass FROM cte_quantity
```

`DELETE`

Delete rows from a table.

Syntax

```
DELETE FROM <table_name> [ AS <alias> ] [ USING <additional_table_or_query> ] [ WHERE <where_conditions> ]
```

Parameters

`<table_name>`
Name of the table.

`USING <additional_table>`
Additional tables for join conditions.

`WHERE <where_conditions>`
Filter for rows to delete.

Examples

Delete with join

```
DELETE FROM orders USING returns WHERE orders.order_id = returns.order_id;
```

Delete with subquery

```
DELETE FROM orders WHERE EXISTS (select 1 from returns where order_id = orders.order_id)
```

`INSERT`

Insert records into a table.

Syntax

```
INSERT INTO <table_name> [ ( <column_name> [ , ... ] ) ]
{ <select_statement> | VALUES ( <value> [ , ... ] ) [ , ... ] }
```

Parameters

`<table_name>`
Name of the table.

`( <column_name> ... )`
Optional column list.

`VALUES`
List of values to insert.

Examples

Insert values

```
INSERT INTO myTable VALUES (21, 'Ruth Asawa'), (38, 'Magdalena Abakanowicz')
```

Insert from select

```
INSERT INTO struct_type VALUES (convert_from('{ x: "hi" }', 'json'))
```

`MERGE`

Run insert or update operations on a target table from the results of a join with a source table.

Syntax

```
MERGE INTO <target_table> [ AS <target_alias> ]
USING <source_table> [ AS <source_alias> ]
ON ( <condition> )
[ WHEN MATCHED THEN UPDATE SET <column> = <value> [ , ... ] ]
[ WHEN NOT MATCHED THEN INSERT ( <column> [ , ... ] ) VALUES ( <value> [ , ... ] ) ]
```

Parameters

`<target_table>`
Table to merge into.

`USING <source_table>`
Source table for data.

`ON ( <condition> )`
Join condition.

`WHEN MATCHED THEN UPDATE`
Update existing rows.

`WHEN NOT MATCHED THEN INSERT`
Insert new rows.

Examples

```
MERGE INTO target_table AS t USING source_table AS s ON (t.id = s.id)
WHEN MATCHED THEN UPDATE SET description = s.description_2
WHEN NOT MATCHED THEN INSERT (id, description) VALUES (s.id, s.description_1);
```

`OPTIMIZE TABLE`

Rewrite data and manifest files to provide peak performance.

Syntax

```
OPTIMIZE TABLE <table_name>
[ REWRITE DATA [ USING BIN_PACK ]
  [ ( { TARGET_FILE_SIZE_MB | MIN_FILE_SIZE_MB | MAX_FILE_SIZE_MB | MIN_INPUT_FILES } =
<value> [, ... ] ) ]
  [ FOR PARTITIONS <predicate> ] ]
[ REWRITE MANIFESTS ]
```

Parameters

`<table_name>`
Name of the table.

`REWRITE DATA`
Rewrite data files.

`TARGET_FILE_SIZE_MB`: Target size.

`MIN_INPUT_FILES`: Min files to trigger optimization.

`FOR PARTITIONS`
Filter partitions to optimize.

`REWRITE MANIFESTS`
Optimize manifest files.

Examples

Optimize data

```
OPTIMIZE TABLE demo.example_table REWRITE DATA USING BIN_PACK (TARGET_FILE_SIZE_MB=512,
MIN_INPUT_FILES=10)
```

Optimize manifests

```
OPTIMIZE TABLE demo.example_table REWRITE MANIFESTS
```

`ROLLBACK TABLE`

Roll back an Iceberg table to a previous snapshot.

Syntax

```
ROLLBACK TABLE <table_name> TO { SNAPSHOT '<snapshot_id>' | TIMESTAMP '<timestamp>' }
```

Parameters

`<table_name>`
Name of the table.

`TO SNAPSHOT '<snapshot_id>'`
Rollback to snapshot ID.

`TO TIMESTAMP '<timestamp>'`
Rollback to timestamp.

Examples

Rollback to snapshot

```
ROLLBACK TABLE demo.example_table TO SNAPSHOT '2489484212521283189'
```

Rollback to timestamp

```
ROLLBACK TABLE demo.example_table TO TIMESTAMP '2022-06-22 17:06:00'
```

`TRUNCATE`

Delete all rows from a table with minimal computation.

Syntax

```
TRUNCATE [ TABLE ] [ IF EXISTS ] <table_name>
```

Parameters

`TABLE`
Optional keyword.

`IF EXISTS`
Prevent error if table missing.

`<table_name>`
Name of the table.

Examples

```
TRUNCATE TABLE IF EXISTS myTable
```

`UPDATE`

Update rows in a table.

Syntax

```
UPDATE <table_name> [ AS <alias> ]
SET <column> = <value> [ , ... ]
[ WHERE <condition> ]
```

Parameters

`<table_name>`
Name of the table.

`SET <column> = <value>`
Columns to update.

`WHERE <condition>`
Filter for rows to update.

Examples

```
UPDATE MYSOURCE.MYTABLE SET EXPR$0 = s.EXPR$1 FROM MYSOURCE.MYTABLE2 AS s
```

SQL Functions

ABS

Returns the absolute value of the argument.

Syntax

```
ABS(numeric_expression NUMERIC) → NUMERIC
```

Parameters

`numeric_expression`: BINARY, DECIMAL, DOUBLE, FLOAT, INTEGER

Examples

```
SELECT ABS(0.0) -- 0.0
SELECT ABS(-2) -- 2
SELECT ABS(NULL) -- null
```

ACOS

Returns the arc cosine of the argument.

Syntax

```
ACOS(numeric_expression NUMERIC) → FLOAT
```

Parameters

`numeric_expression`: The number in radians. This must be DOUBLE, INTEGER, BIGINT, DECIMAL, or FLOAT.

Examples

```
SELECT ACOS(0) -- 1.5707963267948966
SELECT ACOS(1.0) -- 0.0
SELECT ACOS(-1) -- 3.141592653589793
```

AES_DECRYPT

Decrypts a string using AES encryption.

Syntax

```
AES_DECRYPT(ciphertext varchar, key varchar) → varchar
```

Parameters

`ciphertext`: The string to be decrypted.

`key`: The key to use to decrypt the ciphertext. Must be 16, 24, or 32 characters.

Examples

```
SELECT AES_DECRYPT(UNBASE64('UvicDn/xiUDmfSE+KYjjyw=='), 'mypassword123456') -- Dremio
```

AES_ENCRYPT

Encrypts a string using AES encryption.

Syntax

```
AES_ENCRYPT(plaintext varchar, key varchar) → varchar
```

Parameters

`plaintext`: The string to be encrypted.

`key`: The key to use to encrypt the plaintext. Must be 16, 24, or 32 characters.

Examples

```
SELECT BASE64(AES_ENCRYPT('Dremio', 'mypassword123456')) -- UvicDn/xiUDmfSE+KYjjyw==
```

AI_CLASSIFY

Classifies text using a Large Language Model (LLM).

Syntax

```
AI_CLASSIFY( [model_name VARCHAR,] prompt VARCHAR | (prompt VARCHAR, file_reference), categories ARRAY<VARCHAR|INT|FLOAT|BOOLEAN> ) → VARCHAR|INT|FLOAT|BOOLEAN
```

Parameters

˜model_name` (optional): Optional model specification in format 'modelProvider.modelName' (e.g., 'gpt.4o').

˜prompt`: Classification instructions for the LLM. Use (prompt, file_reference) to process files from LIST_FILES.

˜categories`: Array of possible classifications. The LLM will choose one of these values as the result.

Examples

```
SELECT recipe_name, AI_CLASSIFY( 'Determine the difficulty level based on these ingredients and steps', ingredients || ' - Steps: ' || cooking_instructions, ARRAY['Beginner', 'Easy', 'Intermediate', 'Advanced', 'Expert'] ) AS difficulty_level, prep_time, number_of_ingredients FROM recipe_database;
```

AI_COMPLETE

Generates text completion using a Large Language Model (LLM).

Syntax

```
AI_COMPLETE( [model_name VARCHAR,] prompt VARCHAR ) → VARCHAR
```

Parameters

˜model_name` (optional): Optional model specification in format 'modelProvider.modelName'.

˜prompt`: Completion instruction for the LLM. Natural language text describing what you want the model to generate.

Examples

```
SELECT dish_name, AI_COMPLETE( 'Write an appetizing menu description for this dish: ' || dish_name || '. Main ingredients: ' || main_ingredients || '. Cooking style: ' || cuisine_type ) AS menu_description FROM restaurant_dishes;
```

AI_GENERATE

Generates structured data or text using a Large Language Model (LLM).

Syntax

```
AI_GENERATE( [model_name VARCHAR,] prompt VARCHAR | (prompt VARCHAR, file_reference) [WITH SCHEMA data_type] ) → ANY|ROW
```

Parameters

`model_name` (optional): Optional model specification in format 'modelProvider.modelName'.

`prompt`: Natural language instruction for the LLM. Use (prompt, file_reference) to process files from LIST_FILES.

`WITH SCHEMA data_type`: Output structure specification (optional).

Examples

```
WITH recipe_analysis AS ( SELECT file['path'] AS recipe_file, AI_GENERATE( 'gpt.4o', ('Extract recipe details', file) WITH SCHEMA ROW( recipe_name VARCHAR, cuisine_type VARCHAR) ) AS recipe_info FROM TABLE(LIST_FILES('@Cookbooks/cookbook_recipes')) WHERE file['path'] LIKE '%.pdf' ) SELECT recipe_file, recipe_info['recipe_name'] AS recipe, recipe_info['cuisine_type'] AS cuisine FROM recipe_analysis ORDER BY recipe ASC;
```

APPROX_COUNT_DISTINCT

Returns the approximate number of distinct values in a column.

Syntax

```
APPROX_COUNT_DISTINCT(column_name any primitive) → BIGINT
```

Parameters

`column_name`: You can specify a column of any primitive data type.

Examples

```
SELECT                        APPROX_COUNT_DISTINCT(IncidntNum)                        FROM Samples."samples.dremio.com"."SF_incidents2016.json" -- 116696
```

APPROX_PERCENTILE

Returns the approximate percentile of a column.

Syntax

```
APPROX_PERCENTILE(column_name numeric, percentile double) → DOUBLE
```

Parameters

`column_name`: The column for which to compute the approximate percentile.

`percentile`: The percentile to use in the approximation. Must be a number between 0 and 1.

Examples

```
SELECT  APPROX_PERCENTILE(pop,  0.5)  FROM  Samples."samples.dremio.com"."zips.json"  --
2780.17855684608
```

ARRAYS_OVERLAP

Checks if two arrays have any elements in common.

Syntax

```
ARRAYS_OVERLAP(arr1 LIST, arr2 LIST) → BOOLEAN
```

Parameters

`arr1`: The first array.

`arr2`: The second array.

Examples

```
SELECT ARRAYS_OVERLAP(ARRAY['foo', 'bar'], ARRAY['bar', 'baz']) -- true
SELECT ARRAYS_OVERLAP(ARRAY['foo', 'bar'], ARRAY['baz', 'qux']) -- false
```

ARRAY_AGG

Aggregates values into an array.

Syntax

```
ARRAY_AGG ( [ DISTINCT ] expression ) → array
```

Parameters

˜expression`: An expression of any primitive type to aggregate into an array.

Examples

```
SELECT ARRAY_AGG(name) FROM <catalog-name>.people; -- ['Bob', 'Charlie', 'Alice']
```

ARRAY_APPEND

Appends an element to an array.

Syntax

```
ARRAY_APPEND(array LIST, element ANY) → LIST
```

Parameters

˜array`: The array to append to.

˜element`: The element to append to the array.

Examples

```
SELECT ARRAY_APPEND(ARRAY[1, 2], 3); -- [1, 2, 3]
```

ARRAY_AVG

Returns the average of the elements in an array.

Syntax

```
ARRAY_AVG(list_column LIST) → numeric
```

Parameters

˜list_column`: Column that contains a LIST expression. Every element of the list must be a number.

Examples

```
SELECT ARRAY_AVG(array_col) -- 2
```

ARRAY_CAT

Concatenates two arrays.

Syntax

```
ARRAY_CAT(arr1 LIST, arr2 LIST) → list
```

Parameters

`arr1`: The source array.

`arr2`: The array to be appended to the source array.

Examples

```
SELECT ARRAY_CAT(ARRAY[1, 2, 3], ARRAY[4, 5, 6]) -- [1, 2, 3, 4, 5, 6]
```

ARRAY_COMPACT

Removes null values from an array.

Syntax

```
ARRAY_COMPACT(arr LIST) → list
```

Parameters

`arr`: The array from which to remove null values.

Examples

```
SELECT ARRAY_COMPACT(array_col) -- [1, 2]
```

ARRAY_CONTAINS

Checks if an array contains a specific value.

Syntax

```
ARRAY_CONTAINS(list LIST, value any) → boolean
```

Parameters

`list`: The list to search.

`value`: An expression of a type that is comparable with the LIST.

Examples

```
SELECT ARRAY_CONTAINS(CONVERT_FROM('["apple", "pear", "banana"]', 'json'), NULL) -- null
SELECT ARRAY_CONTAINS(CONVERT_FROM('["apple", "pear", "banana"]', 'json'), 'pear') --
true
SELECT ARRAY_CONTAINS(CONVERT_FROM('["apple", "pear", "banana"]', 'json'), 'grape') --
false
```

ARRAY_DISTINCT

Returns an array with distinct elements from the input array.

Syntax

```
ARRAY_DISTINCT(input LIST) → LIST
```

Parameters

`input`: The input array from which to return only distinct elements.

Examples

```
SELECT ARRAY_DISTINCT(ARRAY[1, 2, 3, 1, 2, 3]) -- [2, 3, 1]
```

ARRAY_FREQUENCY

Returns a map where keys are the elements of the array and values are their frequencies.

Syntax

```
ARRAY_FREQUENCY(array LIST) → MAP
```

Parameters

˜array`: The array of values for which to calculate frequency. Accepts primitive types.

Examples

```
SELECT ARRAY_FREQUENCY(ARRAY[2,1,2,1,1,5]); -- {"1":3, "2":2, "5":1}
SELECT ARRAY_FREQUENCY(ARRAY['a','b','ab','b','a']); -- {"a":2, "ab":1, "b":2}
SELECT  ARRAY_FREQUENCY(ARRAY['foo',  'bar',  'FOO',  'foo']);  --  {"FOO":1,  "bar":1,
"foo":2}
SELECT ARRAY_FREQUENCY(array_col); -- {"1":1, "2":2}
```

ARRAY_GENERATE_RANGE

Generates an array of integers in a specified range.

Syntax

```
ARRAY_GENERATE_RANGE(start int32, stop int32, step int32) → list
```

Parameters

˜start`: The first number in the range of numbers to return.

˜stop`: The last number in the range. Note that this number is not included in the range of numbers returned.

˜step`: The amount to increment or decrement each subsequent number in the array. May be a positive or negative number. Cannot be 0. Default value is 1.

Examples

