



Collibra Data Intelligence Cloud

Data Lineage

Collibra Data Intelligence Cloud - Data Lineage

Release date: February 4, 2024

Revision date: February 01, 2024

You can find the most up-to-date technical documentation on our Documentation Center at

<https://productresources.collibra.com/docs/collibra/latest/#cshid=data-lineage>

Contents

Contents	ii
Collibra Data Lineage	iv
What is Collibra Data Lineage?	iv
BI tool integration	viii
Business value	viii
How do I create a technical lineage?	xii
Supported data sources for technical lineage	xiii
Database Owners, BI and ETL Admins, and Collibra Admins	xl
Database Owners	xl
BI and ETL Admins	xlvi
Collibra Admins	xc
Terminology	dcxliii
Methodology	dcxliii
Naming convention	dcxlvi
Business stewards	dclxii
Technical lineage	dclxiii
Automatic stitching for technical lineage	dclxvi
BI tool business logic	dclxviii
Technical lineage and stitching for BI tool integrations	dclxxi
Business lineage	dclxxxii
Differences between Technical lineage and business lineage	dclxxxiv
BI integration concepts	dclxxxvii
Technical stewards	dcciv

Supported SQL syntax	dcciv
Supported transformation details	dccvi
Technical lineage for Snowflake ingestion methods	dccxiv
Transformation logic	dccxviii
Technical lineage export types	dccxix
BI integration concepts	dccxxiv
Technical lineage viewer	dcclxix
Technical lineage troubleshooting	ccxcviii

Collibra Data Lineage

In this topic, we address the following:

- [What is Collibra Data Lineage?](#)
- [BI tool integration](#)
- [Business value](#)
- [How do I create a technical lineage?](#)

Tip Check out our free [Business and Technical Lineage training course](#) in Collibra University.

What is Collibra Data Lineage?

Collibra Data Lineage is a cloud-only product that allows you to trace data from its source system, across the various contact points of your data landscape, to its final destination system.

Ultimately, our objective is to help you establish trust in your reports and use the data to make sound business decisions.

Collibra Data Lineage consists of two components:

- Technical lineage
- Business lineage

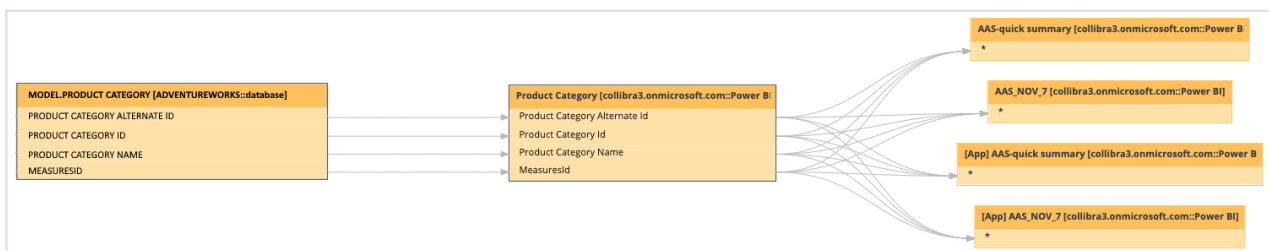
The value of these components are the same, but they are designed for different audiences.

Technical lineage

- Designed for Data Engineers, Data Architects, and other technical stewards.
- A detailed lineage graph that provides complete end-to-end lineage, to visualize the journey of the data objects in your external data sources.
- Allows you to explore data objects, including temporary tables and columns, in your external data sources. You don't need to register data sources in Collibra to include them in a technical lineage.

Tip We use the term "data objects" when referring to columns and tables in your external data sources. We use the term "assets" (specifically Column assets and Table assets) when referring to the representation of data objects in Collibra.

- Includes all source code and data transformation details.
- Shows you in which system data objects are used and how they are transformed from data source to data source.
- Automatically created as part of the technical lineage process.

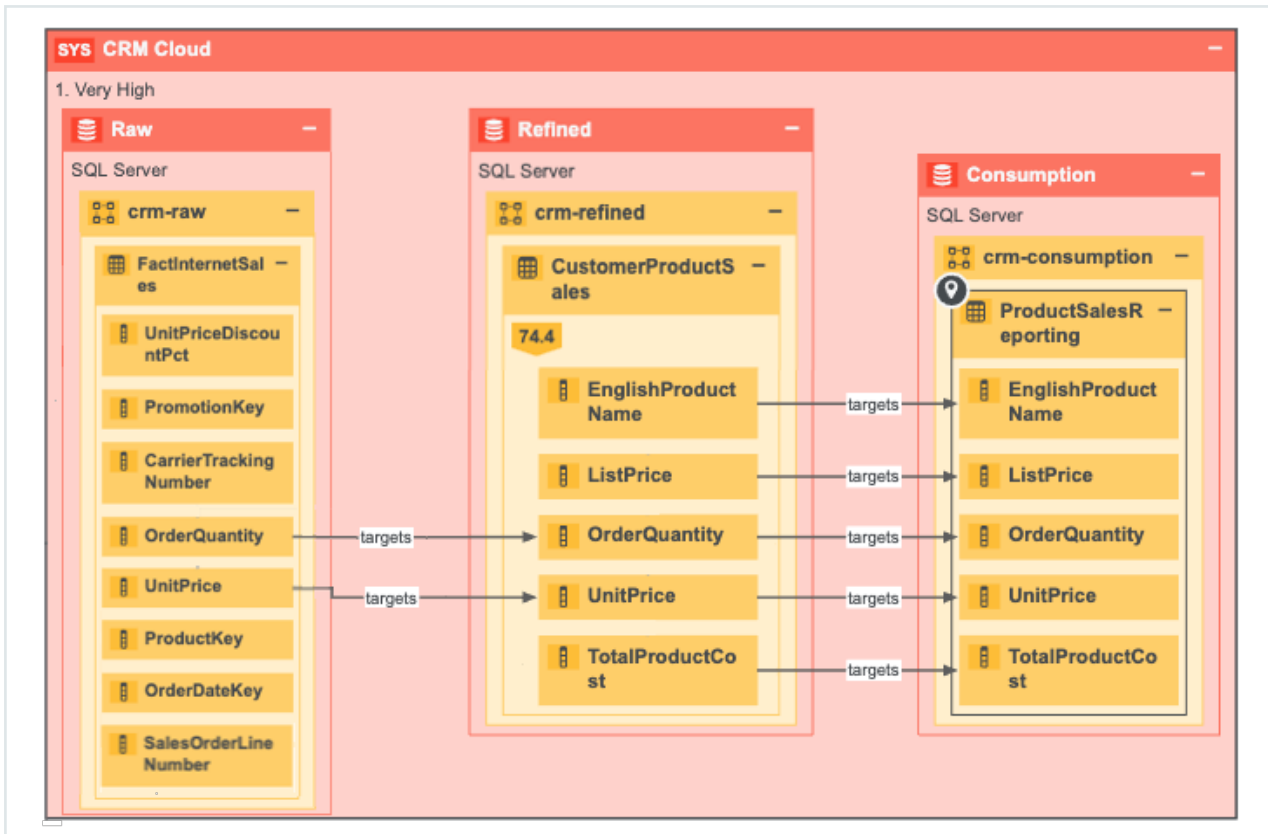


Business lineage

- Designed for Analysts, Governance roles, and other business stewards.
- Shows the relations between assets in Collibra that represent the data objects in your external data sources.
- Refers specifically to the relation type "Data Element targets / sources Data Element" that is drawn between Column assets.
- Shows how [registered data sources](#) relate to each another.

Tip Registering a data source means creating assets (and the relations between the assets) in Collibra that represent the data objects in your external data sources.

- Automatically created as part of the technical lineage process.



Tip The main difference between a technical lineage and a business lineage:

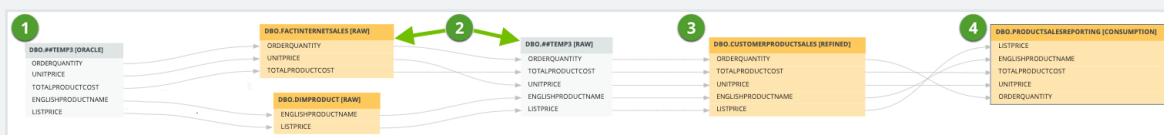
- Technical lineage identifies data objects in your external data sources.
- Business lineage show assets in Collibra that represent some or all of those data objects.

Example

Let's say that you have created a technical lineage for four different databases:

- The first database, *Oracle*, is not registered in Collibra, therefore there are no assets in Data Catalog that represent the Oracle data objects.
- The second database, *Raw*, is registered in Collibra.
 - The yellow background of the first node indicates that Table and Column assets that were created in Data Catalog are **stitched** to their corresponding data objects in the *Raw* database.
 - The other node, the one with the gray background, is a temporary table. No assets are created for temporary data objects and so stitching is not relevant. That is why the node has a gray background.
- The third and fourth databases, *Refined* and *Consumption*, are ingested in Collibra. The assets that were created in Data Catalog are stitched to their corresponding data objects in the two databases.

What we want to point out here is that Technical lineage shows the data flow of all data objects across all four databases, regardless of any assets in Collibra.



The corresponding business lineage shows only the relations between data objects that have corresponding assets in Data Catalog. In the following image, we see the data flow of assets from the second database, to the third, to the fourth. The first database, *Oracle*, which is not registered in Collibra, and , is not shown on the diagram.



For more information on the differences between these two components, go to [Differences between technical lineage and business lineage](#).

For a complete list of supported data sources, go to [Supported data sources for technical lineage](#). If you want to create a technical lineage for a data source that is not currently supported, you can [create a Custom technical lineage](#).

BI tool integration

Business intelligence software helps organizations to collect data from the various data sources across their data ecosystem and present the data in interactive dashboards and reports, to facilitate decision-making and strategic planning.

When you integrate your BI tool in Collibra:

- Metadata about the data objects in your external data sources is created as BI assets in Collibra.
- Relations are created:
 - Between data objects in your external data source and assets in Collibra that represent those data objects.

Tip These assets are created when the data source is registered, which is automatically carried out during the technical lineage process.

- Between BI assets and the assets in Collibra that represent the data objects in your external data source.
- technical lineage and business lineage are automatically created.

Business value

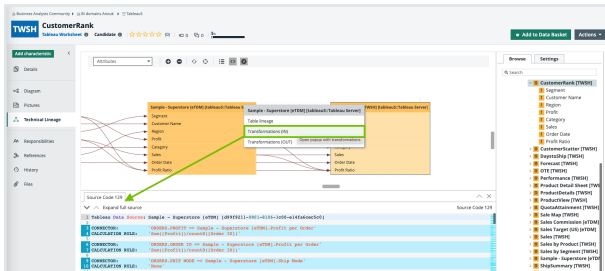
Collibra Data Lineage has many important use cases. Here are a few.

Report certification

By providing transparency and traceability to the data used in a report, data lineage plays a foundational role in the report certification process:

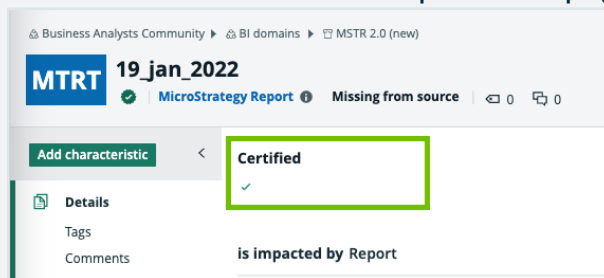
- Review data sources and transformations associated with the data in a report, to help ensure accuracy and reliability.
- Identify the original sources of data used in the report, and how the data moves from the source system to intermediate systems.

- View and analyze the calculation rules that are used to extract and transform the data before it reaches the report.



All critical metadata is ingested during BI integration and shown on the Collibra asset pages. This includes information like data timestamps, quality metrics, data ownership, and other valuable attributes that help you to assess the reliability and quality of the data.

Tip If a report is certified in your BI tool, that metadata is ingested and shown in the Certified attribute on the BI report asset page in Data Catalog.



You can manually synchronize the data in Collibra or set up a synchronization schedule, to help ensure the accuracy and completeness of the data over time. This can help identify inconsistencies or gaps in the data flow and transformation processes.

Impact analysis

Collibra Data Lineage can help you with impact analysis when making changes to data sources, adjusting the calculation rules that drive transformations, migrating data and more. It can help you assess the potential impact of changes on downstream systems, data and reports.

Example Let's say you have data in a Snowflake data source, and you need to move everything to Databricks. After migration, you can create a technical lineage to trace the movement of data from one data source to the other and ensure data integrity throughout the migration process.

Understanding data dependencies and relationships helps you to:

- Anticipate which downstream systems could be impacted if you've made changes to a data source or calculation rule.
- Anticipate how changes to a particular data object or system will propagate across your data landscape.
- Minimize risks and make better informed decisions.

Root cause analysis in data-related issues

Collibra Data Lineage is a valuable tool for helping data analysts and engineers trace the source of data quality issues and anomalies. When you detect a discrepancy in your data, you can examine the lineage and source code to:

- Trace the issue back to the source system or process that is causing the problem.
- Analyze any calculations rules that might have affected the consistency or quality of the data.
- Identify how the issue is affecting downstream systems and reporting.

This can help you identify potential areas where the root cause might exist.

Regulatory Compliance

Compliance with data privacy regulations such as GDPR and CCPA, and various security, auditing and reporting standards, often requires organizations to show end-to-end traceability across their data landscape. In the data privacy context, Collibra Data Lineage can give you a complete view of where sensitive and restricted data is processed, shared, and stored.

Example Let's say that a individual customer of an organization wants to exercise their right to be forgotten, as dictated by GDPR. In compliance with the regulation, the organization has to purge Personally Identifiable Information (PII) about the individual from its systems. Once the organization has identified the PII, it can use data lineage to:

- Trace the information across its systems, data source and processes.
- Monitor any migrations and transformations to the data.
- Identify who has access to the systems and data sources that consume the data.

BI integration: View critical metadata about your reports and dashboards

BI integration in Collibra enables you to view all of the critical metadata about your reports and dashboards on dedicated asset pages in Data Catalog. The many attributes help you to identify the most critical reports that have the highest impact. This can help you effectively allocate your resources and minimize disruptions.

The screenshot shows the Collibra Data Catalog interface for a report named 'covid19_jan_2022'. The breadcrumb trail indicates the path: Business Analysts Community > BI domains Anouk > MSTR 2.0 (new). The report is a 'MicroStrategy Report' and is marked as 'Missing from source'. It has 0 comments and 0 likes.

The left sidebar contains a navigation menu with the following items: Details (selected), Tags, Comments, Diagram, Pictures, Technical Lineage, Responsibilities, References, and History.

The main content area displays the following metadata:

- is impacted by Report**: A table with columns Name, Domain, Definition, and Description.

Name	Domain	Definition	Description
anouk-report-csv	MSTR 2.0 (new)		This is a description of the rep...
Dashboard 1	Nike demo		
- Document creation date**: 6/9/2023
- URL**: <http://ec2-18-216-24-232.us-east-2.compute.amazonaws.com:8080/MicroStrategy/servlet/mstrWeb?evt=4001&reportID=6818584F400F54AA350B978C83C28EA2>
- Owner in Source**: Bill Fischer

A few of the key attributes include the following:

- **Document creation and modification dates**: See when the report was created and updated in your BI tool.
- **Visits count**: See how many people have viewed the report.

Tip Let's say that you have two reports with the same name, but one has 400 views and the other has almost none. That gives a strong indication as to which is the more helpful report.

- **Owner in Source**: Easily identify who owns and who certified a report, to know where to turn for additional help and information
- **Calculation Rule**: See DAX calculations for calculated columns and measures on Power BI Column asset pages.
- **URL**: Easily access the report in your BI tool.
- **Relation types** allow you to immediately identify in which other reports a report is used.

How do I create a technical lineage?

There are two ways to create technical lineage and business lineage:

- [Via Edge](#).
- [Via the lineage harvester](#).

The typical workflow for creating a technical lineage is the same whether you use the lineage harvester or Edge. If you want to use Edge and the lineage harvester together, you must use lineage harvester version 2023.04 or newer. If you want to maintain on Edge the technical lineage that you created by using the lineage harvester, you can add technical lineage capabilities for the data sources with the same source IDs. For details, go to [Migrate the technical lineage of a data source](#).

Edge

You can create a technical lineage and business lineage via Edge, for Tableau, Power BI and all supported JDBC and ETL data sources. Benefits include:

- Seamless integration with Data Catalog.
- The Edge User Interface (UI), instead of Command Line Interface.
- Connections via Edge, instead of lineage harvester drivers.
- Job scheduling via Data Catalog.

The lineage harvester

The lineage harvester is a connectivity tool that allows you to [create](#) a technical lineage and business lineage.

- You can use the lineage harvester for any [supported data source](#).
- You can download the latest lineage harvester from the [Collibra Community Downloads page](#).
- You need to use the Command Line Interface in conjunction with a [lineage harvester configuration file](#).

Supported data sources for technical lineage

Collibra Data Intelligence Cloud supports many data sources and metadata sources, including JDBC data sources, ETL tools and BI tools, for which you can create a technical lineage.

For a complete list of required permissions per supported data source type, see the Requirements and permissions section in [Prepare the lineage harvester configuration file](#).

Note Using an older version of a data source might not work as expected; however, we don't expect problems if you use a newer version.

JDBC data sources

The following tables show the supported JDBC data sources.

Technical lineage via Edge

The following table lists the supported JDBC data sources and connection types you can use when you add capabilities for different data sources. The Shared Storage connection is equivalent to the folder connection type when you use the lineage harvester.

JDBC data source type	Supported versions	Connection type	Scope	Steps to create technical lineage
Amazon Redshift	1.2.34.1058 and newer	JDBC connection, Shared Storage connection	SQL based input without stored procedures.	Create technical lineage for Amazon Redshift on Edge .

JDBC data source type	Supported versions	Connection type	Scope	Steps to create technical lineage
Azure SQL Data Warehouse	Newest version	JDBC connection, Shared Storage connection	SQL based input and stored procedures. Note Technical lineage cannot be created for views and procedures if the SQL definitions are encrypted in the database.	Create technical lineage for Azure SQL Data Warehouse on Edge.
Azure SQL server	Newest version	JDBC connection, Shared Storage connection	SQL based input and stored procedures. Note Technical lineage cannot be created for views and procedures if the SQL definitions are encrypted in the database.	Create technical lineage for Azure SQL server on Edge.

JDBC data source type	Supported versions	Connection type	Scope	Steps to create technical lineage
Azure Synapse Analytics	Newest version	JDBC connection, Shared Storage connection	SQL based input and stored procedures. Note Technical lineage cannot be created for views and procedures if the SQL definitions are encrypted in the database.	Create technical lineage for Azure Synapse Analytics on Edge.
Databricks Unity Catalog (Beta)	Newest version	Databricks connection	Lineage information from the lineage system tables	Create technical lineage for Databricks Unity Catalog on Edge
Google BigQuery	Newest version	JDBC connection, Shared Storage connection	SQL based input without stored procedures.	Create technical lineage for Google BigQuery on Edge.
Greenplum	6.10 and newer	JDBC connection, Shared Storage connection	SQL based input.	Create technical lineage for Greenplum on Edge.

JDBC data source type	Supported versions	Connection type	Scope	Steps to create technical lineage
HiveQL (SQL-like statements)	2.3.5 and newer	JDBC connection, Shared Storage connection	SQL based input and connection via an AWS host.	Create technical lineage for HiveQL on Edge.
IBM Db2	11.5 and newer	JDBC connection, Shared Storage connection	SQL based input without stored procedures.	Create technical lineage for IBM Db2 on Edge.
Oracle	11g, 12c and newer	JDBC connection, Shared Storage connection	SQL based input and stored procedures.	Create technical lineage for Oracle on Edge.
PostgreSQL	9.4, 9.5 and newer	JDBC connection, Shared Storage connection	SQL based input without stored procedures.	Create technical lineage for PostgreSQL on Edge.

JDBC data source type	Supported versions	Connection type	Scope	Steps to create technical lineage
Microsoft SQL Server	2014, 2016 and newer	JDBC connection, Shared Storage connection	SQL based input and stored procedures. Note Technical lineage cannot be created for views and procedures if the SQL definitions are encrypted in the database.	Create technical lineage for Microsoft SQL Server on Edge.
MySQL	5.7, 8 and newer	JDBC connection, Shared Storage connection	SQL based input without stored procedures.	Create technical lineage for MySQL on Edge.
Netezza	7.2.1.0 and newer	JDBC connection, Shared Storage connection	SQL based input without stored procedures.	Create technical lineage for Netezza on Edge.

JDBC data source type	Supported versions	Connection type	Scope	Steps to create technical lineage
SAP HANA, including SAP HANA Classic on-premises and SAP HANA Cloud/Advanced	2.00.40 and newer for SAP HANA Classic on-premises 4/2023 and newer for SAP HANA Cloud/Advanced	JDBC connection, Shared Storage connection	- SQL-based input. - SAP HANA Information views, which includes attributes, analytic views and calculation views from database table or view data sources. - Script-based calculation views and stored procedures are out of scope.	Create technical lineage for SAP HANA on Edge.
Snowflake	Newest version	JDBC connection, Shared Storage connection	- SQL based input without stored procedures. - SQL-API based input with stored procedures. For more information, go to Technical lineage for Snowflake ingestion methods.	Create technical lineage for Snowflake on Edge.

JDBC data source type	Supported versions	Connection type	Scope	Steps to create technical lineage
Spark SQL	2.4.3 and newer	JDBC connection, Shared Storage connection	SQL-based input without stored procedures and connection via an AWS host. For Spark SQL data source, we recommend using the folder connection type to connect to the directory with your SQL queries.	Create technical lineage for Spark SQL on Edge.
Sybase Adaptive Server Enterprise	16.0 SP02 and newer	JDBC connection, Shared Storage connection	SQL based input without stored procedures.	Create technical lineage for Sybase Adaptive Server Enterprise on Edge.
Teradata	15.0, 16.20.07.01 and newer	JDBC connection, Shared Storage connection	SQL based input, including BTEQ scripts.	Create technical lineage for Teradata on Edge.

Lineage harvester

The following table shows the supported JDBC data sources and driver versions that have been tested. You can connect to them via a JDBC driver or by creating a folder.

JDBC data source type	Supported versions	Connection type	Scope	Steps to create technical lineage
Amazon Redshift	1.2.34.1058 and newer	JDBC, Folder	SQL-based input without stored procedures.	Create technical lineage for Amazon Redshift by using the lineage harvester.
Azure SQL Data Warehouse	Newest version	JDBC, Folder	SQL-based input and stored procedures. Note Technical lineage cannot be created for views and procedures if the SQL definitions are encrypted in the database.	Create technical lineage for Azure SQL Data Warehouse by using the lineage harvester.
Azure SQL server	Newest version	JDBC, Folder	SQL-based input and stored procedures. Note Technical lineage cannot be created for views and procedures if the SQL definitions are encrypted in the database.	Create technical lineage for Azure SQL server by using the lineage harvester.

JDBC data source type	Supported versions	Connection type	Scope	Steps to create technical lineage
Azure Synapse Analytics	Newest version	JDBC, Folder	SQL-based input and stored procedures. Note Technical lineage cannot be created for views and procedures if the SQL definitions are encrypted in the database.	Create technical lineage for Azure Synapse Analytics by using the lineage harvester.
Google BigQuery	Newest version	JDBC, Folder	SQL-based input without stored procedures.	Create technical lineage for Google BigQuery by using the lineage harvester.
Greenplum	6.10 and newer	JDBC, Folder	SQL-based input without stored procedures.	Create technical lineage for Greenplum by using the lineage harvester.

JDBC data source type	Supported versions	Connection type	Scope	Steps to create technical lineage
HiveQL (SQL-like statements)	2.3.5 and newer	JDBC, Folder	SQL-based input and connection via an AWS host. Stored procedures are not supported.	Create technical lineage for HiveQL by using the lineage harvester.
IBM Db2	11.5 and newer	JDBC, Folder	SQL-based input without stored procedures.	Create technical lineage for IBM Db2 by using the lineage harvester.
Oracle	11g, 12c and newer	JDBC, Folder	SQL-based input and stored procedures.	Create technical lineage for Oracle by using the lineage harvester.
PostgreSQL	9.4, 9.5 and newer	JDBC, Folder	SQL-based input without stored procedures.	Create technical lineage for PostgreSQL by using the lineage harvester.

JDBC data source type	Supported versions	Connection type	Scope	Steps to create technical lineage
Microsoft SQL Server	2014, 2016 and newer	JDBC, Folder	SQL-based input and stored procedures. Note Only Basic Authentication is supported. NTLM authentication, for example, is not. Note Technical lineage cannot be created for views and procedures if the SQL definitions are encrypted in the database.	Create technical lineage for Microsoft SQL Server by using the lineage harvester.
MySQL	5.7, 8 and newer	JDBC, Folder	SQL-based input without stored procedures.	Create technical lineage for MySQL by using the lineage harvester.

JDBC data source type	Supported versions	Connection type	Scope	Steps to create technical lineage
Netezza	7.2.1.0 and newer	JDBC, Folder	SQL-based input without stored procedures.	Create technical lineage for Netezza by using the lineage harvester.
SAP HANA, including SAP HANA Classic on-premises and SAP HANA Cloud/Advanced	• 2.00.40 and newer for SAP HANA Classic on-premises • 4/2023 and newer for SAP HANA Cloud/Advanced	JDBC, Folder	• SQL-based input. • SAP HANA Information views, which includes attributes, analytic views and calculation views from database table or view data sources. • Script-based calculation views and stored procedures are out of scope.	Create technical lineage for SAP HANA by using the lineage harvester.
Snowflake	Newest version	JDBC, Folder	• SQL-based input without stored procedures. • SQL-API-based input with stored procedures. For more information, go to Technical lineage for Snowflake ingestion methods.	Create technical lineage for Snowflake by using the lineage harvester.

JDBC data source type	Supported versions	Connection type	Scope	Steps to create technical lineage
Spark SQL	2.4.3 and newer	JDBC, Folder	SQL-based input and connection via an AWS host. Stored procedures are not supported. For Spark SQL data source, we recommend using the folder connection type to connect to the directory with your SQL queries.	Create technical lineage for Spark SQL by using the lineage harvester.
Sybase Adaptive Server Enterprise	16.0 SP02 and newer	JDBC, Folder	SQL-based input without stored procedures.	Create technical lineage for Sybase Adaptive Server Enterprise by using the lineage harvester.
Teradata	15.0, 16.20.07.01 and newer	JDBC, Folder	SQL-based input and stored procedures, including BTEQ scripts.	Create technical lineage for Teradata by using the lineage harvester.

ETL tools

The following table shows the supported ETL tools.

Technical lineage via Edge

The following table lists the supported ETL data sources and connection types you can use when you add capabilities for different data sources. The Shared Storage connection is equivalent to the folder connection type when you use the lineage harvester.

ETL tool	Supported versions	Connection type	Scope	Steps to create technical lineage
Azure Data Factory	2 and newer	API	Commonly supported transformations and activities in Azure Data Factory. For details, go to Supported transformation details .	Create technical lineage for Azure Data Factory on Edge .
dbt	1.4 or newer	API for dbt Cloud Shared Storage connection for dbt Core	Commonly supported model types in dbt. For details, go to Supported transformation details .	Create technical lineage for dbt Cloud on Edge . Create technical lineage for dbt Core on Edge .

ETL tool	Supported versions	Connection type	Scope	Steps to create technical lineage
IBM InfoSphere DataStage	11.5 and newer	Shared Storage connection	Commonly used DataStage ETL components including SQL overrides and transformation details. Collibra Data Lineage supports IBM InfoSphere DataStage transformation logic. You have to prepare a folder with all data objects that you want to process.	Create technical lineage for DataStage on Edge.
Informatica Intelligent Cloud Services, specifically Cloud Data Integration Tip Data Integration is one of the Informatica Intelligent Cloud services.	Cloud, newest only	Informatica Intelligent Cloud Services (IICS) connection Note Collibra Data Intelligence Cloud 2023.03 or newer is required to use the Informatica Intelligent Cloud Services (IICS) connection.	Commonly used transformations in Informatica Intelligent Cloud Services: Data Integration, including SQL overrides. Supported data sources are locally stored flat files and databases.	Create technical lineage for IICS on Edge.

ETL tool	Supported versions	Connection type	Scope	Steps to create technical lineage
Informatica PowerCenter	9.6 and newer	Shared Storage connection	Commonly used transformations in Informatica PowerCenter, including SQL overrides. You have to prepare a folder with all data objects that you want to process.	Create technical lineage for Informatica PowerCenter on Edge.
Matillion	Newest version	Matillion connection Note Collibra Data Intelligence Cloud 2023.03 or newer is required to use the Matillion connection.	SQL based input without stored procedures. Technical lineage via Edge can only access Redshift and Snowflake projects.	Create technical lineage for Matillion on Edge.

ETL tool	Supported versions	Connection type	Scope	Steps to create technical lineage
SQL Server Integration Services (SSIS)	2012 and newer Package format version 6 or newer.	Shared Storage connection	All commonly used transformations in SSIS, data flows and mappings, including SQL overrides. Important SQL statements from Excel are not supported. You have to prepare a folder with all data objects that you want to process.	Create technical lineage for SQL Server Integration Services on Edge.

Lineage harvester

The following table shows the supported ETL tools and driver versions that have been tested. You can connect to them via an API or by creating a folder.

ETL tool	Supported versions	Connection type	Scope	Steps to create technical lineage
Azure Data Factory	2 and newer	API	Commonly supported transformations and activities in Azure Data Factory. For details, go to Supported transformation details .	Create technical lineage for Azure Data Factory by using the lineage harvester.
dbt	1.4 or newer	API for dbt Cloud Folder for dbt Core	Commonly supported model types in dbt. For details, go to Supported transformation details .	Create technical lineage for dbt Cloud by using the lineage harvester. Create technical lineage for dbt Core by using the lineage harvester.

ETL tool	Supported versions	Connection type	Scope	Steps to create technical lineage
IBM InfoSphere DataStage	11.5 and newer	Folder	Commonly used DataStage ETL components including SQL overrides and transformation details. Collibra Data Lineagesupports IBM InfoSphere DataStage transformation logic. You have to prepare a folder with all data objects that you want to process.	Create technical lineage for DataStage by using the lineage harvester.
Informatica Intelligent Cloud Services, specifically Cloud Data Integration Tip Data Integration is one of the Informatica Intelligent Cloud services.	Cloud, newest only	API	Commonly used transformations in Informatica Intelligent Cloud Services: Data Integration, including SQL overrides. Supported data sources are locally stored flat files and databases.	Create technical lineage for Informatica Intelligent Cloud Services by using the lineage harvester.

ETL tool	Supported versions	Connection type	Scope	Steps to create technical lineage
Informatica PowerCenter	9.6 and newer	Folder	Commonly used transformations in Informatica PowerCenter, including SQL overrides. You have to prepare a folder with all data objects that you want to process.	Create technical lineage for Informatica PowerCenter by using the lineage harvester.
Matillion	Newest version	API	SQL based input without stored procedures. The lineage harvester can only access Redshift and Snowflake projects.	Create technical lineage for Matillion by using the lineage harvester.

ETL tool	Supported versions	Connection type	Scope	Steps to create technical lineage
SQL Server Integration Services (SSIS)	2012 and newer Package format version 6 or newer.	Folder	All commonly used transformations in SSIS, data flows and mappings, including SQL overrides. Important SQL statements from Excel are not supported. You have to prepare a folder with all data objects that you want to process.	Create technical lineage for SQL Server Integration Services by using the lineage harvester .

BI tools

The following table shows the supported BI tools.

Technical lineage via Edge

The following table lists the supported BI data sources and connection types you can use when you add capabilities for different data sources.

BI tool	Tested versions	Connection type	Capability	Steps to create technical lineage
Tableau	Newest	API	Technical Lineage for Tableau	Create technical lineage for Tableau on Edge.
Power BI	Newest	API	Technical Lineage for Power BI	Create technical lineage for Power BI on Edge.
MicroStrategy	Newest Note Freeform SQL is supported for reports (not cubes or dossiers) if you have MicroStrategy update10 or newer, or MicroStrategy ONE.	API	Technical Lineage for MicroStrategy	Create technical lineage for MicroStrategy on Edge.

Lineage harvester

The following table shows the supported BI tools.

BI tool	Tested versions	Connection type	Steps to create technical lineage
Tableau	Newest	API. You have to prepare: • lineage harvester configuration file for Tableau ingestion. • Optionally, a Tableau <source ID> configuration file.	Create technical lineage for Tableau by using the lineage harvester .
Power BI	Newest	API. Collibra Data Lineage supports: • Power BI on the Microsoft Power Platform. • Power BI on Fabric. The configuration requirements and the integration are the same, regardless of your setup.	Create technical lineage for Power BI by using the lineage harvester .
Looker	Newest	API. Collibra Data Lineage automatically creates a technical lineage, but stitching is not available. You have to prepare a lineage harvester configuration file for Looker ingestion.	Create technical lineage for Looker by using the lineage harvester .