```
SELECT ARRAY_GENERATE_RANGE(1, 5) -- [1, 2, 3, 4]
SELECT ARRAY_GENERATE_RANGE(0, 16, 5) -- [0, 5, 10, 15]
SELECT ARRAY_GENERATE_RANGE(0, -16, -5) -- [0, -5, -10, -15]
SELECT ARRAY_GENERATE_RANGE(2, 2, 4) -- []
SELECT ARRAY_GENERATE_RANGE(8, 2, 2) -- []
SELECT ARRAY_GENERATE_RANGE(2, 8, -2) -- []
SELECT ARRAY_GENERATE_RANGE(2, 2) -- []
```

ARRAY_INSERT

Inserts an element into an array at a specified position.

Syntax

```
ARRAY_INSERT(arr LIST, position INT, new_element ANY) → LIST
```

Parameters

˜arr`: The array to search.

˜position`: The zero-based position in the input array where the new element should be inserted.

˜new_element`: The new element to insert in the specified position.

Examples

```
SELECT ARRAY_INSERT(ARRAY[1, 2, 3, 4, 5], 2, 55); -- [1, 2, 55, 3, 4, 5]
SELECT ARRAY_INSERT(ARRAY[1, 2, 3], 6, 55); -- [1, 2, 3, NULL, NULL, NULL, 55]
SELECT ARRAY_INSERT(ARRAY[1, 2, 3], -1, 55); -- [1, 2, 55, 3]
```

ARRAY_MAX

Returns the maximum value in an array.

Syntax

```
ARRAY_MAX(list_column LIST) → numeric
```

Parameters

˜list_column`: Column that contains a LIST expression. Every element of the list must be a number such as INT, BIGINT, FLOAT4, FLOAT8, or DECIMAL. Cannot be an array literal.

Examples

```
SELECT ARRAY_MAX(array_col) -- 3
SELECT ARRAY_MAX(array_col) -- NULL
```

ARRAY_MIN

Returns the minimum value in an array.

Syntax

```
ARRAY_MIN(list_column LIST) → numeric
```

Parameters

`list_column`: Column that contains a LIST expression. Every element of the list must be a number such as INT, BIGINT, FLOAT4, FLOAT8, or DECIMAL. Cannot be an array literal.

Examples

```
SELECT ARRAY_MIN(array_col) -- 1
SELECT ARRAY_MIN(array_col) -- NULL
```

ARRAY_POSITION

Returns the position of the first occurrence of an element in an array.

Syntax

```
ARRAY_POSITION(element ANY, arr LIST) → numeric
```

Parameters

`element`: Element to find in the array.

`arr`: The array to search.

Examples

```
SELECT ARRAY_POSITION(CAST(3 AS BIGINT), ARRAY[1, 2, 3]) -- 2
SELECT ARRAY_POSITION(4, ARRAY[1, 2, 3]) -- NULL
SELECT ARRAY_POSITION(NULL, array_col) -- 1
SELECT ARRAY_POSITION(ARRAY[2,3], ARRAY[ARRAY[1,2], ARRAY[2,3]]) -- 1
```

ARRAY_PREPEND

Prepends an element to the beginning of an array.

Syntax

```
ARRAY_PREPEND(element ANY, array LIST) → LIST
```

Parameters

`element`: The element to prepend to the array.

˜array`: The array to prepend to.

Examples

```
SELECT ARRAY_PREPEND(1, ARRAY[2, 3]); -- [1, 2, 3]
```

ARRAY_REMOVE

Removes all occurrences of a value from an array.

Syntax

```
ARRAY_REMOVE(list_column LIST, value any) → list
```

Parameters

˜list_column`: Column that contains a LIST expression. Cannot be an array literal.

˜value`: An expression of any data type.

Examples

```
SELECT ARRAY_REMOVE(array_col, 1) -- [2, 3]
SELECT ARRAY_REMOVE(array_col, 2) -- [1,null]
SELECT ARRAY_REMOVE(array_col, null) -- NULL
SELECT ARRAY_REMOVE(array_col, 2) -- NULL
SELECT ARRAY_REMOVE(null, 2) -- NULL
```

ARRAY_REMOVE_AT

Removes the element at a specified position from an array.

Syntax

```
ARRAY_REMOVE_AT(arr LIST, position int32) → list
```

Parameters

˜arr`: Array from which to remove the element at the specified position.

˜position`: The zero-based position of the element to be removed. The function removes the element at the specified position. A negative position is interpreted as an index from the back of the array. For example, the value -1 removes the last element in

the array.

Examples

```
SELECT ARRAY_REMOVE_AT(ARRAY[1, 2, 3], 1) -- [1, 3]
SELECT ARRAY_REMOVE_AT(ARRAY[1, 2, 3], -1) -- [1, 2]
SELECT ARRAY_REMOVE_AT(ARRAY[1, 2, 3], 10) -- [1, 2, 3]
```

ARRAY_SIZE

Returns the number of elements in an array.

Syntax

```
ARRAY_SIZE(arr LIST) → numeric
```

Parameters

`arr`: The source array.

Examples

```
SELECT ARRAY_SIZE(ARRAY[1, 4, 5]) -- 3
```

ARRAY_SLICE

Returns a subset of an array.

Syntax

```
ARRAY_SLICE(arr LIST, from int, to int) → LIST
```

Parameters

`arr`: The input array.

`from`: The zero-based position in the input array of the first element to include in the output array. Elements in positions that are less than the from position are not included in the output array. A negative position is interpreted as an index from the back of the array. For example, the value -1 begins the output array with the last element in the input array.

`to`: The zero-based position in the input array of the last element to include in the

output array. Elements in positions that are equal to or greater than the to position are not included in the resulting array. A negative position is interpreted as an index from the back of the array. For example, the value -1 ends the output array with the second-to-last element in the input array.

Examples

```
SELECT ARRAY_SLICE(array_col) -- [0,1,2]
SELECT ARRAY_SLICE(array_col) -- [0,1,2,3,4]
SELECT ARRAY_SLICE(array_col) -- [2,3]
SELECT ARRAY_SLICE(array_col) -- []
```

ARRAY_SUM

Returns the sum of the elements in an array.

Syntax

```
ARRAY_SUM(list_column LIST) → numeric
```

Parameters

˜list_column`: Column that contains a LIST expression. Every element of the list must be a number such as INT, BIGINT, FLOAT4, FLOAT8, or DECIMAL. Cannot be an array literal.

Examples

```
SELECT ARRAY_SUM(array_col) -- 6
SELECT ARRAY_SUM(array_col) -- 3
SELECT ARRAY_SUM(array_col) -- 0
SELECT ARRAY_SUM(array_col) -- NULL
```

ARRAY_TO_STRING

Converts an array to a string, with elements separated by a delimiter.

Syntax

```
ARRAY_TO_STRING(arr LIST, delimiter VARCHAR) → VARCHAR
```

Parameters

`arr`: The source array.

`delimiter`: The string to place between each element in the array.

Examples

```
SELECT ARRAY_TO_STRING(ARRAY[1, 2, 3], ',') -- 1,2,3
SELECT ARRAY_TO_STRING(array_col, ',') -- 1,,3
```

ASCII

Returns the ASCII code value of the leftmost character of the string.

Syntax

```
ASCII(expression varchar) → int32
```

Parameters

`expression`: The string for which the ASCII code for the first character in the string is returned.

Examples

```
SELECT ASCII ('DREMIO') -- 68
SELECT ASCII ('D') -- 68
SELECT ASCII ('') -- 0
```

ASIN

Returns the arc sine of the argument.

Syntax

```
ASIN(numeric_expression NUMERIC) → FLOAT
```

Parameters

`numeric_expression`: The number in radians. This must be DOUBLE, INTEGER, or FLOAT.

Examples

```
SELECT ASIN(0) -- 0.0
SELECT ASIN(1) -- 1.5707963267948966
SELECT ASIN(-1) -- -1.5707963267948966
```

ATAN

Returns the arc tangent of the argument.

Syntax

```
ATAN(inputValue FLOAT) → FLOAT
```

Parameters

`inputValue`: Floating-point input value, in the range (negative-infinity:positive-infinity)

Examples

```
SELECT ATAN(0) -- 0.0
SELECT ATAN(1) -- 0.7853981633974483
SELECT ATAN(-1) -- -0.7853981633974483
SELECT ATAN(19564.7) -- 1.5707452143321894
```

ATAN2

Returns the arc tangent of the two arguments.

Syntax

```
ATAN2(y NUMERIC, x NUMERIC) → DOUBLE
```

Parameters

`y`: Floating-point input value for the y-coordinate, in the range (negative-infinity:positive-infinity).

`x`: Floating-point input value for the x-coordinate, in the range (negative-infinity:positive-infinity).

Examples

```
SELECT ATAN2(1.0,0.0) -- 1.5707963267948966
SELECT ATAN2(0.0,1.0) -- 0.0
```

```
SELECT ATAN2(0.0,-1.0) -- 3.141592653589793
SELECT ATAN2(-0.00000000001,-1.0) -- -3.141592653579793
SELECT ATAN2(0.0,0.0) -- 0.0
```

AVG

Returns the average of the values in a group.

Syntax

```
AVG(numeric_expression NUMERIC) → DOUBLE
```

Parameters

`numeric_expression`: The values for which to compute the average. Values can be type DOUBLE, NUMERIC, INTEGER, INTERVAL_DATE, or INTERVAL_YEAR.

Examples

```
SELECT AVG(3) -- 3.0
SELECT AVG("val_col"); -- -0.333333
```

BASE64

Encodes a string using Base64.

Syntax

```
BASE64(expression varbinary) → varchar
```

Parameters

`expression`: The string to encode.

Examples

```
SELECT BASE64('Dremio') -- RHJlbWlv
```

BIN

Returns a string representation of the binary value of an integer.

Syntax

```
BIN(expression integer) → varchar
```

Parameters

`expression`: An integer expression to encode.

Examples

```
SELECT BIN(100) -- 1100100
SELECT BIN(-100) -- 11111111111111111111111110011100
SELECT BIN(null) -- null
```

BINARY_STRING

Converts a string to a binary string.

Syntax

```
BINARY_STRING(expression VARCHAR) → BINARY
```

Parameters

`expression`: Varchar expression to convert to binary

Examples

```
SELECT BINARY_STRING('DREMIO') -- RFJFTUlP
SELECT BINARY_STRING('000') -- MDAw
```

BITWISE_AND

Returns the bitwise AND of two numbers.

Syntax

```
BITWISE_AND(op1 NUMERIC, op2 NUMERIC) → NUMERIC
```

Parameters

`op1`: First operand

`op2`: Second operand

Examples

```
SELECT BITWISE_AND(7, 4) -- 4
SELECT BITWISE_AND(1, 2) -- 0
```

BITWISE_NOT

Returns the bitwise NOT of a number.

Syntax

```
BITWISE_NOT(op1 NUMERIC) → NUMERIC
```

Parameters

`op1`: Value to invert.

Examples

```
SELECT BITWISE_NOT(0) -- -1
SELECT BITWISE_NOT(9223372036854775807) -- -9223372036854775808
```

BITWISE_OR

Returns the bitwise OR of two numbers.

Syntax

```
BITWISE_OR(op1 NUMERIC, op2 NUMERIC) → NUMERIC
```

Parameters

`op1`: First operand.

`op2`: Second operand.

Examples

```
SELECT BITWISE_OR(7, 4) -- 7
SELECT BITWISE_OR(1, 2) -- 3
```

BITWISE_XOR

Returns the bitwise XOR of two numbers.

Syntax

```
BITWISE_XOR(op1 NUMERIC, op2 NUMERIC) → NUMERIC
```

Parameters

`op1`: First operand.

`op2`: Second operand.

Examples

```
SELECT BITWISE_XOR(7, 4) -- 3
SELECT BITWISE_XOR(1, 2) -- 3
```

BIT_AND

Returns the bitwise AND of all non-null input values, or null if none.

Syntax

```
BIT_AND(expression int) → int
```

Parameters

`expression`: An expression that evaluates to a data type that can be cast as an integer.

Examples

```
SELECT BIT_AND(passenger_count) FROM Samples."samples.dremio.com"."NYC-taxi-trips" -- 0
```

BIT_LENGTH

Returns the length of the string in bits.

Syntax

```
BIT_LENGTH(expression binary, varchar) → integer
```

Parameters

`expression`: A binary or varchar expression.

Examples

```
SELECT BIT_LENGTH(1010) -- 32
SELECT BIT_LENGTH('DREMIO') -- 48
SELECT BIT_LENGTH('abc') -- 24
SELECT BIT_LENGTH(NULL) -- null
```

BIT_OR

Returns the bitwise OR of all non-null input values, or null if none.

Syntax

```
BIT_OR(expression int) → int
```

Parameters

`expression`: An expression that evaluates to a data type that can be cast as an integer.

Examples

```
SELECT BIT_OR(passenger_count) FROM Samples."samples.dremio.com"."NYC-taxi-trips" -- 255
```

BOOL_AND

Computes the boolean AND of two boolean expressions.

Syntax

```
BOOL_AND(bool_expression1 boolean, bool_expression2 boolean) → boolean
```

Parameters

`bool_expression1`: Boolean input expression.

`bool_expression2`: Boolean input expression.

Examples

```
SELECT BOOL_AND(TRUE, FALSE) -- False
```

BOOL_OR

Computes the boolean OR of two boolean expressions.

Syntax

```
BOOL_OR(bool_expression1 boolean, bool_expression2 boolean) → boolean
```

Parameters

`bool_expression1`: Boolean input expression.

`bool_expression2`: Boolean input expression.

Examples

```
SELECT BOOL_OR(TRUE, FALSE) -- True
```

CBRT

Returns the cube root of a number.

Syntax

```
CBRT(numeric_expression NUMERIC) → FLOAT
```

Parameters

`numeric_expression`: The number (DOUBLE, FLOAT, INTEGER) for which you want to

compute the cube root.

Examples

```
SELECT CBRT(8) -- 2.0
SELECT CBRT(120) -- 4.932424148660941
SELECT CBRT(99.5) -- 4.633839922986558
```

CEIL

Alias for CEILING.

CEILING

Returns the smallest integer not less than the argument.

Syntax

```
CEILING(numeric_expression NUMERIC) → INTEGER
```

Parameters

`numeric_expression`: The number (DOUBLE, FLOAT, INTEGER) for which you want to compute the ceiling.

Examples

```
SELECT CEILING(3.1459) -- 4
SELECT CEIL(37.775420706711) -- 38
SELECT CEIL(-37.775420706711) -- -37
SELECT CEIL(0) -- 0
```

CHAR

Alias for CHR.

CHAR_LENGTH

Returns the number of characters in a string.

Syntax

```
CHAR_LENGTH(expression STRING) → INTEGER
```

Parameters

`expression`: The expression (VARCHAR) to determine character length for.

Examples

```
SELECT CHAR_LENGTH('get the char length') -- 19
SELECT CHAR_LENGTH('DREMIO') -- 6
```

CHARACTER_LENGTH

Returns the number of characters in a string.

Syntax

```
CHARACTER_LENGTH(expression varchar) → int32
```

Parameters

`expression`: String expression to determine the length of.

Examples

```
SELECT CHARACTER_LENGTH('DREMIO') -- 6
```

CHR

Returns the character with the given ASCII code.

Syntax

```
CHR(integer_expression int32) → varchar
```

Parameters

`integer_expression`: Unicode code point to convert to character.

Examples

```
SELECT CHR(72) -- H
SELECT CHR(33) -- null
```

COALESCE

Returns the first non-null expression in the list.

Syntax

```
COALESCE(expression1, expression2, [ ..., expressionN ]) → same as input type
```

Parameters

˜expression`: A combination of symbols and operators that the database evaluates to obtain a single data value. Expressions can be a single constant, variable, column, or scalar function.

Examples

```
SELECT COALESCE(address1, address2, city, state, zipCode) FROM customers -- 123 Main Street
```

COL_LIKE

Returns true if the expression matches the pattern.

Syntax

```
COL_LIKE(expression_col varchar, pattern_col varchar) → boolean
```

Parameters

˜expression_col`: A column containing an expression to compare.

˜pattern_col`: A column containing the pattern to compare to the expression.

Examples

```
-- Assuming table 'names' with columns 'name' and 'pat'
-- values ('john', '%oh%'), ('jacob', '%aco%'), ('bill', '%ob%')
SELECT name FROM names WHERE COL_LIKE(name, pat);
-- john
-- jacob
```

CONCAT

Concatenates two or more strings.

Syntax

```
CONCAT(expression1 string [, expression2 string] [, expressionN string]) → string
```

Parameters

`expression1`: First string expression.

`expression2` (optional): Second string expression.

`expressionN` (optional): Nth string expression.

Examples

```
SELECT CONCAT('CON', 'CAT') -- CONCAT
SELECT CONCAT('con', 'cat', NULL) -- concat
```

CONCAT_WS

Concatenates strings with a separator.

Syntax

```
CONCAT_WS(separator, expression1, expression2, [ ... expressionN ]) → string
```

Parameters

`separator`: An expression of any character type.

`expression`: An expression can be any data type. All arguments must be the same data type.

Examples

```
SELECT CONCAT_WS('-', 'cat', 'dog', 'bird') -- cat-dog-bird
```

CONVERT_FROM

Converts a binary string to a Dremio data type.

Syntax

```
CONVERT_FROM(binary_value value_to_convert, data_type name_of_type) → varies
```

Parameters

`binary_value`: The binary string to convert to a Dremio data type.

`data_type`: The data type of the specified binary string.

Examples

```
SELECT CONVERT_FROM('["apple", "strawberry", "banana"]', 'json') -- ['apple', 'strawberry', 'banana']
SELECT CONVERT_FROM('{"name":"Gnarly", "age":7, "car":null}', 'json') -- {"name:"Gnarly","age":7}
```

CONVERT_TO

Converts a value to a binary string.

Syntax

```
CONVERT_TO(expression value_to_convert, data_type name_of_type) → VARBINARY
```

Parameters

`expression`: The value to convert to a binary string.

`data_type`: The data type to use for the conversion to a binary string.

Examples

```
SELECT CONVERT_TO('this value' ,'UTF8') -- dGhpcyB2YWx1ZQ==
```

CONVERT_TIMEZONE

Converts a timestamp to a different time zone.

Syntax

```
CONVERT_TIMEZONE([sourceTimezone string], destinationTimezone string, timestamp date, timestamp, or string in ISO 8601 format) → timestamp
```

Parameters

`sourceTimezone` (optional): The time zone of the timestamp. If you omit this parameter, Dremio assumes that the source time zone is UTC.

`destinationTimezone`: The time zone to convert the timestamp to.

`timestamp`: The timestamp to convert

Examples

```
select    convert_timezone('America/Los_Angeles',    'America/New_York',    '2021-04-01
15:27:32') -- 2021-04-01 18:27:32
select convert_timezone('America/Los_Angeles', 'America/New_York', timestamp '2021-04-01
15:27:32'); -- 2021-04-01 18:27:32
select convert_timezone('PST', 'EST', '2021-04-01 15:27:32') -- 2021-04-01 18:27:32
select  convert_timezone('America/Los_Angeles',  'America/New_York',  '2021-04-01')  --
2021-04-01 03:00:00
select convert_timezone('America/Los_Angeles', 'America/New_York', date '2021-04-01') --
2021-04-01 03:00:00
select convert_timezone('EDT', '2021-04-01 15:27:32') -- 2021-04-01 11:27:32
select convert_timezone('PST', '+02:00', '2021-04-01 15:27:32') -- 2021-04-02 01:27:32
```

CORR

Returns the coefficient of correlation of a set of number pairs.

Syntax

```
CORR(expression1 numeric, expression2 numeric) → double
```

Parameters

`expression1`: An expression that evaluates to a numeric type. This parameter is the dependent value.

`expression2`: An expression that evaluates to a numeric type. This parameter is the independent value.

Examples

```
SELECT "CORR"(100, 4) -- NaN
```

COS

Returns the cosine of an angle.

Syntax

```
COS(numeric_expression NUMERIC) → FLOAT
```

Parameters

`numeric_expression`: The number in radians. This must be DOUBLE, INTEGER, or FLOAT.

Examples

```
SELECT COS(0) -- 1.0
SELECT COS(1.0) -- 0.5403023058681398
SELECT COS(-1) -- 0.5403023058681398
```

COT

Returns the cotangent of an angle.

Syntax

```
COT(numeric_expression NUMERIC) → FLOAT
```

Parameters

`numeric_expression`: The number in radians. This must be DOUBLE, INTEGER, or FLOAT.

Examples

```
SELECT COT(0) -- 1.0
SELECT COT(1.0) -- 0.5403023058681398
SELECT COT(-1) -- 0.5403023058681398
```

COUNT

Returns the number of rows in the query.

Syntax

```
COUNT(expression ANY) → BIGINT
```

Parameters

`expression`: The expression to evaluate. Can be an asterisk (*) or the column name of any primitive data type. Use an asterisk to include rows that contain NULL. Use a column name to ignore rows that contain NULL.

Examples

```
SELECT COUNT(passenger_count) FROM "Samples"."samples.dremio.com"."NYC-taxi-trips"; --
338293677
```

COVAR_POP

Returns the population covariance of a set of number pairs.

Syntax

```
COVAR_POP(expression1 NUMERIC, expression2 NUMERIC) → DOUBLE
```

Parameters

`expression1`: An expression that evaluates to a numeric type. This parameter is the dependent value.

`expression2`: An expression that evaluates to a numeric type. This parameter is the independent value.

Examples

```
SELECT             COVAR_POP(trip_distance_mi,             fare_amount)             FROM
Samples."samples.dremio.com"."NYC-taxi-trips"; -- 31.705367711861427
SELECT        COVAR_POP(DISTINCT        trip_distance_mi,        fare_amount        FROM
Samples."samples.dremio.com"."NYC-taxi-trips"; -- 302.592806814534
```

COVAR_SAMP

Returns the sample covariance of a set of number pairs.

Syntax

```
COVAR_SAMP(expression1 NUMERIC, expression2 NUMERIC) → DOUBLE
```

Parameters

`expression1`: An expression that evaluates to a numeric type. This parameter is the dependent value.

`expression2`: An expression that evaluates to a numeric type. This parameter is the independent value.

Examples

```
SELECT            COVAR_SAMP(trip_distance_mi,            fare_amount)            FROM
Samples."samples.dremio.com"."NYC-taxi-trips"; -- 31.705367805565245
SELECT       COVAR_SAMP(DISTINCT       trip_distance_mi,       fare_amount)       FROM
Samples."samples.dremio.com"."NYC-taxi-trips"; -- 302.5936880585907
```

CUME_DIST

Calculates the cumulative distribution of a value in a group of values.

Syntax

```
CUME_DIST() OVER ( [PARTITION BY partition_expression] [ORDER BY order_expression]) →
double
```

Parameters

`partition_expression` (optional): An expression that groups rows into partitions.

`order_expression`: An expression that specifies the order of the rows within each partition.

Examples

```
SELECT "Category", "Descript", "DayOfWeek", CUME_DIST() OVER ( PARTITION BY "Category"
ORDER BY "DayOfWeek") FROM Samples."samples.dremio.com"."SF_incidents2016.json"
-- Category, Descript, DayOfWeek, EXPR$3
-- ARSON, ARSON, Friday, 0.13636363636363635
-- EMBEZZLEMENT, EMBEZZLED VEHICLE, Friday, 0.18452380952380953
```

CURRENT_DATE

Returns the current date.

Syntax

```
CURRENT_DATE() → date
```

Examples

```
SELECT CURRENT_DATE() -- 2021-07-02
SELECT CURRENT_DATE -- 2021-07-02
```

CURRENT_TIME

Returns the current time.

Syntax

```
CURRENT_TIME() → TIME
```

Examples

```
SELECT CURRENT_TIME() -- 06:04:31
SELECT CURRENT_TIME -- 06:04:31
```

CURRENT_TIMESTAMP

Returns the current timestamp.

Syntax

```
CURRENT_TIMESTAMP() → TIMESTAMP
```

Examples

```
SELECT CURRENT_TIMESTAMP() -- 2021-06-24 06:11:51.567000
```

DATE_ADD

Adds a specified number of days or a time interval to a date or timestamp.

Syntax

```
DATE_ADD(date_expression string, days integer) → date
DATE_ADD(date_expression date, days integer) → date
DATE_ADD(date_expression string, time_interval interval) → timestamp
DATE_ADD(date_expression date, time_interval interval) → timestamp
DATE_ADD(timestamp_expression string, time_interval interval) → timestamp
```

Parameters

˜date_expression`: A string-formatted date ('YYYY-MM-DD') or a DATE column/literal.

˜days`: The number of days to add.

˜time_interval`: A CAST of a number to an interval (e.g., DAY, MONTH, YEAR).

˜timestamp_expression`: A string-formatted timestamp ('YYYY-MM-DD HH24:MI:SS').

Examples

```
SELECT DATE_ADD('2022-01-01', 2) -- 2022-01-03
SELECT DATE_ADD(DATE '2022-01-01', 30) -- 2022-01-31
SELECT DATE_ADD('2022-01-01', CAST(2 AS INTERVAL DAY)) -- 2022-01-03 00:00:00.000
SELECT  DATE_ADD('2022-01-01  12:00:00',  CAST(30  AS  INTERVAL  DAY))  --  2022-01-31
00:00:00.000
```

DATE_DIFF

Returns the difference between two dates or timestamps.

Syntax

```
DATE_DIFF(date_expression DATE, days INTEGER) → DATE
DATE_DIFF(date_expression DATE, date_expression DATE) → INTERVAL DAY
DATE_DIFF(timestamp_expression TIMESTAMP, timestamp_expression TIMESTAMP) → INTERVAL DAY
DATE_DIFF(time_expression TIME, time_interval INTERVAL) → TIME
```

Parameters

˜date_expression`: The date to subtract from or subtract days from.

˜days`: Number of days to subtract.

˜timestamp_expression`: The timestamp to subtract from.

˜time_expression`: The time to subtract from.

˜time_interval`: Interval to subtract.

Examples

```
SELECT DATE_DIFF(DATE '2022-01-01', 5) -- 2021-12-27
SELECT DATE_DIFF(DATE '2022-04-01', DATE '2022-01-01') -- +090 00:00:00.000
SELECT  DATE_DIFF(TIMESTAMP  '2022-04-01  12:35:23',  TIMESTAMP  '2022-01-01  01:00:00')  --
+090 11:35:23.000
SELECT DATE_DIFF(TIME '12:00:00', CAST(30 AS INTERVAL SECOND)) -- 11:59:30
```

DATE_PART

Extracts a subfield from a date or timestamp.

Syntax

```
DATE_PART(field string, source date or timestamp) → integer
```

Parameters

˜field`: Must be one of: YEAR, QUARTER, MONTH, WEEK, DAY, HOUR, MINUTE, SECOND.

˜source`: The value from which to extract the subfield

Examples

```
select date_part('year', timestamp '2021-04-01 15:27:32') -- 2021
select date_part('month', date '2021-04-01') -- 4
```

DATE_SUB

Subtracts a specified number of days or a time interval from a date or timestamp.

Syntax

```
DATE_SUB(date_expression STRING, days INTEGER) → DATE
DATE_SUB(date_expression DATE, days INTEGER) → DATE
DATE_SUB(date_expression STRING, time_interval INTERVAL) → TIMESTAMP
DATE_SUB(date_expression DATE, time_interval INTERVAL) → TIMESTAMP
DATE_SUB(timestamp_expression STRING, time_interval INTERVAL) → TIMESTAMP
```

Parameters

˜date_expression`: A string-formatted date ('YYYY-MM-DD') or a DATE column/literal.

˜days`: The number of days to subtract.

˜time_interval`: A CAST of a number to an interval.

˜timestamp_expression`: A string-formatted timestamp.

Examples

```
SELECT DATE_SUB('2022-01-01', 2) -- 2021-12-30
SELECT DATE_SUB(DATE '2022-01-01', 30) -- 2021-12-02
SELECT DATE_SUB('2022-01-01', CAST(2 AS INTERVAL DAY)) -- 2021-12-30 00:00:00.000
```

```
SELECT  DATE_SUB('2022-01-01  12:00:00',  CAST(30  AS  INTERVAL  DAY))  --  2021-12-02
00:00:00.000
```

DATE_TRUNC

Truncates a date or timestamp to the specified time unit.

Syntax

```
DATE_TRUNC(time_unit LITERAL, date_timestamp_expression DATE OR TIMESTAMP) → DATE
```

Parameters

`time_unit`: 'YEAR', 'MONTH', 'DAY', 'HOUR', 'MINUTE', or 'SECOND'.

`date_timestamp_expression`: The date or timestamp to truncate.

Examples

```
SELECT DATE_TRUNC('MONTH', '2021-12-24') -- 2021-12-01
SELECT  DATE_TRUNC('MINUTE',  CAST('2021-12-24  12:28:33'  as  TIMESTAMP))  --  2021-12-24
12:28:00
SELECT DATE_TRUNC('HOUR', '2021-12-24 12:28:33') -- 2021-12-24
```

DAY

Returns the day of the month from a date or timestamp.

Syntax

```
DAY(date_timestamp_expression string) → bigint
```

Parameters

`date_timestamp_expression`: A DATE or TIMESTAMP expression.

Examples

```
SELECT "DAY"('2003-02-01 11:43:22') -- 1
```

DAYOFMONTH

Returns the day of the month from a date or timestamp.

Syntax

```
DAYOFMONTH(date_timestamp_expression string) → bigint
```

Parameters

`date_timestamp_expression`: A DATE or TIMESTAMP expression.

Examples

```
SELECT DAYOFMONTH(DATE '2021-02-28') -- 28
SELECT DAYOFMONTH(TIMESTAMP '2021-02-28 11:43:22') -- 28
```

DAYOFWEEK

Returns the day of the week from a date or timestamp.

Syntax

```
DAYOFWEEK(date_timestamp_expression string) → bigint
```

Parameters

`date_timestamp_expression`: A DATE or TIMESTAMP expression.

Examples

```
SELECT DAYOFWEEK(DATE '2021-02-28') -- 1
SELECT DAYOFWEEK(TIMESTAMP '2021-02-27 11:43:22') -- 7
```

DAYOFYEAR

Returns the day of the year from a date or timestamp.

Syntax