BI tool	Tested versions	Connection type	Steps to create technical lineage
SQL Server Reporting Services (SSRS) or Power BI Report Server (PBRS)	- SSRS: 2017 and newer Note Due to a bug in 2017 that is resolved by the newer APIs, we recommend using SQL Server 2019 or newer Reporting Services. - PBRS: 2019 and newer	API. You have to prepare: - A lineage harvester configuration file for SSRS-PBRS ingestion. - Optionally, an SSRS-PBRS <source ID> configuration file. Important There are known limitations to the metadata returned by the API when integrating PBRS. For example, Power BI reports in PBRS are ingested as Power BI Report assets in Data Catalog, but there is no technical lineage for the reports.	Create technical lineage for SSRS or PBRS by using the lineage harvester.

BI tool	Tested versions	Connection type	Steps to create technical lineage
MicroStrategy	Newest	Direct connection to the repository. Stitching is not available and there is no true technical lineage. There is only a diagram view that you can access via a Column or Table asset, but not via MicroStrategy assets. You have to prepare a lineage harvester configuration file for MicroStrategy ingestion. You can access: • Microsoft SQL Server repository. • Any local or remote PostgreSQL database. The MicroStrategy Intelligence Server has an embedded PostgreSQL repository, as its default repository. For complete information on the default, embedded	

BI tool	Tested versions	Connection type	Steps to create technical lineage
		repository, see the MicroStrategy repository documentation. For local database access, only PostgreSQL and Microsoft SQL Server repositories are supported. The MicroStrategy Intelligence Server has an embedded PostgreSQL repository, as its default repository. For complete information, see the MicroStrategy repository documentation. You can access: • Microsoft SQL Server repository. • Any local or remote PostgreSQL database. The MicroStrategy Intelligence Server has an embedded PostgreSQL repository, as its default repository. For complete information on the default, embedded	

BI tool	Tested versions	Connection type	Steps to create technical lineage
		repository, see the MicroStrategy repository documentation .	
MicroStrategy (NEW)	Newest Note Freeform SQL is supported for reports (not cubes or dossiers) if you have MicroStrategy update10 or newer, or MicroStrategy ONE.	You have to prepare a lineage harvester configuration file for MicroStrategy ingestion. Benefits of the new integration method include: • Support for the latest MicroStrategy APIs • Support for technical lineage and stitching. • New operating model. • No longer dependent on a direct connection to the repository.	Create technical lineage for MicroStrategy by using the lineage harvester.

Custom technical lineage

You can create a custom technical lineage to include data objects from data sources that are not listed above.

For information on creating a custom technical lineage via Edge, go to [Create technical lineage via Edge for custom technical lineage](#).

For information on creating technical lineage by using the lineage harvester, go [Custom technical lineage via the lineage harvester](#).

Database Owners, BI and ETL Admins, and Collibra Admins

This section aims to provide information that is most relevant for the following people:

- [Database Owners](#), who work with external data sources, to ensure that Collibra can connect to them.
- [BI and ETL Admins](#), who maintain their organizations' BI and ETL platforms and ensure that Collibra can connect to, and communicate with, BI and ETL tools. They also work with Network and Server Admins to configure proxy servers.
- [Collibra Admins](#), who work with Collibra Data Lineage, as well as with Database Owners and BI Admins, to create a technical lineage.

Database Owners	xl
BI and ETL Admins	xlvi
Collibra Admins	xc
Terminology	dcxliii
Methodology	dcxliii
Naming convention	dcxlviii

Database Owners

This section caters primarily to Database Owners, who work with external data sources, to ensure that Collibra can connect to them. Database Owners create databases and ensure that all of the required [data source-specific permissions](#) are met, so that Collibra can successfully connect to them and ingest the metadata.

Data source-specific permissions

Before you can start ingesting metadata, ensure that you meet the required permissions for your specific data source.

Select a data source,
to show the required
permissions.

Currently, information
is shown for:

Choose another data source

Important

- Ensure that you meet the Azure Data Factory-specific permissions described in [Set up Azure Data Factory](#).
- You need read access on information_schema. Only views that you own are processed.
- You need read access on the SYS schema.
- If you are using the lineage harvester, you need read access on information_schema:
 - bigquery.datasets.get
 - bigquery.tables.get
 - bigquery.tables.list
 - bigquery.jobs.create
 - bigquery.routines.get
 - bigquery.routines.list

If you are using Edge, you also need:

- resourcemanager.projects.get
- bigquery.readsessions.create
- bigquery.readsessions.getData
- SELECT, at table level. Grant this to every table for which you want to create a technical lineage.
- Ensure that your service account token has the Read-Only permission.
- Ensure that you have the permission to copy the **target/** directory, which is generated by running the `dbt compile` command, to a local folder.
- You need Monitoring role permissions.
- A role with the LOGIN option.
- SELECT WITH GRANT OPTION, at Table level.
- CONNECT ON DATABASE
- You need read access on the SYS schema and the View Definition Permission in your SQL Server.
- You need read access on definition_schema.
 - GRANT SELECT, at table level. Grant this to every table for which you want to create a technical lineage.

- The role of the user that you specify in the `username` property in lineage harvester configuration file must be the owner of the views in PostgreSQL.
- You need read access on the DBC.
- You need read access to the following dictionary views:
 - `all_tab_cols`
 - `all_col_comments`
 - `all_objects`
 - `ALL_DB_LINKS`
 - `all_mviews`
 - `all_source`
 - `all_synonyms`
 - `all_views`
- Your user role must have privileges to export assets.
- You must have read permission on all assets that you want to export.
- You have at least a Matillion Enterprise license.
- You have generated the Matillion certificate. For more information, go to [Recreating self-signed SSL certificates on a Matillion ETL instance](#).
- You have added the Matillion certificate to a Java truststore. For more information about adding a certificate to a Java truststore, go to [Add a Certificate to a Truststore Using Keytool](#).
- If you encounter the `javax.net.ssl.SSLHandshakeException: General SSLEngine problem` error message, go to [Data Source Name Failed exception with Tableau & technical lineage](#) in Collibra Support Portal for a solution.

The following permissions are required, regardless of the ingestion mode: SQL or SQL-API.

- Ensure that the Snowflake user has the appropriate allowed host list. For details, go to [Allowing Hostnames](#) in Snowflake documentation.
- You need a role that can access the Snowflake shared read-only database. To access the shared database, the account administrator must grant the `IMPORTED PRIVILEGES` privilege on the shared database to the user that runs the lineage harvester.
- If the default role in Snowflake does not have the `IMPORTED PRIVILEGES` privilege, you can use the `customConnectionProperties` property in the lineage harvester configuration file to assign the appropriate role to the user. For example: `"customConnectionProperties": "role=METADATA"`
- The source code files must be in the same directory as your JSON files. For complete information, go to [Working with custom technical lineage](#).
- To stitch the data objects of your data sources with Data Catalog assets, you need to [register](#) your data sources in Data Catalog. When you then

[prepare the Data Catalog physical data layer](#), ensure that you use a structure that matches the structure of ingested assets in Data Catalog.

- Determine whether you want to use the [single-file or batch definition option](#).
- If you choose the single-file definition option, determine whether you want to create a [simple or advanced custom technical lineage](#).
- Before you start the Power BI integration process, you have to perform a number of tasks in Power BI and Microsoft Azure. These tasks, which are performed outside of Collibra, are needed to enable the lineage harvester to reach your Power BI application and collect its metadata. For complete information, go to [Set up Power BI](#).
- Collibra Data Lineage supports:
 - Power BI on the Microsoft Power Platform.
 - Power BI on Fabric.

The configuration requirements and the integration are the same, regardless of your setup.

Before you start the Tableau integration process, you have to perform a number of tasks in Tableau. For complete information, go to the following topics:

- [Set up Tableau](#)
- [Tableau roles and permissions](#)
- You need the following roles, with user access to the server from which you want to ingest:
 - A system-level role that is at least a System user role.
 - An item-level role that is at least a Content Manager role.

We recommend that you use SQL Server 2019 Reporting Services or newer. We can't guarantee that older versions will work.

Before you start the Looker integration process, you need to [set up Looker](#). The following permissions apply only to MicroStrategy on-premises customers.

- You need Admin API permissions.

The first call we make to MicroStrategy is to authenticate. We connect to `<MSTR URL>:<Port>/MicroStrategyLibrary/api-docs/` and use POST `api/auth/login`. You have to ensure that this API call can be made successfully.
- You need permissions to access the library server.
- The lineage harvester uses port 443. If the port is not open, you also need permissions to access the repository.

- If you have a MicroStrategy on-premises environment, you need the permissions for all of the database objects that the lineage harvester accesses.
- You have to configure the MicroStrategy Modeling Service. For complete information, see the [MicroStrategy documentation](#).
- Necessary permissions to all database objects that the lineage harvester accesses.

Show me the data source-specific permissions

Data source	Required permissions
Amazon Redshift	You need read access on information_schema. Only views that you own are processed.
Azure Data Factory	Ensure that you meet the Azure Data Factory-specific permissions described in Set up Azure Data Factory .
Azure SQL server	You need read access on the SYS schema.
Azure Synapse Analytics	SELECT, at table level. Grant this to every table for which you want to create a technical lineage.
Google BigQuery	If you are using the lineage harvester, you need read access on information_schema: ◦ bigquery.datasets.get ◦ bigquery.tables.get ◦ bigquery.tables.list ◦ bigquery.jobs.create ◦ bigquery.routines.get ◦ bigquery.routines.list If you are using Edge, you also need: ◦ resourcemanager.projects.get ◦ bigquery.readsessions.create ◦ bigquery.readsessions.getData
Greenplum	A role with the LOGIN option.
HiveQL	SELECT WITH GRANT OPTION, at Table level.
IBM DB2	CONNECT ON DATABASE

Data source	Required permissions
Informatica Intelligent Cloud Services	◦ Your user role must have privileges to export assets. ◦ You must have read permission on all assets that you want to export.
Matillion	◦ You have at least a Matillion Enterprise license. ◦ You have generated the Matillion certificate. Ensure that the certificate is signed by a certificate authority. Self-signed certificate is not supported when you create technical lineage via Edge. ◦ You have added the Matillion certificate to a Java truststore. For more information about adding a certificate to a Java truststore, go to Add a Certificate to a Truststore Using Keytool.
Microsoft SQL Server	You need read access on the SYS schema and the View Definition Permission in your SQL Server.
MySQL	SELECT, at table level. Grant this to every table for which you want to create a technical lineage.
Netezza	You need read access on definition_schema.
Oracle	You need read access to the following dictionary views: ◦ all_tab_cols ◦ all_col_comments ◦ all_objects ◦ ALL_DB_LINKS ◦ all_mviews ◦ all_source ◦ all_synonyms ◦ all_views
PostgreSQL	◦ GRANT SELECT, at table level. Grant this to every table for which you want to create a technical lineage. ◦ The role of the user that you specify in the <code>username</code> property in lineage harvester configuration file must be the owner of the views in PostgreSQL.

Data source	Required permissions
Snowflake	The following permissions are required, regardless of the ingestion mode: SQL or SQL-API. You need a role that can access the Snowflake shared read-only database. To access the shared database, the account administrator must grant the IMPORTED PRIVILEGES privilege on the shared database to the user that runs the lineage harvester. If the default role in Snowflake does not have the IMPORTED PRIVILEGES privilege, you can use the <code>customConnectionProperties</code> property in the lineage harvester configuration file to assign the appropriate role to the user. For example: <pre>"customConnectionProperties": "role=METADATA"</pre>
Teradata	You need read access on the DBC.

There are no specific permission requirements for this data source type.

There are no specific permissions requirements for downloaded SQL files.

BI and ETL Admins

This section caters primarily to BI and ETL Admins, who maintain their organizations' BI and ETL platforms and ensure that Collibra can connect to, and communicate with, BI and ETL tools. The following are examples of some BI and ETL Admin roles:

- For Tableau:
 - Tableau Site Administrator
 - Tableau Server Administrator
- For Power BI / Azure Data Factory:
 - Power BI Platform Administrator
 - Global Administrator or Azure Cloud Application Administrator
- For Looker: Looker Administrator
- For Matillion: Matillion Administrator

- For MicroStrategy: MicroStrategy System Administrator
- For SQL Server Reporting Services (SSRS): Member of the local administrator group

Set up Azure Data Factory

The lineage harvester uses Azure APIs to get the information necessary to build technical lineage from Azure Data Factory. This topic guides you through the required tasks for registering Azure Data Factory in the Azure Portal and assigning the necessary permissions and access.

Warning Because the tasks covered in this topic are performed outside of Collibra, it is possible that the content changes without us knowing. We strongly recommend that you carefully read the source documentation.

Topics in this section

- [Required values for your Azure Data Factory configuration file](#)
- [Register your Azure Data Factory instance in the Azure Portal](#)
- [Assign the API permissions](#)
- [Create an authentication secret](#)
- [Create an Azure Active Directory group and add your Azure Data Factory instance](#)
- [Add your Azure Data Factory instance to a resource group](#)
- [Retrieve the subscription ID of the resource group](#)
- [Assign read-only permissions to the resource group](#)

Required values for your Azure Data Factory configuration file

The tasks in this topic help you to identify the values you will need when you are preparing the [lineage harvester configuration file](#) for Azure Data Factory. You need the correct values for the properties shown in the following table.

Important If you want to create a technical lineage for more than one Azure Data Factory instance, you need this information for each instance.

Properties	Description
tenantDomain	The directory ID of your Azure Data Factory instance. To get the directory ID, go to Register your Azure Data Factory instance in the Azure Portal .
applicationId	The application ID of your Azure Data Factory instance. Specifically, this is the associated service principal for Azure Data Factory, not the enterprise application ID. To get the application ID, go to Register your Azure Data Factory instance in the Azure Portal .
resourceGroupName	The name of a resource group with the Reader role for the Azure Data Factory instance. To get the resource group name, go to Add your Azure Data Factory instance to a resource group .
subscriptionId	The subscription ID of the resource group. To get the subscription ID, go to Retrieve the subscription ID of the resource group .
password	The secret value for the application ID. To get the service principal secret, go to Create an authentication secret .

Register your Azure Data Factory instance in the Azure Portal

Follow the Microsoft Azure instructions on how to [register an application](#) and refer to the following table for help with the various settings:

Setting	Description
Name	The name of your Azure Data Factory instance.

Setting	Description
Supported account types	The type of tenant. This indicates who can access the Azure Data Factory instance. Select Single tenant.
Redirect URI	The location to which a user's client is redirected and where security tokens are sent after successful authorization. In this case, the redirect URI must be of the type Web. Leave this field empty. You don't have to specify a web location.

» The Azure Portal creates:

- The Application ID. Use this ID as the value for the `applicationId` property in the lineage harvester configuration file.
- The Directory ID. Use this ID as the value for the `tenantDomain` property in the lineage harvester configuration file.

Note When your Azure Data Factory instance is registered, you can find these two IDs in the **Overview pane** on the Azure Portal or in the upper-right menu.

Assign the API permissions

1. In the Azure Portal, click the **Authentication** pane, and then:
 - a. Click the **Advanced settings** section.
 - b. For the **Allow public client flows** option, click **Yes**.
2. Click the **API permissions** pane, and then:
 - a. For the permission type, click **Delegated permissions**.
 - b. Assign the Azure Data Factory instance in Microsoft Azure the Microsoft Graph **User.Read** permission.

» The user now has the following permissions:

- Microsoft Graph
- User.Read

Create an authentication secret

1. In the sidebar navigation, in the **Manage** section, click **Certificates and secrets**.
2. Click **New client secret**. Note that certificates are not supported.
 - a. Enter a description.
 - b. Use the date picker to specify an expiration date for the authentication secret.
 - c. Click **Add**.

» An authentication secret is shown. The authentication secret is the value you will use when prompted for the password to connect to Azure Data Factory.

Important Make note of the authentication secret. For security purposes, It will not be available later. If you lose the authentication secret, you will need to create a new one.

Create an Azure Active Directory group and add your Azure Data Factory instance

1. Go to the Group Management page for your Azure Data Factory instance.
2. Follow the Microsoft Azure instructions on how to [Create and manage an Azure Active Directory \(AD\) Group](#), and refer to the following table for help with the various settings:

Setting	Description
Group Name	The name of the new Azure AD group that you are creating.
Group Type	The type of the Azure AD group. Select Security .
Service Principal	The identity an application uses to access Azure resources and APIs. Enter the Application ID that was generated when you registered Azure Data Factory in the Azure Portal .

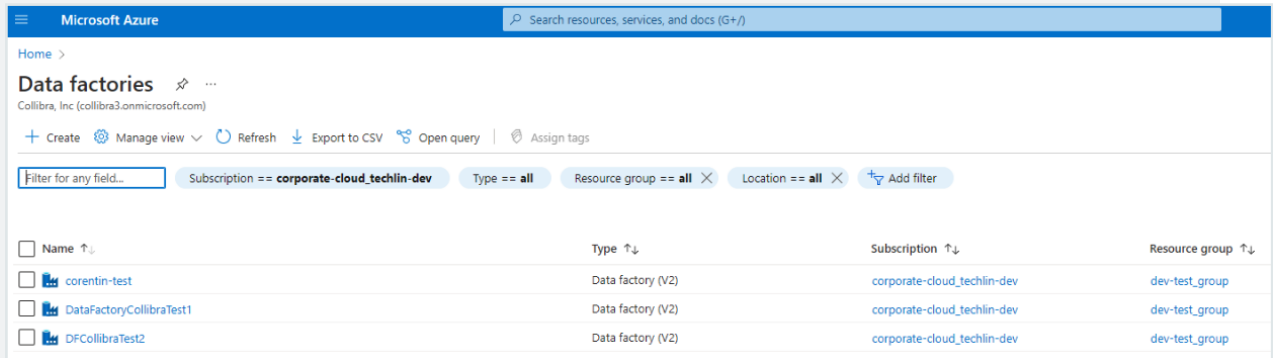
Add your Azure Data Factory instance to a resource group

Your Azure Data Factory instance should already be part of a resource group. If it is, you can skip this step. If it's not, you need to create a resource group and add your Azure Data

Factory instance to it.

The group name is the value you will use for the `resourceGroupName` property in your [Azure Data Factory configuration file](#).

Tip The Data factories page shows all of your Azure Data Factory instances, including their subscriptions and resource groups. Check here to know if your instance is part of a resource group.



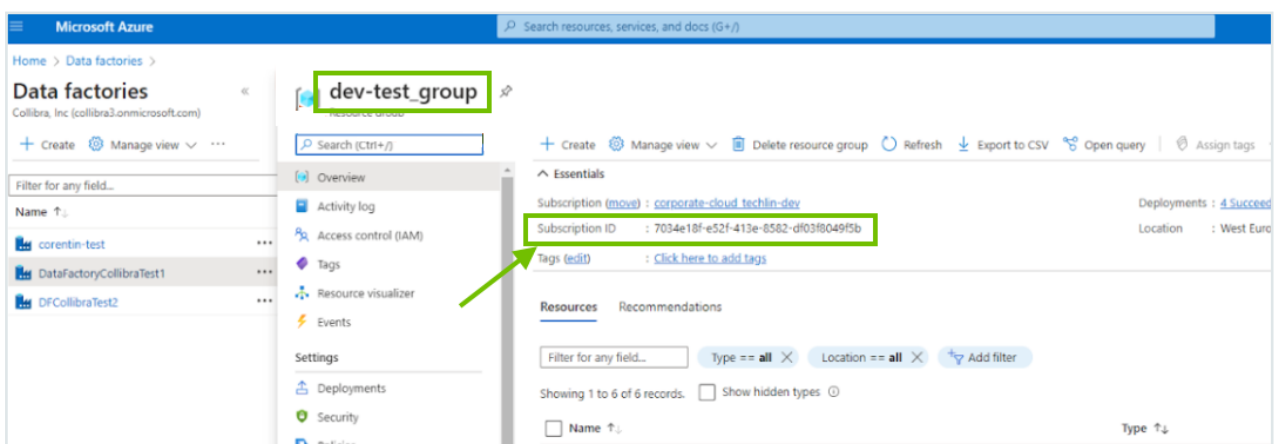
The screenshot shows the 'Data factories' page in the Microsoft Azure portal. At the top, there's a search bar and navigation links. Below, a table lists data factory instances. The table has columns for Name, Type, Subscription, and Resource group. Three instances are listed: 'corentin-test', 'DataFactoryColibraTest1', and 'DFColibraTest2', all of type 'Data factory (V2)' and associated with the 'corporate-cloud_techlin-dev' subscription and 'dev-test_group' resource group.

Name	Type	Subscription	Resource group
corentin-test	Data factory (V2)	corporate-cloud_techlin-dev	dev-test_group
DataFactoryColibraTest1	Data factory (V2)	corporate-cloud_techlin-dev	dev-test_group
DFColibraTest2	Data factory (V2)	corporate-cloud_techlin-dev	dev-test_group

Retrieve the subscription ID of the resource group

On the Data factories page, click the resource group for the Azure Data Factory instance for which you want to create a technical lineage, and make note of the subscription ID.

The subscription ID is the value you will use for the `subscriptionId` property, in your Azure Data Factory configuration file.



The screenshot shows the 'dev-test_group' resource group details page in the Microsoft Azure portal. The 'Subscription ID' is highlighted with a green box and a green arrow pointing to it. The subscription ID is '7034e18f-e52f-413e-8562-df03f0049f5b'. The page also shows the 'Subscription' as 'corporate-cloud_techlin-dev' and the 'Location' as 'West Euro'.

dev-test_group

Subscription (move): corporate-cloud_techlin-dev

Subscription ID : 7034e18f-e52f-413e-8562-df03f0049f5b

Tags (edit) : Click here to add tags

Resources Recommendations

Filter for any field... Type == all Location == all Add filter

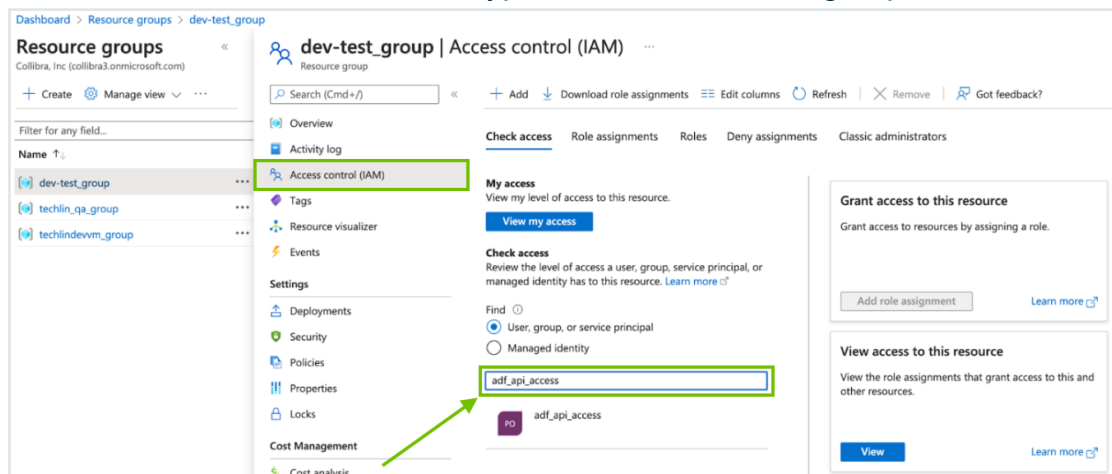
Showing 1 to 6 of 6 records. Show hidden types

Name Type

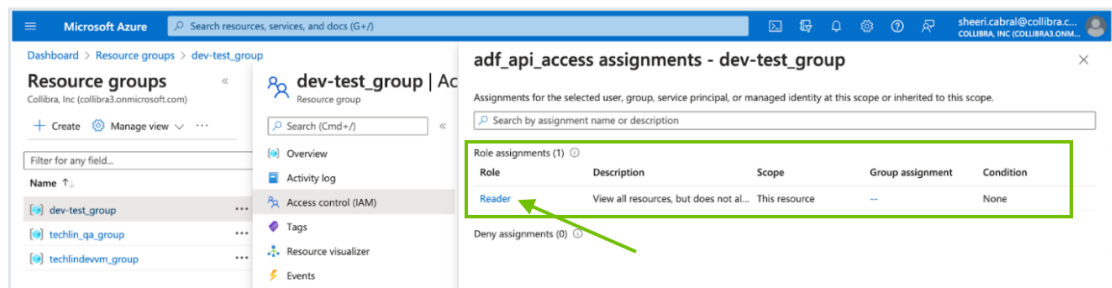
Assign read-only permissions to the resource group

To gather the information needed for technical lineage, the resource group needs permission to read the APIs.

1. Check to see which permissions the resource group has.
 - a. On the Resource groups page, click **Access control (IAM)**.
 - b. In the **Check access** search box, type the name of the AD group.

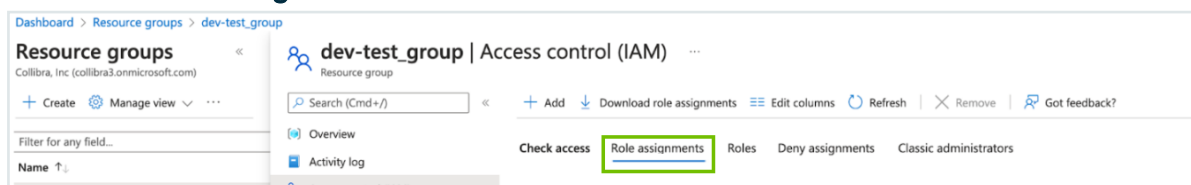


- c. In the search results, click on the AD group to see the access assignments.



- » If your resource group already has the Reader role, as shown in the previous image, this task is complete.
2. If your resource group does not have the Reader role, click **X** in the upper-right corner, to close the Access assignments page.

» The Access control (IAM) page again appears.
 3. Click the **Role assignments** tab.



4. Click **Add > Add role assignment** and follow the Microsoft Azure instructions on how to [add a role assignment](#). Refer to the following table for help with the various settings:

Setting	Description
Roles	The role assignment for the resource group. Select Reader . The lineage harvester only needs read access.
Members	Ensure that the User, group, or service principal radio button is selected. Search for and select the AD group.
Conditions	No conditions are necessary. Click Next .
Review + assign	Click Review +assign , to assign the Reader role to the resource group.

» After a few moments, the read-only permission is assigned to the resource group.

Set up Looker

Before you start the Looker integration, you have to enable Collibra to access your Looker data. The Looker integration process uses a [Looker API](#). To access the Looker metadata, the Looker API uses API3 credentials for authorization and access control.

Prerequisite

- You have the necessary permissions in Looker to see the Looker data.

Steps

1. Create a user with the [Admin role](#).

Tip Only a user with a role that has the Admin permission set can create API3 credentials. Some Looker API calls also require a role that has the Admin permission set.

2. Create the [API3 credentials](#).
3. Use the API3 credentials in your [lineage harvester configuration file](#).

Note API3 credentials are always linked to a Looker user account. As a result, calls to the API only return data that the user is allowed to see.

Tip For more information, see the [Looker documentation](#).

Set up MicroStrategy

The MicroStrategy integration supports both MicroStrategy Cloud and MicroStrategy on-premises environments. Before you start the integration, you have to enable Collibra to access your MicroStrategy data.

Requirements and permissions

The following permissions apply only to MicroStrategy on-premises customers.

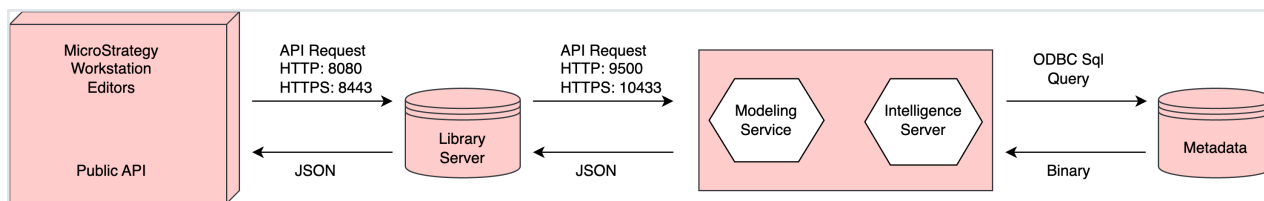
- You need Admin API permissions.
The first call we make to MicroStrategy is to authenticate. We connect to *<MSTR URL>:<Port>/MicroStrategyLibrary/api-docs/* and use POST *api/auth/login*. You have to ensure that this API call can be made successfully.
- You need permissions to access the library server.
- The lineage harvester uses port 443. If the port is not open, you also need permissions to access the repository.
- If you have a MicroStrategy on-premises environment, you need the permissions for all of the database objects that the lineage harvester accesses.
- You have to configure the MicroStrategy Modeling Service. For complete information, see the [MicroStrategy documentation](#).
- Necessary permissions to all database objects that the lineage harvester accesses.

Show me the data source-specific permissions

Data source	Required permissions
Amazon Redshift	You need read access on information_schema. Only views that you own are processed.
Azure Data Factory	Ensure that you meet the Azure Data Factory-specific permissions described in Set up Azure Data Factory .

Data source	Required permissions
Azure SQL server	You need read access on the SYS schema.
Azure Synapse Analytics	SELECT, at table level. Grant this to every table for which you want to create a technical lineage.
Google BigQuery	If you are using the lineage harvester, you need read access on information_schema: - bigquery.datasets.get - bigquery.tables.get - bigquery.tables.list - bigquery.jobs.create - bigquery.routines.get - bigquery.routines.list If you are using Edge, you also need: - resourcemanager.projects.get - bigquery.readsessions.create - bigquery.readsessions.getData
Greenplum	A role with the LOGIN option.
HiveQL	SELECT WITH GRANT OPTION, at Table level.
IBM DB2	CONNECT ON DATABASE
Informatica Intelligent Cloud Services	- Your user role must have privileges to export assets. - You must have read permission on all assets that you want to export.
Matillion	- You have at least a Matillion Enterprise license. - You have generated the Matillion certificate. Ensure that the certificate is signed by a certificate authority. Self-signed certificate is not supported when you create technical lineage via Edge. - You have added the Matillion certificate to a Java truststore. For more information about adding a certificate to a Java truststore, go to Add a Certificate to a Truststore Using Keytool.
Microsoft SQL Server	You need read access on the SYS schema and the View Definition Permission in your SQL Server.

Data source	Required permissions
MySQL	SELECT, at table level. Grant this to every table for which you want to create a technical lineage.
Netezza	You need read access on definition_schema.
Oracle	You need read access to the following dictionary views: ◦ all_tab_cols ◦ all_col_comments ◦ all_objects ◦ ALL_DB_LINKS ◦ all_mviews ◦ all_source ◦ all_synonyms ◦ all_views
PostgreSQL	◦ GRANT SELECT, at table level. Grant this to every table for which you want to create a technical lineage. ◦ The role of the user that you specify in the <code>username</code> property in lineage harvester configuration file must be the owner of the views in PostgreSQL.
Snowflake	The following permissions are required, regardless of the ingestion mode: SQL or SQL-API. You need a role that can access the Snowflake shared read-only database. To access the shared database, the account administrator must grant the IMPORTED PRIVILEGES privilege on the shared database to the user that runs the lineage harvester. If the default role in Snowflake does not have the IMPORTED PRIVILEGES privilege, you can use the <code>customConnectionProperties</code> property in the lineage harvester configuration file to assign the appropriate role to the user. For example: <pre>"customConnectionProperties": "role=METADATA"</pre>
Teradata	You need read access on the DBC.



Important To access MicroStrategy data, you have to use the In-memory Dataset connection method in MicroStrategy, not the Live Connect connection method. If the data is not stored in memory, the MicroStrategy APIs can't access it.

Note Freeform SQL is supported for reports (not cubes or dossiers) if you have MicroStrategy update 10 or newer, or MicroStrategy ONE.

Using a proxy server

If you are using a proxy server, confirm with your Collibra Admin that your [proxy server is configured](#) to access the library server.

Using a custom URL to connect to the MicroStrategy Library Server

Important This requirement applies only if you are [connecting to a proxy server](#).

If you are using a custom URL to connect to the MicroStrategyLibrary Server, you have to specify the URL.

If you are creating a technical lineage:

- [Via Edge](#), specify the URL in the **MicroStrategy Library URL** field, in your Edge capability.
- [Via the lineage harvester](#), use the `microStrategyLibraryUrl` property in your [lineage harvester configuration file](#), to specify the URL.

Set up Power BI

Before you start the Power BI integration process, you have to perform a number of tasks in Power BI and Microsoft Azure. These tasks, which are performed outside of Collibra, are needed to enable the lineage harvester to reach your Power BI application and collect its metadata.

The tasks include the following:

- Attain [authentication](#).
- [Register](#) your Power BI application in Microsoft Azure and set permissions.
- Fulfill the Power BI dedicated capacities and roles requirements for [Power BI workspaces](#).
- Create a cloud security group and add your Power BI application as a member of that group. For complete information on creating a security group and adding members, see [Microsoft's documentation](#).
- Ensure that the lineage harvester can connect to the following URLs:
 - <https://login.microsoftonline.com:443>
 - <https://api.powerbi.com:443>
 - The URL of your Power BI tenant, part of which you have to specify in the `tenantDomain` property of your [lineage harvester configuration file](#).

The [metadata harvesting process](#) explains in detail the prerequisites for enabling the lineage harvester to collect the Power BI metadata.

Note There are some [limitations](#) to the metadata harvesting process. Ensure that you understand these limitations before you start the harvesting process.

Warning Because these tasks are performed outside of Collibra, it is possible that the content changes without us knowing. We strongly recommend that you carefully read the source documentation.

Supported Power BI subscriptions

You need one of the following subscriptions to ingest Power BI metadata in Data Catalog. The metadata collected by the lineage harvester is the same, regardless of your

subscription.

- Power BI Pro.
- Power BI Premium.
- Power BI Premium Per User.

Tip We highly recommend you to have a Power BI Premium subscription.

Power BI ingestion considerations and limitations

There are a few considerations and limitations that you should be aware of when you use the Power BI metadata connector and lineage feature.

General considerations

- Ensure that the lineage harvester can connect to the following URLs:
 - <https://login.microsoftonline.com:443>
 - <https://api.powerbi.com:443>
 - The URL of your Power BI tenant, part of which you have to specify in the `tenantDomain` property of your [lineage harvester configuration file](#).
- The assets created in Collibra have the same names as their counterparts in Power BI. Full names and Display names cannot be changed in Data Catalog.
- Asset types are only created if you have all specific Power BI and Data Catalog permissions.
- The Power BI assets are created in the domain (or domains) that you specify in the Power BI [<source ID> configuration file](#).
- Relations that were created between Power BI assets and other assets via a relation type in the Power BI operating model, are deleted upon synchronization. The same is true of any attribute types in the operating model that you add to Power BI assets. To ensure that the characteristics you add to Power BI assets are not deleted upon synchronization, be sure to use characteristics that are not part of the Power BI operating model.

Supported subscriptions

You need one of the following subscriptions to ingest Power BI metadata in Data Catalog. The metadata collected by the lineage harvester is the same, regardless of your subscription.

- Power BI Pro.
- Power BI Premium.
- Power BI Premium Per User.

Other Power BI subscriptions are currently not supported.

Power BI metadata

Certified data sets and reports

If a data set or report in Power BI is certified, the corresponding Power BI Data Model and Power BI Report assets in Collibra are automatically certified, as identified by the Certified attribute. If, however, certification of a data set or report in Power BI is rescinded, the corresponding assets in Collibra still identify as being certified.

Important Collibra Data Lineage can connect only to datasets that are hosted by Power BI. It cannot connect to externally hosted datasets or models. For complete information, consult Microsoft's [Power BI documentation](#).

Partial access to metadata of certain Power BI elements

The lineage harvester can only partially access metadata of the following Power BI elements:

- [Classic](#) Power BI workspaces, which include My Workspace. Only a full ingestion of new Power BI workspaces is supported.
- Descriptions of most Power BI elements.
- [Power BI apps](#) are not ingested. They can, however, be ingested as Power BI Reports.

Note The prefix "[App]" in the name of a Power BI Report asset indicates that the report is distributed as part of an app, in Power BI. Direct links to app reports are only available if the name of the original report matches the name of the app report, and if the name is unique. In all other cases, the URL on the asset page links to the app, not to the app report.

The lineage harvester cannot access metadata of the following Power BI elements:

- [Tile](#) subtitles.
- Data from external sources supplying the input for the [Power Query](#) expressions in Power BI.

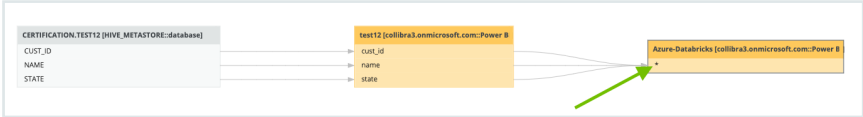
The Power BI API doesn't provide information about the dataset ID for Paginated Reports, therefore lineage for Paginated Reports is not available.

Important The Collibra Data Lineage service can process most, but not all, complex Power BI metadata. This means that the success rate of a Power BI ingestion can be very high, but almost never 100%.

Known issues

The following table presents the known issues of the Power BI integration in Collibra Data Intelligence Cloud.

Known issue	Description
The data set <i>Report Usage Metrics Model</i> cannot be ingested.	The <i>Report Usage Metrics Model</i> is a data set that is automatically created by Power BI. This data set does not contain actual data, which means that they contain nothing to ingest into Data Catalog. However, the lineage harvester still tries to access the metadata and, since there is nothing to access, shows an error message. All error messages about the <i>Report Usage Metrics</i> can be ignored.

Known issue	Description
Report attributes are not returned by the API.	When harvesting Power BI, report attributes are not returned by the API. Therefore, for a given report, Collibra Data Lineage creates a dummy report attribute. This dummy report attribute is identified in the technical lineage by an asterisk (*), as shown in the following example image. Links are drawn from all data attributes in the data set that were used to create the report, to the dummy report attribute.  <pre> graph LR A["CERTIFICATION.TEST12 [HIVE_METASTORE:database] CUST_ID NAME STATE"] --> B["test12 [collibra.onmicrosoft.com:Power BI] cust_id name state"] B --> C["Azure-Databricks [collibra.onmicrosoft.com:Power BI] *"] </pre>
The Power BI API does not return metadata from data sources that are deployed by tabular models in Power BI.	Tabular models are created in Microsoft Visual Studio with the Analysis Services projects extension. They aren't supported for metadata extraction. The API returns the reason for not getting the metadata from the dataset. It's found in the schemaRetrievalError field.
The Power BI API does not support real-time streaming, for example streaming models and data flows.	Real-time datasets, semantic models with object-level security, semantic models with a live connection to AS-Azure and AS on-premises, and Excel full fidelity datasets aren't supported for sub-artifact metadata. For complete information, consult Microsoft's Real-time streaming in Power BI documentation.
Power BI assets that are moved to a different domain are deleted after synchronization.	Warning We highly recommend that you do not move the ingested assets to other domains. If you do, the assets will be deleted and recreated in the initial Data Catalog BI domain (or domains) when you synchronize Power BI. As a result, any manually added characteristics of those assets are lost.
You have successfully ingested Power BI metadata, but calculated tables and columns are not shown in the Technical lineage or in the browse tab pane .	Calculated columns are virtually the same as a non-calculated columns, with one exception: their values are calculated using DAX formulas and values from other columns. Collibra Data Lineage currently does not support internal transformations via DAX language, and any data objects derived via DAX are not shown in the technical lineage or in the browse tab pane. Currently, only M Query/Power Query expressions are supported.

Known issue	Description
You get an error message that mentions one of the following: • “... function not implemented” • “invalid lexical element”	This means that the specific integration feature is not currently supported. Tip You can add your ideas for product enhancements and new features in the Collibra Integrations Ideation Portal.

Power BI authentication

You have to attain authentication to access Power BI metadata. Your authentication method determines how you [retrieve the metadata](#). The lineage harvester supports two authentication methods:

- [Username and password](#)
- [Service principal](#)

The [metadata harvesting process](#) is different for each authentication method. Therefore, different [configurations](#) in Microsoft Azure and Power BI are required.

Note We highly recommend that you use the service principal authentication, as detailed metadata scanning in Power BI is designed for use with service principal authentication.

Tip

You can use a cURL command to check whether or not you can use username and password authentication.

Show me how

Run the following command, where the bolded text refers to your information:

```
curl -v "https://login.microsoftonline.com/<your  
environment>.onmicrosoft.com/oauth2/v2.0/token" -F client_  
id=<your ID> -F "username=<your username>" -F "password=<your  
password"scope=https://analysis.windows.net/powerbi/api/.default" -F  
grant_type=password
```

To check on Windows, follow these steps:

1. Download and [install](#) the cURL Command-Line Tool.
2. In Windows, click **Start > Run**, and then enter *cmd* in the **Run** dialog box.
3. Run the following command, where the bolded text refers to your information:
"https://login.microsoftonline.com/<your
environment>.onmicrosoft.com/oauth2/v2.0/token" -F client_
id=<your ID> -F "username=<your username>" -F
"password=<your **password"scope=https://analysis.windows.net/powerbi/api/.default" -
F grant_type=password**

To ingest Power BI dataflows:

- You need access to the Power BI environment in which the data flow is stored.
- The data set in the data flow must exist in a premium workspace.

Username and password

The username and password authentication method relies on the username, in the form of an email address, and a password you provide to access the Power BI metadata.

To use the username and password authentication method, you need to be an Azure Active Directory user with a Power BI admin role in Power BI.

When you become an Azure Active Directory user, a new email address is created. This email address is the username you use to [sign in](#) to Power BI. You can store the username and password you use to sign in to Power BI in the [lineage harvester configuration file](#).

Note Only Azure Administrators can create users and require them to authenticate via username and password. The Azure Administrator also assigns the user the Power BI admin role. This user is only created for the purpose of Power BI integration in Collibra Data Intelligence Cloud. The user in Azure should have a Member user type.

Service principal

The service principal authentication method allows an Azure Active Directory application to automatically access Power BI content and APIs.

Service principal authentication relies on the Power BI Tenant ID and the Azure Active Directory application ID that you provide in the lineage harvester configuration file. The password you need to access Power BI is the client secret key of the Azure Active Directory application.

To use service principal authentication, you need to [embed Power BI content with a Service Principal and an application secret](#). This entails the following steps:

- In the [Power BI Admin portal](#):
 - [Enable](#) the **Allow service principals to use read-only Power BI admin APIs** option.
 - [Enable](#) the **Allow service principal to use Power BI APIs** option in the Developer settings.

Note This option is no longer required. You can leave it enabled, but you can also safely disable it, if you prefer.

- [Enable](#) the **Enhance admin APIs responses with detailed metadata** option.
- [Enable](#) the **Enhance admin APIs responses with DAX and mashup expressions** option.

Note You need Power BI administrator rights to access the Power BI Admin portal.

Tip Do not confuse the **Allow service principals to use read-only Power BI admin APIs** option with the **Allow service principal to use Power BI APIs** option. You need to enable both options.

Register Power BI in Microsoft Azure and set permissions

Before you set up the lineage harvester, make sure that the harvester can reach Power BI by registering Power BI in Azure and setting the necessary permission to harvest the metadata.

We highly recommend that you read about [supported authentication methods](#) before you register Power BI in Microsoft Azure.

Warning This procedure is performed outside of Collibra. A third-party might change the software without notification, which can render this documentation out-of-date. We highly recommend that you carefully read the source documentation.

Steps

Tip The content in this topic is different for the username / password authentication method or service principal authentication method. We highly recommend that you read the following instructions carefully before you register Power BI in Microsoft Azure:

- [Service principal instructions](#)
- [Username / password instructions](#)

1. Register Power BI in the Azure Portal using the following settings:

Setting	Description
Name	The name of your Power BI application.
Supported account types	The type of tenant. This indicates who can access the Power BI application. In this case, the supported account type must be <i>Single tenant</i>.
Redirect URI	The location to which a user's client is redirected and where security tokens are sent after a successful authorization. In this case, the redirected URI must be <i>Web</i>, but you do not have to specify any web location.

» When you have registered Power BI, the Azure portal creates two important IDs that you need in the [lineage harvester configuration file](#):

- The Application (client) ID
- The Directory (tenant) ID

Note We highly recommend that you store these IDs for further use. You can find the IDs in the **Overview** pane on the Azure portal or in the top right menu.

2. Create a user with the [Power BI Administrator role](#) (only for username / password authentication).

Note

- The user must have administrator rights (such as Office 365 Global Administrator or Power BI Service Administrator) in Power BI. (only for username / password [authentication](#))
- Delegated permissions are supported.

3. In the Azure portal, go to the **Authentication** pane and do the following:
 - a. Go to the **Advanced settings** section.
 - b. Set the **Treat application as a public client** to **Yes**.

Note When Power BI is registered in Microsoft Azure, the **Treat application as a public client** setting label changes to **Allow public client flows**.

4. Go to the **API permissions** pane and do the following:
 - a. Select **Delegated permissions** as permission type.
 - b. Grant the Power BI application in Microsoft Azure the Microsoft Graph `User-Read` permission.
 - c. Grant the Power BI application in Microsoft Azure all Power BI Service permissions (only for username / password [authentication](#)).
 - d. Set **Admin consent required** for `Tenant.ReadAll` permission to **Yes** (only for username / password [authentication](#)).

Note Also ensure that the user who runs the lineage harvester has been granted the Admin consent.

» The user now has the following permissions:

- Microsoft Graph
 - `User.Read`

Important You cannot have any API permissions with Admin consent set to Yes.

- Power BI Service (only for username / password authentication)
 - `App.Read.All`
 - `Capacity.Read.All`
 - `Dashboard.Read.All`
 - `Dataflow.Read.All`
 - `Group.Read.All`
 - `Report.Read.All`
 - `Workspace.Read.All`
 - `Tenant.Read.All`: You need explicit Admin consent. If you have explicit Admin consent, "granted for" is shown in the Status column.

API / Permissions name	Type	Description	Admin consent requ...	Status
▼ Power BI Service (6)				
<code>Tenant.Read.All</code>	Delegated	View all content in tenant	Yes	✔ Granted for Collibra, Inc
<code>Tenant.ReadWrite.All</code>	Delegated	Read and write all content in tenant	Yes	✔ Granted for Collibra, Inc

5. In the [Power BI Admin portal](#), do the following (only for service principal authentication):

- a. [Enable](#) the **Allow service principals to use read-only admin APIs** option.
- b. [Enable](#) the **Allow service principals to use Power BI APIs** option in the Developer settings.

Note This option is no longer required. You can leave it enabled, but you can also safely disable it, if you prefer.

- c. [Enable](#) the **Enhance admin APIs responses with detailed metadata** option.
- d. [Enable](#) the **Enhance admin APIs responses with DAX and mashup expressions** option.
- e. Apply the option to specific security groups.
- f. Enter the name of the security group to which you want to add the service principal.

Warning The Power BI APIs do not support mail-enabled security groups.

Note You need Power BI administrator rights to access the Power BI Admin portal.

6. In the [Power BI Admin portal](#), do the following(Only for username / password authentication):

Note Apply the integration setting to the entire organization (default) or to the specific security group to which your [workspaces](#) belong.

- a. [Enable](#) the **Enhance admin APIs responses with detailed metadata** option.
- b. [Enable](#) the **Enhance admin APIs responses with DAX and mashup expressions** option.

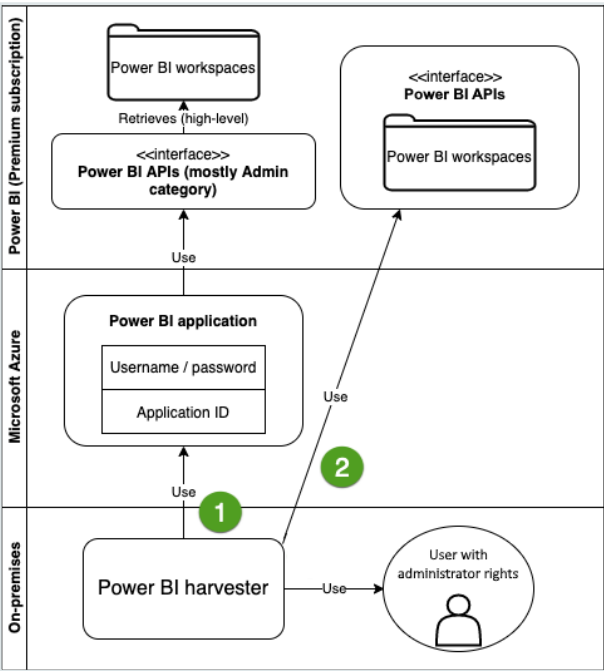
The metadata harvesting process

Collibra uses Power BI REST APIs to harvest Power BI metadata.

To enable the lineage harvester to access metadata in [Power BI workspaces](#), you must have the correct configurations in [Microsoft Azure](#).

Note There are some [limitations](#) to the metadata harvesting process. Ensure that you understand these limitations before you start the harvesting process.

Overview of the metadata harvesting process with username / password authentication

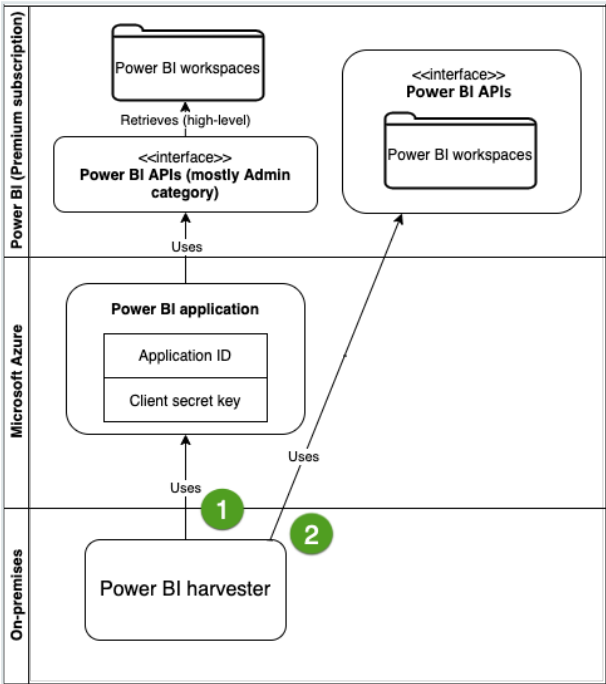


Step	Description
1	The lineage harvester uses the username, password and application ID to access the Power BI APIs. These APIs retrieve basic Power BI metadata, for example metadata in the Power BI tenant or server and reports.
2	The lineage harvester uses Power BI API calls to retrieve more specific metadata, for example Power BI columns and lineage.

Important The Power BI application in Microsoft Azure must be granted administrator rights, such as Office 365 Global Administrator or Power BI Service Administrator. Delegated permissions are supported.

Note The lineage harvester accesses the metadata of all Power BI workspaces. If you don't use filtering, all workspaces are ingested in Collibra. We recommend that you use filtering and domain mapping to structure your Power BI assets in Collibra.

Overview of the metadata harvesting process with service principal authentication



Step	Description
1	The lineage harvester uses the application ID and the client secret key of the Azure Active Directory application to access the Power BI APIs. These APIs retrieve basic Power BI metadata, for example metadata in the Power BI tenant or server and reports.
2	The lineage harvester uses Power BI API calls to retrieve more specific metadata, for example Power BI columns and lineage.

Note The lineage harvester accesses the metadata of all Power BI workspaces. If you don't use filtering, all workspaces are ingested in Collibra. We recommend that you use filtering and domain mapping to structure your Power BI assets in Collibra.

Set up SSRS-PBRS

Before you start the SSRS-PBRS integration, you have to enable Collibra to access your SSRS-PBRS data.

You need the following roles, with user access to the server from which you want to ingest:

- A system-level role that is at least a System user role.
- An item-level role that is at least a Content Manager role.

We recommend that you use SQL Server 2019 Reporting Services or newer. We can't guarantee that older versions will work.

Limitations

- Transformations are not included in the integration. Therefore, no transformations details are shown on the [Sources tab page](#).
- Some of the more complex SQL queries might not be supported.
- There are known limitations to the metadata returned by the API when integrating PBRS. For example, Power BI reports in PBRS are ingested as Power BI Report assets in Data Catalog, but there is no technical lineage for the reports.

SSRS-PBRS roles and permissions

The lineage harvester uses REST API version 2.0 to ingest SQL Server Reporting Services (SSRS) and Power BI Report Server (PBRS) metadata. You need at least the minimum permissions in SSRS-PBRS to enable the lineage harvester to access the SSRS-PBRS metadata and ingest it in Data Catalog.

Important There are known limitations to the metadata returned by the API when integrating PBRS. For example, Power BI reports in PBRS are ingested as Power BI Report assets in Data Catalog, but there is no technical lineage for the reports.

Roles in SSRS-PBRS

Role definitions are either item-level or system-level. SSRS-PBRS include predefined roles that correspond to different levels of user interaction.

- Item-level role definitions determine your access to the content in SSRS-PBRS.
- System-level role definitions determine your access to site operations.

Predefined role	Scope	Description
Content Manager	Item-level	Full access to reports, report models, folders, and other items within the folder hierarchy on the Report Server.
Publisher	Item-level	Add content to a Report Server. Intended for users who author reports or models in Report Designer or Model Designer and then publish those items to a Report Server.
Browser	Item-level	View folders and reports, and subscribe to reports, but not author or manage reports.
Report Builder	Item-level	Load reports in Report Builder, and view and navigate the folder hierarchy. Note Report Builder is a tool for authoring paginated reports that you can publish to the Power BI service. To create and modify reports in Report Builder, you must also have a system role assignment that includes the "Execute report definitions" task. Both predefined system-level roles include this task.
My Reports	Item-level	Publish reports and linked reports. Manage folders, reports, and resources in your My Reports folder. Intended for users of the My Reports feature. It includes tasks that grant administrative permissions to users only over the My Reports folder that they own.
System Administrator	System-level	Gives users full administrative permissions over a Report Server, but not the content in it.
System User	System-level	Allows users to view basic information about a Report Server. It also includes support for loading a report in Report Builder.

For complete information, see the [SSRS documentation](#).

Minimum roles and permissions

The following item-level roles in SSRS provide the required access to content in SSRS-PBRS. With any of the following item-level roles, you can harvest the metadata, ingest the corresponding SSRS assets in Data Catalog and view the lineage between those assets:

- Content Manager
- Publisher
- Report Builder: Only the content contained in your paginated reports.
- My Reports: Only the content in the My Reports folders that you own.

Recommended roles

The Content Manager role is often used with the System Administrator role. The System Administrator role includes tasks that are useful for a Report Server administrator who has overall responsibility for a Report Server, but not necessarily for the content within it. For users who require access to both site-wide operations and items stored on the Report Server, create a second role assignment on the Home folder that includes the Content Manager role. Together, the two role definitions provide a complete set of tasks for users who require full access to all items on a Report Server.

Set up Tableau

Before you start the Tableau integration in Data Catalog, make sure that:

- You have the necessary [role and permissions in Tableau](#).
- You've configured the [mandatory settings in Tableau](#).

Warning Because some of these tasks are performed outside of Collibra, it is possible that the content changes without us knowing. We strongly recommend that you carefully read the source documentation.

Before you run the lineage harvester, you also need to [test](#) your connectivity with the Tableau server.

Important By default, your Tableau token is valid for one year; however, you must use the token at least once every two weeks for it to remain valid. If a token is unused for two weeks, it expires. The life of a token can never be greater than one year, but you can configure it to be less than a year. The two-week usage frequency condition remains valid, regardless.

Tableau ingestion considerations

There are currently three supported methods for integrating Tableau metadata in Data Catalog:

- [Via Edge](#)
- [Via the lineage harvester](#)
- [Via the Data Catalog user interface](#)

Warning As of October 2022, Tableau is enforcing multi-factor authentication for Tableau Cloud Admin users. However, the lineage harvester doesn't support multi-factor authentication. Therefore, Tableau Cloud users with an Admin role must use token-based authentication. This does not affect Tableau Server users or Tableau Cloud users with an Explorer role.

- Data Catalog uses Tableau's REST API to get metadata information and follows Tableau's requirements regarding authentication methods. As such, you need a Tableau user with access to the relevant Tableau sites. For more information, see the [Tableau documentation](#).
- If you use custom SQL that is not supported by the Tableau metadata API, the technical lineage might not be complete. For complete information, see the Tableau documentation on [Tableau Catalog support for custom SQL](#) and [Tableau Lineage and custom SQL connections](#).
- If you use stored procedures, lineage is shown between the Tableau Data Source and the Tableau Worksheet, but the database information is missing, so stitching cannot be achieved.
- Collibra Data Lineage partially supports Unions and Joins. For example, Unions created via the Tableau UI are not represented in Data Catalog. Tableau Data Sources created via custom SQL are supported.

- Hidden Tableau worksheets are currently ingested in Collibra. You can find them by filtering on the attribute “Visible on server”. Hidden worksheets have the value “false”.
- Data fields are ingested with their actual names. Labels and aliases are not returned by the APIs.

Tableau versions and licenses

Before you ingest Tableau metadata in Data Catalog via the lineage harvester, you must ensure that the lineage harvester can access and harvest the Tableau metadata.

Important If you want to create a technical lineage and stitch your Tableau assets to assets in Data Catalog, you must [enable](#) the Tableau metadata API in Tableau.

Supported versions

We will continue to update this list of supported versions, but we don't expect any issues with future versions of Tableau.

- 2023.1
- 2022.x
- 2021.4
- 2021.3
- 2021.2
- 2021.1
- 2020.4
- 2020.3
- 2020.2

License

[Tableau ingestion results](#) depend, in part, on whether or not you have the Data Management Add-on, which requires licensing. For more information about licensing the Data Management Add-on, see the [Tableau documentation](#).

Tableau roles, permissions, and mandatory settings

The lineage harvester uses the Tableau Rest APIs and Tableau Metadata API to ingest the Tableau metadata. You need at least the minimum permissions in Tableau to enable the lineage harvester to access the Tableau metadata and ingest it in Data Catalog.

In this topic

- [First things first: A word about Data Management](#)
- [Permissions on metadata](#)
- [Roles in Tableau](#)
- [Minimum roles and permissions](#)
- [Recommended roles and permissions](#)
- [Mandatory settings in Tableau](#)
- [Best practice: Necessary permissions when using the Explorer role](#)

First things first: A word about Data Management

Consider the following facts:

- To do just about anything with your data in Tableau, including access data objects and visualize lineage, you must have the Data Management add-on and it must be enabled for your Tableau Online account or Tableau Server.
- The Metadata API and Data Management are enabled together, so when you enable one, you automatically enable the other.
- The Metadata API must be enabled in Tableau to create a technical lineage and benefit from automatic stitching.

In short, to integrate Tableau and create a technical lineage, you must have Data Management enabled. Therefore, the following information is built on the assumption that you meet that requirement.

Permissions on metadata

Permissions control who is allowed to see and manage external assets and which metadata (for both Tableau content and external assets) is shown through lineage.

Tip In Tableau, the term "external asset" refers to databases, files and tables that act as Tableau data sources. You need to be able to access external assets if you want to ingest lineage information and benefit from stitching. If you only want to ingest Tableau assets and view the lineage between those assets, it is sufficient to have access only to data objects in Tableau.

No particular role or permissions are needed to allow the lineage harvester access to data objects in Tableau and external assets for which you are the owner. The lineage harvester can automatically access all such data.

Roles in Tableau

The different roles in Tableau allow for different levels of access to data objects in Tableau and external assets.

Viewer role

With the Viewer role, you cannot access external assets, regardless of any other factors, for example even if you are the Project Leader for the projects you want to ingest.

Tableau Data Attributes and Tableau Data Models are ingested as assets in Data Catalog and you can view the lineage for the ingested assets up until the table level only.

Explorer role

With the Explorer role, your access to external assets depends on the following combined factors:

- Whether or not you are a Project Leader for the projects you want to ingest.
- Whether or not derived permissions are turned on in Tableau.

Important If you use the Explorer role, ensure that you configure the [mandatory settings in Tableau](#), as described further on in this topic.

Here are a few tested configurations for the Explorer role:

Combination of accessibility factors	You can access...
• Project leader: Yes • Derived permissions: No	• All Tableau data objects.
• Project leader: Yes • Derived permissions: Yes	• All Tableau data objects. • External assets for which you have derived permissions.
• Project leader: No • Derived permissions: Yes	If you have manually granted permissions for all projects you want to ingest, on all levels, including databases and tables, you can access: • All Tableau data objects for which you have permissions. • External assets for which you have permissions.

Note Data Management allows you (as the person running the lineage harvester) to view in Tableau the external assets for which you are the owner. Without Data Management, some databases, tables and even files might come through, but you can't see them. If you don't have Data Management, we highly recommend that you use a Tableau Server Administrator or Tableau Site Administrator role.

For complete information, see the [Tableau documentation](#).

Tableau Server Administrator or Tableau Site Administrator

With either of these roles, you can access all Tableau data objects and external assets, regardless of any other factors. No permissions need to be configured.

Note Tableau users with a Server Administrator role have access to the entire Tableau Server. Tableau users with a Site Administrator role can only be assigned to specific Tableau sites. As a result, if you have the Site Administrator role, only metadata from specific Tableau sites can be ingested in Data Catalog.

Minimum roles and permissions

To harvest Tableau metadata, you need the following minimum roles and permissions in Tableau:

- You have a View permission on the Tableau projects, workbooks and data sources you want to ingest.
- You have a Viewer role with access to the Tableau REST API.

Important With the minimum roles and permissions, you can harvest Tableau metadata, ingest the corresponding Tableau assets and view the lineage between those assets. However, you cannot access external assets, meaning the databases, files and tables that act as Tableau data sources. Therefore, stitching is not possible.

Recommended roles and permissions

For a full ingestion, you have to be able to access the external assets. We recommend the following roles and permissions in Tableau:

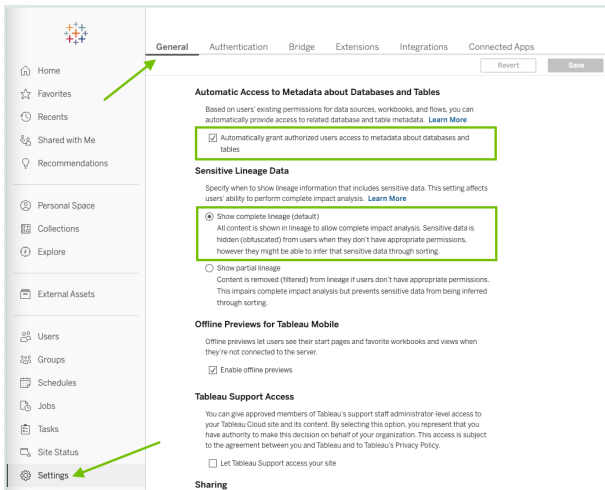
- You have at least a View permission on the Tableau projects, workbooks, data sources, and external assets you want to ingest.
- You have an Administrator role or you have the Explorer role with a sufficient combination of accessibility factors, as previously described in [Explorer role](#).

Mandatory settings in Tableau

If you use the Explorer role, you have to ensure that the lineage harvester can access all of the lineage information. Specifically, as a Tableau administrator, click **Settings > General**, and ensure that the following options are selected:

- **Automatically grant authorized users access to metadata about databases and tables**
- **Show complete lineage (default)**

Show me an image



You also have to select the **Turn on Tableau Catalogue** option, to:

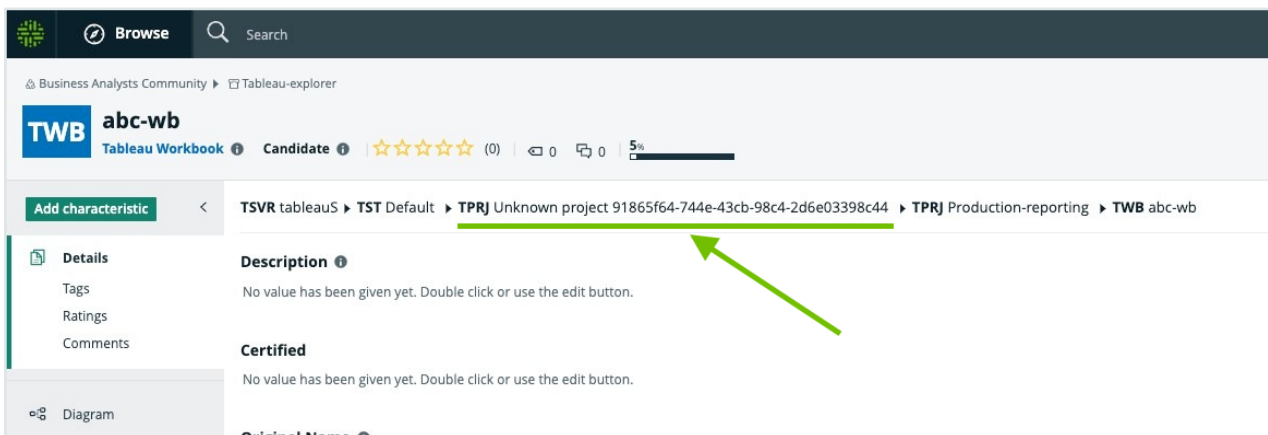
- View lineage and external assets.
- Set permissions on external assets.

Tableau Catalogue

Turning on Catalogue access enables users with appropriate permissions to see metadata, lineage, data quality information and more. In addition to this site setting, you must enable the Metadata API. [Learn more](#)

☒ Turn on Tableau Catalogue

As shown in the following image, if you use the Explorer role and you have access to a subproject, but not the parent project, the parent project is ingested with the Tableau UUID, to maintain the hierarchy of assets.



Best practice: Necessary permissions when using the Explorer role

In this section, we address the permissions you need to attain full lineage and automatic stitching when using the Explorer role in Tableau.

As with any Tableau role, you need to ensure that the [mandatory Tableau settings](#) are configured.

Option 1: Explorer role with Project Leader rights

Project Leaders have full administrative access to the project and its content. This option is better suited for big projects and projects in which the contents frequently change.

Benefits	Limitations
• Permissions are inherited across the entire hierarchy of the project. • The entire project is ingested, so there is no risk of missing anything that you might have missed if you manually applied permissions to the various data objects in the project.	• The entire project is ingested, although you might not want to ingest everything.