```
DAYOFYEAR(date_timestamp_expression string) → bigint
```

Parameters

`date_timestamp_expression`: A DATE or TIMESTAMP expression.

Examples

```
SELECT DAYOFYEAR(DATE '2021-02-28') -- 59
SELECT DAYOFYEAR(TIMESTAMP '2021-03-15 11:43:22') -- 74
```

DEGREES

Converts radians to degrees.

Syntax

```
DEGREES(numeric_expression NUMERIC) → FLOAT
```

Parameters

`numeric_expression`: The number of radians. This must be an DOUBLE, INTEGER, or FLOAT.

Examples

```
SELECT DEGREES(PI()) -- 180.0
SELECT DEGREES(0) -- 0.0
SELECT DEGREES(1) -- 57.29577951308232
```

DENSE_RANK

Returns the rank of a value in a group of values.

Syntax

```
DENSE_RANK() OVER ( [PARTITION BY partition_expression] [ORDER BY order_expression]) →
bigint
```

Parameters

`partition_expression` (optional): An expression that groups rows into partitions.

`order_expression`: An expression that specifies the order of the rows within each partition.

Examples

```
SELECT "Category", "Descript", "DayOfWeek", DENSE_RANK() OVER ( PARTITION BY "Category"
ORDER BY "DayOfWeek") FROM Samples."samples.dremio.com"."SF_incidents2016.json"
-- Category, Descript, DayOfWeek, EXPR$3
-- ARSON, ARSON, Friday, 1
-- ARSON, ARSON, Monday, 2
```

E

Returns the base of the natural logarithm (e).

Syntax

```
E() → float
```

Examples

```
SELECT E() -- 2.718281828459045
```

EXP

Returns e raised to the power of a specified number.

Syntax

```
EXP(numeric_expression NUMERIC) → FLOAT
```

Parameters

`numeric_expression`: The exponent value to raise e to. This must be an DOUBLE, INTEGER, or FLOAT.

Examples

```
SELECT EXP(1) -- 2.718281828459045
SELECT EXP(10.0) -- 22026.465794806718
```

EXTRACT

Extracts a part of a date, time, or timestamp.

Syntax

```
EXTRACT(time_unit KEYWORD, date_time_expression DATE, TIME, TIMESTAMP) → INTEGER
```

Parameters

`time_unit`: The time unit to extract (EPOCH, YEAR, MONTH, DAY, HOUR, MINUTE, SECOND).

`date_time_expression`: The date, time, or timestamp.

Examples

```
SELECT EXTRACT(HOUR FROM CAST('05:33:44' AS TIME)) -- 5
SELECT EXTRACT(MONTH FROM CAST('2021-03-22 05:33:44.2' AS TIMESTAMP)) -- 3
SELECT EXTRACT(SECOND FROM CAST('2021-03-22 05:33:44.2' AS TIMESTAMP)) -- 44
SELECT EXTRACT(YEAR FROM CAST('2021-03-22' AS DATE)) -- 2021
SELECT EXTRACT(EPOCH FROM CAST('2021-03-22 05:33:44.2' AS TIMESTAMP)) -- 1616391224
SELECT EXTRACT(EPOCH FROM CAST('2021-03-22' AS DATE)) -- 1616371200
```

FIRST_VALUE

Returns the first value in an ordered set of values.

Syntax

```
FIRST_VALUE(expression VARCHAR, order_subclause VARCHAR) → VARCHAR
```

Parameters

`expression`: The expression that determines the return value.

`order_subclause`: A subclause that specifies the order of the rows within each partition of the result set.

Examples

```
SELECT city, state, pop, FIRST_VALUE(pop) OVER (PARTITION BY state ORDER BY city) FROM Samples."samples.dremio.com"."zips.json";
```

FLATTEN

Explodes a compound value into multiple rows.

Syntax

```
FLATTEN(expression list) → list
```

Parameters

`expression`: The expression that will be unpacked into rows. The expression must be of data type LIST.

Examples

```
SELECT FLATTEN(CONVERT_FROM ('["Ford", "BMW", "Fiat"]', 'json'))
-- Ford
-- BMW
-- Fiat
```

FLOOR

Returns the largest integer not greater than the argument.

Syntax

```
FLOOR(numeric_expression NUMERIC) → INTEGER
```

Parameters

`numeric_expression`: The number (DOUBLE, FLOAT, INTEGER) for which you want to compute the floor.

Examples

```
SELECT FLOOR(0) -- 0
SELECT FLOOR(45.76) -- 45
SELECT FLOOR(-1.3) -- -2
```

FROM_HEX

Converts a hexadecimal string to a binary value.

Syntax

```
FROM_HEX(in string) → binary
```

Parameters

`in`: A hexadecimal string

Examples

```
select from_hex('3fd98a3c') -- P9mKPA==
```

GREATEST

Returns the largest value from a list of expressions.

Syntax

```
GREATEST(expression) → same as input type
```

Parameters

`expression`: The arguments must include at least one expression. All the expressions should be of the same type or compatible types.

Examples

```
SELECT GREATEST(1, 5, 3, 8) -- 8
```

HASH

Computes a hash value for an expression.

Syntax

```
HASH(expression any) → numeric
```

Parameters

`expression`: Can be a general expression of any Dremio-supported data type.

Examples

```
SELECT         HASH(host_id)         FROM         "Samples"."samples.dremio.com"."Dremio
University"."airbnb_listings.csv" LIMIT 5
```

HASH64

Computes a 64-bit hash value for an expression.

Syntax

```
HASH64(value ANY [, seed BIGINT]) → BIGINT
```

Parameters

`value`: Input value for hash calculation.

`seed` (optional): Optional seed for hash calculation.

Examples

```
SELECT HASH64('abc') -- -5434086359492102041
SELECT HASH64(5.127) -- -1149762993205326574
SELECT HASH64(null) -- 0
SELECT HASH64('abc',123) -- 1489494923063836066
```

HEX

Returns the hexadecimal representation of a value.

Syntax

```
HEX(expression any primitive) → varchar
```

Parameters

`expression`: The expression to encode.

Examples

```
SELECT HEX('Dremio') -- 4472656D696F
SELECT HEX(2023) -- 7E7
```

HOUR

Extracts the hour from a time, timestamp, or date.

Syntax

```
EXTRACT(HOUR FROM date_timestamp_expression string) → bigint
```

Parameters

`date_timestamp_expression`: A TIME, TIMESTAMP, or DATE expression.

Examples

```
SELECT EXTRACT(HOUR FROM TIMESTAMP '2019-08-12 01:10:30.123456') -- 1
SELECT EXTRACT(HOUR FROM TIME '01:10:30.123456') -- 1
SELECT EXTRACT(HOUR FROM CAST('2019-08-12 01:10:30' AS TIMESTAMP)) -- 1
```

IFNULL

Alias for COALESCE or NVL.

ILIKE

Compares two strings for equality, ignoring case.

Syntax

```
ILIKE(expression varchar, pattern varchar) → boolean
ILIKE(expression varchar, pattern varchar, escape_character varchar) → boolean
```

Parameters

`expression`: The expression to compare.

`pattern`: The pattern that is compared to the expression.

`escape_character`: Putting escape_character before a wildcard in pattern makes ILIKE treat the wildcard as a regular character when it appears in expression.

Examples

```
SELECT ILIKE ('pancake', '%Cake') -- True
SELECT ILIKE ('50%_Off', '%50!%%','!') -- True
```

INITCAP

Returns the string with the first letter of each word in uppercase and all other letters in lowercase.

Syntax

```
INITCAP(expression varchar) → varchar
```

Parameters

`expression`: Input string.

Examples

```
SELECT INITCAP('a guide to data lakehouses') -- A Guide To Data Lakehouses
SELECT INITCAP('a guide to data lakeHouses') -- A Guide To Data Lakehouses
```

IS_MEMBER

Checks if the current user is a member of a specific role.

Syntax

```
IS_MEMBER(expression varchar) → boolean
```

Parameters

`expression`: String expression identfying a role in Dremio.

Examples

```
SELECT IS_MEMBER ('public') -- True
SELECT IS_MEMBER ('non-role') -- False
```

IS_UTF8

Checks if a binary string is a valid UTF-8 string.

Syntax

```
IS_UTF8(in any) → boolean
```

Parameters

`in`: an expression

Examples

```
select is_utf8('hello') -- True
```

IS_VARCHAR

Checks if an expression is a VARCHAR.

Syntax

```
IS_VARCHAR(expression any) → boolean
```

Parameters

`expression`: Input expression.

Examples

```
SELECT IS_VARCHAR(column_name) -- True
```

LAG

Returns the value of a column at a specified offset before the current row.

Syntax

```
LAG(expression, [offset]) OVER ([PARTITION BY partition_expression] [ORDER BY
order_expression]) → same as input type
```

Parameters

`expression` (optional): An expression that is returned.

`offset` (optional): The number of rows before the current row from which to obtain a value.

Examples

```
SELECT "Category", "Descript", "DayOfWeek", LAG(DayOfWeek, 3) OVER ( PARTITION BY "Category" ORDER BY "DayOfWeek") FROM Samples."samples.dremio.com"."SF_incidents2016.json"
-- Category, Descript, DayOfWeek, EXPR$3
-- ARSON, ARSON, Friday, null
-- ARSON, ARSON, Friday, null
-- ARSON, ARSON OF AN INHABITED DWELLING, Friday, null
-- ARSON, ARSON, Friday, Friday
```

LAST_DAY

Returns the last day of the month for a given date or timestamp.

Syntax

```
LAST_DAY(date_timestamp_expression string) → date
```

Parameters

˜date_timestamp_expression`: A DATE or TIMESTAMP expression.

Examples

```
SELECT LAST_DAY('2009-01-12 12:58:59') -- 2009-01-31
```

LAST_VALUE

Returns the last value in an ordered set of values.

Syntax

```
LAST_VALUE(expression VARCHAR, order_subclause VARCHAR) → VARCHAR
```

Parameters

˜expression`: The expression that determines the return value.

˜order_subclause`: A subclause that specifies the order of the rows within each partition of the result set.

Examples

```
SELECT city, state, pop, LAST_VALUE(pop) OVER (PARTITION BY state ORDER BY city) FROM
```

```
Samples."samples.dremio.com"."zips.json"
```

LCASE

Returns the string in lowercase.

Syntax

```
LCASE(expression varchar) → varchar
```

Parameters

`expression`: String to convert to lowercase.

Examples

```
SELECT LCASE('A GUIDE to data Lakehouses') -- a guide to data lakehouses
```

LEAD

Returns the value of a column at a specified offset after the current row.

Syntax

```
LEAD(expression, [offset]) OVER ([PARTITION BY partition_expression] [ORDER BY order_expression]) → same as input type
```

Parameters

`expression` (optional): An expression that is returned.

`offset` (optional): The number of rows after the current row from which to obtain a value.

Examples

```
SELECT "Category", "Descript", "DayOfWeek", LEAD(DayOfWeek, 3) OVER ( PARTITION BY "Category" ORDER BY "DayOfWeek") FROM Samples."samples.dremio.com"."SF_incidents2016.json"
```

LEFT

Returns the specified number of characters from the left of a string.

Syntax

```
LEFT(expression varchar, length int64) → varchar
```

Parameters

`expression`: String input parameter

`length`: Number of characters on the left to return.

Examples

```
SELECT "LEFT"('Dremio - SQL Engine', -12) -- Dremio
SELECT "LEFT"('Dremio - SQL Engine', 6) -- Dremio
```

LENGTH

Returns the length of a string.

Syntax

```
LENGTH([expression varchar]) → int32
```

Parameters

`expression` (optional): String expression to determine the length of.

Examples

```
SELECT LENGTH('DREMIO') -- 6
```

LISTAGG

Concatenates the values of a column for each group.

Syntax

```
LISTAGG ( [ALL | DISTINCT] measure_expr [, 'delimiter'] ) [WITHIN GROUP ( ORDER BY
measure_expr [ASC | DESC] )]
```

Parameters

˜ALL` (optional): Keeps duplicate values in the return list. This is the default behavior.

˜DISTINCT` (optional): Removes duplicate values from the return list.

˜measure_expr`: A string column or value.

˜delimiter` (optional): Designates a string literal to separate the measure column values. If a delimiter is not specified, will default to NULL.

˜WITHIN GROUP` (optional): Determines the order in which the concatenated values are returned.

Examples

```
SELECT LISTAGG(city, '; ') FROM "Samples"."samples.dremio.com"."zips.json"
```

LN

Returns the natural logarithm of a number.

Syntax

```
LN(numeric_expression double) → float8
```

Parameters

˜numeric_expression`: A number greater than 0.

Examples

```
SELECT LN(0), LN(.1525), LN(1), LN(5.35), LN(5269853105789632584), LN(-1)
-- null, -1.8805906829346708, 0.0, 1.6770965609079151, 43.10853416239341, null
```

LOCALTIME

Returns the current time in the local time zone.

Syntax

```
LOCALTIME() → time
```

Examples

```
SELECT LOCALTIME() -- 05:07:01
```

LOCALTIMESTAMP

Returns the current timestamp in the local time zone.

Syntax

```
LOCALTIMESTAMP() → timestamp
```

Examples

```
SELECT LOCALTIMESTAMP() -- 2021-06-29 05:17:44.703000
```

LOCATE

Returns the position of the first occurrence of a substring in a string.

Syntax

```
LOCATE(substring varchar, expression varchar [, start int32]) → int32
```

Parameters

`substring`: Substring to search for in the expression.

`expression`: The input expression to search.

`start` (optional): Position to start the search from.

Examples

```
SELECT LOCATE('no','banana') -- 0
SELECT LOCATE('an','banana') -- 2
SELECT LOCATE('an','banana', 3) -- 4
```

LOG

Returns the logarithm of a number.

Syntax

```
LOG([base_expression float], expression float) → double
LOG([base_expression double], expression double) → double
LOG(expression int64) → double
LOG([base_expression int64], expression int64) → double
LOG(expression int32) → double
LOG([base_expression int32], expression int32) → double
LOG(expression float) → double
LOG(expression double) → double
```

Parameters

`base_expression` (optional): The base to use.

`expression`: The value for which you want to calculate the log.

Examples

```
SELECT LOG(20.5, 1.5) -- 0.1342410830900514
SELECT LOG(10) -- 2.302585092994046
SELECT LOG(10, 2) -- 0.30102999566398114
SELECT LOG(12.5) -- 2.5257286443082556
```

LOG10

Returns the base-10 logarithm of a number.

Syntax

```
LOG10(expression double) → double
LOG10(expression int64) → double
LOG10(expression int32) → double
LOG10(expression float) → double
```

Parameters

`expression`: The value for which you want to calculate the log.

Examples

```
SELECT LOG10(20.5) -- 1.3117538610557542
SELECT LOG10(100) -- 2.0
```

LOWER

Returns the string in lowercase.

Syntax

```
LOWER(expression varchar) → varchar
```

Parameters

˜expression`: String to convert to lowercase.

Examples

```
SELECT LOWER('A GUIDE to data Lakehouses') -- a guide to data lakehouses
```

LPAD

Pads a string on the left with a specified character.

Syntax