Permissions for Project "Snowflake demo"

Asset permissions: Customisable [Edit](#)

[Learn more about permissions](#)

Projects

Workbooks

Data Sources

Data Roles

Flows

Ask Data Lenses

More

Permission Rules

Group/User	Template		
Collibra	Administer	☒	☒
Explorer	Administer	☒	☒

+ Add Group/User Rule

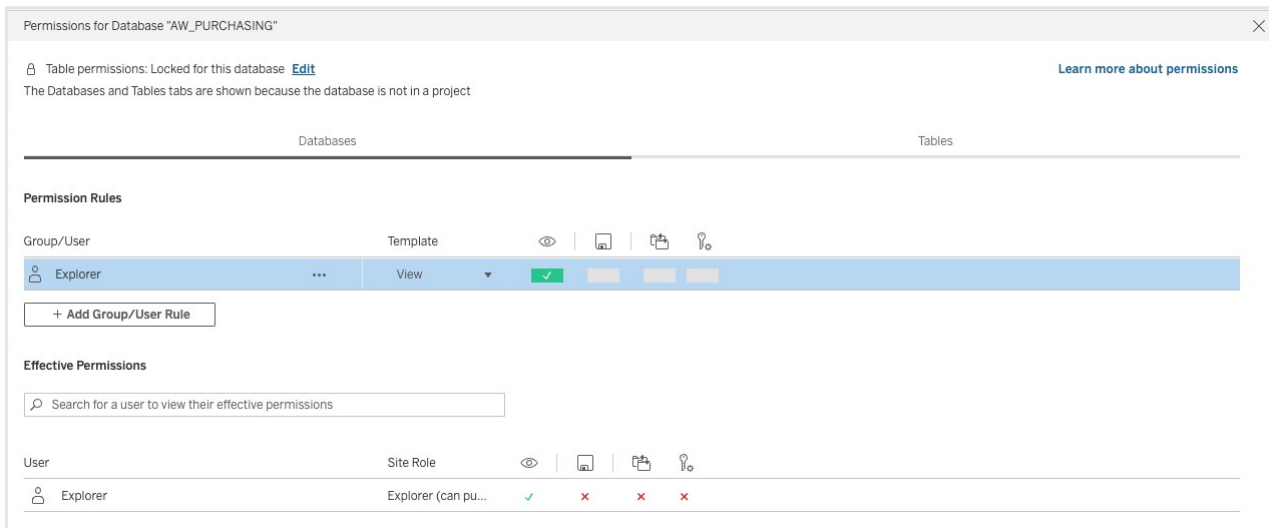
Effective Permissions

Search for a user to view their effective permissions

User	Site Role		
Explorer	Explorer (can pu...	☒	☒

Project Leader status alone doesn't give you access to external assets, which you need to benefit from stitching.

- If Data Management is enabled, you automatically have access to external assets.
- If Data Management is not enabled, you have to manually give the Explorer user access to the database and tables.



Option 2: Explorer role with View permissions

If you have the Explorer role, but you are not the Project Leader for the project you want to ingest, then ensure that you have access to the entire project hierarchy. For example, if you want to ingest a specific worksheet, make sure that you also have permissions to ingest the parent workbook, (sub)projects and related data sources. For each data source, ensure that you have derived permissions to the external assets.

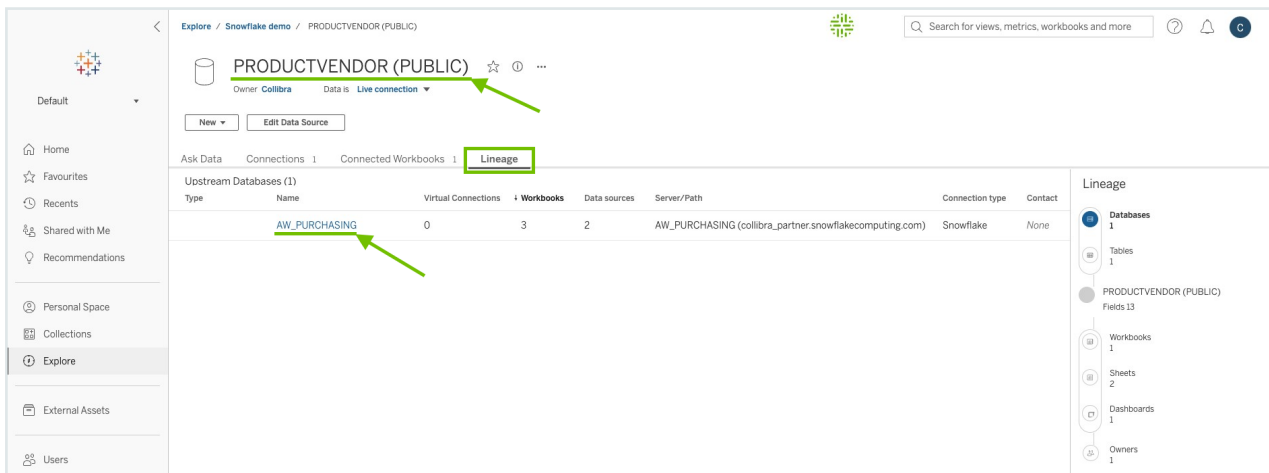
This option is better suited for smaller projects and projects in which the contents don't change very frequently.

Benefits	Limitations
• Highly customizable.	• Difficult to set up and maintain.

How to find the database and tables of a data source

If you don't have Data Management and you, therefore, have to manually give the user access to the database and tables, follow these steps to find the database and tables for a specific data source.

1. Click the Tableau data source or workbook, to open it.
2. Click the **Lineage** tab.
3. Grant access to the database. View permissions are sufficient.
4. Do one of the following:
 - a. Grant access to all tables in the database.
 - b. In the **Lineage** tab, click **Tables** to view the tables that are used in the data source and grant access only to the tables you want to ingest.



Ingestion results per Tableau role

The following tables show the ingestion results based on your [role and permissions in Tableau](#).

By default, the lineage harvester uses both the Tableau REST API and the Tableau Metadata API, but you can limit the ingestion by allowing the lineage harvester to use only the Tableau REST API.

Important The Metadata API must be [enabled](#) to:

- Create a technical lineage and allow the automatic stitching of Column assets to Tableau Data Attribute assets.
- Harvest owner information for Tableau projects, workbooks and data models.

Note If you ingest a Tableau dataset that doesn't have any attributes, asterisks (*) are shown as the Tableau Data Attribute asset names in Collibra.

Tableau site role	Result in Data Catalog
Viewer	Tableau reports and data sources are ingested into Data Catalog, but with a limited scope. Resulting asset types: • Tableau Dashboard • Tableau Data Model • Tableau Project • Tableau Server • Tableau Site • Tableau Workbook • Tableau Worksheet • Tableau Data Attributes Warning Tableau Data Attributes are only ingested if the Metadata API is enabled in Tableau. Important Collibra Data Lineage cannot retrieve lineage information or perform automatic stitching.
Explorer, without access to external assets. For more information, see Tableau roles and permissions.	Tableau reports and data sources are ingested into Data Catalog, but with a limited scope. Resulting asset types: • Tableau Server • Tableau Site • Tableau Project • Tableau Dashboard • Tableau Data Model • Tableau Workbook • Tableau Worksheet • Tableau Data Attributes Warning Tableau Data Attributes are only ingested if the Metadata API is enabled in Tableau. Important We cannot retrieve lineage information or perform automatic stitching. This is the case if you don't have the Data Management add-on or derived permissions for the external assets.

Tableau site role	Result in Data Catalog
One of the following: • Tableau Server Administrator • Tableau Site Administrator • Explorer with access to external assets. For more information, see Tableau roles and permissions.	Data Catalog creates new assets according to your content in Tableau using metadata in Tableau databases and tables. Resulting asset types: • Tableau Server • Tableau Site • Tableau Project • Tableau Dashboard • Tableau Data Model • Tableau Workbook • Tableau Worksheet • Tableau Data Attributes Warning Tableau Data Attributes are only ingested if the Metadata API is enabled in Tableau. Warning The Metadata API must be enabled in Tableau to retrieve lineage information or perform automatic stitching.

Prepare an external directory folder for the lineage harvester

If you want to create a technical lineage for Informatica PowerCenter, SQL Server Integration Services (SSIS), IBM InfoSphere DataStage, or dbt core data sources, you have to prepare a folder with the external directory's data source files.

Tip Go to the [online version](#) of the user guide for more detailed steps and examples.

Prerequisites

- You have IBM InfoSphere Information Server version 11.5 or newer.
- You have Informatica PowerCenter version 9.6 or newer.

- You have SQL Server Integration Services 2012 or newer with package format version 6 or newer.
- You have Microsoft Visual Studio version 2012 or newer.
- You have [downloaded](#) the lineage harvester and you have the necessary [system requirements](#) to run it.
- You have [prepared the physical data layer](#) in Data Catalog.

Note To [stitch](#) the data objects in the source and target data sources in external directories with Data Catalog assets, you first have to [register](#) those data sources in Data Catalog.

Steps to create a technical lineage for dbt Core (Beta)

1. Set the profile of your dbt Core to the environment that you want to retrieve the lineage information from.
2. Take one of the following actions to enable Collibra Data Lineage to access the necessary SQL files and manifest JSON file for creating technical lineage:
 - If the lineage harvester has access to the location where the `dbt run` command is run, the SQL files and manifest JSON file are in the **target/** directory. Specify the `path` property with the path to the **target/** directory in [the lineage harvester configuration file](#).
 - If the lineage harvester does not have access to the location where the `dbt run` command is run, complete the following steps:
 - a. Create a local folder.
 - b. In the dbt Core environment, use the `dbt compile` command to generate SQL files. You can find these compiled SQL files and the manifest JSON file in the **target/** directory of your dbt project. For details, go to [About dbt compile command](#) and [Manifest JSON file](#) in dbt documentation.
 - c. Copy and paste the **target/** directory in your local folder and ensure that you maintain the folder structure. Your local folder must contain all files and subdirectories, such as **target/manifest.json** and **target/models/**.
 - d. Specify the `path` property with the path to the **target/** directory in [the lineage harvester configuration file](#).

Steps to create a technical lineage for Informatica PowerCenter

1. Create a local folder.
2. Export the Informatica objects or repository for which you want to create a technical lineage to the local folder. Make sure to export all objects, parameter files, mappings and sessions at the same time.

Note

- If your folder contains previous versions of the parameter files, objects might be duplicated across different file versions. The duplicated objects cause CollibraData Lineage to ignore some transformations, resulting in missing lineage and error messages. For example, if a parameter file is exported after a column was added to a table, duplicated objects exist if the previous version of the parameter file remains in the folder. To avoid missing lineage, export all objects and parameter files at the same time.
- All XML and parameter files, for example PAR, TXT or PRM files in this folder and its subfolders are taken into account when you create a technical lineage, but Collibra Data Lineage only shows a technical lineage for workflows that have mappings with sources, transformations and targets. Collibra [supports](#) the most common Informatica PowerCenter transformations. For more information, see the [Informatica PowerCenter documentation](#).
- When you export a workflow, ensure that all dependencies are included in the same export file. This applies whether you export the XML file manually or by using the command line. If any dependencies are missing for a workflow, the lineage information for the workflow will be missing in the technical lineage.
- A technical lineage is created when the following tags are present in your XML file:
 - <REPOSITORY>
 - <FOLDER>
 - <SOURCE> / <TARGET>
 - <SESSION>
 - <MAPPING>
 - <TRANSFORMATION> (within a <MAPPING> tag)
- If parameters are missing from the parameter files, the UNRESOLVED PARAMETERS analyze error will be displayed in the analysis results in the Sources tab page. For details, go to [Analyze errors and possible solutions](#) in Technical lineage Sources tab page.

3. In the local folder, create a folder named *techlin-param* and put the parameter files in the *techlin-param* folder.

Steps to create a technical lineage for SQL Server Integration Services

1. Create a local folder.
2. Export the SSIS files for which you want to create a technical lineage.

Tip You can export them directly from the SQL Server Integration Services repository or via Microsoft Visual Studio. For more information, see the [SQL Server Integration Services documentation](#).

3. Store the SSIS files to your local folder. Typically, the folder contains the following files:
 - SSIS package files (DTSX), containing the SQL Server Integration Services source code.
 - Connection manager files (CONMGR), containing environment and connection information.
 - Parameter files (PARAMS), if applicable.

Note

- All files in this folder and subfolders are taken into account when you create a technical lineage. The lineage harvester automatically detects data sources in the SSIS files.
- Not all SSIS files are processed and shown in the technical lineage. The lineage harvester retrieves all of the SSIS package files from the server, but only the files that contain lineage information, meaning those that contain a data flow, or Pipeline, are processed.

Steps to create a technical lineage for DataStage

1. Create a local folder.
2. Export the DataStage project files (DSX) for which you want to create a technical lineage. If you want to include runtime parameters in parameter set files in the

technical lineage, ensure to export DataStage files with executables.

Tip You can either export a DataStage project [manually](#) or automatically via [command line](#).

3. Store the DataStage files in your local folder.
4. Optionally, if your DataStage project uses environment variables, [manually export the environment files](#) (ENV).
5. Give the environment files the same name as the DataStage project files. For example, if your project file is named *datastage-project-1.dmx*, name your environment file *datastage-project-1.env*.
6. Store the environment files in the same local folder.

Important

- Collibra Data Lineage only supports DSX and ENV files.
- You can have one DSX file per DataStage project.
- You can have more than one DSX file in the local folder.
- You can have one or none ENV file per DSX file.
- The name of the DSX file and the ENV file has to be the same.

What's next

For DataStage, Informatica PowerCenter, and SQL Server Integration Services, if the external directory files do not have the necessary information, such as a database and a schema, you can provide the connection definitions manually [in a <source ID> configuration file](#). The information is required for stitching the data sources. If you have multiple connections, provide connection definitions for each connection, regardless of whether the `useCollibraSystemName` property in the [lineage harvester configuration file](#) is set to `true` or `false`.

If the external directory files have all necessary information, you can [specify the remaining lineage harvester configuration file](#) and run the lineage harvester to create a technical lineage.

When you run the lineage harvester, the content in your local folder is sent to Collibra Data Lineage service for processing.

Collibra Admins

This section caters primarily to Collibra Admins, who work with Collibra Data Lineage, as well as with Database Owners and BI Admins, to create a technical lineage.

Create a technical lineage via Edge

This topic provides an overview of the necessary steps to create a technical lineage via Edge.

You can also use the [Collibra Catalog Cloud Ingestions API](#) to create or update a technical lineage capability and start or schedule a synchronization to create a technical lineage. For more information about using APIs, go to [Collibra Developer Portal](#).

To view the steps to create technical lineage for your data source, select the data source and connection type, if applicable. For a listed of supported data sources and their corresponding connection types, go to [Supported data sources for technical lineage](#).

You can use a vaults when creating a connection for your technical lineage capability. For steps on how to use a vault with your connection, see the [online version of this guide](#).

Important Collibra Data Lineage support for Databricks Unity Catalog is in [Beta testing](#), because the system tables feature in Databricks Unity Catalog is in Public Preview. Collibra Data Lineage support for Databricks Unity Catalog will go GA when the system tables feature in Databricks Unity Catalog goes GA.

Before you begin

- Use Collibra Data Intelligence Cloud 2024.02 or later
- Use Collibra Data Intelligence Cloud 2023.03 or later.
- Use Collibra Data Intelligence Cloud 2023.08 or later

- [Create an Edge site](#) in Collibra Data Intelligence Cloud. Ensure that you use Edge 2024.02 or newer.
- [Install an Edge site](#).
- Enable technical lineage via Edge.

To use technical lineage via Edge, you must enable the technical lineage for different data sources. You can also define some specific technical lineage settings on Edge. The technical lineage settings are general configurations that apply to all technical lineage features on Edge.

Prerequisites and permissions

The settings on the **Services Configuration** page are also available in the [DGC service settings](#) in Collibra Console.

Note The **Services Configuration** page is not available by default. For more information, contact [Collibra Support](#).

Steps

- a. Open the DGC service settings for editing:
 - i. Open Collibra Console.
 - » Collibra Console opens with the **Infrastructure** page.
 - ii. In the tab pane, expand an environment to show its services.
 - iii. In the tab pane, click the Data Governance Center service of that environment.
 - iv. Click **Configuration**.
 - v. Click **Edit configuration**.
- b. In the **Lineage on Edge** section, select the value of ✓ True for the technical lineage features that you want to enable.

Setting	Description
Databricks Unity Catalog synchronization via Edge	Enable the function of creating technical lineage for Databricks Unity Catalog. ✓ True Enable this feature. ✗ False Disable this feature.
Power BI and Tableau synchronization enabled	Enable Data Catalog ingestion and technical lineage for Power BI and Tableau via Edge. ✓ True Enable this feature. ✗ False Disable this feature.

Setting	Description
Technical lineage via JDBC connection enabled	Enable the function of creating technical lineage for data sources that are connected by using JDBC connections on Edge. For a list of data sources that use the JDBC connection type, go to Supported data sources for technical lineage. ✓ True Enable this feature. ✗ False Disable this feature.

Setting	Description
Technical lineage for SQL via folder, ETL tools, and custom lineage on Edge enabled	Enable the function of creating technical lineage for data sources that are connected by using Shared Storage connections and APIs. With this function enabled, you can create custom technical lineage and technical lineage for ETL tools and JDBC data sources that are connected by using Shared Storage connections. For a list of data sources and connection types, go to Supported data sources for technical lineage. ✓ True Enable this feature.

Setting	Description
	✕ False Disable this feature.

- c. In the same **Lineage on Edge** section, enter the following information:

Setting	Description
DGC user name	The user name to be used to ingest metadata in Data Catalog via Colibra Data Lineage service.
DGC user password	The password for the user to be used to ingest metadata.

Setting	Description
Collibra system name	Indicates whether you want to use the system or server name of a data source to match to the System asset that you create in Data Catalog. Only set the value to ✓ True when you have multiple databases with the same name. ✗ False - The technical lineage via Edge ignores all system or server names that you specify for the Collibra system name settings when you add technical lineage capabilities. This is the default value. ✓ True - The technical lineage via Edge uses the system or server names that you specify for the Collibra system name settings when you add technical lineage capabilities. Important If you use technical lineage via Edge and the lineage harvester to generate the technical lineage, the following values must be the same: ■ The value for this setting. ■ The value of the <code>useCollibraSystemName</code> property in the lineage harvester configuration file. If the values are different, the technical lineage synchronization ends with an error.

- d. Click **Save all**.
- Enable technical lineage via Edge.
To use technical lineage via Edge, you can define some specific technical lineage settings on Edge. The technical lineage settings are general configurations that apply to all technical lineage features on Edge.

Prerequisites and permissions

The settings on the **Services Configuration** page are also available in the [DGC service settings](#) in Collibra Console.

Note The **Services Configuration** page is not available by default. For more information, contact [Collibra Support](#).

Steps

- a. Open the DGC service settings for editing:
 - i. Open Collibra Console.
 - » Collibra Console opens with the **Infrastructure** page.
 - ii. In the tab pane, expand an environment to show its services.
 - iii. In the tab pane, click the Data Governance Center service of that environment.
 - iv. Click **Configuration**.
 - v. Click **Edit configuration**.

- b. In the same **Lineage on Edge** section, enter the following information:

Setting	Description
DGC user name	The user name to be used to ingest metadata in Data Catalog via Collibra Data Lineage service.
DGC user password	The password for the user to be used to ingest metadata.
Collibra system name	Indicates whether you want to use the system or server name of a data source to match to the System asset that you create in Data Catalog. Only set the value to ✓ True when you have multiple databases with the same name. ✗ False - The technical lineage via Edge ignores all system or server names that you specify for the Collibra system name settings when you add technical lineage capabilities. This is the default value. ✓ True - The technical lineage via Edge uses the system or server names that you specify for the Collibra system name settings when you add technical lineage capabilities. Important If you use technical lineage via Edge and the lineage harvester to generate the technical lineage, the following values must be the same: ■ The value for this setting. ■ The value of the <code>useCollibraSystemName</code> property in the lineage harvester configuration file. If the values are different, the technical lineage synchronization ends with an error.

- c. Click **Save all**.
- Ensure that you meet the minimum network requirements.
The lineage harvester uses the HTTPS protocol by default and uses port 443.

Ensure that you satisfy your organization's firewall rules so that the Edge harvester can connect to all [Collibra Data Lineage service instances](#) in your geographic location:

- 15.222.200.199 (techlin-aws-ca.collibra.com)
- 18.198.89.106 (techlin-aws-eu.collibra.com)
- 13.228.38.245 (techlin-aws-sg.collibra.com)
- 54.242.194.190 (techlin-aws-us.collibra.com)
- 51.105.241.132 (techlin-azure-eu.collibra.com)
- 20.102.44.39 (techlin-azure-us.collibra.com)
- 35.197.182.41 (techlin-gcp-au.collibra.com)
- 34.152.20.240 (techlin-gcp-ca.collibra.com)
- 35.205.146.124 (techlin-gcp-eu.collibra.com)
- 34.87.122.60 (techlin-gcp-sg.collibra.com)
- 35.234.130.150 (techlin-gcp-uk.collibra.com)
- 34.73.33.120 (techlin-gcp-us.collibra.com)

Note Edge connects to different Collibra Data Lineage service instances based on your geographic location and cloud provider. If your location or cloud provider changes, Edge rescans all your data sources. You have to allow all Collibra Data Lineage service instances in your geographic location. In addition, we highly recommend that you always allow the techlin-aws-us instance as a backup, in case Edge cannot connect to other Collibra Data Lineage service instances.

- Connect to a proxy server, if needed.

CollibraData Lineage supports proxy server connection and authentication.

Specifically, the technical lineage capability automatically applies all proxy settings that are configured on Edge.

Warning If you intend to use a custom certificate to allow the Edge capability to connect to your data source, you must enable the `--ca` flag on Edge. However, the `--ca` flag only works in conjunction with the `--proxy` flag. If you enable the `--proxy` flag, you must use a proxy between the Edge capability and the Collibra Data Lineage service instance, or else ingestion will fail. This means that if you use a proxy or custom certificate between the Edge

capability and your data source, you must also use a proxy between Edge and the Collibra Data Lineage service instance, or else ingestion will fail.

Important You can use a man-in-the-middle (MITM) proxy between Edge and the Collibra Data Lineage service instances.

View the results of the testing we did on different proxy combinations for JDBC data source integration

Proxy between Edge and the data source	Proxy between Edge and the Collibra Data Lineage Service instance	Configuration	Connection to the service instance result	Connection to the data source result
No proxy	No proxy	- No proxy.properties - No ca.pem	Supported	Supported
No proxy	Proxy (not MITM)	- No proxy.properties - No ca.pem	Supported	Supported
No proxy	MITM proxy	- No proxy.properties - No ca.pem	Supported	Supported
Proxy (not MITM)	No proxy	- proxy.properties - Added <code>techlin-gcp-dev.collibra.com</code> to <code>noProxy</code> param	Supported	Supported
MITM proxy	No proxy	- proxy.properties - Added <code>techlin-gcp-dev.collibra.com</code> to <code>noProxy</code> param - ca.pem	Supported	Supported
Proxy (not MITM)	Proxy (not MITM)	proxy.properties	Supported	Supported

Proxy between Edge and the data source	Proxy between Edge and the Collibra Data Lineage Service instance	Configuration	Connection to the service instance result	Connection to the data source result
MITM proxy	MITM proxy	◦ proxy.properties ◦ ca.pem	Supported	Supported

For Collibra Data Lineage to connect to a data source that requires a proxy, you can configure a forward proxy on Edge when you install your Edge site. For detailed steps, go to [configure a forward proxy on Edge](#).

When you add and synchronize the technical lineage capability on Edge, the technical lineage capability uses these proxy settings to establish connections to your data source to collect the metadata and create the technical lineage.

Proxy server support is added to each technical lineage capability. If you encounter a connection error while synchronizing a technical lineage capability for a data source with a forward proxy configured, contact [Collibra Support](#).

- Connect to a proxy server, if needed.

CollibraData Lineage supports proxy server connection and authentication.

Specifically, Collibra Data Lineage automatically applies all proxy settings that are configured on Edge.

Warning If you intend to use a custom certificate to allow the Edge capability to connect to your data source, you must enable the `--ca` flag on Edge. However, the `--ca` flag only works in conjunction with the `--proxy` flag. If you enable the `--proxy` flag, you must use a proxy between the Edge capability and the Collibra Data Lineage service instance, or else ingestion will fail. This means that if you use a proxy or custom certificate between the Edge capability and your data source, you must also use a proxy between Edge and the Collibra Data Lineage service instance, or else ingestion will fail.

Important You can use a man-in-the-middle (MITM) proxy between Edge and the Collibra Data Lineage service instances.

View the results of testing we did on different proxy combinations for Power BI integration

Proxy between Edge and Power BI	Proxy between Edge and the Collibra Data Lineage Service instance	Configuration	Connection to the service instance result	Connection to Power BI result
No proxy	No proxy	- No proxy.properties - No ca.pem	Supported	Supported
No proxy	Proxy (not MITM)	proxy.properties Added *.powerbi.com, login.microsoftonline.com to noProxy param	Supported	Supported
No proxy	MITM proxy	- proxy.properties Added *.powerbi.com, login.microsoftonline.com to noProxy param - ca.pem	Supported	Supported
Proxy (not MITM)	No proxy	proxy.properties Added techlin-gcp-dev.collibra.com to noProxy param	Supported	Supported
MITM proxy	No proxy	- proxy.properties Added techlin-gcp-dev.collibra.com to noProxy param - ca.pem	Couldn't verify	Not supported

Proxy between Edge and Power BI	Proxy between Edge and the Collibra Data Lineage Service instance	Configuration	Connection to the service instance result	Connection to Power BI result
Proxy (not MITM)	Proxy (not MITM)	proxy.properties	Supported	Supported
MITM proxy	MITM proxy	- proxy.properties - ca.pem	Supported	Not supported, even with custom certificate in the connection.

For Collibra Data Lineage to connect to a data source that requires a proxy, you can configure a forward proxy on Edge when you install your Edge site. For detailed steps, go to [configure a forward proxy on Edge](#).

When you add and synchronize the technical lineage capability on Edge, Collibra Data Lineage uses these proxy settings to establish connections to your data source to collect the metadata and create the technical lineage.

Proxy server support is added to each technical lineage capability. If you encounter a connection error while synchronizing a technical lineage capability for a data source with a forward proxy configured, contact [Collibra Support](#).

- Connect to a proxy server, if needed.

CollibraData Lineage supports proxy server connection and authentication. Specifically, Collibra Data Lineage automatically applies all proxy settings that are configured on Edge.

Warning If you intend to use a custom certificate to allow the Edge capability to connect to your data source, you must enable the `--ca` flag on Edge. However, the `--ca` flag only works in conjunction with the `--proxy` flag. If you enable the `--proxy` flag, you must use a proxy between the Edge capability and the Collibra Data Lineage service instance, or else ingestion will fail. This means that if you use a proxy or custom certificate between the Edge capability and your data source, you must also use a proxy between Edge and the Collibra Data Lineage service instance, or else ingestion will fail.

Important You can use a man-in-the-middle (MITM) proxy between Edge and the Collibra Data Lineage service instances.

View the results of testing we did on different proxy combinations for MicroStrategy integration

Proxy between Edge and MicroStrategy	Proxy between Edge and the Collibra Data Lineage Service instance	Configuration	Connection to the service instance result	Connection to MicroStrategy result
No proxy	No proxy	- No proxy.properties - No ca.pem	Supported	Supported
No proxy	Proxy (not MITM)	proxy.properties Added ec2-18-216-24-232.us-east-2.compute.amazonaws.com to noProxy param	Supported	Supported, but verified for HTTP source only, not HTTPS.

Proxy between Edge and MicroStrategy	Proxy between Edge and the Collibra Data Lineage Service instance	Configuration	Connection to the service instance result	Connection to MicroStrategy result
No proxy	MITM proxy	◦ proxy.properties Added <code>ec2-18-216-24-232.us-east-2.compute.amazonaws.com</code> to <code>noProxy</code> param ◦ ca.pem	Supported	Supported, but verified for HTTP source only, not HTTPS.
Proxy (not MITM)	No proxy	◦ proxy.properties Added <code>techlin-gcp-dev.collibra.com</code> to <code>noProxy</code> param	Supported	Supported
MITM proxy	No proxy	◦ proxy.properties Added <code>techlin-gcp-dev.collibra.com</code> to <code>noProxy</code> param ◦ ca.pem	Not supported	Supported, but verified for HTTP source only, not HTTPS.
Proxy (not MITM)	Proxy (not MITM)	◦ proxy.properties	Supported	Supported, but verified for HTTP source only, not HTTPS.
MITM proxy	MITM proxy	◦ proxy.properties ◦ ca.pem	Supported	Not verified

For Collibra Data Lineage to connect to a data source that requires a proxy, you can configure a forward proxy on Edge when you install your Edge site. For detailed steps, go to [configure a forward proxy on Edge](#).

When you add and synchronize the technical lineage capability on Edge, Collibra Data Lineage uses these proxy settings to establish connections to your data source to collect the metadata and create the technical lineage.

Proxy server support is added to each technical lineage capability. If you encounter a connection error while synchronizing a technical lineage capability for a data source with a forward proxy configured, contact [Collibra Support](#).

- Connect to a proxy server, if needed.
- CollibraData Lineage supports proxy server connection and authentication. Specifically, the technical lineage capability automatically applies all proxy settings that are configured on Edge.

Warning If you intend to use a custom certificate to allow the Edge capability to connect to your data source, you must enable the `--ca` flag on Edge. However, the `--ca` flag only works in conjunction with the `--proxy` flag. If you enable the `--proxy` flag, you must use a proxy between the Edge capability and the Collibra Data Lineage service instance, or else ingestion will fail. This means that if you use a proxy or custom certificate between the Edge capability and your data source, you must also use a proxy between Edge and the Collibra Data Lineage service instance, or else ingestion will fail.

Important You can use a man-in-the-middle (MITM) proxy between Edge and the Collibra Data Lineage service instances.

View the results of testing we did on different proxy combinations for Azure Data Factory integration

Proxy between Edge and Azure Data Factory	Proxy between Edge and the Collibra Data Lineage Service instance	Configuration	Connection to the service instance result	Connection to Azure Data Factory result
No proxy	No proxy	◦ No proxy.properties ◦ No ca.pem	Supported	Supported

Proxy between Edge and Azure Data Factory	Proxy between Edge and the Collibra Data Lineage Service instance	Configuration	Connection to the service instance result	Connection to Azure Data Factory result
No proxy	Proxy (not MITM)	◦ proxy.properties Added <code>management.azure.com</code> to <code>noProxy</code> param	Supported	Supported
No proxy	MITM proxy	◦ proxy.properties Added <code>management.azure.com</code> to <code>noProxy</code> param ◦ <code>ca.pem</code>	Not verified	Not supported
Proxy (not MITM)	No proxy	◦ proxy.properties Added <code>techlin-gcp-dev.collibra.com</code> to <code>noProxy</code> param	Supported	Supported
MITM proxy	No proxy	◦ proxy.properties Added <code>techlin-gcp-dev.collibra.com</code> to <code>noProxy</code> param ◦ <code>ca.pem</code>	Not supported	Fails to harvester metadata, but no error is raised. Empty SQLITE file is generated.
Proxy (not MITM)	Proxy (not MITM)	◦ proxy.properties	Supported	Supported

Proxy between Edge and Azure Data Factory	Proxy between Edge and the Collibra Data Lineage Service instance	Configuration	Connection to the service instance result	Connection to Azure Data Factory result
MITM proxy	MITM proxy	◦ proxy.properties ◦ ca.pem	Supported	Fails to harvester metadata, but no error is raised. Empty SQLITE file is generated.

For Collibra Data Lineage to connect to a data source that requires a proxy, you can configure a forward proxy on Edge when you install your Edge site. For detailed steps, go to [configure a forward proxy on Edge](#).

When you add and synchronize the technical lineage capability on Edge, the technical lineage capability uses these proxy settings to establish connections to your data source to collect the metadata and create the technical lineage.

Proxy server support is added to each technical lineage capability. If you encounter a connection error while synchronizing a technical lineage capability for a data source with a forward proxy configured, contact [Collibra Support](#).

- Connect to a proxy server, if needed.
Collibra Data Lineage supports proxy server connection and authentication. Specifically, the technical lineage capability automatically applies all proxy settings that are configured on Edge. Technical lineage for Databricks Unity Catalog also supports a forward proxy that decrypts TLS traffic, a so-called man-in-the-middle (MITM) proxy.

Warning If you intend to use a custom certificate to allow the Edge capability to connect to your data source, you must enable the `--ca` flag on Edge. However, the `--ca` flag only works in conjunction with the `--proxy` flag. If you enable the `--proxy` flag, you must use a proxy between the Edge capability and the Collibra Data Lineage service instance, or else ingestion will fail. This means that if you use a proxy or custom certificate between the Edge capability and your data source, you must also use a proxy between Edge and the Collibra Data Lineage service instance, or else ingestion will fail.

The following proxies are supported when you create technical lineage for Databricks Unity Catalog:

- Path through (No authentication)
- Path through (Basic authentication)
- MITM (No authentication)
- MITM (Basic authentication)
- No proxy for noProxy hosts defined by Edge

For Collibra Data Lineage to connect to a data source that requires a proxy, configure a forward proxy on Edge when you install your Edge site. For detailed steps, go to [configure a forward proxy on Edge](#).

When you add and synchronize the technical lineage capability on Edge, the technical lineage capability uses these proxy settings to establish connections to your data source to collect the metadata and create the technical lineage.

Important Proxy server support is added to each technical lineage capability. If you encounter a connection error while synchronizing a technical lineage capability for a data source with a forward proxy configured, contact [Collibra Support](#).

- Connect to a proxy server, if needed.
CollibraData Lineage supports proxy server connection and authentication. Specifically, Collibra Data Lineage automatically applies all proxy settings that are configured on Edge.

Warning If you intend to use a custom certificate to allow the Edge capability to connect to your data source, you must enable the `--ca` flag on Edge. However, the `--ca` flag only works in conjunction with the `--proxy` flag. If you enable the `--proxy` flag, you must use a proxy between the Edge capability and the Collibra Data Lineage service instance, or else ingestion will fail. This means that if you use a proxy or custom certificate between the Edge capability and your data source, you must also use a proxy between Edge and the Collibra Data Lineage service instance, or else ingestion will fail.

Important You can use a man-in-the-middle (MITM) proxy between Edge and the Collibra Data Lineage service instances.

View the results of testing we did on different proxy combinations for Tableau integration

Proxy between Edge and Tableau	Proxy between Edge and the Collibra Data Lineage Service instance	Configuration	Connection to the service instance result	Connection to Tableau result
No proxy	No proxy	- No <code>proxy.properties</code> - No <code>ca.pem</code>	Supported	Supported
No proxy	Proxy (not MITM)	<code>proxy.properties</code> Added <code>prod-useast-b.online.tableau.com</code> to <code>noProxy</code> param	Supported	Supported
No proxy	MITM proxy	<code>proxy.properties</code> Added <code>prod-useast-b.online.tableau.com</code> to <code>noProxy</code> param <code>ca.pem</code>	Supported	Supported

Proxy between Edge and Tableau	Proxy between Edge and the Collibra Data Lineage Service instance	Configuration	Connection to the service instance result	Connection to Tableau result
Proxy (not MITM)	No proxy	◦ proxy.properties Added <code>techlin-gcp-dev.collibra.com</code> to <code>noProxy</code> param	Supported	Supported
MITM proxy	No proxy	◦ proxy.properties Added <code>techlin-gcp-dev.collibra.com</code> to <code>noProxy</code> param ◦ ca.pem	Not supported	Supported
Proxy (not MITM)	Proxy (not MITM)	◦ proxy.properties	Supported	Supported
MITM proxy	MITM proxy	◦ proxy.properties ◦ ca.pem	Supported	Supported

For Collibra Data Lineage to connect to a data source that requires a proxy, you can configure a forward proxy on Edge when you install your Edge site. For detailed steps, go to [configure a forward proxy on Edge](#).

When you add and synchronize the technical lineage capability on Edge, Collibra Data Lineage uses these proxy settings to establish connections to your data source to collect the metadata and create the technical lineage.

Proxy server support is added to each technical lineage capability. If you encounter a connection error while synchronizing a technical lineage capability for a data source with a forward proxy configured, contact [Collibra Support](#).

- [Register the data source via Edge](#). Before you register the data source, ensure that you [add the Catalog JDBC ingestion capability](#), so that CollibraData Lineage can stitch the data objects in your technical lineage to the assets in Data Catalog.
- [Integrate Databricks Unity Catalog or register a Databricks file system](#).

Requirements and permissions

The following requirements and permissions are needed for the technical lineage process. Additional, Edge-related roles and resources, are mentioned in each of the specific steps.

- A [global role](#) with the following [global permissions](#):
 - Data Stewardship Manager
 - Manage all resources
 - System administration
 - Technical lineage
- A [resource role](#) with the following [resource permission](#) on the community level in which you created the BI Data Catalog domain:
 - Asset: add
 - Attribute: add
 - Domain: add
 - Attachment: add
- As a technical lineage user, ensure that your Catalog Author [global role](#) has the following [global permissions](#). With these permissions, Collibra Data Lineage can process the lineage and synchronize the results to Data Catalog to create technical lineage.
 - Catalog > Advanced Data Type > Add
 - Catalog > Advanced Data Type > Remove
 - Catalog > Advanced Data Type > Update
 - Catalog > Technical lineage
- As a Data Catalog user, ensure that your Edge integration engineer global role has the following global permissions. With these permissions, you can create connections and capabilities on Edge, configure the integration, and synchronize the integration.
 - Manage connections and capabilities
 - View Edge connections and capabilities
- As a Databricks Unity Catalog user, ensure that you have the following permissions in Databricks. The access token of this user must be specified in the Databricks

connection so that Colibra Data Lineage can access the system tables (Public Preview) after connecting to Databricks Unity Catalog.

- Enable the lineage system tables.
- Have the USE CATALOG privilege to the `system` catalog.

Example

Catalogs >

 **system**

Owner: System user 

Tags: Add tags

System catalog (auto-created)  

Schemas

Details

Permissions

Grant

Revoke

Type to filter by

Principal	Privilege	Object
account users	USE CATALOG	 system

- USE_SCHEMA, and SELECT privileges to the `system.access` schema.

Example

Catalogs > system >

system.access

Owner: System user 

Tags:

Comment:

Tables Volumes Models Functions Details **Permissions**

Grant

Revoke

Principal	Privileges
account users	SELECT
account users	USE SC

For details, go to [Enable system tables](#) and [Grant access to system tables](#) in Databricks documentation.

How to verify whether you have the right accesses in Databricks?

- Log into Databricks with the credentials that you use to [create a Databricks connection](#) on Edge.
- Run the following SQL query against your `system.access` catalog and schema in Unity Catalog:

```
SELECT count(*) FROM column_lineage
```

If you do not have the right accesses, the [Could not get column lineage data](#) error occurs when you synchronize the Technical Lineage for

Databricks Unity Catalog capability. Contact Databricks support if you encounter issues on getting access to the system tables.

- Necessary permissions to all [database objects](#) that technical lineage via Edge accesses.

Tip Some data sources require specific permissions. For the data source selected above:

You need read access on the SYS schema.

You need read access on the SYS schema and the View Definition

Permission in your SQL Server.

You need read access on information_schema:

- `bigquery.datasets.get`
- `bigquery.tables.get`
- `bigquery.tables.list`
- `bigquery.jobs.create`
- `resourcemanager.projects.get`
- `bigquery.routines.get`
- `bigquery.routines.list`
- `bigquery.readsessions.create`
- `bigquery.readsessions.getData`
- GRANT SELECT, at table level. Grant this to every table for which you want to create a technical lineage.
- The role of the user that you specify in the `username` property in lineage harvester configuration file must be the owner of the views in PostgreSQL.

The role of the user must be the owner of the views in PostgreSQL, and the username of the user must be specified in [the JDBC connection](#) that you use to access PostgreSQL.

You need read access on information_schema. Only views that you own are processed.

Ensure that your service account token has the Read-Only permission.

Ensure that you have the permission to copy the **target/** directory, which is generated by running the `dbt compile` command, to a Shared Storage connection folder. For more information about the Shared Storage connection folder, go to Step 1 Create a Shared Storage connection.

SELECT, at table level. Grant this to every table for which you want to create a technical lineage.

A role with the LOGIN option.

SELECT WITH GRANT OPTION, at Table level.

CONNECT ON DATABASE

The following permissions are required, regardless of the ingestion mode: SQL or SQL-API.

- Ensure that the Snowflake user has the appropriate allowed host list. For details, go to [Allowing Hostnames](#) in Snowflake documentation.
- You need a role that can access the Snowflake shared read-only database. To access the shared database, the account administrator must grant the [IMPORTED PRIVILEGES](#) privilege on the shared

database to the user. The username of the user must be specified in [the JDBC connection](#) that you use to access Snowflake.

- If the default role in Snowflake does not have the IMPORTED PRIVILEGES privilege, you can click the **Add property** button to add a custom parameter with the following values specified:

Field	Value
Name	customConnectionProperties
Type	Text
Encryption	Select one of the following encryption methods: ◦ Not encrypted (plain text) ◦ Encrypted with public key ◦ To be encrypted by Edge management server
Value	role=METADATA

You need read access on the DBC.

You need read access to the following dictionary views:

- all_tab_cols
- all_col_comments
- all_objects
- ALL_DB_LINKS
- all_mviews
- all_source
- all_synonyms
- all_views

You need read access on definition_schema.

- Your user role must have privileges to export assets.
- You must have read permission on all assets that you want to export.
- You have at least a Matillion Enterprise license.
- You have generated the Matillion certificate. Ensure that the certificate is signed by a certificate authority. Self-signed certificate is not supported when you create technical lineage via Edge.
- You have added the Matillion certificate to a Java truststore. For more information about adding a certificate to a Java truststore, go to [Add a Certificate to a Truststore Using Keytool](#).

The following permissions apply only to MicroStrategy on-premises customers.

- You need Admin API permissions.

The first call we make to MicroStrategy is to authenticate. We connect to

<MSTR URL>:<Port>/MicroStrategyLibrary/api-docs/ and use POST api/auth/login. You have to ensure that this API call can be made successfully.

- You need permissions to access the library server.
- The lineage harvester uses port 443. If the port is not open, you also need permissions to access the repository.
- If you have a MicroStrategy on-premises environment, you need the permissions for all of the database objects that the lineage harvester accesses.
- You have to configure the MicroStrategy Modeling Service. For complete information, see the [MicroStrategy documentation](#).
- Necessary permissions to all database objects that the lineage harvester accesses.

Show me the data source-specific permissions

Data source	Required permissions
Amazon Redshift	You need read access on information_schema. Only views that you own are processed.
Azure Data Factory	Ensure that you meet the Azure Data Factory-specific permissions described in Set up Azure Data Factory .
Azure SQL server	You need read access on the SYS schema.
Azure Synapse Analytics	SELECT, at table level. Grant this to every table for which you want to create a technical lineage.
Google BigQuery	If you are using the lineage harvester, you need read access on information_schema: ◦ bigquery.datasets.get ◦ bigquery.tables.get ◦ bigquery.tables.list ◦ bigquery.jobs.create ◦ bigquery.routines.get ◦ bigquery.routines.list If you are using Edge, you also need: ◦ resourcemanager.projects.get ◦ bigquery.readsessions.create ◦ bigquery.readsessions.getData
Greenplum	A role with the LOGIN option.

Data source	Required permissions
HiveQL	SELECT WITH GRANT OPTION, at Table level.
IBM DB2	CONNECT ON DATABASE
Informatica Intelligent Cloud Services	◦ Your user role must have privileges to export assets. ◦ You must have read permission on all assets that you want to export.
Matillion	◦ You have at least a Matillion Enterprise license. ◦ You have generated the Matillion certificate. Ensure that the certificate is signed by a certificate authority. Self-signed certificate is not supported when you create technical lineage via Edge. ◦ You have added the Matillion certificate to a Java truststore. For more information about adding a certificate to a Java truststore, go to Add a Certificate to a Truststore Using Keytool.
Microsoft SQL Server	You need read access on the SYS schema and the View Definition Permission in your SQL Server.
MySQL	SELECT, at table level. Grant this to every table for which you want to create a technical lineage.
Netezza	You need read access on definition_schema.
Oracle	You need read access to the following dictionary views: ◦ all_tab_cols ◦ all_col_comments ◦ all_objects ◦ ALL_DB_LINKS ◦ all_mviews ◦ all_source ◦ all_synonyms ◦ all_views

Data source	Required permissions
PostgreSQL	◦ GRANT SELECT, at table level. Grant this to every table for which you want to create a technical lineage. ◦ The role of the user that you specify in the <code>username</code> property in lineage harvester configuration file must be the owner of the views in PostgreSQL.
Snowflake	The following permissions are required, regardless of the ingestion mode: <code>SQL</code> or <code>SQL-API</code>. You need a role that can access the Snowflake shared read-only database. To access the shared database, the account administrator must grant the IMPORTED PRIVILEGES privilege on the shared database to the user that runs the lineage harvester. If the default role in Snowflake does not have the IMPORTED PRIVILEGES privilege, you can use the <code>customConnectionProperties</code> property in the lineage harvester configuration file to assign the appropriate role to the user. For example: <pre>"customConnectionProperties": "role=METADATA"</pre>
Teradata	You need read access on the DBC.

Important

- Before you start the Power BI integration process, you have to perform a number of tasks in Power BI and Microsoft Azure. These tasks, which are performed outside of Collibra, are needed to enable the lineage harvester to reach your Power BI application and collect its metadata. For complete information, go to [Set up Power BI](#).
- Collibra Data Lineage supports:
 - Power BI on the Microsoft Power Platform.
 - Power BI on Fabric.

The configuration requirements and the integration are the same, regardless of your setup.

Important

Before you start the Tableau integration process, you have to perform a number of tasks in Tableau. For complete information, go to the following topics:

- [Set up Tableau](#)
- [Tableau roles and permissions](#)

Steps

1. [Set up Tableau](#).
2. [Set up Power BI](#).
3. [Set up MicroStrategy](#).
4. Which custom lineage definition option are you using?

What's next?

[View the technical lineage](#).

About Technical lineage via Edge

You can use Edge to collect metadata from your data sources and create new relations between data elements from your data source and existing assets into Data Catalog. Edge collects transformation logic like SQL scripts and ETL scripts from a specified location, for example a database table or a folder on a file system.

Just like the lineage harvester, Edge connects to different [Collibra Data Lineage service instances](#) based on your geographical location and cloud provider.

For a list of the supported data sources and the technical lineage capabilities and connection types for each data source, go to [Supported data sources for technical lineage](#). For specific steps to create a technical lineage on Edge, go to [Creating technical lineage via Edge](#).

You can also use Edge to create a custom technical lineage. For complete information, go to [Create technical lineage via Edge](#) and select Custom technical lineage.

Important If you want to use technical lineage via Edge together with the lineage harvester, ensure that you use the lineage harvester version 2023.04 or newer. For more information, go to [Migrate the technical lineage of a data source](#).

Tip Check out our free [Why you should use Lineage on Edge today training course](#) in Collibra University.

Enabling and configuring technical lineage via Edge

To create a technical lineage via Edge, you must [enable](#) the feature in the Collibra settings or in Collibra Console.

You can define how Edge accesses the data sources by creating different connections. The following connection types are supported on Edge:

- JDBC connection, for JDBC data sources and ETL tools.
- Shared Storage connection, for JDBC data sources and some ETL tools.
- APIs, for MicroStrategy, Power BI, Tableau, Informatica Intelligent Cloud Services, and Matillion.

Configurations for technical lineage via Edge include the following:

- General configuration settings in the Collibra settings or Collibra Console, which apply to all data sources for which you create the technical lineage. For example, you can enter your Collibra Data Intelligence Cloud username and user password in the general configuration settings.
- Specific configuration settings for each data source. You can add a technical lineage capability for each data source to provide specific configurations.

Specific permissions might be required to access different data sources. For complete information, go to [Data source-specific permissions](#).

BI tool ingestion via Edge

Note BI ingestion via Edge is currently available for MicroStrategy, Power BI, and Tableau. To integrate other [supported BI tools](#), you need to use the [lineage harvester](#). You can also use the lineage harvester to integrate MicroStrategy, Power BI, and Tableau.

During the technical lineage process, relations of the type "Data Element targets / sources Data Element" are automatically created:

- Between data objects in your data source and their corresponding assets in Data Catalog, including the asset that you create when [preparing the Data Catalog physical data layer](#).
- If you are integrating a BI tool, between ingested assets from BI sources and Data Catalog assets from registered data sources.

Note

- You can't work with Edge via the REST API.
- You can't migrate from Jobserver to Edge to preserve the metadata that you manually added to the assets that you ingested via Jobserver.

Delete the technical lineage of a data source on Edge

You can delete the technical lineage of a data source by updating the capability for the data source and synchronizing the technical lineage again.

Note

- If you want to use technical lineage via Edge together with the lineage harvester, ensure that you use the lineage harvester version 2023.04 or newer.
- If you want to delete the technical lineage of a data source by using the lineage harvester, ensure that you use the lineage harvester version 2023.04 or newer. For details, go to [Delete the technical lineage of a data source](#).

Steps

1. Open an Edge site.
 - a. On the main toolbar, click , and then click  **Settings**.
 - » The [Collibra settings page](#) opens.
 - b. In the tab pane, click **Edge**.
 - » The **Sites** tab opens and shows a table with an overview of the Edge sites.
 - c. In the Edge site overview, click the name of the Edge site where you created the technical lineage capability for the data source.
 - » The Edge site page appears.
2. In the **Capabilities** section, locate and click the technical lineage capability that you added for the data source when you [created the technical lineage](#).
3. Clear the **Active** check box.
4. Click **Save**.
 - » The capability is updated.
5. [Synchronize the technical lineage capability for the data source](#).
 - » The data source is marked as ignored internally and will be excluded when the technical lineage is synchronized again.
6. Synchronize your technical lineage by taking any of the following actions:
 - On Edge, synchronize the technical lineage capability for any of your data sources that are active.
 - With the lineage harvester, run any of the following commands:
 - The `sync` command:
 - For Windows: `.\bin\lineage-harvester.bat sync`
 - For other operating systems: `./bin/lineage-harvester sync`
 - The `full-sync` command:
 - For Windows: `.\bin\lineage-harvester.bat full-sync`
 - For other operating systems: `./bin/lineage-harvester full-sync`

For more information, go to [Lineage harvesting app command options and arguments](#).

 - » When synchronization is complete, the technical lineage of the data source is deleted.

What's next?

If you want to delete the technical lineage capability for the data source, ensure that the technical lineage of the data source is removed successfully after synchronization. For more information, go to [Delete an Edge capability from an Edge site](#).

You can [view a summary of the results](#) from the Activities list to see whether the technical lineage is synchronized successfully.

If the synchronization fails or completes with errors, you can use the [technical lineage via Edge common error messages](#) in [Collibra Support Portal](#) to fix the errors.

Migrate the technical lineage of a data source

You can use the lineage harvester and technical lineage via Edge together. You can migrate a data source from lineage harvester to technical lineage via Edge, and also from technical lineage via Edge to the lineage harvester.

Prerequisites and permissions

- A [global role](#) that has the following [global permission](#):
 - The Catalog, for example Catalog Author
 - View Edge connections and capabilities
- A [resource role](#) with Configure external system [resource permission](#), for example Owner.
- The permissions to retrieve the metadata of the following database components through the JDBC Driver Database Metadata methods:
 - Schemas
 - Tables
 - Columns
- [The lineage harvester version 2023.04 newer](#).

Migrate to technical lineage via Edge

1. Open [the lineage harvester configuration file](#) in the config folder of your lineage harvester.
2. For the data source that you want to move to Edge, remove the section of the data source from the lineage harvester configuration file and save the configuration file.
3. If needed, start the lineage harvester in the console and run the following command to ignore the data source.
 - For Windows: `.\bin\lineage-harvester.bat ignore-source <source_ID>`, where `<source_id>` is the ID of the data source that you want to ignore.
 - For other operating systems: `./bin/lineage-harvester ignore-source <source_ID>`, where `<source_id>` is the ID of the data source that you want to ignore.

» The data source is excluded from the list of data sources that are used to create the technical lineage.

Important This step is required only in the following cases:

- If you use a different source ID for the data source on Edge.
 - If you are migrating SAP HANA data sources from the lineage harvester to Edge, regardless of the source IDs you use.
- When you created technical lineage for SAP HANA by using the lineage harvester, different sources IDs were required if you used the `hana` and `hana-cviews`, or `hana` and `hana-cviews-v2` dialects. However, in [the Technical Lineage for SAP HANA capability](#), you can use one source ID for both SQL based and calculated views input. Technical lineage via Edge adds suffixes to the source ID automatically and internally. When you synchronize the Technical Lineage for SAP HANA capability, an error occurs if the source IDs from the lineage harvester exist for the same data source.

4. On Edge, [add the technical lineage capability](#) for the data source with the same configurations, for example, the same source ID.
5. [Synchronize the technical lineage](#).
 - » When the synchronization completes, the technical lineage is created for the data source.

Migrate to the lineage harvester

1. Open an Edge site.
 - a. On the main toolbar, click , and then click  **Settings**.
 - » The [Collibra settings page](#) opens.
 - b. In the tab pane, click **Edge**.
 - » The **Sites** tab opens and shows a table with an overview of the Edge sites.
 - c. In the Edge site overview, click the name of the Edge site where you created the technical lineage capability for the data source.
 - » The Edge site page appears.
2. In the **Capabilities** section, locate and click the technical lineage capability for the data source.
 - » The technical lineage capability page opens.
3. Clear the **Active** check box.
4. Click **Save**.
 - » The capability is updated.
5. [Synchronize the technical lineage](#). If you added a synchronization schedule for the technical lineage capability, ensure that you delete the schedule.
 - » When the synchronization completes, the technical lineage of the data source is deleted.
6. Open [the lineage harvester configuration file](#) in the config folder of your lineage harvester.
7. Specify the properties in the lineage harvester configuration file for the data source with the same configurations of the capability, for example, the same source ID, and save the configuration file.
8. [Run the lineage harvester](#).
 - » When the lineage harvester finishes processing, the technical lineage is created for the data source.

For the overall steps to create technical lineage, go to [Create a technical lineage via the lineage harvester](#) or [Create a technical lineage via Edge](#).

What's next?

[View the technical lineage graph](#).

You can check the progress of the technical lineage creation in [Activities](#) in your Collibra Data Intelligence Cloud environment. The **Results** field indicates how many relations were imported into Data Catalog. Go to the [status page](#) to see the log files of the SQL analysis.

If the lineage harvester log shows an error message or the [harvesting process](#) fails, you can use the technical lineage common errors and issues in [Collibra Support Portal](#) to fix the error.

For technical lineage via Edge, if the synchronization fails or completes with an error message, you can use [the technical lineage via Edge common error messages](#) in [Support Portal](#) to fix the error.

Create a technical lineage via the lineage harvester

This topic describes the general steps on how to use the lineage harvester to create a technical lineage.

Select a data source,
to show the relevant
integration steps.

Currently, information
is shown for:

[Choose another data source](#)

Important Amazon Redshift Azure SQL server Azure Synapse
Analytics Greenplum Hive IBM Db2 PostgreSQL Microsoft SQL
Server MySQL Netezza SAP HANA Teradata requirements:

- Ensure that you meet the Azure Data Factory-specific permissions described in [Set up Azure Data Factory](#).
- You need read access on information_schema. Only views that you own are processed.
- You need read access on the SYS schema.
- If you are using the lineage harvester, you need read access on information_schema:
 - bigquery.datasets.get
 - bigquery.tables.get

- `bigquery.tables.list`
- `bigquery.jobs.create`
- `bigquery.routines.get`
- `bigquery.routines.list`

If you are using Edge, you also need:

- `resourcemanager.projects.get`
- `bigquery.readsessions.create`
- `bigquery.readsessions.getData`
- `SELECT`, at table level. Grant this to every table for which you want to create a technical lineage.
- Ensure that your service account token has the Read-Only permission.
- Ensure that you have the permission to copy the **target/** directory, which is generated by running the `dbt compile` command, to a local folder.
- You need Monitoring role permissions.
- A role with the `LOGIN` option.
- `SELECT WITH GRANT OPTION`, at Table level.
- `CONNECT ON DATABASE`
- You need read access on the `SYS` schema and the View Definition Permission in your SQL Server.
- You need read access on `definition_schema`.
 - `GRANT SELECT`, at table level. Grant this to every table for which you want to create a technical lineage.
 - The role of the user that you specify in the `username` property in lineage harvester configuration file must be the owner of the views in PostgreSQL.
- You need read access on the `DBC`.
- You need read access to the following dictionary views:
 - `all_tab_cols`
 - `all_col_comments`
 - `all_objects`
 - `ALL_DB_LINKS`
 - `all_mviews`
 - `all_source`
 - `all_synonyms`
 - `all_views`
 - Your user role must have privileges to export assets.
 - You must have read permission on all assets that you want to export.
 - You have at least a Matillion Enterprise license.
 - You have generated the Matillion certificate. For more information, go to [Recreating self-signed SSL certificates on a Matillion ETL instance](#).

- You have added the Matillion certificate to a Java truststore. For more information about adding a certificate to a Java truststore, go to [Add a Certificate to a Truststore Using Keytool](#).
- If you encounter the `javax.net.ssl.SSLHandshakeException: General SSLEngine problem` error message, go to [Data Source Name Failed exception with Tableau & technical lineage](#) in Collibra Support Portal for a solution.

The following permissions are required, regardless of the ingestion mode: SQL or SQL-API.

- Ensure that the Snowflake user has the appropriate allowed host list. For details, go to [Allowing Hostnames](#) in Snowflake documentation.
- You need a role that can access the Snowflake shared read-only database. To access the shared database, the account administrator must grant the **IMPORTED PRIVILEGES** privilege on the shared database to the user that runs the lineage harvester.
- If the default role in Snowflake does not have the **IMPORTED PRIVILEGES** privilege, you can use the `customConnectionProperties` property in the lineage harvester configuration file to assign the appropriate role to the user. For example:
`"customConnectionProperties": "role=METADATA"`
- The source code files must be in the same directory as your JSON files. For complete information, go to [Working with custom technical lineage](#).
- To stitch the data objects of your data sources with Data Catalog assets, you need to [register](#) your data sources in Data Catalog. When you then [prepare the Data Catalog physical data layer](#), ensure that you use a structure that matches the structure of ingested assets in Data Catalog.
- Determine whether you want to use the [single-file or batch definition option](#).
- If you choose the single-file definition option, determine whether you want to create a [simple or advanced custom technical lineage](#).
- Before you start the Power BI integration process, you have to perform a number of tasks in Power BI and Microsoft Azure. These tasks, which are performed outside of Collibra, are needed to enable the lineage harvester to reach your Power BI application and collect its metadata. For complete information, go to [Set up Power BI](#).
- Collibra Data Lineage supports:
 - Power BI on the Microsoft Power Platform.
 - Power BI on Fabric.

The configuration requirements and the integration are the same, regardless of your setup.

Before you start the Tableau integration process, you have to perform a number of tasks in Tableau. For complete information, go to the following topics:

- [Set up Tableau](#)
- [Tableau roles and permissions](#)
- You need the following roles, with user access to the server from which you want to ingest:
 - A system-level role that is at least a System user role.
 - An item-level role that is at least a Content Manager role.

We recommend that you use SQL Server 2019 Reporting Services or newer. We can't guarantee that older versions will work.

Before you start the Looker integration process, you need to [set up Looker](#). The following permissions apply only to MicroStrategy on-premises customers.

- You need Admin API permissions.
The first call we make to MicroStrategy is to authenticate. We connect to `<MSTR URL>:<Port>/MicroStrategyLibrary/api-docs/` and use POST `api/auth/login`. You have to ensure that this API call can be made successfully.
- You need permissions to access the library server.
- The lineage harvester uses port 443. If the port is not open, you also need permissions to access the repository.
- If you have a MicroStrategy on-premises environment, you need the permissions for all of the database objects that the lineage harvester accesses.
- You have to configure the MicroStrategy Modeling Service. For complete information, see the [MicroStrategy documentation](#).
- Necessary permissions to all database objects that the lineage harvester accesses.

Show me the data source-specific permissions

Data source	Required permissions
Amazon Redshift	You need read access on information_schema. Only views that you own are processed.
Azure Data Factory	Ensure that you meet the Azure Data Factory-specific permissions described in Set up Azure Data Factory .
Azure SQL server	You need read access on the SYS schema.

Data source	Required permissions
Azure Synapse Analytics	SELECT, at table level. Grant this to every table for which you want to create a technical lineage.
Google BigQuery	If you are using the lineage harvester, you need read access on information_schema: - bigquery.datasets.get - bigquery.tables.get - bigquery.tables.list - bigquery.jobs.create - bigquery.routines.get - bigquery.routines.list If you are using Edge, you also need: - resourcemanager.projects.get - bigquery.readsessions.create - bigquery.readsessions.getData
Greenplum	A role with the LOGIN option.
HiveQL	SELECT WITH GRANT OPTION, at Table level.
IBM DB2	CONNECT ON DATABASE
Informatica Intelligent Cloud Services	- Your user role must have privileges to export assets. - You must have read permission on all assets that you want to export.
Matillion	- You have at least a Matillion Enterprise license. - You have generated the Matillion certificate. Ensure that the certificate is signed by a certificate authority. Self-signed certificate is not supported when you create technical lineage via Edge. - You have added the Matillion certificate to a Java truststore. For more information about adding a certificate to a Java truststore, go to Add a Certificate to a Truststore Using Keytool.
Microsoft SQL Server	You need read access on the SYS schema and the View Definition Permission in your SQL Server.

Data source	Required permissions
MySQL	SELECT, at table level. Grant this to every table for which you want to create a technical lineage.
Netezza	You need read access on definition_schema.
Oracle	You need read access to the following dictionary views: ◦ all_tab_cols ◦ all_col_comments ◦ all_objects ◦ ALL_DB_LINKS ◦ all_mviews ◦ all_source ◦ all_synonyms ◦ all_views
PostgreSQL	◦ GRANT SELECT, at table level. Grant this to every table for which you want to create a technical lineage. ◦ The role of the user that you specify in the username property in lineage harvester configuration file must be the owner of the views in PostgreSQL.
Snowflake	The following permissions are required, regardless of the ingestion mode: SQL or SQL-API. You need a role that can access the Snowflake shared read-only database. To access the shared database, the account administrator must grant the IMPORTED PRIVILEGES privilege on the shared database to the user that runs the lineage harvester. If the default role in Snowflake does not have the IMPORTED PRIVILEGES privilege, you can use the customConnectionProperties property in the lineage harvester configuration file to assign the appropriate role to the user. For example: <pre>"customConnectionProperties": "role=METADATA"</pre>
Teradata	You need read access on the DBC.

Steps

Note Only Basic Authentication is supported. NTLM authentication, for example, is not.

1. Optionally, [connect](#) to a proxy server.
2. Ensure that you meet the [Azure Data Factory prerequisites](#).
3. Ensure that you have the correct Tableau versions and permissions, as described in the [Set up Tableau](#) topics.
4. Complete the tasks in Power BI and Microsoft Azure, as described in the [Set up Power BI](#) topics.
5. If you are a MicroStrategy on-premises customer, ensure that you have enabled Collibra to access your MicroStrategy data, as described in [Set up MicroStrategy](#).
6. Ensure that you have API3 credentials for authorization and access control. For complete information, go to [Set up Looker](#).
7. [Prepare](#) the Data Catalog physical data layer.
8. [Prepare](#) an external directory folder for the lineage harvester.
9. [Prepare](#) a domain for BI asset ingestion.
10. Optionally, assign the attribute type State to the global assignment of the Power BI Workspace asset type. For complete information, go to [Power BI workspaces](#).
11. [Download](#) and [install](#) the lineage harvester.
12. [Prepare](#) the lineage harvester configuration file.

Note The project name in the configuration file must be the same as the full name of the Database asset.

13. If necessary, [prepare](#) a <source ID> configuration file.
14. Manually refresh your Power BI datasets.

Important The first time you integrate Power BI, you need to make sure that the data in your Power BI datasets is up-to-date. Carry out this step only if this is the first time you're integrating Power BI in Data Catalog. After that, Microsoft automatically refreshes the datasets every 90 days. For complete information, see:

- The [Microsoft documentation](#).
- The [Microsoft Power BI Blog](#).

15. [Run](#) the lineage harvester.

You can define your Custom technical lineage via batch or single-file definition.

Batch definition (Beta)

1. Create a local folder.
2. Create the following:
 - A single metadata file.
Name the file **metadata.json**.
 - Optionally, one or more assets files. These are required if you want to achieve stitching.
Name the files **assets<something unique>.json**.
 - One or more lineage files.
Name the files **lineage<something unique>.json**.
 - Optionally, a folder for your source code files. This folder must be put in the same local folder.

For guidance on creating these files, go to [custom technical lineage JSON file examples](#).

Single-file definition

1. Create a local folder.
2. Create a JSON file in the local folder and name the JSON file **lineage.json**.
For guidance on creating the **lineage.json**, go to [custom technical lineage JSON file examples](#).

Note The JSON file must be named as **lineage.json**; otherwise, the process fails. You can have other types of files in this folder.

3. If you want to create an advanced custom technical lineage, store all of the source code files that you want to reference in the JSON file in the same local folder. For more information about the simple and advanced custom technical lineage, go to [Custom technical lineage](#).

What's next?

You can check the progress of the ingestion in [Activities](#). The results field indicates how many relations were imported into Data Catalog.

What are "heartbeat" messages in the lineage harvester log?

"Heartbeat" messages in the lineage harvester logs indicate that the connection to your Power BI service is active and that metadata is still loading. Heartbeat messages, like this example, can be shown throughout the harvesting process.

```
2023-07-14T15:43:36+02 INFO [src=PowerBITenant] Heartbeat
```

After the metadata is ingested in Data Catalog, you can go to the domain that you specified in your lineage harvester configuration file and view the newly created assets. These assets are automatically stitched to existing assets in Data Catalog.

You can also view the Tableau technical lineage.