```
LPAD(base_expression varchar, length int64) → varchar
LPAD(base_expression varchar, length int64 [, pad_expression varchar]) → varchar
```

Parameters

˜base_expression`: The expression to pad.

˜length`: The number of characters to return.

˜pad_expression` (optional): Characters to pad the base_expression with.

Examples

```
SELECT LPAD('parameter', 11) -- parameter
SELECT LPAD('engineering', 6) -- engine
select LPAD('parameter', 11, '-') -- --parameter
```

LTRIM

Removes leading characters from a string.

Syntax

```
LTRIM(expression varchar, trim_expression varchar) → varchar
```

Parameters

`expression`: The expression to be trimmed.

`trim_expression`: Leading characters to trim. If this parameter is not specified, then spaces will be trimmed from the input expression.

Examples

```
SELECT LTRIM('pancake', 'pan') -- cake
SELECT LTRIM('pancake', 'abnp') -- cake
SELECT LTRIM(' dremio') -- dremio
```

MASK_FIRST_N

Masks the first N characters of a string.

Syntax

```
MASK_FIRST_N(expression varchar [, num_chars int] [, uc_mask varchar] [, lc_mask varchar] [, num_mask varchar]) → varchar
```

Parameters

`expression`: The string to mask.

`num_chars` (optional): The number of characters to mask.

`uc_mask` (optional): Controls the mask character for upper case letters.

`lc_mask` (optional): Controls the mask character for lower case letters.

`num_mask` (optional): Controls the mask character for numbers.

Examples

```
SELECT MASK_FIRST_N('abcd-ABCD-1234') -- xxxx-ABCD-1234
SELECT MASK_FIRST_N('abcd-ABCD-1234', 2) -- xxcd-ABCD-1234
SELECT MASK_FIRST_N('Aa12-ABCD-1234', 4, 'U', 'u', '#') -- Uu##-ABCD-1234
SELECT MASK_FIRST_N('abcd-ABCD-1234', 7, '', 'u', '') -- uuuu-XXCD-1234
```

MASK_LAST_N

Masks the last N characters of a string.

Syntax

```
MASK_LAST_N(expression varchar [, num_chars int] [, uc_mask varchar] [, lc_mask varchar] [, num_mask varchar]) → varchar
```

Parameters

˜expression`: The string to mask.

˜num_chars` (optional): The number of characters to mask.

˜uc_mask` (optional): Controls the mask character for upper case letters.

˜lc_mask` (optional): Controls the mask character for lower case letters.

˜num_mask` (optional): Controls the mask character for numbers.

Examples

```
SELECT MASK_LAST_N('abcd-ABCD-1234') -- abcd-ABCD-nnnn
SELECT MASK_LAST_N('abcd-ABCD-1234', 2) -- abcd-ABCD-12nn
SELECT MASK_LAST_N('abcd-ABCD-Aa12', 4, 'U', 'u', '#') -- abcd-ABCD-Uu##
SELECT MASK_LAST_N('abcd-ABCD-1234', 7, '', 'u', '') -- abcd-ABXX-nnnn
```

MASK_SHOW_FIRST_N

Masks all but the first N characters of a string.

Syntax

```
MASK_SHOW_FIRST_N(expression varchar [, num_chars int] [, uc_mask varchar] [, lc_mask varchar] [, num_mask varchar]) → varchar
```

Parameters

˜expression`: The string to mask.

˜num_chars` (optional): The number of characters to unmask.

˜uc_mask` (optional): Controls the mask character for upper case letters.

˜lc_mask` (optional): Controls the mask character for lower case letters.

˜num_mask` (optional): Controls the mask character for numbers.

Examples

```
SELECT MASK_SHOW_FIRST_N('abcd-ABab-1234') -- abcd-XXxx-nnnn
SELECT MASK_SHOW_FIRST_N('abcd-ABab-1234', 2) -- abxx-XXxx-nnnn
SELECT MASK_SHOW_FIRST_N('Aa12-ABab-1234', 4, 'U', 'u', '#') -- Aa12-UUuu-####
SELECT MASK_SHOW_FIRST_N('abcd-ABCD-1234', 2, '', 'u', '') -- abuu-XXXX-nnnn
```

MASK_SHOW_LAST_N

Masks all but the last N characters of a string.

Syntax

```
MASK_SHOW_LAST_N(expression varchar [, num_chars int] [, uc_mask varchar] [, lc_mask varchar] [, num_mask varchar]) → varchar
```

Parameters

˜expression`: The string to mask.

˜num_chars` (optional): The number of characters to unmask.

˜uc_mask` (optional): Controls the mask character for upper case letters.

˜lc_mask` (optional): Controls the mask character for lower case letters.

˜num_mask` (optional): Controls the mask character for numbers.

Examples

```
SELECT MASK_SHOW_LAST_N('ab12-ABab-1234') -- xxnn-XXxx-1324
SELECT MASK_SHOW_LAST_N('ab12-ABab-1234', 2) -- xxnn-XXxx-nn34
SELECT MASK_SHOW_LAST_N('Aa12-ABab-1234', 4, 'U', 'u', '#') -- Uu##-UUuu-1234
SELECT MASK_SHOW_LAST_N('abcd-ABCD-1234', 2, '', 'u', '') -- uuuu-XXXX-nn34
```

MAX

Returns the maximum value of an expression across all rows.

Syntax

```
MAX(expression NUMERIC) → NUMERIC
```

Parameters

`expression`: The expression from which to take the maximum value, across all rows.

Examples

```
SELECT  MAX("total_amount")  FROM  "Samples"."samples.dremio.com"."NYC-taxi-trips";  --
685908.1
```

MD5

Computes the MD5 hash of a string.

Syntax

```
MD5(expression varchar) → varchar
```

Parameters

`expression`: The string to hash.

Examples

```
SELECT MD5('Dremio') -- 288e0e9ab8b8ac8737afefecf16f61fd
```

MEDIAN

Computes the median value of a numeric column.

Syntax

```
MEDIAN(num_col numeric) → double precision
```

Parameters

`num_col`: A numeric column whose median value you want to compute.

Examples

```
SELECT MEDIAN(pop) FROM Samples."samples.dremio.com"."zips.json" -- 2783.0
```

MIN

Returns the minimum value of an expression across all rows.

Syntax

```
MIN(expression NUMERIC) → NUMERIC
```

Parameters

`expression`: The expression from which to take the minimum value, across all rows.

Examples

```
SELECT  MIN("total_amount")  FROM  "Samples"."samples.dremio.com"."NYC-taxi-trips";  --
-1430.0
```

MOD

Returns the remainder of a division.

Syntax

```
MOD(numeric_expression int64, numeric_expression int64) → int64
MOD(numeric_expression int64, numeric_expression int32) → int32
MOD(numeric_expression decimal(0,0), numeric_expression decimal(0,0)) → decimal(0,0)
```

Parameters

`numeric_expression`: The dividend.

`numeric_expression`: The divisor.

Examples

```
SELECT MOD(50, 7) -- 1
SELECT MOD(35, 5) -- 0
SELECT MOD(47.6, 5.2) -- 0.8
```

MONTH

Extracts the month from a time, timestamp, or date.

Syntax

```
EXTRACT(MONTH FROM date_timestamp_expression string) → bigint
```

Parameters

`date_timestamp_expression`: A DATE or TIMESTAMP expression.

Examples

```
SELECT EXTRACT(MONTH FROM TIMESTAMP '2019-08-12 01:00:00.123456') -- 8
SELECT EXTRACT(MONTH FROM DATE '2019-08-12') -- 8
SELECT EXTRACT(MONTH FROM CAST('2019-08-12 01:00:00' AS TIMESTAMP)) -- 8
```

MULTIPLY

Use the `*` operator for multiplication.

NEAR

Not a supported function.

NEXT_DAY

Returns the date of the first specified day of the week that occurs after the input date.

Syntax

```
NEXT_DAY(date_timestamp_expression string, day_of_week string) → date
```

Parameters

`date_timestamp_expression`: A DATE or TIMESTAMP expression.

`day_of_week`: A string expression identifying a day of the week (e.g., 'SU', 'SUN', 'SUNDAY').

Examples

```
SELECT NEXT_DAY('2015-01-14 12:05:55', 'TU') -- 2015-01-20
```

NOW

Returns the current timestamp.

Syntax

```
NOW() → timestamp
```

Examples

```
SELECT NOW() -- 2021-07-02 04:55:55.267000
```

NTILE

Divides an ordered data set into a number of buckets indicated by `buckets` and assigns the appropriate bucket number to each row.

Syntax

```
NTILE(buckets) OVER (PARTITION BY partition_expression ORDER BY order_expression) → int
```

Parameters

˜buckets`: A positive integer literal.

˜partition_expression` (optional): An expression that groups rows into partitions.

˜order_expression`: An expression that specifies the order of the rows within each partition.

Examples

```
SELECT "Category", "Descript", "DayOfWeek", NTILE(1) OVER ( PARTITION BY "Category" ORDER BY "DayOfWeek") FROM Samples."samples.dremio.com"."SF_incidents2016.json"
```

NULLIF

Returns NULL if the two expressions are equal, otherwise returns the first expression.

Syntax

```
NULLIF(expression1, expression2) → same as input type
```

Parameters

˜expression1`: The first expression.

˜expression2`: The second expression.

Examples

```
SELECT NULLIF(user_id, customer_id)
```

NVL

Returns the first non-null expression. Alias for COALESCE.

Syntax

```
NVL(expression1, expression2) → same as input type
```

Parameters

˜expression1`: The first expression.

˜expression2`: The second expression.

Examples

```
SELECT NVL(NULL, 2) -- 2
SELECT NVL(5, 2) -- 5
```

NVL2

Not supported. Use `CASE` or `NVL`/`COALESCE`.

OCTET_LENGTH

Returns the length of a string in bytes (octets).

Syntax

```
OCTET_LENGTH(input varchar) → int32
```

Parameters

˜input`: The string for which the length is returned.

Examples

```
SELECT OCTET_LENGTH('abc') -- 3
```

OVERLAY

Not supported. Use `SUBSTR` or `SUBSTRING`.

PERCENT_RANK

Calculates the percent rank of a value in a group of values.

Syntax

```
PERCENT_RANK() OVER ( [PARTITION BY partition_expression] [ORDER BY order_expression]) → double
```

Parameters

˜partition_expression` (optional): An expression that groups rows into partitions.

˜order_expression`: An expression that specifies the order of the rows within each partition.

Examples

```
SELECT "Category", "Descript", "DayOfWeek", PERCENT_RANK() OVER ( PARTITION BY "Category" ORDER BY "DayOfWeek") FROM Samples."samples.dremio.com"."SF_incidents2016.json"
```

PI

Returns the value of PI.

Syntax

```
PI() → double
```

Examples

```
SELECT PI() -- 3.141592653589793
```

POSITION

Returns the position of a substring within a string.

Syntax

```
POSITION(substr string IN expression string) → integer
```

Parameters

˜substr`: The substring to search for in the expression.

˜expression`: The input expression to search.

Examples

```
select position('an' in 'banana') -- 2
select position('no' in 'banana') -- 0
```

POW

Alias for POWER.

POWER

Returns the value of a number raised to a power.

Syntax

```
POWER(numeric_expression, power) → double
```

Parameters

˜numeric_expression`: The base number.

˜power`: The exponent.

Examples

```
SELECT POWER(5, 2) -- 25.0
SELECT POWER(10, -2) -- 0.01
```

QUARTER

Extracts the quarter from a time, timestamp, or date.

Syntax

```
EXTRACT(QUARTER FROM date_timestamp_expression string) → bigint
```

Parameters

˜date_timestamp_expression`: A DATE or TIMESTAMP expression.

Examples

```
SELECT EXTRACT(QUARTER FROM TIMESTAMP '2019-08-12 01:00:00.123456') -- 3
SELECT EXTRACT(QUARTER FROM DATE '2019-08-12') -- 3
SELECT EXTRACT(QUARTER FROM CAST('2019-08-12 01:00:00' AS TIMESTAMP)) -- 3
```

RADIANS

Converts degrees to radians.

Syntax

```
RADIANS(x number) → float
```

Parameters

˜x`: The number in degrees.

Examples

```
select radians(45) -- 0.7853981633974483
```

RAND

Alias for RANDOM.

RANK

Returns the rank of a value in a group of values.

Syntax

```
RANK() OVER ( [PARTITION BY partition_expression] [ORDER BY order_expression]) → bigint
```

Parameters

`partition_expression` (optional): An expression that groups rows into partitions.

`order_expression`: An expression that specifies the order of the rows within each partition.

Examples

```
SELECT "Category", "Descript", "DayOfWeek", RANK() OVER ( PARTITION BY "Category" ORDER BY "DayOfWeek") FROM Samples."samples.dremio.com"."SF_incidents2016.json"
```

REGEXP_COL_LIKE

Matches a string against a regular expression contained in a column.

Syntax

```
REGEXP_COL_LIKE(input string, regex string) → boolean
```

Parameters

`input`: The string to test.

`regex`: The column containing the Perl-compatible regular expression (PCRE) to use for the test.

Examples

```
SELECT Category, REGEXP_COL_LIKE('WARRANTS', Category) FROM Samples."samples.dremio.com"."SF_incidents2016.json" LIMIT 3
```

REGEXP_LIKE

Matches a string against a regular expression.

Syntax

```
REGEXP_LIKE(input string, regex string) → boolean
```

Parameters

˜input`: The string to test.

˜regex`: The Perl-compatible regular expression (PCRE) to use for the test. Must be a literal.

Examples

```
SELECT REGEXP_LIKE('the data lakehouse', '.*?\\Qlake\\E.*?') -- True
```

REGEXP_MATCH

Alias for REGEXP_MATCHES or REGEXP_LIKE.

REGEXP_REPLACE

Replaces substrings matching a regular expression.

Syntax

```
REGEXP_REPLACE(input string, regex string, replacement_string string) → string
```

Parameters

˜input`: The expression to search for a matching string.

˜regex`: The Perl-compatible regular expression (PCRE) to match against.

˜replacement_string`: The string with which to replace the matching string.

Examples

```
SELECT REGEXP_REPLACE('8AM-4PM', '\\Q-\\E', ' to ') -- 8AM to 4PM
```

REGEXP_SPLIT

Splits a string using a regular expression.

Syntax

```
REGEXP_SPLIT(input string, regex string, keyword string, integer integer) → array
```

Parameters

`input`: The string that you want to split by means of the regular expression.

`regex`: The regular expression to use to split the string.

`keyword`: The keyword that determines where or how many times to use the regular expression to split the string. Can be FIRST, LAST, INDEX, or ALL.

`integer`: The value specified for the keyword.

Examples

```
SELECT REGEXP_SPLIT('REGULAR AIR', 'R', 'FIRST', -1) AS R_LESS_SHIPMENT_TYPE -- ['', 'EGULAR AIR']
```

REPEAT

Repeats a string a specified number of times.

Syntax

```
REPEAT(expression varchar, nTimes int32) → varchar
```

Parameters

`expression`: The input string from which the output string is built.

`nTimes`: The number of times the input expression should be repeated.

Examples

```
SELECT REPEAT('abc', 3) -- abcabcabc
```

REPLACE

Replaces all occurrences of a specified string.

Syntax

```
REPLACE(string_expression varchar, pattern varchar, replacement varchar) → varchar
```

Parameters

`string_expression`: String expression in which to do the replacements.

`pattern`: The substring you want replaced in the string_expression.

`replacement`: The string to replace the occurrences of the pattern substring with.

Examples

```
SELECT REPLACE('THE CATATONIC CAT', 'CAT', 'DOG')
```

REVERSE

Reverses a string.

Syntax

```
REVERSE(expression varchar) → varchar
```

Parameters

`expression`: The string to reverse.

Examples

```
SELECT REVERSE('Hello, world!'); -- !dlrow ,olleH
```

RIGHT

Returns the specified number of characters from the right of a string.

Syntax

```
RIGHT(string varchar, length int64) → varchar
```

Parameters

`string`: String input parameter.

`length`: Number of characters on the right to return.

Examples

```
SELECT "RIGHT"('Dremio - SQL Engine', 6) -- Engine
```

ROUND

Rounds a number to a specified number of decimal places.

Syntax

```
ROUND(numeric_expression decimal(0,0), scale int32) → decimal(0,0)
ROUND(numeric_expression int32, scale int32) → int32
ROUND(numeric_expression int32) → int32
ROUND(numeric_expression double) → double
```

Parameters

˜numeric_expression`: Numeric value to round.

˜scale`: The decimal place to round.

Examples

```
SELECT ROUND(-24.35, -1) -- -24.4
SELECT ROUND(24.35, 1) -- 24.4
SELECT ROUND(24, 0) -- 0
```

ROW_NUMBER

Returns the row number for the current row in a partition.

Syntax

```
ROW_NUMBER() OVER ( [PARTITION BY partition_expression] [ORDER BY order_expression]) →
bigint
```

Parameters

˜partition_expression` (optional): An expression that groups rows into partitions.

˜order_expression` (optional): An expression that specifies the order of the rows within each partition.

Examples

```
SELECT "Category", "Descript", "DayOfWeek", ROW_NUMBER() OVER ( PARTITION BY "Category"
ORDER BY "DayOfWeek") FROM Samples."samples.dremio.com"."SF_incidents2016.json"
```

RPAD

Right-pads a string with another string to a specified length.

Syntax

```
RPAD(base_expression varchar, length int64 [, pad_expression varchar]) → varchar
```

Parameters

`base_expression`: The expression to pad.

`length`: The number of characters to return.

`pad_expression` (optional): Characters to pad the base_expression with.

Examples

```
select RPAD('dremio', 9, '!') -- dremio!!!
select RPAD('base_', 9, 'expression') -- base_expr
select RPAD('dremio', 9) -- dremio
```

RTRIM

Removes trailing characters from a string.

Syntax

```
RTRIM(expression varchar [, trim_expression varchar]) → varchar
```

Parameters

`expression`: The expression to be trimmed.

`trim_expression` (optional): Trailing characters to trim. If this parameter is not specified, then spaces will be trimmed from the input expression.

Examples

```
SELECT RTRIM('pancake', 'cake') -- pan
SELECT RTRIM('pancake pan', 'abnp') -- pancake
SELECT RTRIM('dremio ') -- dremio
```

SEARCH

Not a direct function. Use LOCATE or POSITION.

SEC

Alias for SECOND.

SECOND

Extracts the second from a time, timestamp, or date.

Syntax

```
EXTRACT(SECOND FROM date_timestamp_expression string) → bigint
```

Parameters

˜timestamp_expression`: A TIME, TIMESTAMP, or DATE expression.

Examples

```
SELECT EXTRACT(SECOND FROM TIMESTAMP '2019-08-12 01:10:30.123456') -- 1
SELECT EXTRACT(SECOND FROM TIME '01:10:30.123456') -- 1
SELECT EXTRACT(SECOND FROM CAST('2019-08-12 01:10:30' AS TIMESTAMP)) -- 1
```

SHA1

Computes the SHA-1 hash of a string.

Syntax

```
SHA1(expression varchar) → varchar
```

Parameters

˜expression`: The string to hash.

Examples

```
SELECT SHA1('Dremio') -- dda3f1ef53d1e82a4845ef5b2893b9d9c04bd3b1
```

SHA256

Computes the SHA-256 hash of a string.

Syntax

```
SHA256(expression varchar) → varchar
```

Parameters

`expression`: The string to hash.

Examples

```
SELECT                                    SHA256('Dremio')                                    --
ffae26c65c486a4d9143cbb1a6829166f17dab711910fdfc5787b1a249bd9921
```

SIGN

Returns the sign of a number.

Syntax

```
SIGN(numeric_expression double) → int
SIGN(numeric_expression int32) → int32
SIGN(numeric_expression int64) → int64
SIGN(numeric_expression float) → int
```

Parameters

`numeric_expression`: Input expression.

Examples

```
SELECT SIGN(10.3) -- 1
SELECT SIGN(-5) -- -1
SELECT SIGN(24) -- 1
SELECT SIGN(0.0) -- 0
```

SIN

Returns the sine of a number (in radians).

Syntax

```
SIN(numeric_expression int32) → double
SIN(numeric_expression float) → double
SIN(numeric_expression int64) → double
SIN(numeric_expression double) → double
```

Parameters

`numeric_expression`: The number in radians.

Examples

```
SELECT SIN(360) -- 0.9589157234143065
SELECT SIN(510.89) -- 0.9282211721815067
```

SINH

Returns the hyperbolic sine of a number.

Syntax

```
SINH(numeric_expression int32) → double
SINH(numeric_expression float) → double
SINH(numeric_expression double) → double
SINH(numeric_expression int64) → double
```

Parameters

`numeric_expression`: Input expression.

Examples

```
SELECT SINH(1) -- 1.1752011936438014
SELECT SINH(1.5) -- 2.1292794550948173
```

SIZE

Returns the number of entries in a map.

Syntax

```
SIZE(input map) → int
```

Parameters

`input`: A map expression for which to return the number of entries.

Examples

```
SELECT SIZE(properties)
```

SPLIT_PART

Splits a string by a delimiter and returns the specified part.

Syntax

```
SPLIT_PART(expression varchar, delimiter varchar, part_number int32) → varchar
```

Parameters

`expression`: Input expression.

`delimiter`: String representing the delimiter to split the input expression by.

`part_number`: Requested part of the split. Must be an integer greater than zero.

Examples

```
SELECT SPLIT_PART('127.0.0.1', '.', 1) -- 127
```

SQRT

Returns the square root of a number.

Syntax

```
SQRT(numeric_expression double) → double
SQRT(numeric_expression int64) → int64
SQRT(numeric_expression int32) → int32
SQRT(numeric_expression float) → float
```

Parameters

`numeric_expression`: Numeric expression to calculate the square root for.

Examples

```
SELECT SQRT(25.25) -- 5.024937810560445
SELECT SQRT(-25.25) -- NaN
SELECT SQRT(25) -- 5
```

STDDEV

Returns the sample standard deviation of a numeric column.

Syntax

```
STDDEV(col_name NUMERIC) → DOUBLE
```

Parameters

`col_name`: The name of the column for which to return the standard deviation. The values in the column must be numbers, such as INT, DOUBLE, or FLOAT.

Examples

```
SELECT STDDEV(tip_amount) FROM Samples."samples.dremio.com"."NYC-taxi-trips"; --
2.2596650338662974
```

STDDEV_POP

Returns the population standard deviation of a numeric column.

Syntax

```
STDDEV_POP(col_name NUMERIC) → DOUBLE
```

Parameters

`col_name`: The name of the column for which to return the population standard deviation. The values in the column must be numbers, such as INT, DOUBLE, or FLOAT.

Examples

```
SELECT STDDEV_POP(tip_amount) FROM Samples."samples.dremio.com"."NYC-taxi-trips" --
2.259665030506379
```

STDDEV_SAMP

Returns the sample standard deviation of a numeric column.

Syntax

```
STDDEV_SAMP(col_name NUMERIC) → DOUBLE
```

Parameters

`col_name`: The name of the column for which to return the sample standard deviation. The values in the column must be numbers, such as INT, DOUBLE, or FLOAT.

Examples

```
SELECT STDDEV_SAMP(tip_amount) FROM Samples."samples.dremio.com"."NYC-taxi-trips" --
2.259665033866297
```

STRING_BINARY

Converts a binary value to a string.

Syntax

```
STRING_BINARY(bytes BYTES) → STRING
```

Parameters

`bytes`: Bytes to convert to a string.

Examples

```
SELECT STRING_BINARY(BINARY_STRING('Dremio')) -- Dremio
SELECT STRING_BINARY(FROM_HEX('54455354111213')) -- TEST\x11\x12\x13
```

SUBSTR

Extracts a substring from a string.

Syntax

```
SUBSTR(string_expression varchar, offset int64) → varchar
```

```
SUBSTR(string_expression varchar, offset int64, length int64) → varchar
SUBSTR(string_expression varchar, pattern varchar) → varchar
```

Parameters

`string_expression`: Base expression to extract substring from.

`offset`: The offset from which the substring starts.

`length` (optional): The length limit of the substring.

`pattern`: Regex pattern to match.

Examples

```
SELECT SUBSTR('dremio user 1 2 3', 12) -- 1 2 3
SELECT SUBSTR('base expression', 6, 4) -- expr
SELECT SUBSTR('dremio user 123', '[0-9]+') -- 123
```

SUBSTRING

Extracts a substring from a string.

Syntax

```
SUBSTRING(string_expression varchar, offset int64) → varchar
SUBSTRING(string_expression varchar FROM offset int64) → varchar
SUBSTRING(string_expression varchar, offset int64, length int64) → varchar
```

Parameters

`string_expression`: Base expression to extract substring from.

`offset`: The offset from which the substring starts.

`length` (optional): The length limit of the substring.

Examples

```
SELECT SUBSTRING('dremio user 1 2 3', 12) -- 1 2 3
SELECT SUBSTRING('dremio user 1 2 3' FROM 12) -- 1 2 3
SELECT SUBSTRING('base expression', 6, 4) -- expr
```

SUM

Returns the sum of values in a column.

Syntax

```
SUM(col_name NUMERIC) → same as input except for INT, which returns BIGINT
```

Parameters

`col_name`: The name of the column for which to return the sum. The values in the column must be numbers, such as INT, DOUBLE, or FLOAT.

Examples

```
SELECT SUM(trip_distance_mi) FROM Samples."samples.dremio.com"."NYC-taxi-trips"; --
9.858134477692287E8
```

TAN

Returns the tangent of a number (in radians).

Syntax

```
TAN(numeric_expression double) → double
TAN(numeric_expression int64) → double
TAN(numeric_expression int32) → double
TAN(numeric_expression float) → double
```

Parameters

`numeric_expression`: The number in radians.

Examples

```
SELECT TAN(180.8) -- -6.259341891872157
SELECT TAN(1200) -- -0.08862461268886584
```

TANH

Returns the hyperbolic tangent of a number.

Syntax

```
TANH(numeric_expression double) → double
TANH(numeric_expression int64) → double
TANH(numeric_expression float) → double
TANH(numeric_expression int32) → double
```

Parameters

˜numeric_expression`: Input expression to calculate tanh for.

Examples

```
SELECT TANH(1.5); -- 0.9051482536448664
SELECT TANH(1); -- 0.7615941559557649
```

TIMESTAMPADD

Adds a specified number of units to a timestamp.

Syntax

```
TIMESTAMPADD(unit symbol, count integer, givenTime date or timestamp) → date or timestamp
```

Parameters

˜unit`: The unit of the interval. Must be one of the following: YEAR, QUARTER, MONTH, WEEK, DAY, HOUR, MINUTE, SECOND.

˜count`: Number of units to be added (or subtracted) from givenTime. To subtract units, pass a negative number.

˜givenTime`: Value to which to add units (either a database column in DATE or TIMESTAMP format, or literal value explicitly converted to DATE or TIMESTAMP).

Examples

```
SELECT TIMESTAMPADD(DAY, 1, DATE '2021-04-01') -- 2021-04-02
SELECT TIMESTAMPADD(HOUR, -2, TIMESTAMP '2021-04-01 17:14:32') -- 2021-04-01 15:14:32
```

TIMESTAMPDIFF

Returns the difference between two timestamps in the specified unit.

Syntax

```
TIMESTAMPDIFF(unit symbol, giventime1 date or timestamp, givenTime2 date or timestamp) → integer
```

Parameters

`unit`: The unit of the interval. Must be one of the following: YEAR, QUARTER, MONTH, WEEK, DAY, HOUR, MINUTE, SECOND.

`giventime1`: The first DATE or TIMESTAMP (subtrahend).

`givenTime2`: The second DATE or TIMESTAMP (minuend).

Examples

```
SELECT TIMESTAMPDIFF(MONTH, DATE '2021-02-01', DATE '2021-05-01'); -- 3
SELECT TIMESTAMPDIFF(DAY, TIMESTAMP '2003-02-01 11:43:22', TIMESTAMP '2005-04-09 12:05:55'); -- 798
```

TO_BINARY

Not a direct function. Use `CONVERT_TO` or `BINARY_STRING`.

TO_CHAR

Converts an expression to a string using a specified format.

Syntax

```
TO_CHAR(expression time, format varchar) → varchar
TO_CHAR(expression date, format varchar) → varchar
TO_CHAR(expression int32, format varchar) → varchar
TO_CHAR(expression float, format varchar) → varchar
TO_CHAR(expression int64, format varchar) → varchar
TO_CHAR(expression double, format varchar) → varchar
TO_CHAR(expression timestamp, format varchar) → varchar
```

Parameters

`expression`: Expression to convert to a string.

`format`: Format to use for the conversion.

Examples

```
SELECT TO_CHAR(CAST('01:02:03' AS TIME) , 'HH:MI'); -- 01:02
SELECT TO_CHAR(CAST('2021-02-11' AS DATE) , 'yyyy.mm.dd'); -- 2021.02.11
SELECT TO_CHAR(10, '#') -- 10
SELECT TO_CHAR(7.5, '#.#') -- 7.5
```

TO_DATE

Converts an expression to a date.

Syntax

```
TO_DATE(in timestamp) → date
TO_DATE(numeric_expression int32) → date
TO_DATE(numeric_expression float) → date
TO_DATE(numeric_expression int64) → date
TO_DATE(string_expression varchar, format varchar, replaceErrorWithNull int32) → date
TO_DATE(string_expression varchar, format varchar) → date
TO_DATE(numeric_expression double) → date
```

Parameters

˜in`: The date is extracted from the timestamp.

˜numeric_expression`: A Unix epoch timestamp.

˜string_expression`: The string from which to extract the date.

˜format`: String to specify format of the date.

˜replaceErrorWithNull`: If 0, the function will fail when given malformed input. If 1, the function will return NULL when given the malformed input.

Examples

```
SELECT TO_DATE(TIMESTAMP '2022-05-17 19:15:00.000') -- 2022-05-17
SELECT TO_DATE(1640131200) -- 2021-12-22
SELECT TO_DATE('05/24/22', 'MM/DD/YY') -- 2022-05-24
```

TO_HEX

Converts a binary value to a hexadecimal string.

Syntax

```
TO_HEX(in binary) → string
```

Parameters

`in`: A binary value

Examples

```
select to_hex(binary_string('hello')) -- 68656C6C6F
```

TO_NUMBER

Converts a string to a number.

Syntax