Warning We strongly recommend that you not edit the full names of any BI assets. Doing so will likely lead to errors during the synchronization process.

Warning We highly recommend that you do not move the ingested assets to a different domain. If you do, the assets will be deleted and recreated in the initial BI Catalog domain when you synchronize. As a consequence, any manually added data of those assets is lost.

About Technical lineage via the lineage harvester

The lineage harvester is a connectivity tool that allows you to create a technical lineage. The lineage harvester collects metadata from your data sources. Collibra then analyzes and processes the metadata, and creates new Table and Column assets in Data Catalog, with names that match those of the data objects in your data sources. If you integrate a BI tool, new BI assets are also created.

You can download the lineage harvester from the [Collibra Community downloads page](#).

Important

- If you want to use technical lineage via Edge together with the lineage harvester, ensure that you use the lineage harvester version 2023.04 or newer. For more information, go to [Migrate the technical lineage of a data source](#).
- To [delete the technical lineage of a data source](#) in the lineage harvester version 2023.04 or newer, you must remove the section of the data source from [the lineage harvester configuration file](#) and also run [the ignore-source command](#) with the source ID specified.

For complete information on technical lineage, see the Colibra Data Intelligence Cloud User Guide.

Prepare the Data Catalog physical data layer for technical lineage

Important This topic does not apply if you register a data source via Edge because in that case, Colibra automatically creates the system > database > schema > table > column hierarchy.

To stitch data objects in your data sources to their corresponding assets in Colibra Data Intelligence Cloud, the full names of the data objects and assets must match exactly. The full names are constructed according to the full path of the data objects in your data source:

(system name) > database name > schema name > table name > column name

However, when you register a data source via Jobserver or by running the lineage harvester, only assets of the following asset types are created in Data Catalog:

- Schema
- Table
- Column

Therefore, you have to create a Database asset and create a relation between it and the relevant Schema asset, to construct the full path hierarchy required for full name matching. If you set the `useColibraSystemName` property to `true` in your [lineage harvester configuration file](#), you also need to create a System asset and create a relation between it

and the Database asset. We refer to this as preparing the Data Catalog physical data layer.

For more information, see [Automatic stitching for technical lineage](#).

Prerequisites

- You have a [global role](#) with the Catalog [global permission](#), for example, Catalog Author.
- You have a resource role with the following [resource permissions](#) on the **Schema** community if you use a Jobserver and on the **Database** community if you use Edge.
 - Asset > add
 - Attribute > add
 - Domain > add
 - Attachment > add

Additional prerequisites for JDBC data source types

If you are working with a JDBC data source type, you also need to meet the following prerequisites:

- You have the permissions to retrieve the metadata of the following database components through the JDBC Driver Database Metadata methods:
 - Schemas
 - Tables
 - Columns
- You have [set up the JDBC driver](#) of your source data, for example MySQL.
- You have [registered](#) a data source.

Tip The full name of your Schema asset must match the exact name of the schema (including for case-sensitivity) in the data source that you register in the configuration file.

If you use Jobservers in Collibra Console and there is no available Jobserver, the **Register data source** actions will be grayed out in the global create menu in Collibra.

Steps

1. Create a System asset:

Important This is only required if you set the `useCollibraSystemName` property to `true` in your [lineage harvester configuration file](#).

Tip The full name of the System asset must match (including for case-sensitivity) the exact name of the system of the data source that you register in the configuration file.

Show me how

- a. Open the product for which you want to create an asset, for example, **Business Glossary**.
- b. On the main toolbar, click **+**.
 - » The **Create** dialog box appears.
- c. On the **Assets** tab, click Database.
 - » The **Create Asset** dialog box appears.
- d. Enter the required information.

Field	Description
Asset Type	The asset type of the asset that you are creating.
Domain	The domain to which the asset will belong. Tip Ensure that the domain type of the selected domain is assigned to the selected asset type.

Field	Description
Asset Name	A name to identify the asset. Do not include the > symbol in the name. If an asset name contains the > symbol, Colibra Data Lineage fails to stitch the ingested data objects to the asset. Tip You can simultaneously create multiple assets. To do so, after typing the name, press <code>Enter</code>, and then type the next name. Depending on the settings, asset names may need to be unique in their domain. If you enter a name that already exists, it appears in the strike-through style.

e. Click **Create**.

» A message stating that one or more assets are created appears in the upper-right corner of the page.

2. Create a Database asset:

Tip The full name of your Database asset must match (including for case-sensitivity) the exact name of the database or project, in case of Google BigQuery, that you register in the configuration file. The names are case-sensitive.

Show me how

- Open the product for which you want to create an asset, for example, **Business Glossary**.
- On the main toolbar, click **+**.

» The **Create** dialog box appears.
- On the **Assets** tab, click Database.

» The **Create Asset** dialog box appears.
- Enter the required information.

Field	Description
Asset Type	The asset type of the asset that you are creating.

Field	Description
Domain	The domain to which the asset will belong. Tip Ensure that the domain type of the selected domain is assigned to the selected asset type.
Asset Name	A name to identify the asset. Do not include the > symbol in the name. If an asset name contains the > symbol, Collibra Data Lineage fails to stitch the ingested data objects to the asset. Tip You can simultaneously create multiple assets. To do so, after typing the name, press Enter, and then type the next name. Depending on the settings, asset names may need to be unique in their domain. If you enter a name that already exists, it appears in the strike-through style.

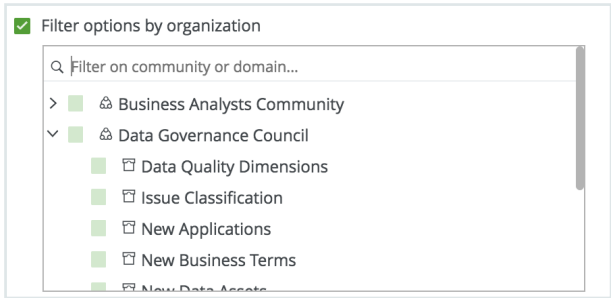
- e. Click **Create**.
 - » A message stating that one or more assets are created appears in the upper-right corner of the page.
3. Create a relation between the System asset and the Database asset using the "Technology Asset groups / is grouped by Technology Asset" relation type.

Important This step is only relevant if you created a System asset, in step 1.

Show me how

- a. Open the asset page of the System asset.
- b. In the tab pane, click **Add characteristic**.
 - » The **Add a Characteristic** dialog box appears.
- c. Click **Relations**.
- d. Search for and click **groups Technology Asset**.
 - » The **Add groups Technology Asset** dialog box appears.
- e. Enter the required information.

Option	Description
Assets	The name of the Database asset.

Option	Description
Filter suggested assets by organization	Option to filter the suggestions based on selected communities and domains. If this option is selected, the organization tree appears. You can then filter and select domains and communities. 
Start date	Optionally enter the date on which the relation between the assets becomes applicable.
End date	Optionally enter the date on which the relation between the assets is no longer applicable. Leave this field empty to create a permanent relation.

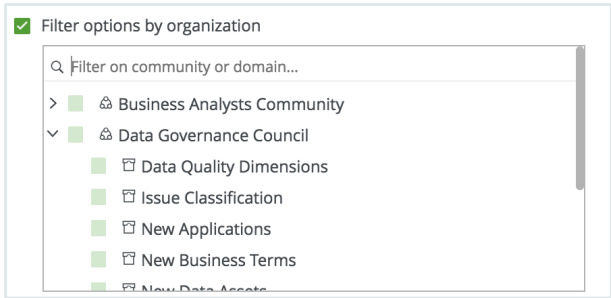
f. Click **Save**.

4. Create a relation between the Database asset and the relevant Schema asset using the "Technology Asset has / belongs to Schema" relation type.

Show me how

- a. In the tab pane, click **Add characteristic**.
 - » The **Add a Characteristic** dialog box appears.
- b. Click **Relations**.
- c. Search for and click **has Schema**.
 - » The **Add has Schema** dialog box appears.
- d. Enter the required information.

Option	Description
Assets	The name of the Schema asset.

Option	Description
Filter suggested assets by organization	Option to filter the suggestions based on selected communities and domains. If this option is selected, the organization tree appears. You can then filter and select domains and communities. 
Start date	Optionally enter the date on which the relation between the assets becomes applicable.
End date	Optionally enter the date on which the relation between the assets is no longer applicable. Leave this field empty to create a permanent relation.

e. Click **Save**.

What's next?

If you haven't [created](#) a configuration file yet, you are now required to create it.

If you created the configuration file and prepared the physical data layer, you can [run](#) the lineage harvester to start the technical lineage process.

When the technical lineage process is finished and you have the required permissions, you can go to the [asset page](#) of a Table or Column asset from the data source that you added in the configuration file and visualize the technical lineage. At the same time, new relations of the type "Data Element targets / sources Data Element" between assets in Data Catalog are created.

Set up the lineage harvester

The lineage harvester is a software application that is needed to create a technical lineage and import metadata into Data Catalog.

The lineage harvester

You use the [lineage harvester](#) to collect source code from your [data sources](#) and create new relations between data elements from your data source and existing assets into Data Catalog.

The lineage harvester runs close to the data source and can harvest [transformation logic](#) like SQL scripts and ETL scripts from a specific location, for example a database table or a folder on a file system.

The lineage harvester connects to different [Collibra Data Lineage service instances](#) based on your geographical location and cloud provider. Ensure you have the correct [system requirements](#) before you run the lineage harvester. If your location or cloud provider changes, the lineage harvester re-harvests all your data sources.

Note Technical lineage is created by a cloud-based service. You only connect to the cloud via an API call that is triggered by the lineage harvester.

The lineage harvester configuration file

The lineage harvester uses a configuration file to connect to JDBC data sources, BI tools and ETL tools. The configuration file contains references to the data sources for which you want to create a technical lineage. You have to [prepare the configuration file](#) if you want to create a technical lineage and add new relations of the type "Data Element targets / sources Data Element" between existing assets in Data Catalog, and "Column is target of / is source of Data Attribute" between assets from ingested BI sources and assets in Data Catalog.

Warning You can only use UTF-8 or ISO-8859-1 characters in all lineage harvester files.

The lineage harvester components

The lineage harvester consists of components that harvest the metadata from the data sources specified in your configuration file and send their metadata to the [Collibra Data Lineage service](#).

Using the lineage harvester

If you want to separately process data sources on different servers, you can use more than one lineage harvester connected to a single Collibra Data Intelligence Cloud instance. In this case, you can create a [configuration file](#) for the lineage harvester on each server and configure different data sources in each configuration file.

Note You can use different [command options and arguments](#) to perform various actions with the lineage harvester.

Permissions

You need a global role with the System Administration [global permission](#), for example Sysadmin. This role must have access to all assets in the data sources in the [configuration file](#) and be able to create new relations between these assets.

Typical workflow

You use the lineage harvester to run the `full-sync` command. That triggers the following actions:

1. The lineage harvester:
 - Harvests the metadata from the data sources that are specified in the [configuration file](#).
 - Uploads metadata collected from all configured data sources to Collibra Data Lineage's Metadata Ingest Pipeline.
 - Triggers the Sync Pipeline after all metadata has been completely processed.
2. The Metadata Ingest Pipeline:
 - Parses the metadata for all lineage assets and relations.
 - Stores the assets and relations in the cloud storage.

3. The Sync Pipeline:

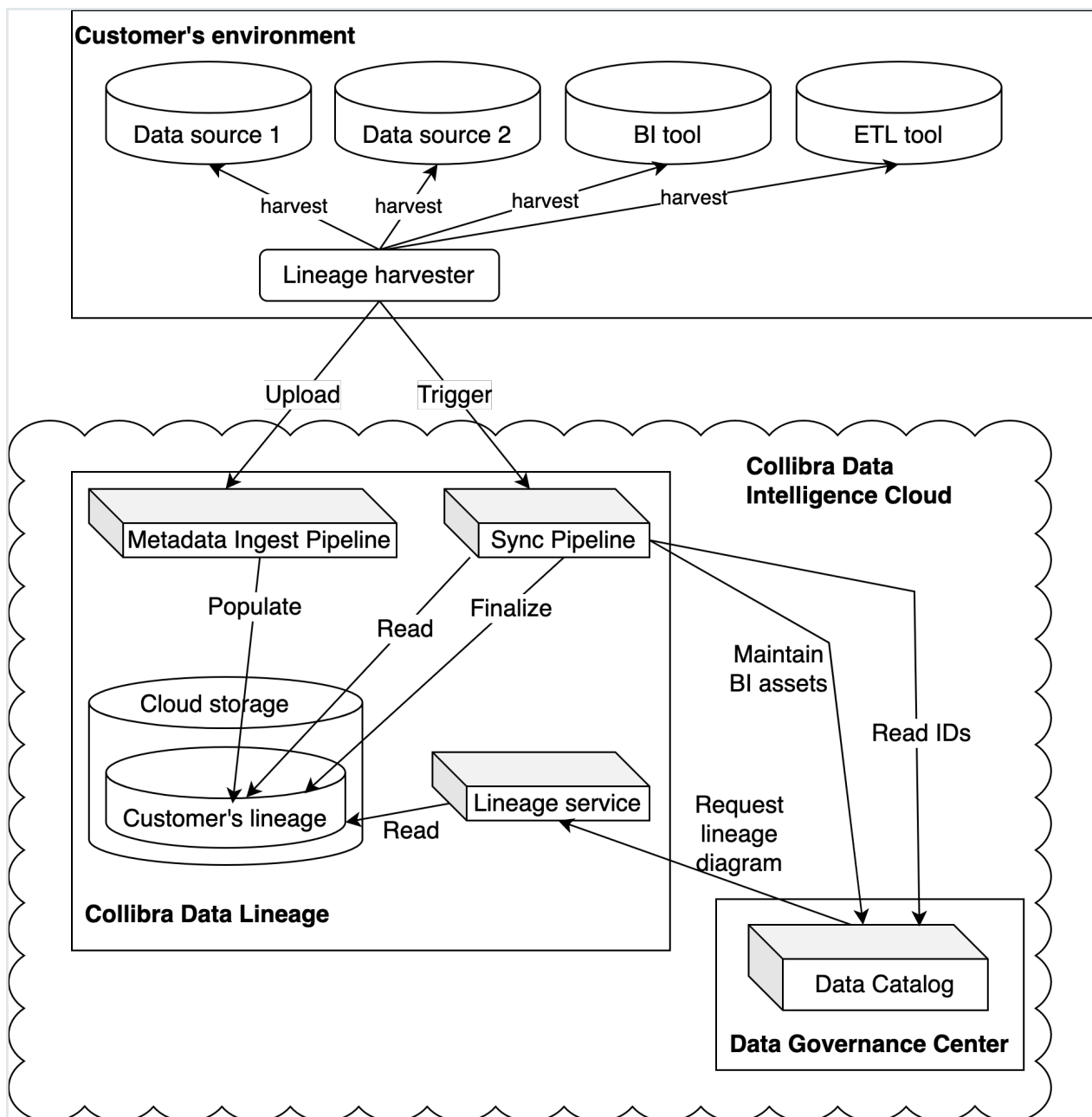
- Merges all partial lineages into a single data store.
- Publishes discovered BI assets to Data Catalog.
- Matches asset IDs from Data Catalog to the assets discovered from the metadata (stitching).
- Stores the complete lineage in the cloud storage.
- Publishes newly discovered relations to Data Catalog.

4. The Lineage Service:

- Upon request, creates HTML diagrams of the lineage.

5. Data Catalog:

- Connects to the lineage service to get the technical lineage to be shown in the technical lineage viewer.



Note The lineage harvester can only create Power BI, Tableau, Looker and other BI tool specific assets, if you included a reference to the specific BI tool in the configuration file. No other assets are created during the process. Only new relations between existing and newly created BI assets (for example between two Tableau Data Attribute assets), and between BI column and Column assets (for example between Power BI Column and Column assets) are created.

The lineage harvester change log

Collibra Data Lineage is updated and improved on a regular basis. On this page, you can see the most important changes between different versions of the lineage harvester. For a complete list, see the release notes.

Note We highly recommend that you download and use the newest lineage harvester from the [Collibra downloads page](#), even if you are on an older version of Collibra Data Intelligence Cloud.

The following list contains the most important changes to the lineage harvester and the [lineage harvester configuration file](#).

Changed in version	New lineage harvester improvements
2024.02	- Improved folder filtering for Looker integrations. The following improvements are applicable when filtering by folder ID. - Filtering is now carried out via the API, instead of on the Collibra Data Lineage service instances. This means that the lineage harvester accesses only the specified folders, and sends the metadata to the Collibra Data Lineage service instance for processing and ingestion in Data Catalog. Previously, metadata from all folders was sent to the Collibra Data Lineage service instance, where filtering was then carried out. This change greatly improves Looker integration flexibility, speed, and performance. - The log files now include more detailed information about the filter results. - When integrating JDBC data sources via the lineage harvester, with the "Folder connection" method, you can now include the new "databaseSystemMapping" property in your configuration file, to map specific databases to specific systems, to obtain stitching. Previously, stitching was missing, as all databases were associated with the default system name provided in the "collibraSystemName" property. - CollibraData Lineagesupport for Databricks Unity Catalog is now in beta testing. You can create technical lineage for Databricks Unity Catalog by using technical lineage via Edge. - When integrating Power BI, you can now enable DAX parsing (in beta), to allow for column-level lineage of calculated columns and measures.

Changed in version	New lineage harvester improvements
2024.01	• The lineage harvester now uses Transport Layer Security v1.3 encryption, for heightened security. • When integrating MicroStrategy: ◦ Intelligent Cubes are now ingested even if some properties are missing in the API call. ◦ MicroStrategy Data Model assets have a relation to MicroStrategy Folder assets. • When integrating Power BI: ◦ The Owner in Source attribute on Power BI Dataflow asset pages now correctly reflects the real owner of the dataflow in Power BI. ◦ If a data set in Power BI has a description, the description is now ingested and shown on the asset page of the corresponding Power BI Data Model asset. ◦ If reports are ingested from multiple tenants, the report URLs on the asset pages in Collibra now include the tenant ID. Previously, without this differentiation in the URL, the links accessed the page of the same, default tenant, even if a given report was not located in that tenant. • When integrating Looker and filtering on a folder, metadata processing no longer fails if there are Looks that don't pass the filter, but are related to a Tile that is ingested. In this case, processing continues and the following warning message is included in the source code: "Look <Look ID> was not found in folder <folder ID>." • When integrating Looker, the lineage harvester now includes a retry mechanism, to reduce the chances of experiencing a GOAWAY error when trying to connect to Looker. • Implemented streaming by the lineage harvester when synchronizing Looker, to help reduce the chances of Out Of Memory (OOM) errors. • When integrating Tableau, the relation "Report uses / used in Data Attribute" is now created between Tableau Dashboard assets and Tableau Data Attribute assets. • When ingesting Snowflake data sources, the Collibra Data Lineage service instances now support SELECT... EXCLUDE statements. • Technical lineage for IBM DataStage has improved analysis of INSERT statements that use ORCHESTRATE. • Technical lineage for Informatica Intelligent Cloud Services (IICS) now includes lineage from presql and postsql commands. • For technical lineage for IICS, a bug is fixed regarding case sensitivity. The technical lineage graph now includes more end-to-end lineage for IICS, and stitching to Data Catalog assets is more accurate. • Technical lineage for Informatica PowerCenter now supports sessions in

Changed in version	New lineage harvester improvements
	Folders and Workflows. Previously, when a TaskInstance of a Workflow was set to REUSABLE, Collibra Data Lineage looked for sessions only in the Folder. • Technical lineage for Azure Data Factory can now resolve global parameters into their values. With this enhancement, you won't experience missing lineage or DEFAULT databases or schemas caused by global parameters.

Changed in version	New lineage harvester improvements
2023.11	• CollibraData Lineage support for dbt Cloud and dbt Core is now generally available. You can create technical lineage for dbt Cloud and dbt Core by using the lineage harvester and technical lineage via Edge. • When you create technical lineage for Looker, you can now define the number of concurrent request and connection timeout in the configuration file. Previously, you might experience HTTP 401 Unauthorized errors due to too many concurrent requests. • When creating a custom technical lineage, you can now opt for the new batch definition option, which includes the following benefits: ◦ Support for indirect and table-level lineage as well as column-level lineage. ◦ Support for any asset type with stitching. ◦ You can now submit multiple JSON files. ◦ You no longer have to define every asset used in lineage relation in a separate section. The new batch definition option is in beta. After a successful beta period, we will announce the deprecation of the single-file definition option. • When integrating Power BI, tables and columns are correctly identified for the Power Query M function AzureAnalysis.Database. (ticket #121494) • You can now configure a proxy connection when integrating Power BI via the lineage harvester. Important TLS proxy connection is not supported. • When you create technical lineage for SQL Server Integration Services (SSIS), CollibraData Lineage creates a summary of all connection definition errors in a analyze error in the Sources tab page. • When integrating Power BI with a Snowflake data source, the “Kind” parameter is now optional in your Power Query Snowflake.Databases function. Previously, if the “Kind” parameter did not follow the database name, an analysis error was produced. (ticket #120494) • When integrating Power BI, you can now have more than one upstream dataset as the source of other datasets. (ticket #124002, 127341, 127825, 127825, 127877) • When integrating MicroStrategy, Facts, and Fact Expressions are ingested in Data Catalog as MicroStrategy Data Entity, and MicroStrategy Data Attribute assets, respectively. Relations between Facts and Fact Expressions are now shown on the respective asset pages. • When integrating MicroStrategy and specifying the Collibra system name, stitching is again achieved as long as the system name you specify in your <source ID> configuration file matches the name of the System asset in Data

Changed in version	New lineage harvester improvements
	Catalog. The names are not case-sensitive. When you create technical lineage for Informatica Intelligent Cloud Services, CollibraData Lineage can now parse parameter files that are organized in a flattened structure in the root directory in the ZIP file, rather than in their corresponding sub-directories.
2023.10	- You can now configure a proxy connection when integrating Power BI via the lineage harvester. Important TLS proxy connection is not supported. - When ingesting any of the supported JDBC data sources, the Collibra Data Lineage service instances now support “ALTER TABLE ... RENAME COLUMN” commands. - When integrating MicroStrategy: - You can now use domain mapping to specify into which domains in Collibra you want to ingest MicroStrategy project metadata. - Intelligent Cubes are now harvested and ingested in Collibra as MicroStrategy Data Model assets, with the Source Type attribute value "Intelligent Cube". - Collibra Data Lineage now creates technical lineage for paginated reports in Power BI. - Schema mapping is now available for schemas that come from Power Query connections. - Tableau sites without views can now be successfully processed. - Collibra Data Lineage can now handle asset names that include characters other than alphabetical letters and numeric digits. Previously, if an asset name contained non-alphanumeric characters, the technical lineage viewer failed to display the technical lineage and returned an internal server error. - The technical lineage for dbt Cloud (Beta) on Edge is using a new API to get information, as the previous API used is deprecated.

Changed in version	New lineage harvester improvements
2023.09	• Technical lineage for dbt Cloud and Core is now in beta. For dbt Cloud, you can create technical lineage on Edge or by using the lineage harvester. For dbt Core, you can create technical lineage by using the lineage harvester only. • Collibra Data Lineage now supports the Power Query M function Teradata.Database. • The MicroStrategy integration now supports MicroStrategy Freeform SQL. To benefit from this enhancement, you need MicroStrategy version 10 or newer, or MicroStrategy ONE. • When integrating Tableau, the import job no longer fails if an ingested calculation rule exceeds the size limit of 100,000 bytes, which is approximately 100,000 characters in length. • When integrating MicroStrategy, Metric Attributes are now identified in the technical lineage and in Data Catalog by their names, instead of an asterisk (*), and their source type is now “Metric Attribute”.

Changed in version	New lineage harvester improvements
2023.08	• The new MicroStrategy integration method, via the lineage harvester, is now generally available. The new integration method has the following benefits: ◦ Supports technical lineage with stitching. ◦ Supports the latest MicroStrategy APIs. ◦ Supports project filtering. ◦ Allows you to view the source code for all tables and transformations. • When you integrate MicroStrategy via the new integration method, you can now view the source code for all tables and transformations, in the technical lineage Sources tab page. The source code shows information about the processes visible in the technical lineage and shows warnings and errors where a process has failed. This enhancement does not affect the success rate of metadata analysis. • The Power BI and MicroStrategy global assignments are updated to show more details on respective asset pages. • The Source Type attribute is now included on MicroStrategy Data Entity and MicroStrategy Data Attribute asset pages, to identify the MicroStrategy data object type, for example Attribute, Fact, Table, or Column. • Collibra Data Lineage now supports the following Power Query M functions: ◦ AnalysisServices.Databases ◦ AnalysisServices.Database Note ■ This function is fully supported if no MDX queries are used. ■ If MDX queries are used and they resemble SQL, they will be parsed by the SQL parser. ■ We don't currently support this function if used with MDX queries that resemble DAX, as the Collibra Data Lineage service instances can't parse such queries. ◦ GoogleAnalytics.Accounts • The relation “Data Asset contained in BI Folder” is now available between all Tableau Data Model and Tableau Project assets. • When you integrate Tableau: ◦ Tableau Dashboard, Worksheet and Workbook asset pages now show the number of views in the Visits count attribute type. ◦ The Tableau API analysis documentation is updated with the visits count. • The Tableau hostname mapping feature is now generally available. When integrating Tableau, you can use the optional “hostnameMapping” section in your <source ID> configuration file, to map Tableau technical database, server and schema names to the respective real names, to preserve stitching. • When integrating Power BI, datamart metadata is now ingested in Collibra as

Changed in version	New lineage harvester improvements
	assets of the new asset type Power BI Data Mart. • When ingesting PostgreSQL data sources, the Collibra Data Lineage service instances now support "x::typename" cast constructs, where "typename" contains a dot (.), for example "SELECT 'null'::qwerty.qwerty". • When ingesting Snowflake data sources, the Collibra Data Lineage service instances now support LEVEL and CONNECT BY keywords. • Previously, when creating technical lineage for SQL Server Integration Services, Collibra Data Lineage filtered out some queries due to legacy limitations. Collibra Data Lineage no longer filters out queries. You may find increased successful lineage as well as increased parsing or analysis errors, as Collibra Data Lineage tries to parse more queries. This is a backend change, and the new behavior will be seen during the next synchronization of the technical lineage for SQL Server Integration Services. • Collibra Data Lineage now processes and generates technical lineage for Informatica PowerCenter four times faster with the following changes: ◦ Data Lineage now pre-processes data into pydantic models instead of using the slower xpath solution that existed previously. ◦ Shortcuts are handled faster, by keeping necessary objects in memory on the Collibra Data Lineage service instances. ◦ The Analysis Error messages are enhanced by adding information that is related to rejected files and unresolved parameters. • The new analyzeTemporaryTables property is added to the Snowflake SQL-API source ID file. With this property, you can decide whether CollibraData Lineage should analyze the CREATE TEMPORARY TABLE statement

Changed in version	New lineage harvester improvements
2023.07	• Collibra Data Lineage now supports the following Power Query M functions: ◦ Cube.Transform ◦ Cube.AddAndExpandDimensionColumn ◦ Table.FromList ◦ AnyalysisServices.Databases These functions enable technical lineage between SAP HANA (using SapHana.Database) and Analysis Services (using AnalysisServices.Databases), and improve the success rate of metadata analysis. • When you create technical lineage for Snowflake with the SQL-API ingestion method, you can use the displaySampleQueries property in the new Snowflake source ID configuration file to control whether a question mark (?) is displayed in place of certain static values, such as numbers or dates. • When ingesting Spark SQL data sources, the Collibra Data Lineage service instances now support PARTITIONED BY parameters in CREATE TABLE statements. • Collibra Data Lineage now supports the following Power Query M functions: • When you create technical lineage for Matillion, the lineage harvester now supports multiple data sources. Previously, the lineage harvester could only generate technical lineage from one source. • When you create technical lineage for Informatica Intelligent Cloud Services and set the useCollibraSystemName property as true, the Collibra system name is used as root of the tree in the technical lineage graph. Previously, IICS was used. See an example.

Changed in version	New lineage harvester improvements
	Browse Settings Q Search ✱ All data objects > DATABASE > FILE SYSTEM ▼ IICS ▼ iics_source ▼ /Explore/Prj_EDW_CRM > mt_audit_Check_pbc • When you create technical lineage for Azure Data Factory, global parameters are now taken into consideration. • SQL Extension now supports queries that have a WITH clause. • When ingesting Oracle data sources, the Collibra Data Lineage service instances now correctly handle database links even when the remote database has a dot (.) in the name. • When ingesting Snowflake data sources, the Collibra Data Lineage service instances now benefit from the following parsing enhancements: ◦ Undocumented usage of UPDATE FROM statement, when the FROM clause comes before the SET clause. ◦ IDENTIFIER keyword appears as a column name. • When integrating MicroStrategy, any facts that don't have expressions are now skipped. Previously, Collibra Data Lineage attempted to process such forms, which resulted in errors.

Changed in version	New lineage harvester improvements
2023.06	• When integrating SQL Server Reporting Services (SSRS) or Power BI Report Server (PBRS): ◦ You no longer get an error if you filter on a folder to which you don't have access. ◦ You no longer get an error if the "rd" namespace is not specified at the top level of a report (an RDL file). In that case, it is now taken from the child level. ◦ The Collibra Data Lineage service instances now support CommandText with SQL that starts with "=". CommandText is split by either "+" or "&" and merged into a single parseable SQL command. ◦ If you filter on a specific folder, paginated reports at the root level of the folder are now correctly ingested. • When integrating Tableau, backticks "" in a query no longer result in missing columns when processing a CREATE TECHLIN VIEW. • When integrating Power BI, you can now use HTTP1 streams if you are experiencing timeout issues with the default HTTP2 streams. To do so, include the new optional property "useHttp1" in your lineage harvester configuration file, and set the value to "true". • When you integrate MicroStrategy via the new integration method (beta), you can now view the source code for all tables and transformations, in the technical lineage Sources tab page. The source code shows information about the processes visible in the technical lineage and shows warnings and errors where a process has failed. This enhancement does not affect the success rate of metadata analysis. • When ingesting CSV files as part of a Tableau integration, the "database > schema > table" structure in the technical lineage now matches the structure of the ingested CSV file in Data Catalog. This ensures that stitching can be achieved for CSV files. • Collibra Data Lineage now supports the Power Query M function Table.CombineColumns. • When integrating MicroStrategy, any forms that don't have expressions are now skipped. Previously, Collibra Data Lineage attempted to process such forms, resulting in errors. • When integrating Power BI via the Power BI harvester (integration method v1), which has been deprecated since 2022, the Power BI source code now includes an end-of-life message. Please migrate to Power BI via the lineage harvester (integration method v2) by August 1, 2023. • We have upgraded: ◦ The Snowflake driver to address the CVE-2023-30535 vulnerability. ◦ The BigQuery driver to mitigate the CVE-2022-45688 vulnerability.

Changed in version	New lineage harvester improvements
	• When ingesting Redshift data sources, the Collibra Data Lineage service instances now support the COLLATE function. • When ingesting HiveQL metadata, the Collibra Data Lineage service instances now support the TBLPROPERTIES parameter with an empty list, for Hive CREATE TABLE statements. • When ingesting Spark SQL data sources, the Collibra Data Lineage service instances now support identifiers that start with a number. • When ingesting HiveQL, the Collibra Data Lineage service instances now support Hive extension for the multiple inserts clause. • When ingesting Snowflake data sources, the Collibra Data Lineage service instances now support: ◦ Aliases in combination with the FLATTEN function. ◦ The DATA_RETENTION_TIME_IN_DAYS parameter for CREATE TABLE statements.