```
TO_NUMBER(expression varchar, format varchar) → double
```

Parameters

`expression`: String to convert to a number.

`format`: Format for number conversion.

Examples

```
SELECT TO_NUMBER('12374.0023', '#####.###') -- 12374.002
SELECT TO_NUMBER('12374', '#####') -- 12374.0
```

TO_TIME

Converts an expression to a time.

Syntax

```
TO_TIME(numeric_expression int32) → time
TO_TIME(numeric_expression int64) → time
TO_TIME(string_expression varchar, format varchar, replaceErrorWithNull int32) → time
TO_TIME(string_expression varchar, format varchar) → time
TO_TIME(numeric_expression double) → time
TO_TIME(numeric_expression float) → time
```

Parameters

`numeric_expression`: A Unix epoch timestamp.

˜string_expression`: The string from which to extract the time.

˜format`: String to specify format of the time.

˜replaceErrorWithNull`: If 0, the function will fail when given malformed input. If 1, the function will return NULL when given malformed input.

Examples

```
SELECT TO_TIME(1665131223) -- 08:27:03
SELECT TO_TIME('09:15:00', 'HH:MI:SS') -- 09:15:00
```

TO_TIMESTAMP

Converts an expression to a timestamp.

Syntax

```
TO_TIMESTAMP(numeric_expression double) → timestamp
TO_TIMESTAMP(string_expression varchar, format varchar [, replaceErrorWithNull int32]) →
timestamp
TO_TIMESTAMP(numeric_expression int64) → timestamp
TO_TIMESTAMP(numeric_expression int32) → timestamp
TO_TIMESTAMP(numeric_expression float) → timestamp
```

Parameters

˜numeric_expression`: A Unix epoch timestamp.

˜string_expression`: The string from which to extract the timestamp.

˜format`: String to specify format of the timestamp.

˜replaceErrorWithNull` (optional): If 0, the function will fail when given malformed input. If 1, the function will return NULL when given malformed input.

Examples

```
SELECT TO_TIMESTAMP(52 * 365.25 * 86400) -- 2022-01-01 00:00:00
SELECT TO_TIMESTAMP(1640131200) -- 2021-12-22 00:00:00
```

TRANSACTION_TIMESTAMP

Returns the timestamp of the start of the current transaction.

Syntax

```
TRANSACTION_TIMESTAMP() → timestamp
```

Examples

```
SELECT TRANSACTION_TIMESTAMP() -- 2021-07-13 06:52:10.694000
```

TRANSLATE

Replaces a sequence of characters in a string with another set of characters.

Syntax

```
TRANSLATE(base_expression varchar, source_characters varchar, target_characters varchar)
→ varchar
```

Parameters

˜base_expression`: The string to translate.

˜source_characters`: A string with all the characters in the base expression that need translating.

˜target_characters`: A string containing all the characters to replace the original characters with.

Examples

```
SELECT TRANSLATE('*a*bX*dYZ*','XYZ*','cef'); -- abcdef
```

TRIM

Removes leading, trailing, or both spaces or specified characters from a string.

Syntax

```
TRIM(LEADING or TRAILING or BOTH trim_expression varchar FROM expression varchar) →
varchar
```

Parameters

˜trim_expression` (optional): The characters to trim.

`expression`: The expression to be trimmed.

Examples

```
SELECT TRIM(' pancake ') -- pancake
SELECT TRIM(leading 'pan' from 'pancake') -- cake
SELECT TRIM(trailing 'cake' from 'pancake') -- pan
SELECT TRIM(both 'pan' from 'pancake pan') -- cake
```

TRUNC

Alias for TRUNCATE (numeric) or DATE_TRUNC (date).

TRUNCATE

Truncates a number to a specified scale.

Syntax

```
TRUNCATE(numeric_expression float) → int
TRUNCATE(numeric_expression double) → int
TRUNCATE(numeric_expression int32) → int32
TRUNCATE(numeric_expression int64) → int64
TRUNCATE(numeric_expression decimal(0,0) [, scale_expression int32]) → decimal(0,0)
TRUNCATE(numeric_expression float [, scale_expression int32]) → float
TRUNCATE(numeric_expression double [, scale_expression int32]) → double
```

Parameters

`numeric_expression`: The numeric expression to truncate.

`scale_expression` (optional): The decimal place to round to.

Examples

```
SELECT TRUNCATE(987.65) -- 987
SELECT TRUNCATE(89.2283211, 2) -- 89.22
SELECT TRUNCATE(2021, -1) -- 2020
```

UPPER

Converts a string to uppercase.

Syntax

```
UPPER(expression varchar) → varchar
```

Parameters

`expression`: String to convert to uppercase.

Examples

```
SELECT UPPER('a guide to data lakehouses') -- A GUIDE TO DATA LAKEHOUSES
```

VAR_POP

Returns the population variance of a numeric column.

Syntax

```
VAR_POP(col_name NUMERIC) → NUMERIC
```

Parameters

`col_name`: The name of the column for which to return the population variance. The values in the column must be numbers, such as INT, DOUBLE, or FLOAT.

Examples

```
SELECT      VAR_POP(pop)      FROM      Samples."samples.dremio.com"."zips.json";      --
1.5167869917122573E8
```

VAR_SAMP

Returns the sample variance of a numeric column.

Syntax

```
VAR_SAMP(col_name NUMERIC) → NUMERIC
```

Parameters

`col_name`: The name of the column for which to return the sample variance. The values in the column must be numbers, such as INT, DOUBLE, or FLOAT.

Examples

```
SELECT VAR_SAMP(passenger_count) FROM Samples."samples.dremio.com"."NYC-taxi-trips"; --
1.868747683518558
```

VARIANCE

Alias for VAR_SAMP.

WEEK

Extracts the week from a date or timestamp.

Syntax

```
EXTRACT(WEEK FROM date_timestamp_expression string) → bigint
```

Parameters

˜date_timestamp_expression`: A DATE or TIMESTAMP expression.

Examples

```
SELECT EXTRACT(WEEK FROM TIMESTAMP '2019-08-12 01:00:00.123456') -- 33
SELECT EXTRACT(WEEK FROM DATE '2019-08-12') -- 33
```

YEAR

Extracts the year from a date or timestamp.

Syntax

```
EXTRACT(YEAR FROM date_timestamp_expression string) → bigint
```

Parameters

˜date_timestamp_expression`: A DATE or TIMESTAMP expression.

Examples

```
SELECT EXTRACT(YEAR FROM TIMESTAMP '2019-08-12 01:00:00.123456') -- 2019
SELECT EXTRACT(YEAR FROM DATE '2019-08-12') -- 2019
```

System Tables

System tables make up Dremio's system-created catalog to store metadata for the objects in your Dremio organization.

sys.organization.model_usage

Contains metadata for LLM model usage through Dremio.

Field	Data Type	Description
organization_id	nvarchar	The UUID of the organization.
project_id	nvarchar	The UUID of the project.
user_id	nvarchar	The UUID of the user.
model_name	nvarchar	The name of the model.
operation	nvarchar	The operation performed. Enum: `CHAT`, `SQL`
input_tokens	integer	The number of input tokens.
output_tokens	integer	The number of output tokens.
timestamp	timestamp	The timestamp of the usage.

sys.organization.privileges

Contains metadata for privileges at the organization-level.

Field	Data Type	Description
grantee_id	nvarchar	The UUID of the user or role that has been granted the privilege.
grantee_type	nvarchar	The type of grantee. Enum: `user`, `role`
privilege	nvarchar	The privilege that has been granted.
object_id	nvarchar	The UUID of the object on which the privilege has been granted.
object_type	nvarchar	The type of the object on which the privilege has been granted.

sys.organization.projects

Contains metadata for projects in an organization.

Field	Data Type	Description
project_id	nvarchar	The UUID to identify the project.
project_name	nvarchar	The name of the project.
project_state	nvarchar	The state of the project. Enum: `COMMISSIONING`, `ACTIVE`, `FAILED`, `MARK_DELETE`
description	nvarchar	The description of the project.
created	timestamp	The timestamp for when the project was created.
organization_id	nvarchar	The UUID of the organization.
identity_type	nvarchar	The type of identity. Enum: `ACCESS_KEY`, `IAM_ROLE`
owner_id	nvarchar	The UUID of the owner.
owner_type	nvarchar	The type of owner. Enum: `USER`, `ROLE`

sys.organization.roles

Contains metadata for roles in an organization.

Field	Data Type	Description
role_id	nvarchar	The UUID to identify the role.
role_name	nvarchar	The name of the role.
role_type	nvarchar	The type of role. Enum: `SYSTEM`, `INTERNAL`, `USER`
owner_id	nvarchar	The UUID of the owner (user or role) of the role.
owner_type	nvarchar	The type of owner of the role. Enum: `USER`, `ROLE`
created	timestamp	The timestamp for when the role was created.
created_by	nvarchar	The method of creation. Enum: `LOCAL`, `SCIM`

sys.organization.usage

Contains data about an organization's usage.

Field	Data Type	Description
organization_id	nvarchar	The UUID of the organization.
project_id	nvarchar	The UUID of the project.
edition	nvarchar	The edition of Dremio.
job_id	nvarchar	The UUID of the job.
user_id	nvarchar	The UUID of the user.
start_time	timestamp	The start time of the job.
end_time	timestamp	The end time of the job.
engine_id	nvarchar	The UUID of the engine.
engine_name	nvarchar	The name of the engine.
engine_size	nvarchar	The size of the engine (e.g., m5d.4xlarge).
dcu	double	The number of Dremio Capacity Units (DCUs) consumed.
job_type	nvarchar	The type of job.
status	nvarchar	The status of the job.
considered_reflection_count	integer	The number of reflections considered.
matched_reflection_count	integer	The number of reflections matched.
chosen_reflection_count	integer	The number of reflections chosen.

sys.organization.users

Contains metadata for users in an organization.

Field	Data Type	Description
user_id	nvarchar	The UUID to identify the user.
user_name	nvarchar	The email of the user is used as the username.
first_name	nvarchar	The first name of the user.
last_name	nvarchar	The last name of the user.
status	nvarchar	The state of the user depending on if they have accepted the invite to the organization and have logged in to the application. Enum: `active`, `invited`
user_type	nvarchar	The type of user based on how it was created. Enum: `EXTERNAL`, `LOCAL`
created	timestamp	The timestamp for when the user was created.
owner_id	nvarchar	The UUID for the owner (user or role) of the user. This UUID

corresponds to the `user_id` or `role_id` in the `users` or `roles` table. |
| owner_type | nvarchar | The type of owner of the user. Enum: `user`, `role` |
| created_by | nvarchar | The method of creation. Enum: `LOCAL`, `SCIM` |

sys.project.engines

Contains metadata for engines in a project.

Field	Data Type	Description
engine_id	nvarchar	The UUID to identify the engine.
engine_name	nvarchar	The name of the engine.
engine_size	nvarchar	The size of the engine. Enum: `XX_SMALL_V1`, `X_SMALL_V1`, `SMALL_V1`, `MEDIUM_V1`, `LARGE_V1`, `X_LARGE_V1`, `XX_LARGE_V1`, `XXX_LARGE_V1`
engine_state	nvarchar	The state of the engine. Enum: `DELETING`, `DISABLED`, `DISABLING`, `ENABLED`
min_replicas	integer	The minimum number of replicas for the engine.
max_replicas	integer	The maximum number of replicas for the engine.
current_replicas	integer	The current number of replicas for the engine.
instance_family	nvarchar	The instance family of the engine.
tag	nvarchar	The tag of the engine.

sys.project.jobs

Contains the metadata for the jobs in a project.

Field	Data Type	Description
job_id	nvarchar	The UUID to identify the job.
job_type	nvarchar	The type of job. Enum: `ACCELERATOR_CREATE`, `ACCELERATOR_DROP`, `ACCELERATOR_EXPLAIN`, `FLIGHT`, `INTERNAL_ICEBERG_METADATA_DROP`, `JDBC`, `UI_EXPORT`, `UI_INTERNAL_PREVIEW`, `UI_INTERNAL_RUN`, `UI_PREVIEW`, `UI_RUN`, `METADATA_REFRESH`, `ODBC`, `PREPARE_INTERNAL`, `REST`, `UNKNOWN`
status	nvarchar	The status of the job. Enum: `SETUP`, `QUEUED`, `ENGINE START`, `RUNNING`
user_name	nvarchar	The username of the user who submitted the job.
submitted_ts	timestamp	The timestamp for when the job was submitted.
submitted_epoch	bigint	The epoch timestamp for when the job was submitted.
is_accelerated	boolean	Whether the job was accelerated.
accelerated_by_substitution	boolean	Whether the job was accelerated by substitution.
queried_datasets	array	The datasets that were queried.
scanned_datasets	array	The datasets that were scanned.
attempt_count	integer	The number of attempts for the job.
error_msg	nvarchar	The error message if the job failed.
query_type	nvarchar	The type of query.

sys.project.pipes

Contains the metadata for autoingest pipes in a project.

Column Name	Data Type	Description
pipe_name	nvarchar	The name of the pipe.
pipe_id	nvarchar	The unique identifier of the pipe.
pipe_state	nvarchar	The current state of the pipe. Enum: `Running`, `Paused`, `Stopped_Missing_Table_or_Branch`, `Stopped_Storage_Location_Altered`, `Stopped_Access_Denied`, `Stopped_Missing_Dremio_Source`, `Unhealthy`, `Stopped_Internal_Error`
dedupe_lookback_period	integer	The number of days to look back for deduplication.
notification_provider	nvarchar	The notification provider for the pipe.
notification_queue_reference	nvarchar	The reference to the notification queue.
source_root_path	nvarchar	The root path of the source data.
target_table	nvarchar	The target table for the pipe.
file_format	nvarchar	The file format of the source data.
pipe_owner	nvarchar	The owner of the pipe.
created_at	timestamp	The timestamp for when the pipe was created.
last_updated_at	timestamp	The timestamp for when the pipe was last updated.
cloud_settings	nvarchar	The cloud settings for the pipe.

sys.project.privileges

Contains metadata for privileges at the project-level.

Field	Data Type	Description
grantee_id	nvarchar	The UUID of the user or role that has been granted the privilege.
grantee_type	nvarchar	The type of grantee. Enum: `user`, `role`
privilege	nvarchar	The privilege that has been granted.
object_id	nvarchar	The UUID of the object on which the privilege has been granted.
object_type	nvarchar	The type of the object on which the privilege has been granted. (e.g., VDS for view)

sys.project.reflection_dependencies

Contains metadata for Reflection dependencies in the current project.

Field	Data Type	Description
reflection_id	nvarchar	The UUID of the Reflection.
dependency_id	nvarchar	The UUID of the dependency.
dependency_type	nvarchar	The type of dependency. Enum: `DATASET`, `REFLECTION`
dependency_path	array	The path of the dependency.

sys.project.reflections

Contains metadata for Reflections in a project.

| Field | Data Type | Description |

| --- | --- | --- |
reflection_id	nvarchar	The UUID to identify the Reflection.
reflection_name	nvarchar	The name of the Reflection.
type	nvarchar	The type of Reflection. Enum: `AGGREGATION`, `RAW`
status	nvarchar	The status of the Reflection. Enum: `CAN_ACCELERATE`, `CAN_ACCELERATE_WITH_FAILURES`, `REFRESHING`, `FAILED`, `EXPIRED`, `DISABLED`, `INVALID`, `CANNOT_ACCELERATE_SCHEDULED`, `CANNOT_ACCELERATE_MANUAL`
dataset_id	nvarchar	The UUID of the dataset that the Reflection is defined on.
dataset_name	nvarchar	The name of the dataset.
dataset_type	nvarchar	The type of dataset. Enum: `PHYSICAL_DATASET_HOME_FILE`, `PHYSICAL_DATASET_SOURCE_FILE`, `PHYSICAL_DATASET_SOURCE_FOLDER`, `VIRTUAL_DATASET`
sort_columns	array	The columns that the Reflection is sorted by.
partition_columns	array	The columns that the Reflection is partitioned by.
distribution_columns	array	The columns that the Reflection is distributed by.
dimensions	array	The dimensions of the Reflection.
measures	array	The measures of the Reflection.
external_reflection	nvarchar	The name of the external reflection.
arrow_caching_enabled	boolean	Whether Arrow caching is enabled.
partition_distribution_strategy	nvarchar	The partition distribution strategy.
measure_fields	array	The measure fields.
dimension_fields	array	The dimension fields.
display_columns	array	The display columns.
num_failures	integer	The number of failures.
created	timestamp	The timestamp for when the Reflection was created.
modified	timestamp	The timestamp for when the Reflection was last modified.
refresh_method	nvarchar	The refresh method. Enum: `Manual`, `Autonomous`
refresh_status	nvarchar	The refresh status. Enum: `NONE`

sys.project."tables"

Contains metadata for tables in a project.

Field	Data Type	Description
table_id	nvarchar	The UUID to identify the table.
table_name	nvarchar	The name of the table.
schema_id	nvarchar	The UUID for the schema/folder in which the table is contained.
path	nvarchar	The string array representation of the path of the table.
tag	nvarchar	The UUID that is generated to identify the instance of the table. Dremio changes this tag whenever a change is made to the table.
type	nvarchar	The type of table. Enum: `PHYSICAL_DATASET`, `SYSTEM_TABLE`, `NESSIE_TABLE`
format	nvarchar	The format of the table. Enum: `DELTA`, `EXCEL`, `ICEBERG`, `JSON`, `PARQUET`, `TEXT`, `UNKNOWN`, `XLS`
created	timestamp	The date and time that the table was created.
owner_id	nvarchar	The UUID for the owner (user or role) of the table.
owner_type	nvarchar	The type of owner of the table. Enum: `USER_OWNER`, `ROLE_OWNER`
record_count	bigint	The number of records in the table.
column_count	integer	The number of columns in the table.

| is_approximate_stats | boolean | Whether the statistics are approximate. |

sys.project.views

Contains metadata for views in a project.

Field	Data Type	Description
view_id	nvarchar	The UUID to identify the view.
space_id	nvarchar	The UUID to identify the parent space that the view is saved under.
view_name	nvarchar	The user- or system-defined name of the view.
schema_id	nvarchar	The UUID for the schema/folder in which the view is contained.
path	nvarchar	The string array representation of the path of the view.
tag	nvarchar	The UUID that is generated to identify the instance of the view. Dremio changes this tag whenever a change is made to the view.
type	nvarchar	The type of view. Enum: `VIRTUAL_DATASET`, `NESSIE_VIEW`
created	timestamp	The date and time that the view was created.
sql_definition	nvarchar	The DDL statement that was used to create the view.
sql_context	nvarchar	The context for the SQL definition.
owner_id	nvarchar	The UUID for the owner (user or role) of the view.
owner_type	nvarchar	The type of owner of the view. Enum: `USER_OWNER`, `ROLE_OWNER`

sys.project.copy_errors_history

Contains metadata for copy errors history.

Column Name	Data Type	Description
executed_at	timestamp	The timestamp when the copy command was executed.
job_id	nvarchar	The UUID of the job that executed the copy command.
table_name	nvarchar	The name of the table.
user_name	nvarchar	The name of the user who executed the command.
file_path	nvarchar	The path of the file that caused the error.
line_number	bigint	The line number in the file where the error occurred.
error_message	nvarchar	The error message.

sys.project.copy_file_history

Contains metadata for copy file history.

Column Name	Data Type	Description
executed_at	timestamp	The timestamp when the copy command was executed.
job_id	nvarchar	The UUID of the job that executed the copy command.
table_name	nvarchar	The name of the table.
user_name	nvarchar	The name of the user who executed the command.
file_path	nvarchar	The path of the file that was copied.
file_state	nvarchar	The state of the file copy. Enum: `LOADED`, `PARTIALLY_LOADED`, `SKIPPED`

| records_loaded_count | bigint | The number of records loaded. |
| records_rejected_count | bigint | The number of records rejected. |

sys.project.history.autonomous_reflections

Contains metadata for autonomous reflections history.

Column Name	Data Type	Description
reflection_id	nvarchar	The UUID of the Reflection.
reflection_name	nvarchar	The name of the Reflection.
dataset_id	nvarchar	The UUID of the dataset.
dataset_name	nvarchar	The name of the dataset.
status	nvarchar	The status of the Reflection.
event_type	nvarchar	The type of event.
event_timestamp	timestamp	The timestamp of the event.
details	nvarchar	The details of the event.

sys.project.history.events

Contains metadata for historical events.

Field	Data Type	Description
event_id	nvarchar	The UUID of the event.
event_type	nvarchar	The type of event.
event_timestamp	timestamp	The timestamp of the event.
user_id	nvarchar	The UUID of the user who triggered the event.
entity_id	nvarchar	The UUID of the entity related to the event.
entity_type	nvarchar	The type of entity.
details	nvarchar	The details of the event.

sys.project.history.jobs

Contains metadata for historical jobs.

Field	Data Type	Description
job_id	nvarchar	The UUID to identify the job.
job_type	nvarchar	The type of job.
status	nvarchar	The status of the job.
user_name	nvarchar	The username of the user who submitted the job.
submitted_ts	timestamp	The timestamp for when the job was submitted.
submitted_epoch	bigint	The epoch timestamp for when the job was submitted.
is_accelerated	boolean	Whether the job was accelerated.
accelerated_by_substitution	boolean	Whether the job was accelerated by substitution.
queried_datasets	array	The datasets that were queried.
scanned_datasets	array	The datasets that were scanned.
attempt_count	integer	The number of attempts for the job.
error_msg	nvarchar	The error message if the job failed.
query_type	nvarchar	The type of query.

sys.project.materializations

Contains metadata for materializations.

Field	Data Type	Description
materialization_id	nvarchar	The UUID of the materialization.
reflection_id	nvarchar	The UUID of the Reflection.
job_id	nvarchar	The UUID of the job that created the materialization.
created	timestamp	The timestamp for when the materialization was created.
expires	timestamp	The timestamp for when the materialization expires.
state	nvarchar	The state of the materialization.
footprint	bigint	The size of the materialization.
original_cost	double	The original cost of the query.
reflection_type	nvarchar	The type of Reflection.
series_id	nvarchar	The series ID.
series_ordinal	integer	The series ordinal.
join_analysis	nvarchar	The join analysis.

sys.project.pipe_summary

Contains metadata for pipe summary.

Column Name	Data Type	Description
pipe_name	nvarchar	The name of the pipe.
pipe_id	nvarchar	The unique identifier of the pipe.
jobs_count	integer	The number of jobs completed for autoingestion related to this pipe.
files_loaded_count	integer	The number of files loaded by the pipe.
files_skipped_count	integer	The number of files skipped by the pipe.
files_partially_loaded_count	integer	The number of files partially loaded by the pipe.
pipe_status	nvarchar	The current status of the pipe.
error_message	nvarchar	The error message if the pipe is in an error state.
last_updated_at	timestamp	The timestamp of the last update to the pipe summary.
total_records_count	integer	The total number of records processed by the pipe.

Table Functions

Table functions in Dremio return a table as a result and can be used in the `FROM` clause of a query.

sys.recommend_reflections

Returns a list of recommendations for Reflections that can be created to accelerate queries.

Syntax:

```
SELECT * FROM TABLE(sys.recommend_reflections())
```

Columns:

Column	Data Type	Description
query_id	VARCHAR	The job ID of the query that would benefit from the reflection.
dataset_id	VARCHAR	The ID of the dataset.
dataset_name	VARCHAR	The name of the dataset.
reflection_type	VARCHAR	The type of reflection (RAW or AGGREGATION).
display_columns	LIST	The list of display columns.
dimension_columns	LIST	The list of dimension columns.
measure_columns	LIST	The list of measure columns.
sort_columns	LIST	The list of sort columns.
partition_columns	LIST	The list of partition columns.
distribution_columns	LIST	The list of distribution columns.
acceleration_count	BIGINT	The number of times the reflection would have accelerated queries.
ratio	DOUBLE	The ratio of acceleration.
error_message	VARCHAR	Any error message associated with the recommendation.

sys.reflection_lineage

Return a list of the Reflections that will also be refreshed if a refresh is triggered for a particular Reflection.

Syntax:

```
SELECT * FROM TABLE(sys.reflection_lineage('<reflection_id>'))
```

Parameters:

`<reflection_id>` (String): The ID of the Reflection.

Columns:

Column	Data Type	Description
reflection_id	nvarchar	The ID of the Reflection.
reflection_name	nvarchar	The name of the Reflection.
dataset_name	nvarchar	The name of the dataset.
dataset_id	nvarchar	The ID of the dataset.
reflection_type	nvarchar	The type of Reflection. Enum: `RAW`, `AGG`
status	nvarchar	The status of the Reflection. Enum: `INVALID`, `REFRESHING`, `METADATA_REFRESH`, `COMPACTING`, `FAILED`, `DEPRECATED`, `CANNOT_ACCELERATE`, `MANUAL_REFRESH`, `DONE`
num_failures	integer	The number of consecutive failures for the Reflection.
is_active	boolean	Indicates whether the Reflection is active (true) or inactive (false).
can_view	boolean	Indicates whether the current user has permission to view the Reflection.

sys.reflection_refresh_settings

Returns the refresh settings for a Reflection, including settings inherited from the

datasets that the Reflection depends on.

Syntax:

```
SELECT * FROM TABLE(sys.reflection_refresh_settings('<reflection_id>'))
```

Columns:

Column	Data Type	Description
table_type	nvarchar	Defines the type of table. Enum: `DATASET` or `EXTERNAL_QUERY`
table_path	nvarchar	Identifies the path to the dataset or external query source that the Reflection depends on.
table_version_context	nvarchar	Specifies the versioning context for datasets, stored in JSON format.
overrides_source	boolean	Indicates whether settings are inherited from the source (true) or set on the table (false).
refresh_method	nvarchar	Shows the method used for the most recent refresh of the Reflection. Enum: `FULL`, `INCREMENTAL`, `AUTO`
refresh_policy	nvarchar	Identifies the type of refresh policy. Enum: `PERIOD`, `SCHEDULE`, `LIVE_REFRESH`, `NEVER_REFRESH`
refresh_period_seconds	double	Specifies the time in seconds (truncated from milliseconds) between refreshes.
refresh_schedule	nvarchar	Provides the cron expression (UTC) that defines the refresh schedule for the Reflection.
never_expire	boolean	Indicates whether the Reflection never expires (true) or uses the expiration setting (false).
expiration_seconds	double	Defines the expiration time in seconds (truncated from milliseconds), after which the system removes the Reflection.

Information Schema

Dremio stores metadata for the objects in your project in the Information Schema, which is a set of system-generated read-only views.

INFORMATION_SCHEMA.CATALOGS

Returns metadata for the catalogs.

Fields:

Field	Data Type	Description
CATALOG_NAME	nvarchar	The name of the catalog, which is always DREMIO.
CATALOG_DESCRIPTION	nvarchar	The description for the catalog that contains metadata.
CATALOG_CONNECT	nvarchar	The connection permissions to the catalog that contains metadata information. This is an inherited field and is always empty.

INFORMATION_SCHEMA.COLUMNS

Returns metadata for columns in tables and views.

Fields:

Field	Data Type	Description
TABLE_CATALOG	nvarchar	The name of the catalog, which is always DREMIO.
TABLE_SCHEMA	nvarchar	The path (source, space, folders) to the table or view.
TABLE_NAME	nvarchar	The name of the table or view that the column belongs to.
COLUMN_NAME	nvarchar	The name of the column in the table or view.
ORDINAL_POSITION	integer	This represents the position at which the column appears in the table or view.
COLUMN_DEFAULT	nvarchar	The default value of the column.
IS_NULLABLE	nvarchar	The value is YES if null values can be stored in the column and the value is NO if null values cannot be stored in the column.
DATA_TYPE	nvarchar	The system-defined data type of the column in the table or view.
COLUMN_SIZE	integer	The size of the table or view column in bytes.
CHARACTER_MAXIMUM_LENGTH	integer	The maximum length in characters for binary data, character data, or text and image data.
CHARACTER_OCTET_LENGTH	integer	The maximum length in bytes for binary data, character data, or text and image data.
NUMERIC_PRECISION	integer	The precision of approximate numeric data, exact numeric data, integer data, or monetary data.
NUMERIC_PRECISION_RADIX	integer	The precision radix of approximate numeric data, exact numeric data, integer data, or monetary data.
NUMERIC_SCALE	integer	The scale of approximate numeric data, exact numeric data, integer data, or monetary data.
DATETIME_PRECISION	integer	The supported precision for datetime and interval data types.
INTERVAL_TYPE	integer	If the data type is interval, then specified fields (year) are returned.
INTERVAL_PRECISION	integer	If the data type is interval, then the declared precision is displayed.

INFORMATION_SCHEMA.SCHEMATA

Returns metadata for schemas (folders/spaces).

Fields:

Field	Data Type	Description
CATALOG_NAME	nvarchar	Name of the catalog, which is always DREMIO.
SCHEMA_NAME	nvarchar	Path (source, space, folders) that contains datasets.
SCHEMA_OWNER	nvarchar	Owner of the schema. This is an inherited field and <owner> is always returned.
TYPE	nvarchar	Type of the schema, which is always SIMPLE.
IS_MUTABLE	nvarchar	The value in this column is YES if the schema can be modified. NO if it's immutable.

INFORMATION_SCHEMA."TABLES"

Returns metadata for tables and views.

Fields:

Field	Data Type	Description
TABLE_CATALOG	nvarchar	Name of the catalog, which is always DREMIO.
TABLE_SCHEMA	nvarchar	The path (source, space, folders) to the table or view.
TABLE_NAME	nvarchar	The name of the table or view.
TABLE_TYPE	nvarchar	The type of the object. Enum: `SYSTEM_TABLE`, `TABLE`, `VIEW`

INFORMATION_SCHEMA.VIEWS

Returns metadata for views.

Fields:

Field	Data Type	Description
TABLE_CATALOG	nvarchar	The name of the catalog, which is always DREMIO.
TABLE_SCHEMA	nvarchar	The path (source, space, folders) to the view.
TABLE_NAME	nvarchar	The name of the view.
VIEW_DEFINITION	nvarchar	The original SQL query (underlying DDL statement) used to define the view.