Changed in version	New lineage harvester improvements
2023.05	• When you integrate Power BI, reports and dashboards that are part of an app in Power BI are ingested as Power BI Dashboard and Power BI Report assets, respectively. The URLs on these asset pages now correctly link to the corresponding dashboards and reports in the Power BI app. • When integrating Power BI, the full names of Power BI capacities now include their unique identifiers. This helps to distinguish two capacities with the same name. Upon the first synchronization after this fix, if you use only one Power BI tenant, the Shared Capacity asset is deleted and recreated with the new naming format. If you have multiple Power BI tenants, a Shared Capacity asset with the new naming format is created for each tenant. • If a data set or report in Power BI is certified, the corresponding Power BI Data Model and Power BI Report assets in Collibra are now automatically certified. • When integrating Power BI, if you use the Databricks.Query query without specifying the database name, the database name in the technical lineage is “Default”. • When you integrate Tableau, the lineage harvester now automatically connects to the REST API version that matches your Tableau Server or Tableau Online environment. • When integrating Tableau: ◦ Filtering on sub-projects no longer results in FOREIGN KEY constraint errors. ◦ Custom SQL is now successfully processed when Tableau object names contain quotes. • When ingesting Spark SQL data sources, the Collibra Data Lineage service instances now benefit from the following parsing enhancements: ◦ CREATE VIEW to support TBLPROPERTIES ◦ SELECT allowed as column name ◦ TABLE allowed as column name ◦ CREATE TABLE to support the USING clause ◦ CREATE TBAL to support the OPTIONS clause • When ingesting Oracle data sources, SQL queries to extract views no longer include views for which the owner has a user name that start with “APEX”. • Collibra Data Lineage support for creating technical lineage for Azure Data Factory by using the lineage harvester is now generally available. • When you create technical lineage for Azure Data Factory, you can filter the factories that the lineage harvester collects and processes by using the new <code>factories</code> property in the lineage harvester configuration file to filter. • The new MicroStrategy integration method, via the lineage harvester, is now in beta. The new integration method allows for technical lineage, supports the latest MicroStrategy APIs, and is no longer dependent on a direct connection

Changed in version	New lineage harvester improvements
	to the repository. - When you create technical lineage for Snowflake by using the SQL-API ingestion method, - QUERY_TAG values are now shown in the transformation window for lineage queries. - The lineage harvester optimized the results of the columns_joined query. Previously, the view definition would be saved for each column of a view. Now, a view definition is only saved once. This enhancement results in faster processing of lineage for your Snowflake database that has views with many columns. - When you create technical lineage for Informatica PowerCenter, an error message is logged if any of the following issues occur: - A parameter file cannot be parsed. - A workflow XML file cannot be parsed or is invalid. - The list-sources command is enhanced to: - Indicate how each data source was ingested, by using the lineage harvester or technical lineage via Edge. - List the <code>useSystemName</code> value to each data source. - List up to 500 data sources. With this enhancement, you can determine which page to be displayed and also the number of data sources to be listed on certain pages.
2023.04	- Power BI Data Flow asset pages now show the description of the data flow. - When you integrate Looker, corrupt Looker Dashboards are now skipped. - You can now include the new Calculation Rule attribute type on Power BI Column asset pages, to show DAX calculations. - When ingesting Microsoft SQL Server data sources, the Collibra Data Lineage service instances can now parse QUALIFY statements. - A partial technical lineage that is generated from queries that have analyze errors no longer produces a foreign key error. - With the lineage harvester, you can now use the new ignore-source command to ignore the specified data source from the list of data sources to be used to create the technical lineage.

Changed in version	New lineage harvester improvements
2023.03	- When you create technical lineage for IBM DataStage, CollibraData Lineage now parses the following: - The inner stage SQL statements, such as the INSERT and UPDATE statements, which are used to bind DataStage columns and target database objects in some stages. - Parameters that are passed by when a Job is called or used. The parameters are typically passed to the Job activity in a Sequence Job. Previously, CollibraData Lineage only parsed and created technical lineage for default parameters. - DataStage containers, including local and shared containers. - Runtime column propagation that is enabled on stages. - The account loop and stage variables of the Transformer stage. - When you integrate Power BI or Azure Data Factory (ADF integration is currently in beta), the lineage harvester now connects to the Microsoft cloud instance, instead of the login.microsoftonline.com host. - When you integrate Power BI, parameters are now ingested even if the “Enable load” option is not selected for the relevant parameters. Note To harvest columns and create a full technical lineage, the “Enable load” option must be selected. If it is not, the Power BI APIs will recognise parameterised tables, but not the columns in the tables. In which case, only table-level lineage is created; columns cannot be shown. - When you integrate Tableau, - And configure site filtering in your <source ID> configuration file, Tableau sites that are not mentioned in the filter are now correctly included in the ingestion. They are ingested in the default domain. - The lineage harvester now ingests parameters. - With a Google BigQuery data source, all BigQuery data objects now correctly appear in the technical lineage. - You can use the new list-sources command to list all data sources that were ingested to create the technical lineage via the lineage harvester and technical lineage via Edge. - When you integrate SQL Server Reporting Services (SSRS) or Power BI Report Server (PBRS), folder filtering no longer fails when you want to ingest everything. - When ingesting Snowflake data sources, the correct source table is now shown in the technical lineage when the UNION operator is used. - When ingesting MySQL data sources, the Collibra Data Lineage service instances now support the “as” keyword as optional in “create table” state-

Changed in version	New lineage harvester improvements
	ments. Previously, parsing failed if the “as” keyword was missing.
2023.02	• When you integrate Tableau: ◦ Performance is significantly improved for Tableau users with the Explorer role. ◦ The lineage harvester now filters out data objects for which you do not have permissions. ◦ You can now view the source code in the technical lineage Sources tab page. The source code shows information about the processes visible in the technical lineage and shows warnings and errors where a process has failed. This enhancement does not affect the success rate of metadata analysis. ◦ UUIDs no longer appear in the names of Tableau assets in a technical lineage, with the following exception: if Tableau data objects in a technical lineage hierarchy have the same full name, Collibra Data Lineage adds the UUIDs of the corresponding assets to the names in the technical lineage, to maintain uniqueness. For complete information, including how to resolve UUIDs in the names, see Technical lineage for Tableau. • The Power BI integration can now connect to Power BI for US government customers. • When you integrate Power BI or Azure Data Factory (currently in public beta), the lineage harvester now connects to the Microsoft cloud instance, instead of the login.microsoftonline.com host. • When you synchronize any supported BI tool, if a corresponding data object of an asset in Data Catalog can no longer be found in the data source, the asset is no longer deleted from Data Catalog. Instead, the status of the asset changes to “Missing from source”. • The lineage harvester logic is now based on the UUIDs of attribute types in the BI tool operating models, instead of the attribute type names. This means that when you integrate any of the supported BI tools, you can now change the names of the ingested attribute types. • Collibra Data Lineage now supports Power Query parameters (public beta). For complete information, including how to ensure that the Power BI APIs return all parameters that are loaded in a report, see Working with Power Query parameters. • When you run a full-sync of a Snowflake data source, the lineage harvester automatically refreshes the authentication token, to avoid a time-out error. • Fixed the ordering of columns for Power BI technical lineage custom queries.

Changed in version	New lineage harvester improvements
2023.01	- When you integrate Power BI, - Collibra Data Lineage now supports the Power Query M function Table.Combine. If Collibra Data Lineage can't determine the column names in multiple sources, a dummy column with the value "*" is now created in the sources and Power BI tables, which preserves the technical lineage at the table level. For complete details, see Supported Power Query M functions. If you use this function, Table.Combine function is used. You can now view a technical lineage at the table level, where previously analyze error "Cannot determine source table for column". - The technical lineage now correctly shows a yellow background when columns and tables are stitched. - If you use a <source ID> configuration file, you no longer have to include the filters section. - When you integrate Tableau: - If a Tableau worksheet is hidden in Tableau, the "Visible on server" attribute of the Tableau Worksheet asset in Collibra now has the value false. If it is not hidden, the attribute has the value true. - Metadata batches no longer fail if CREATE TECHLIN VIEW statements fail due to analysis errors. - Collibra Data Lineage service benefits from improved parsing of BigQuery quoted identifiers, for example `a.b`.`c`. - Tableau filtering now works as intended. Previously, filtering didn't work if, for example, you moved an older Tableau project under a newer project. - Fixed the ordering of columns for Tableau technical lineage custom queries. - Tableau Data Attributes are no longer shown twice, once with the UUID in the name and once without, in the technical lineage Browse tab pane. - The "Document size" attribute type and value are now shown for Tableau Workbook assets. - If you don't have permissions to access a parent project, but the lineage harvester identifies published data sources that belong to the project, the lineage harvester creates an 'Unknown project' that has the UUID of the inaccessible parent project. To avoid an error, the lineage harvester can now correctly link the published data sources to the unknown project. - Collibra Data Lineage service now supports the Power Query M function Value.NativeQuery. Note Query parameters are supported, but core parameters are not. - When you integrate Power BI or Azure Data Factory (currently in public beta),

Changed in version	New lineage harvester improvements
	the lineage harvester now connects to the Microsoft cloud instance, instead of the login.microsoftonline.com host. • When you ingest SQL Server Reporting Services (SSRS) and you set the “useCollibraSystemName” property to “true”, SSRS now has its own node in the navigation tree of the Technical lineage Browse tab pane. • When you ingest Oracle data sources using the DatabaseOracle source type, passwords are now stored per url, username and db instead of just url and username. With this enhancement, you can connect to Oracle Pluggable Databases, for which a single user can have the same username and different passwords for each of their pluggable databases. • For Informatica PowerCenter technical lineage, when a PowerCenter mapplet had an associated shortcut, technical lineage in Collibra would be broken up. Now, there is end-to-end lineage within PowerCenter even when a mapplet has an associated shortcut. • Fixed a ValidationError related to the unsupported Exasol dialect. The Postgres dialect is now used in place of Exasol dialect.

Changed in version	New lineage harvester improvements
2022.11	• When you integrate Power BI: ◦ Inactive workspaces and personal workspaces are no longer ingested. ◦ Filtering is improved. You can now use the optional properties <code>excludeWorkspaceNames</code> and <code>excludeWorkspaceIds</code> to exclude specified workspaces. Before configuring your filters, ensure that you read all about the advantages, limitations and configuration considerations in Power BI workspaces. ◦ The ownership information (admin and creator email addresses only) for reports is now ingested in Collibra. The "Owner in source" attribute is included on Power BI Report asset pages. ◦ The email addresses of all admins and creators of Power BI data models and workspaces are now ingested. Previously only a single email address was ingested, even if there were multiple admins or creators of the data object in Power BI. • When you ingest Snowflake data sources, the <code>databaseNames</code> property is now correctly taken into consideration. • When you integrate Tableau: ◦ Previously, when you filtered on a site, a Tableau Site asset was created in Collibra, but no metadata was ingested. Now, when you filter on a site, all metadata in the site is ingested in the specified domain. If, however, a site is specified in the lineage harvester configuration file, but not in the <code>filters</code> and <code>domainMapping</code> properties in the Tableau <source ID> configuration file, the metadata is ingested in the default domain. ◦ You can now use wildcards in the <code>filters</code> property in the Tableau <source ID> configuration file. Also, the <code>filters</code> property is no longer case-sensitive. ◦ You can now ingest sites that don't have workbooks. ◦ Ownership information (email addresses only) for projects, data models, workbooks and dashboard is now ingested in Collibra. The Owner in source attribute is included on Tableau Project, Tableau Data Model, Tableau Workbook and Tableau Dashboard asset pages. • When you ingest Informatica PowerCenter data sources, the lineage harvester now correctly processes session mapplets. Previously, this failed with error message "'NoneType' object has no attribute 'lower'". • When you ingest Informatica Intelligent Cloud Services data sources and the <code>useCollibraSystemNames</code> property is set to <code>true</code>, databases are now shown in the Technical lineage Browse tab pane with the specified system name or as "UNDEFINED", if a database could not be mapped to a system name. If set to <code>false</code>, then all databases are now shown directly under the

Changed in version	New lineage harvester improvements
	DATABASE node. • When you ingest metadata from Oracle data sources, you can now add a new <code>DatabaseOracle</code> section in your lineage harvester configuration file, to specify the Oracle database name and ensure stitching without any work-arounds. • If you integrate SSRS-PBRS and use a <source ID> configuration file, the <code>CustomDataSource</code> section in the <code><source ID></code> configuration file is no longer mandatory. • The lineage harvester now uses Looker 4.0 APIs, with paging options.

Changed in version	New lineage harvester improvements
2022.10	• The lineage harvester now supports the following IBM DB2 constructs: PREVVAL FOR <sequence>, PREVIOUS VALUE FRO <sequence>, NEXTVAL FOR <sequence> and NEXT VALUE FOR <sequence>. • You can now use the new optional "deleteRawMetadataAfterProcessing" property in your lineage harvester configuration file. With this property, you can delete your raw metadata from the Collibra Data Lineage service after processing. This property is applicable for all supported data sources. • When you specify a Data Catalog URL in the lineage harvester configuration file, it no longer matters whether you include a trailing slash (/) in the URL. • The Collibra Data Lineage service now supports the following transformations: Table.FromRecords and Table.IsEmpty. • Collibra Data Lineage now supports key-pair authentication when ingesting Snowflake data sources. • The PostgreSQL JDBC Driver is upgraded to version 42.4.1. • The Collibra Data Lineage service can now compute indirect lineage from set queries, which are queries with the UNION keyword with the ORDER BY clause. • When you integrate Power BI, the lineage harvester is now more resilient to OutOfMemory errors. • When you integrate Tableau and filter on a sub-project, the metadata of the parent project is no longer ingested in Collibra. However, the parent Tableau Project asset is created in the default domain, to preserve the hierarchy required for stitching. • Looker integration no longer fails if the "collibraSystemName" property is not included in the lineage harvester configuration file. If you want to specify the system name of a database in Looker, use the "collibraSystemName" property in the Looker source ID configuration file. If you don't specify a system name in the source ID configuration file, the system name in the technical lineage graph will be Default. • In the case of a lookup procedure when ingesting Informatica Intelligent Cloud Services data sources, if the CONNECTIONSUBTYPE parameter is empty, the Collibra Data Lineage service now looks to the CONNECTIONREFERENCE parameter for the name. If that is also empty, then the name in the VARIABLE parameter is used. The ensures the correct detection of the SQL dialect. • Fixed an issue related to dialect extraction when ingesting Informatica Intelligent Cloud Services data sources.

Changed in version	New lineage harvester improvements
2022.09	• Previously, when you created a technical lineage for Power BI, SQL Server Reporting Services (SSRS) or Power BI Report Server (PBRS), the nodes in the technical lineage graph had a gray background, even if the data objects from your data source were stitched to assets in Data Catalog. Data objects now have the intended yellow background when creating a technical lineage for Power BI, SSRS or PBRS. We introduced this enhancement for Tableau and Looker in Collibra 2022.07. • When you integrate Tableau, for every Tableau Workbook that you have permission to ingest, all Tableau Dashboards in the Workbooks are now correctly shown in the technical lineage graph. If you do not have permission on the Workbook or Dashboard level, the metadata of these data objects is not ingested. • When integrating Power BI, the ownership information (email address only) for reports is now ingested in Collibra. The new Owner in source attribute is included on Power BI Report asset pages. • The lineage harvester now uses Looker 4.0 APIs, with paging options. • When you integrate Power BI, the lineage harvester is now more resilient against OutOfMemory errors. • When you integrate Tableau and use domain mapping, subprojects are now ingested in the domains of their parent projects. • The Collibra Data Lineage service instances now benefit from the following parsing enhancements when integrating Snowflake data sources: ◦ Support for the COLLATE keyword. ◦ Support for EXTERNAL TABLE syntax. • When integrating Power BI, the descriptions of Data Set Tables and Data Set Columns in Power BI are now harvested. • Fixed an issue that was resulting in a processing error when a column referenced in an ORDER BY clause references a repeated column in the SELECT column list. • When integrating Tableau, you can now ingest sub-projects for which you have permission to ingest, even if you don't have permission to ingest the parent projects.

Changed in version	New lineage harvester improvements
2022.08	• Previously, when you created a technical lineage for a supported BI tool, the nodes in the technical lineage graph had a gray background, even if the data objects from your data source were stitched to assets in Data Catalog. Data objects now have the intended yellow background when creating a technical lineage for Power BI. This enhancement was introduced for Tableau or Looker in Collibra 2022.07. Soon, the enhancement will also apply to SSRS and PBRS. • When synchronizing Tableau, the synchronization no longer fails if two data sources in the same project with the same name are returned from the Tableau API. The assets of both data sources are now synchronized in Collibra. • You can now filter on the Tableau project level. • When integrating Power BI, you can now ingest measures and show them in the technical lineage. Measures are included as the value in the Role in Report attribute on Power BI Column asset pages. • When attempting to integrate Power BI with invalid Power BI credentials, the lineage harvester log file now provides a more helpful error message. • When you specify the Power BI workspaces for ingestion, the filters are not case sensitive now. • When integrating Looker, the ownership information (email address only) for folders, Looks and dashboards is now ingested in Collibra. The new Owner in source attribute is included on Looker Folder, Looker Look and Looker Dashboard asset pages. • When integrating Power BI, the ownership information (email address only) for data sets and workspaces is now ingested in Collibra. The new Owner in source attribute is included on Power BI Data Model and Power BI Workspace asset pages. • The lineage harvester log file now identifies whether you are using Tableau Online or Tableau Server, and the version of your Tableau environment.

Changed in version	New lineage harvester improvements
2022.07	• The lineage harvester now retries to get a batch status again if the first HTTP call failed due to a network error. • Fixed an issue that was causing custom SQL queries to be identified as belonging to two different Tableau data sources. This resulted in a "Unique constraint failed" error. • Fixed an issue that was resulting in the No asset matches the specified criteria error. • When the lineage harvester fetches an access key for a data store, only active records are now fetched. Inactive records are ignored. • The lineage harvester is more resilient against authorization expiration when ingesting Looker metadata. • The lineage harvester log file now includes the following information: ◦ Your Tableau environment type: Tableau Online or Tableau Server type ◦ The version of your Tableau environment
2022.06	• When synchronizing Power BI, the last sync time is now correctly shown in the Sources tab page. • Fixed an issue that was causing the processing of harvested metadata batches to run without coming to completion. • When ingesting Power BI, if there are Oracle data sources, the Oracle service name is now used, instead of the database name. • When processing Tableau metadata, the Collibra Data Lineage servers no longer replace ">>" by "<}", which was resulting in parsing errors. • Fixed an [SQLITE_ERROR] issue that was breaking the technical lineage when attempting to synchronize a data source. • When processing Power BI metadata, SQL statements are now in upper case. • When creating a technical lineage for Tableau, any unnecessary brackets "[[" in the names of schemas are now removed. • When integrating Power BI, you can now ingest measures without DAX. They are shown as attribute type Role in Report on Power BI Column asset pages.

Changed in version	New lineage harvester improvements
2022.05	Warning The lineage harvester 2022.05 includes an internal format change to the password manager <code>pwd.conf</code> file. This means that if you use Lineage harvester 2022.05, you can no longer use the <code>pwd.conf</code> file with an older harvester. - You can now integrate Power BI in Data Catalog via the lineage harvester, meaning you no longer need to use the Power BI harvester. Additional benefits include the following: - Support for Power BI Data Flows. - Descriptions of Power BI Reports. - Statuses of Power BI Workspaces. - Filtering and domain mapping. Note The new Power BI integration method is specifically for new integrations. For those who have been ingesting Power BI via the Power BI harvester, we will soon release a migration script. - Collibra Data Lineage now also supports the following BI integrations: - MicroStrategy - SQL Server Reporting Services and Power BI Report Server. - You can now use token-based authentication when creating a technical lineage for Matillion. Warning This enhancement is not backwards compatible. You must update your configuration file. If you use the lineage harvester 2022.05, you can no longer use the <code>pwd.conf</code> file with an older harvester. - The <code>useCollibraSystemName</code> property is now solely used for the configuration of the system name. - If you set the <code>useCollibraSystemName</code> property to <code>true</code> in your lineage harvester configuration file, but don't define the system name in the Tableau <source ID> configuration file, the system name in the Tableau technical lineage shows DEFAULT as the system name. - If using a Tableau <source ID> configuration file: - You can now use wildcards throughout the file. - The <code>hostname</code> and <code>connectorUrl</code> properties are no longer case-sensitive. - The PostgreSQL JDBC driver is now upgraded from 42.3.2 to 42.3.3. - The Apache Hive JDBC driver is now upgraded from 2.6.17.1020 to 2.6.19.2022.

Changed in version	New lineage harvester improvements
	• The lineage harvester no longer hangs when harvesting metadata from certain data sources. • The lineage harvester automatically refreshes Tableau tokens. • You can now use the optional <code>concurrencyLevel</code> property in the lineage harvester configuration file, to specify the internal sizing, meaning the amount of tasks that can be executed at the same time.
2022.04	• You can now use the <code>databaseMapping</code> property in your Tableau <source ID> configuration file, to map a Tableau technical database name to the real database name. • When providing connection definitions for Informatica PowerCenter, the <code>dbname</code> property is no longer case-sensitive. • When integrating Informatica PowerCenter data sources, Collibra Data Lineage now correctly creates a technical lineage when <code>useCollibraSystemName</code> is set to <code>true</code>.
2022.03	• By default, the lineage harvester no longer harvests images. If you want to include images, include the optional <code>excludeImages</code> property in your configuration file and set the value to <code>false</code>. • When ingesting Tableau metadata, you can now leave empty the <code>collibraSystemName</code> property in your configuration file, even if the <code>useCollibraSystemName</code> property is set to <code>true</code>. • The lineage harvester now correctly shows the help overview when you run the <code>--help</code> command. • Hive source now skips harvesting DDL of exclusively locked tables. • When you change the domain reference ID in the lineage harvester configuration file, Tableau assets are now successfully deleted from the previous domain and recreated in the new domain. • You no longer see a Fiber Failed error while running the lineage harvester. • Protobuf is upgraded to version 3.19.3. • Fixed an issue that was causing incomplete technical lineage and stitching issues when using custom SQL in Tableau. • Fixed an issue that resulted in a <code>TableauHarvesterError</code> when ingesting Tableau metadata via the lineage harvester. • Fixed a <code>NullPointerException</code> when no column data type is harvested. • Fixed an issue that was causing the ingestion of Looker metadata to fail. • Fixed an issue that was causing a <code>JsonParseError</code> when ingesting Tableau metadata.

Changed in version	New lineage harvester improvements
2022.02	

Lineage harvester system requirements

To [install](#) and run the lineage harvester, you have to meet the following requirements.

Software requirements

Java Runtime Environment version 11.0.18 or newer, or OpenJDK 11.0.18 or newer.

Note To ingest Snowflake data sources, the minimum requirement is Java Runtime Environment version 16 or newer, or OpenJDK 16 or newer. For the lineage harvester to function properly, set the `JAVA_OPTS` environment variable when you [run the lineage harvester](#). For example, to process data from all data sources including the Snowflake data sources, take the following steps:

On Windows

1. Enter one of the following commands:
 - If you use OpenJDK 16:

```
set JAVA_OPTS="-Djdk.moduleillegalAccess=permit"
```

- If you use OpenJDK 17 or higher:

```
set JAVA_OPTS="--add-opens=java.base/java.nio=ALL-UNNAMED"
```

2. In the same command line, enter the following command:

```
.\bin\lineage-harvester.bat full-sync
```

Note The `set` command is specific to the Windows Command Shell. The command is different if you are using PowerShell.

On Linux

Enter the following command:

- If you use OpenJDK 16:

```
JAVA_OPTS="-Djdk.moduleillegalAccess=permit"
./bin/lineage-harvester full-sync
```

- If you use OpenJDK 17 or higher:

```
JAVA_OPTS="--add-opens=java.base/java.nio=ALL-UNNAMED"  
./bin/lineage-harvester full-sync
```

Hardware requirements

You need to meet the hardware requirements to install and run the lineage harvester.

Minimum hardware requirements

You need the following minimum hardware requirements:

- 2 GB RAM
- 1 GB free disk space

Recommended hardware requirements

The minimum requirements are most likely insufficient for production environments. We recommend the following hardware requirements:

- 4 GB RAM

Tip 4 GB RAM is sufficient in most cases, but more memory could be needed for larger harvesting tasks. For instructions on how to increase the maximum heap size, see the advice on how to [resolve Java heap space errors](#), in the Colibra Support Portal.

- 20 GB free disk space

Network requirements

The lineage harvester uses the HTTPS protocol by default and uses port 443.

You need the following minimum network requirements:

- Firewall rules so that the lineage harvester can connect to:
 - The host names of all data sources in your lineage harvester [configuration file](#).
 - All [Collibra Data Lineage service instances](#) in your geographic location:
 - 15.222.200.199 (techlin-aws-ca.collibra.com)
 - 18.198.89.106 (techlin-aws-eu.collibra.com)
 - 13.228.38.245 (techlin-aws-sg.collibra.com)
 - 54.242.194.190 (techlin-aws-us.collibra.com)
 - 51.105.241.132 (techlin-azure-eu.collibra.com)
 - 20.102.44.39 (techlin-azure-us.collibra.com)
 - 35.197.182.41 (techlin-gcp-au.collibra.com)
 - 34.152.20.240 (techlin-gcp-ca.collibra.com)
 - 35.205.146.124 (techlin-gcp-eu.collibra.com)
 - 34.87.122.60 (techlin-gcp-sg.collibra.com)
 - 35.234.130.150 (techlin-gcp-uk.collibra.com)
 - 34.73.33.120 (techlin-gcp-us.collibra.com)

Note The lineage harvester connects to different Collibra Data Lineage service instances based on your geographic location and cloud provider. If your location or cloud provider changes, the lineage harvester rescans all your data sources. You have to allow all Collibra Data Lineage service instances in your geographic location. In addition, we highly recommend that you always allow the techlin-aws-us instance as a backup, in case the lineage harvester cannot connect to other Collibra Data Lineage service instances.

Install the lineage harvester

Before you can use the lineage harvester, you need to download and install it. You can download the lineage harvester from the [Collibra Community downloads page](#).

Requirements and permissions

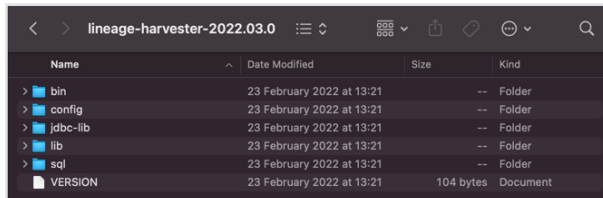
- Collibra Data Intelligence Cloud.
- You have purchased Collibra Data Lineage.
- A [global role](#) with the following [global permissions](#):
 - Catalog, for example Catalog Author
 - Data Stewardship Manager

- Manage all resources
- System administration
- Technical lineage
- A [resource role](#) with the following [resource permissions](#) on the community level in which you created the domain:
 - Asset: add
 - Attribute: add
 - Domain: add
 - Attachment: add
- Necessary permissions to all [database objects](#) that the lineage harvester accesses.
- You meet the [minimum system requirements](#).
- You have added Firewall rules so that the lineage harvester can connect to:
 - The host names of all databases in the lineage harvester configuration file.
 - All [Collibra Data Lineage service instances](#) within your geographical location:
 - 15.222.200.199 (techlin-aws-ca.collibra.com)
 - 18.198.89.106 (techlin-aws-eu.collibra.com)
 - 13.228.38.245 (techlin-aws-sg.collibra.com)
 - 54.242.194.190 (techlin-aws-us.collibra.com)
 - 51.105.241.132 (techlin-azure-eu.collibra.com)
 - 20.102.44.39 (techlin-azure-us.collibra.com)
 - 35.197.182.41 (techlin-gcp-au.collibra.com)
 - 34.152.20.240 (techlin-gcp-ca.collibra.com)
 - 35.205.146.124 (techlin-gcp-eu.collibra.com)
 - 34.87.122.60 (techlin-gcp-sg.collibra.com)
 - 35.234.130.150 (techlin-gcp-uk.collibra.com)
 - 34.73.33.120 (techlin-gcp-us.collibra.com)

Note The lineage harvester connects to different instances based on your geographic location and cloud provider. If your location or cloud provider changes, the lineage harvester rescans all your data sources. You have to allow all Collibra Data Lineage service instances in your geographic location. In addition, we highly recommend that you always allow the techlin-aws-us instance as a backup, in case the lineage harvester cannot connect to other Collibra Data Lineage service instances.

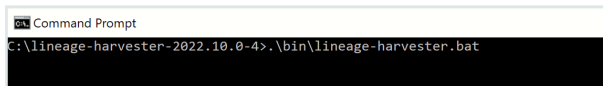
Steps

1. [Download](#) the newest lineage harvester.
2. Unzip the archive.
 - » You can now access the lineage harvester folder. The lineage harvester folder name is unique per version.

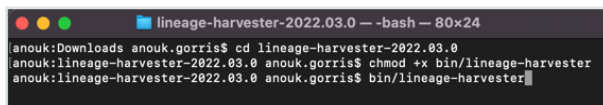


3. Start the lineage harvester to create an empty lineage harvester configuration file by entering the following command:

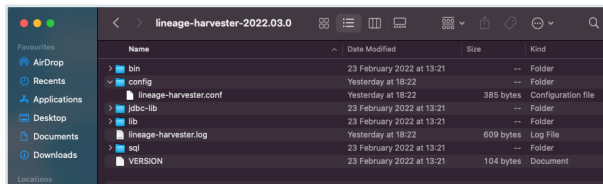
- Windows: `.\bin\lineage-harvester.bat`



- For other operating systems: `chmod +x bin/lineage-harvester` and then `bin/lineage-harvester`



- » An empty configuration file is created in the config folder.



- » The lineage harvester is installed automatically. You can check the installation by running `./bin/lineage-harvester --help`.

What's next?

[Prepare](#) the lineage harvester configuration file.

Lineage harvesting app command options and arguments

After creating a [configuration file](#), you can use the lineage harvester to perform specific actions with the data sources that are defined in your configuration file.

Tip If you run the lineage harvester in command line, you will see an overview of possible command options and arguments that you can use. If the [lineage harvester process](#) fails, you can use the [technical lineage troubleshooting guide](#) to fix your issue.

Typical command options and arguments

The following table shows the most commonly used command options and arguments. You can see a full list of commands by entering the `--help` command in the command line. Note that commands that are not listed in this table are intended for internal use.

Command	Description
full-sync	Uploads all of the metadata from the data sources mentioned in your configuration file to the Collibra Data Lineage service, where the metadata is then processed and uploaded to Data Catalog. After you enter this command, the lineage harvester starts synchronization processing and displays the total number of data sources that are being ingested. Synchronization processing ends with an error in the following situations: • The lineage harvester does not find any data sources, • The <code>useSystemName</code> value is not the same for all data sources. The value of <code>useSystemName</code> is based on the following settings: ◦ The <code>useCollibraSystemName</code> property in the lineage harvester configuration file for different data sources. ◦ The Collibra system name setting on Edge.

Command	Description
<code>-s "<ID of data source>"</code>	Uploads only the metadata from a specified data source. For example, <code>full-sync -s "myOracleDataSource"</code>. The specified data source must be mentioned in your configuration file. This command allows you to process data from a newly added data source or to refresh a data source in the configuration file, without refreshing the other data sources. This reduces the time you need to upload your data sources, since you only upload specific ones without affecting the others. If you want to process multiple data sources, add <code>-s "ID of another data source"</code> per data source to the command. You can concurrently synchronize as many sources as you like. On the Collibra Data Lineage service instance, the sources are processed sequentially. Note You can use this argument multiple times to include multiple data sources.
<code>--no-matching</code>	Uploads a technical lineage without stitching the data objects in your technical lineage to the corresponding Column and Table assets in Data Catalog. Note As a result, you won't see the technical lineage of a specific Table or Column asset, but you can still see and browse the full technical lineage.

Command	Description
sync	Whereas <code>full-sync</code> ingests metadata onto the Collibra Data Lineage service, processes the metadata and syncs it with assets in Data Catalog, the <code>sync</code> command only performs this last part: it syncs the metadata—as it exists on the Collibra Data Lineage service—and your assets in Data Catalog. After you enter this command, the lineage harvester starts synchronization processing and displays the total number of data sources that are being ingested. Synchronization processing ends with an error in the following situations: • The lineage harvester does not find any data sources, • The <code>useSystemName</code> value is not the same for all data sources. The value of <code>useSystemName</code> is based on the following settings: ◦ The <code>useCollibraSystemName</code> property in the lineage harvester configuration file for different data sources. ◦ The Collibra system name setting on Edge. Tip See the following example for advice on how to use the <code>sync</code> command to add a new data source without re-harvesting all data sources. Example Let's say you've run <code>bin/lineage-harvester full-sync</code>, to upload from all data sources, process the metadata and sync with Data Catalog. You then decide that you want to add a new data source, but not harvest all data sources again.

Command	Description
	1. Reference the new data source in the lineage harvester configuration file. Let's say that the new data source has the ID "MyNewSource". 2. Run <code>bin/lineage-harvester load-sources -s MyNewSource</code>, to load the new data source and create the ZIP file. 3. Run <code>bin/lineage-harvester analyze \${zip_file_from_step_2}</code>, to analyze the new data source on the Colibra Data Lineage service. 4. Run <code>bin/lineage-harvester sync</code>, to sync all of the data sources referenced in your configuration file and Data Catalog.

Command	Description
<pre>-s "<ID of data source>"</pre>	Syncs only the metadata on the Collibra Data Lineage service, from a specified data source. For example, <code>sync -s "myOracleDataSource"</code>. The specified data source must be mentioned in your configuration file. This command allows you to sync data from one data source without refreshing the other data sources. You must have previously uploaded the metadata to the Collibra Data Lineage service. Warning Only the sources you specify are synced. This means that any previously ingested metadata from non-specified sources, in Data Catalog, is deleted, along with its existing technical lineage. If this is not your intention, consider using <code>full-sync -s</code>. With <code>full-sync -s</code>, all sources are synced, regardless of which sources are specified by the <code>-s</code> command. Therefore, any previously ingested metadata from non-specified data sources remains, as do the respective technical lineages. Note You can use this argument multiple times to include multiple data sources.
<pre>analyze \${name-of-zip-file}</pre>	Analyzes a specified batch (ZIP file) of metadata on the Collibra Data Lineage service instance. The Sources tab page shows the transformation details or source code that was analyzed and the results of the analysis.

Command	Description
load-sources	Downloads all your data sources in a separate ZIP file, per data source, to the lineage harvester output folder.
-s <ID of data source>	Downloads only the data source with a specific ID. For example, <code>load-sources -s "myOracleDataSource"</code>. Note You can use this argument multiple times to include multiple data sources.

Command	Description
<code>list-sources</code>	Lists all of the data sources that will be used to create a technical lineage. When you enter this command, up to 500 data sources are listed per page by default. The list includes the following details for each data source: Source ID <ID of data source> (from edge: false true) (useSystemName: false true). Source ID <ID of data source> The source ID of your data source. from edge: false true Indicates whether the data source is ingested by using technical lineage via Edge. If the value is <code>true</code>, the data source is ingested by using technical lineage via Edge. If the value is <code>false</code>, the data source is ingested by using the lineage harvester. useSystemName: false true Indicates whether Collibra Data Lineage uses the system or server name of the data source to match the System asset in Data Catalog. If the value is <code>true</code>, the system or server name of the data source is used. If the value is <code>false</code>, the system or server name of the data source is not used. The value of <code>useSystemName</code> is based on the following settings: • The <code>useCollibraSystemName</code> property in the lineage harvester configuration file for the data source. • The Collibra system name setting for the data source on Edge.

Command	Description
	Example Source ID <code>1redshift</code> (<code>from edge: false</code>) (<code>useSystemName: false</code>) indicates that the data source with the <code>1redshift</code> source ID was ingested by using the lineage harvester, and the system name of the data source is not used to match the System asset in Data Catalog.
<code>-p <page number></code>	Specifies the page to be displayed. The value of <code><page number></code> must be greater than 0. This option is optional. For example, if you enter <code>list-sources -p 2</code>, page 2 is displayed with a default page size of 500 data sources listed. If there are less than 500 data sources in total, an error message is issued. Note To use the <code>-p</code>, <code>-s</code>, and <code>-all</code> options, you must have the lineage harvester version 2023.05 or newer.

Command	Description
<code>-s <number of data sources></code>	Specifies the number of data sources to be listed on one page. The value of <code><number of data sources></code> must be in the range 0 - 500. This option is optional. For example, if you enter <code>list-sources -s 40</code>, default page 1 is displayed with 40 data sources listed. If there are 80 data sources in total, you see the Displaying page 1 of 2 message and a list of 40 data sources. If you enter <code>list-sources -p 3 -s 20</code>, page 3 is displayed with 20 data sources listed. If there are 80 data sources, you see the Displaying page 3 of 4 message and a list of 20 data sources. Note To use the <code>-p</code>, <code>-s</code>, and <code>-all</code> options, you must have the lineage harvester version 2023.05 or newer.
<code>-all</code>	Lists all data sources. The data sources are not formatted in pages. If you enter this option with the <code>-p</code> and <code>-s</code> options, this option overrides the <code>-p</code> and <code>-s</code> options. For example, if you enter <code>list-sources -p 3 -s 20 --all</code>, all data sources are listed. Note To use the <code>-p</code>, <code>-s</code>, and <code>-all</code> options, you must have the lineage harvester version 2023.05 or newer.

Command	Description
<pre>ignore-source <source_id></pre>	Ignores the specified data source from the list of data sources that will be used to create the technical lineage, where <code><source_id></code> is the ID of the data source that you want to ignore. When you create the technical lineage again by entering the <code>sync</code> command or synchronizing a technical lineage capability via Edge, the specified data source is ignored. You can specify only one source ID at a time. If your source ID includes spaces, enclose the source ID in double or single quotation marks, for example <code>ignore-source "Source A"</code>. You can use this command to delete the technical lineage of a data source by using the lineage harvester. For details, go to Delete the technical lineage of a data source if you use the lineage harvester and Delete the technical lineage of a data source on Edge for technical lineage via Edge. Note To use the <code>ignore-source</code> command, you must have the lineage harvester version 2023.04 or newer.
<pre>cat passwords.json ./bin/lineage-harvester <command-like-full-sync> --passwords-stdin</pre>	Provides passwords of your Collibra Data Intelligence Cloud instance and the data sources in your configuration file to the lineage harvester without storing the passwords in the lineage harvester folder. You can replace <code>cat passwords.json</code> by a string generated by your password manager.

Command	Description
<code>test-connection</code>	Checks the connectivity to the Collibra Data Lineage service instance and to Data Catalog. The logs will also show the IP addresses of the Collibra Data Lineage service instances that you have to allow. This command is mostly used for troubleshooting purposes.
<code>--help</code>	Shows an overview of all supported command options and arguments that you can use in the lineage harvester.
<code>--version</code>	Shows the version of the lineage harvester that you are using.
<code>-Dlineage-harvester.log.dir=path/to/log/dir</code>	Determine the path of the log file.

Technical lineage password manager integration design

When you run the lineage harvester, you can either:

- Enter the passwords in the console. The passwords are then encrypted and stored in `/config/pwd.conf`.

Note Lineage harvester 2022.05 includes an internal format change to the password manager `pwd.conf` file. This means that if you use Lineage harvester 2022.05, you can no longer use the `pwd.conf` file with an older lineage harvester version.

- Provide the passwords via [command line](#), in a prescribed JSON structure via [stdin](#). This allows you to store the passwords locally in your password manager, instead of in your lineage harvester folder.

This topic provides guidance on how to structure the JSON file and which commands to use, to store the passwords locally in your password manager.

Structure of the JSON file

If you prepare a JSON file with your passwords, you have to name the file *passwords.json*.

The JSON file must have two sections:

- The `catalogs` section defines the connection information and credentials to your Collibra Data Intelligence Cloud instance.
- The `sources` section defines the connection information and credentials to your data sources. You use the same "id" as the `id` property in the lineage harvester configuration file.

The JSON file must have the following structure:

```
{
  "catalogs": [
    {
      "url" : "<url-to-collibra-cloud>",
      "username": "<username-to-sign-in-to-collibra>",
      "password": "<password-to-sign-in-to-collibra>"
    }
  ],
  "sources": [
    {
      "id": "<id-of-your-database>",
      "username": "<database-username>",
      "password": "<database-password>"
    }
  ]
}
```

Examples of commands

When you run the lineage harvester, you can use one of the following commands to provide the passwords:

Passwords location	Command
a locally stored JSON file	<code>cat passwords.json ./bin/lineage-harvester full-sync --passwords-stdin</code>

Passwords location	Command
a custom script, for example from a password manager	<pre><prepare-passwords-command> ./bin/lineage-harvester full-sync --passwords-stdin</pre> Note Depending on your password manager, you may need different parameters. For example, see the LastPass documentation for the parameters needed by LastPass.

Connecting to a proxy server

You can connect to a proxy server when you use the lineage harvester. Collibra Data Lineage supports proxy server connection and authentication.

Set the environment variable on Windows or set the system properties on other operating systems with the following parameters specified to connect to a proxy server. See the following steps for code examples.

- `-Dhttps.proxyHost`
- `-Dhttps.proxyPort`
- `-Dhttps.proxyUser`
- `-Dhttps.proxyPassword`
- `-Dhttp.nonProxyHosts`

The `-Dhttps.proxyUser` and `-Dhttps.proxyPassword` parameters are optional.

On Windows

1. Set the `-D` parameter to the `JAVA_OPTS` environment variable.

Example

```
set JAVA_OPTS=-Dhttps.proxyHost="azusquid.imf.org" -Dhttps.proxyPort="8080" -Dhttps.proxyUser="myusername" -Dhttps.proxyPassword="mypassword"
```

2. Run the lineage harvester in the same command line window: `.\bin\lineage-harvester.bat`

On other operating systems

1. To access the hosts via a proxy server, run the following command: `bin/lineage-harvester -Dhttps.proxyHost=<Hostname or IP address of the proxy> -Dhttps.proxyPort=<port number> -Dhttps.proxyUser=<username> -Dhttps.proxyPassword=<password> full-sync`

Example If you want to use a proxy with hostname *proxy.example.com* and port number *443*, run the following command:

```
bin/lineage-harvester -Dhttps.proxyHost=proxy.example.com
-Dhttps.proxyPort=443 -Dhttps.proxyUser=myusername -
Dhttps.proxyPassword=mypassword
```

2. To exclude hosts that should be accessed without going through the proxy server, add the following parameter: `-Dhttp.nonProxyHosts=<host to exclude>`. You can exclude multiple hosts by using the pipe character (`|`) to separate the hostnames or IP addresses to exclude. You can also use an asterisk (`*`) as a wildcard to match multiple hostnames or IP addresses.

Example If you want to exclude hosts with hostname *localhost* and hosts with IP address *127.0.0.1* and all IP addresses starting with *192.168**, run the following command:

```
bin/lineage-harvester -Dhttps.proxyHost=proxy.example.com
-Dhttps.proxyPort=443 -
Dhttp.nonProxyHosts=localhost|127.0.0.1|192.168*
```

Important In your configuration file, the value of the source "url" or "hostname" property (depending on the data source), and the value in your `-Dhttp.nonProxyHosts` parameter, as described above, must both be either an IP address or a host name. You will get an error if, for example, you have a host name in the "hostname" property and an IP address in the `-Dhttp.nonProxyHosts` parameter.

Prepare the lineage harvester configuration file

Before you can visualize the technical lineage, you have to create a [configuration file](#) for the (meta)data sources that you want to process. This configuration file is used by the [lineage harvester](#) to extract data from (meta)data sources for which you want to create a technical lineage or you want to ingest.

If you use multiple lineage harvesters on different servers, you can create a separate configuration file for the lineage harvester on each server and configure different data sources in each configuration file.

Note

- Technical lineage [supports](#) a limited list of (meta)data sources.
- In all lineage harvester files, you must use UTF-8 or ISO-8859-1 characters, with the exception of SQL files, which can only be UTF-8 encoded.
- Each data source has an ID property. The ID string must be unique and human readable. The ID can be anything and is only used to identify the batch of metadata that is processed on the Collibra Data Lineage service.
- The lineage harvester connects to different [Collibra Data Lineage service instances](#) based on your geographical location and cloud provider. Make sure you have the correct [system requirements](#) before you run the lineage harvester. If your location or cloud provider changes, the lineage harvester rescans all your data sources.
- Comments in the lineage harvester configuration file are not supported.

Before you begin

- [Download](#) and [install](#) the lineage harvester.

Tip You can use the [configuration file generator](#) to create an example configuration file to accommodate the data sources you specify in the generator. You can then copy the example code to your configuration file and replace the values of the properties to suit your needs.

Steps

1. Open the lineage-harvester.conf file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
general	This section describes the connection between Collibra Data Lineage and Data Catalog.
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.
url	The URL of the Collibra Data Lineage service instance. Example “url”: “https://techlin-gov.collibra.com” Warning This section applies only to US government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.

Properties	Description
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a JDBC data source to match the System asset that you created when you prepared the physical data layer. The names are case-sensitive. Specify one of the following values: <code>false</code> The lineage harvester ignores all system or server names that you specify on the <code>collibraSystemName</code> properties in the <source ID> file. This is the default value. <code>true</code> The lineage harvester reads the system and server names that you specify on the <code>collibraSystemName</code> properties in all sections of the <source ID> file. Note Specify this property with the value of <code>true</code> only when you have multiple databases with the same name.

Properties	Description
sources	This section describes the data sources for which you want to create the technical lineage. You have to create a configuration section for each data source.
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable. For example, <i>my_adf</i> .
type	The type of data source. The value must be <i>AzureDataFactory</i> .
collibraSystemName (Deprecated)	This property is deprecated. If you specify a value for this property, it is ignored. To override the default Collibra System asset name, use the <code>useCollibraSystemName</code> property in the source id file .
tenantDomain	The directory ID of the Azure Data Factory instance.
loginFlow	This section contains the login application information.
applicationId	The application ID of the Azure Data Factory instance.
type	The identity of the application. The value has to be <i>ServicePrincipal</i> .
resourceGroupName	The name of the resource group with the Reader role for the Azure Data Factory instance.
subscriptionId	The subscription ID of the resource group.

Properties	Description
factories	The Azure Data Factory factories that the lineage harvester collects and processes. Specify this property with an array of Azure Data Factory factory names. This property is optional. The following rules apply when you specify this property: ◦ Enter the factory names in square brackets ([]), enclose each factory name in double quotes (" "), and separate them by a comma, for example, ["MyFirstFactory", "MySecondFactory"]. ◦ The factory name is not case-sensitive. For example, the MyFactory and myfactory factories are considered the same by Azure Data Factory and the lineage harvester. ◦ If you do not specify any factory name, the lineage harvester collects and processes all factories that have datasets and pipelines in them.
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

- 2. Save the configuration file.

Steps

- 1. Open the lineage-harvester.conf file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
general	This section describes the connection between Collibra Data Lineage and Data Catalog.
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.
url	The URL of the Collibra Data Lineage service instance. Example “url”: “https://techlin-gov.collibra.com” Warning This section applies only to US government customers.

Properties	Description
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.

Properties	Description
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a JDBC data source to match the System asset that you created when you prepared the physical data layer. The names are case-sensitive. Specify one of the following values: false The lineage harvester ignores all system or server names that you specify on the <code>collibraSystemName</code> properties in the configuration file. This is the default value. true The lineage harvester reads the system and server names that you specify on the <code>collibraSystemName</code> properties in all sections of the configuration file. Only specify this value when you have multiple databases with the same name. Note For SQL data sources, if this property is: ◦ <code>false</code>, system or server names in table references in analyzed SQL code are ignored. This means that a table that exists in two different systems or servers is identified (either correctly or incorrectly) as a single data object, with a single asset name. ◦ <code>true</code>, system or server names in table references are considered to be represented by different System assets in Data Catalog. The value of the <code>collibraSystemName</code> property is used as the default system or server name.

Properties	Description
sources	This section describes the data sources for which you want to create the technical lineage. You have to create a configuration section for each data source. This configuration section contains the required information of one individual SQL directory with connection type "Folder". Note You can add multiple data sources to the same configuration file.
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable.
type	The kind of data source. In this case, the value has to be <code>SqlDirectory</code> .
path	The full path to the folder where you added SQL files , for example, <code>C:\path\to\config\dir</code> .
mask	The pattern of the file names in the directory. By default, this is <code>*</code> .
recursive	Indication of the files you want to harvest: ◦ <code>false</code> (default): Only harvest the files in directly under the folder in the SQL directory path. ◦ <code>true</code>: Harvest all files under the folder in the SQL directory path and subdirectories.

Properties	Description
dialect	The dialect of the database: redshift azure bigquery greenplum hive db2 oracle postgres mssql mysql netezza snowflakesybasesparkteradata hana, for an SAP HANA data source. hana-cviews, for getting lineage from calculated views in an SAP HANA Classic on-premises data source. hana-cviews-v2, for getting lineage from calculated views in an SAP HANA Cloud/Advanced data source. Important To get technical lineage including calculated views, you must harvest SAP HANA by specifying two data sources in the lineage harvester configuration file. In one data source, specify the <code>hana</code> dialect, and in the other, specify the <code>hana-cviews</code> or <code>hana-cviews-v2</code> dialect. The value your put for this property has to match the dialect you provide with in the directory with your SQL files.

Properties	Description
database	The name of your database, which is the name of your Database asset. Note - You have to use the same database name as the name of the Database asset that you create when you prepare the physical data layer in Data Catalog. The names are case-sensitive. - The database and schema names in the SQL statements in your SQL files take precedence over the values that you provide for the <code>database</code> and <code>schema</code> properties in the lineage harvester configuration file. If your SQL statements contain database and schema names, Collibra Data Lineage uses them for stitching. If your SQL statements do not contain database and schema names, Collibra Data Lineage uses the values of the <code>database</code> and <code>schema</code> properties in the configuration file for stitching.. For more information, go to Prepare an SQL directory and Automatic stitching for technical lineage. Important HiveQL data sources don't have schemas. Therefore, HiveQL databases are stored in Data Catalog and technical lineage as Schema assets. The technical lineage Browse tab pane shows the following names: - The database name is the name that you enter for the <code>collibraSystemName</code> property. - The schema name is the name that you enter for the <code>database</code> property.

Properties	Description
	Important MySQL data sources don't have schemas. Therefore, MySQL databases are stored in Data Catalog and technical lineage as Schema assets. The technical lineage Browse tab pane shows the following names: ◦ The database name is the name that you enter for the <code>database</code> property. Important Teradata data sources don't have schemas. Therefore, Teradata databases are stored in Data Catalog and technical lineage as Schema assets. The technical lineage Browse tab pane shows the following names: ◦ The database name is the name that you enter for the <code>collibraSystemName</code> property. ◦ The schema name is the name that you enter for the <code>database</code> property.
collibraSystemName	The name of the data source's system or server. This is also the name of your System asset in Data Catalog. Specify this property with the same name as the name of the System asset that you create when you prepare the physical data layer in Data Catalog. If you don't prepare the physical data layer, Collibra Data Lineage cannot stitch the data objects in your technical lineage to the assets in Data Catalog. Specify this property with the same name as the name of the System asset that you created when you registered the data source.

Properties	Description
databaseSystemMapping	This optional property allows you to map databases to their rightful systems, to obtain stitching. This resolves missing stitching, which occurs when Collibra Data Lineage associates multiple databases with the default system name that you provide in the <code>collibraSystemName</code> property. More information and example configuration Let's say you use the <code>collibraSystemName</code> property to specify the system name <code>SystemA</code>. This system name will be associated with all of the databases mentioned in your SQL statements in your SQL files. Now let's say that you have a SQL statement that selects data from a database <code>DB2</code> (which is in <code>SystemB</code>) and inserts the data in database <code>DB3</code> (which is part of <code>SystemC</code>). Both of these databases will be associated with <code>SystemA</code>, and both will appear under <code>SystemA</code> in the Browse tab pane. Therefore, stitching fails and duplicate databases might appear. To resolve this issue and obtain stitching, you can use this property to map <code>DB2</code> to its rightful <code>SystemB</code>, and <code>DB3</code> to its rightful <code>SystemC</code>. <pre> { "id": "sql_dir", "type": "SqlDirectory", "path": "/tmp/sql_dir", "dialect": "bigquery", "collibraSystemName": "SystemA", "database": "DB", "schema": "SCH", "databaseSystemMapping": { "DB2": "SystemB", "DB3": "SystemC" } }</pre>

Properties	Description
schema	The name of the default schema, if not specified in the data source itself. This corresponds to name of your Schema asset. Note You must use the same schema name as the name of the Schema asset that you create when you prepare the physical data layer in Data Catalog.
verbose	Indication whether you want to enable verbose logging. By default this is set to <code>True</code>. If you don't want to use verbose logging, set it to <code>False</code>.
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

Properties	Description
general	This section describes the connection between Collibra Data Lineage and Data Catalog.
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.
url	The URL of the Collibra Data Lineage service instance. Example "url": "https://techlin-gov.collibra.com" Warning This section applies only to US government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.

Properties	Description
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.

Properties	Description
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a JDBC data source to match the System asset that you created when you prepared the physical data layer. The names are case-sensitive. Specify one of the following values: false The lineage harvester ignores all system or server names that you specify on the <code>collibraSystemName</code> properties in the configuration file. This is the default value. true The lineage harvester reads the system and server names that you specify on the <code>collibraSystemName</code> properties in all sections of the configuration file. Only specify this value when you have multiple databases with the same name. Note For SQL data sources, if this property is: ◦ <code>false</code>, system or server names in table references in analyzed SQL code are ignored. This means that a table that exists in two different systems or servers is identified (either correctly or incorrectly) as a single data object, with a single asset name. ◦ <code>true</code>, system or server names in table references are considered to be represented by different System assets in Data Catalog. The value of the <code>collibraSystemName</code> property is used as the default system or server name.

Properties	Description
sources	This section describes the data sources for which you want to create the technical lineage. You have to create a configuration section for each data source. This section contains the required information of one individual data source with <code>connection type "JDBC"</code>. Note You can add multiple data sources to the same configuration file.
id	The unique ID of the data source. For example, <code>my_first_data_source</code> .
type	The kind of data source. In this case, the value has to be <code>Database</code> .
username	The username that you use to sign in to your data source.

Properties	Description
dialect	The dialect of the database: redshift azure bigquery greenplum hive db2 oracle postgres mssql mysql netezza snowflakesybasesparkteradata. hana, for an SAP HANA data source. hana-cviews, for getting lineage from calculated views in an SAP HANA Classic on-premises data source. hana-cviews-v2, for getting lineage from calculated views in an SAP HANA Cloud/Advanced data source. Important To get technical lineage including calculated views, you must harvest SAP HANA by specifying two data sources in the lineage harvester configuration file. In one data source, specify the <code>hana</code> dialect, and in the other, specify the <code>hana-cviews</code> or <code>hana-cviews-v2</code> dialect. The value your put for this property has to match the dialect you provide with in the directory with your SQL files.

Properties	Description
databaseNames	The names or IDs of your databases. Enter the database names of your data source between double quotes (") and put everything between square brackets. If you want to include more than one database, separate them by a comma. For example, ["MyFirstDatabase", "MySecondDatabase"]. Note Ensure that you use the same database names as the names of the Database assets. The names are case-sensitive. Important HiveQL, Spark SQL, and Teradata are database-less data sources. Therefore, HiveQL, Spark SQL, and Teradata databases are stored in Data Catalog and technical lineage as Schema assets. The technical lineage Browse tab pane shows the following names: ◦ The database name is the name that you enter for the <code>externalDbName</code> property. ◦ The schema name is the name that you enter for the <code>database</code> property. If you do not specify a value for the <code>externalDbName</code> property, Collibra Data Lineage uses the value of the <code>collibraSystemName</code> property as the database name. For details, see the <code>externalDbName</code> property above.

Properties	Description
externalDbName	This property can be considered a means of database mapping, to help preserve stitching. Note This property is relevant only for HiveQL, Spark SQL, and Teradata data sources, specifically because they are database-less data sources. You can add the key/value pair to the configuration file, as follows: "externalDbName": "<dbname>", where <dbname> is one of the following values: ◦ CData, which Cdata drivers returned as a placeholder. Use this value if you did not create a custom database name by using the CustomizedDefaultCatalogName property when you registered your data source. ◦ The custom database name that you specified for the CustomizedDefaultCatalogName property when you registered your data source. For more information about the CustomizedDefaultCatalogName connection property, go to Customizing the database name for database-less data sources. See an example Let's say you ingest a HiveQL data source via Edge. If you do not customize the database name, Edge gives the name "CData" for the database. The full path to a column is something like: <pre>Hive_123 (system) > CData (database) > Hive_ABC (schema) > Table > Column</pre> Now, because HiveQL is database-less, the value

Properties	Description
	that you give for the <code>databaseNames</code> property in your configuration file is used as the schema name in the technical lineage, and the value you give for <code>collibraSystemName</code> is used as the database name. But if <code>useCollibraSystemName</code> is set to <code>true</code>, then the value of <code>collibraSystemName</code> is also used as the system name. In that case, in the full path to the column, the system name and the database name are the same: <pre>Hive_123 (system) > Hive_123 (database) > Hive_ABC (schema) > Table > Column</pre> Notice the mismatch between the database names. The <code>externalDbName</code> property tells the lineage harvester to use the value that you specify here for the database name in the technical lineage, specifically "CData". This ensures that the full paths match and stitching is preserved. See code examples The following examples show the full paths when you set the <code>useCollibraSystemName</code> and <code>externalDbName</code> properties differently. Code example 1 The <code>useCollibraSystemName</code> property is set to <code>false</code> and the <code>externalDbName</code> property is specified In the following example, the <code>useCollibraSystemName</code> property is set to <code>false</code>, and the <code>externalDbName</code> property is set as <code>CData</code>. The data objects from the HiveQL data source are stitched to the assets in Data Catalog with the following path:

Properties	Description
	CData (database) > Hive_ABC (schema) > Table > Column <pre>{ "general": { : "useCollibraSystemName" : false }, "sources" : [{ "id" : "database_source", "type" : "Database", "username" : "MyUsername", "dialect" : "hive", "externalDbName": "CData" "databaseNames" : ["Hive_ ABC"], "collibraSystemName" : ["Hive_123"], : }] }</pre> Code example 2 The useCollibraSystemName property is set to true and the externalDbName property is specified In the following example, the useCollibraSystemName property is set to true, and the externalDbName property is set as CData. The data objects from the HiveQL data source are stitched to the assets in Data Catalog with the following path: Hive_123 (system) > CData (database) > Hive_ABC (schema) > Table > Column

Properties	Description
	<pre>{ "general": { : "useCollibraSystemName" : false }, "sources" : [{ "id" : "database_source", "type" : "Database", "username" : "MyUsername", "dialect" : "hive", "externalDbName": "CData" "databaseNames" : ["Hive_ ABC"], "collibraSystemName" : ["Hive_123"], : }] }</pre> Code example 3 The <code>useCollibraSystemName</code> property is set to <code>true</code> and the <code>externalDbName</code> property is not specified In the following example, the <code>useCollibraSystemName</code> property is set to <code>true</code>, but the <code>externalDbName</code> property is not specified. The data objects from the HiveQL data source are not stitched to the assets in Data Catalog because of the following mismatch: In Data Catalog, the full path is <code>Hive_123 (system) > CData (database) > Hive_ABC (schema) > Table > Column</code>. However, the lineage harvester has <code>Hive_123 (system) > Hive_123 (database) > Hive_ABC (schema) > Table > Column</code>

Properties	Description
	<pre>{ "general": { : "useCollibraSystemName" : false }, "sources" : [{ "id" : "database_source", "type" : "Database", "username" : "MyUsername", "dialect" : "hive", "externalDbName": "" "databaseNames" : ["Hive_ ABC"], "collibraSystemName" : ["Hive_123"], : }] }</pre>
hostname	The name of your database host.

Properties	Description
<code>collibraSystemName</code>	The name of the data source's system or server. This is also the name of your System asset in Data Catalog. Specify this property with the same name as the name of the System asset that you create when you prepare the physical data layer in Data Catalog. If you don't prepare the physical data layer, Collibra Data Lineage cannot stitch the data objects in your technical lineage to the assets in Data Catalog. Specify this property with the same name as the name of the System asset that you created when you registered the data source. If the <code>useCollibraSystemName</code> property is: ◦ <code>false</code> (default), system or server names in table references in analyzed SQL code are ignored. This means that a table that exists in two different systems or servers is identified (either correctly or incorrectly) as a single data object, with a single asset name. ◦ <code>true</code>, system or server names in table references are considered to be represented by different System assets in Data Catalog. The value of the <code>collibraSystemName</code> field is used as the default system or server name.
<code>port</code>	The port number.
<code>customConnectionProperties</code>	An option to enable the lineage harvester to read additional connection parameters. This parameter is only required in very specific situations. If you don't need it, you can remove it from the configuration file.

Properties	Description
<code>deleteRawMetadataAfterProcessing</code>	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

2. Save the configuration file.

Steps

1. Open the `lineage-harvester.conf` file that was created when you installed the lineage harvester, and enter the values for each property.

Code example

In the following example, the path of `/opt/Collibra/techlin/dbt-core-files/` contains a **target/** directory, which includes the compiled SQL files and `manifest.json` file:

```
ls /opt/Collibra/techlin/dbt-core-files/target
ls /opt/Collibra/techlin/dbt-core-files/target compiled
manifest.json
```

```

{
  "general" : {
    "catalog" : {
      "url" : "https://<organization>.collibra.com",
      "username" : "<your-collibra-username>"
    },
    "useCollibraSystemName" : false
  },
  "sources" : [
    {
      "id" : "my_dbt_source",
      "collibraSystemName" : "",
      "type" : "ExternalDirectory",
      "dirType" : "dbt",
      "path" : "/opt/Collibra/techlin/dbt-core-files/",
      "recursive" : true,
      "deleteRawMetadataAfterProcessing" : false
    }
  ]
}

```

Properties	Description	Required
general	This section describes the connection between Collibra Data Lineage and Data Catalog.	Yes
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.	Yes for US government customers

Properties	Description	Required
url	The URL of the Collibra Data Lineage service instance. Example “url”: “https://techlin-gov.collibra.com” Warning This section applies only to US government customers.	Yes for US government customers
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.	Yes for US government customers
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.	Yes

Properties	Description	Required
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.	Yes
username	The username that you use to sign in to Collibra.	Yes
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a JDBC data source to match the System asset that you created when you prepared the physical data layer. The names are case-sensitive. Specify one of the following values: false The lineage harvester ignores all system or server names that you specify on the <code>collibraSystemName</code> properties in the configuration file. This is the default value. true The lineage harvester reads the system and server names that you specify on the <code>collibraSystemName</code> properties in all sections of the configuration file. Only specify this value when you have multiple databases with the same name.	No

Properties	Description	Required
sources	This configuration section contains the required information of dbt Core data source. Note Make sure that you have prepared a local folder with the SQL files and Manifest JSON file for which you want to create a technical lineage.	Yes
collibraSystemName	The system or server name of the data source. Use this property with the <code>useCollibraSystemName</code> property in the configuration file to override the default Collibra System asset name for this data source. Specify this property with the same name as the name of the System asset that you create when you prepare the physical data layer in Data Catalog. If you don't prepare the physical data layer, Collibra Data Lineage cannot stitch the data objects in your technical lineage to the assets in Data Catalog	No
id	The unique ID that is used to identify the data source on the Collibra Data Lineage service instances. For example, <i>my_dbt</i> .	Yes
type	The kind of data source. In this case, the value has to be <i>ExternalDirectory</i> .	Yes
dirType	The type of external directory. The value has to be <i>dbt</i> .	Yes

Properties	Description	Required
path	The full path to the external directory that you created , for example, <code>/opt/dbt/my-project/</code> or <code>/opt/Collibra/techlin/dbt-core-files</code> . Ensure that the target/ directory is in the external directory.	Yes
mask	The pattern of the file names in the directory. By default, this is <code>*</code> , which sends the SQL and JSON files to the Collibra Data Lineage service instance.	No
recursive	Indication whether you want to use recursive queries. You must set the value to <code>True</code> . By default, this is set to <code>False</code> .	Yes

Properties	Description	Required
<code>deleteRawMetadataAfterProcessing</code>	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.	No

2. Save the configuration file.

Steps

1. Open the `lineage-harvester.conf` file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
<code>general</code>	This section describes the connection between Collibra Data Lineage and Data Catalog.

Properties	Description
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.
url	The URL of the Collibra Data Lineage service instance. Example "url": "https://techlin-gov.collibra.com" Warning This section applies only to US government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.

Properties	Description
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.

Properties	Description
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a JDBC data source to match the System asset that you created when you prepared the physical data layer. The names are case-sensitive. Specify one of the following values: false The lineage harvester ignores all system or server names that you specify on the <code>collibraSystemName</code> properties in the configuration file. This is the default value. true The lineage harvester reads the system and server names that you specify on the <code>collibraSystemName</code> properties in all sections of the configuration file. Only specify this value when you have multiple databases with the same name. Note For SQL data sources, if this property is: ◦ <code>false</code>, system or server names in table references in analyzed SQL code are ignored. This means that a table that exists in two different systems or servers is identified (either correctly or incorrectly) as a single data object, with a single asset name. ◦ <code>true</code>, system or server names in table references are considered to be represented by different System assets in Data Catalog. The value of the <code>collibraSystemName</code> property is used as the default system or server name.

Properties	Description
sources	This section describes the data sources for which you want to create the technical lineage. You have to create a configuration section for each data source. This configuration section contains the required information of one individual SQL directory with connection type "Folder". Note You can add multiple data sources to the same configuration file.
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable.
type	The kind of data source. In this case, the value has to be <code>SqlDirectory</code> .
path	The full path to the folder where you added SQL files , for example, <code>C:\path\to\config\dir</code> .
mask	The pattern of the file names in the directory. By default, this is <code>*</code> .
recursive	Indication of the files you want to harvest: ◦ <code>false</code> (default): Only harvest the files in directly under the folder in the SQL directory path. ◦ <code>true</code>: Harvest all files under the folder in the SQL directory path and subdirectories.
dialect	The dialect of the database. For example, <i>bigquery</i>. The value your put for this property has to match the dialect you provide with in the directory with your SQL files.

Properties	Description
database	The name of your database, which is the name of your Database asset. Note - You have to use the same database name as the name of the Database asset that you create when you prepare the physical data layer in Data Catalog. The names are case-sensitive. - The database and schema names in the SQL statements in your SQL files take precedence over the values that you provide for the <code>database</code> and <code>schema</code> properties in the lineage harvester configuration file. If your SQL statements contain database and schema names, Collibra Data Lineage uses them for stitching. If your SQL statements do not contain database and schema names, Collibra Data Lineage uses the values of the <code>database</code> and <code>schema</code> properties in the configuration file for stitching.. For more information, go to Prepare an SQL directory and Automatic stitching for technical lineage. Important MySQL data sources don't have schemas. Therefore, MySQL databases are stored in Data Catalog and technical lineage as Schema assets. The technical lineage Browse tab pane shows the following names: - The database name is the name that you enter for the <code>database</code> property.

Properties	Description
collibraSystemName	The name of the data source's system or server. This is also the name of your System asset in Data Catalog. Specify this property with the same name as the name of the System asset that you create when you prepare the physical data layer in Data Catalog. If you don't prepare the physical data layer, Collibra Data Lineage cannot stitch the data objects in your technical lineage to the assets in Data Catalog. Specify this property with the same name as the name of the System asset that you created when you registered the data source.

Properties	Description
databaseSystemMapping	This optional property allows you to map databases to their rightful systems, to obtain stitching. This resolves missing stitching, which occurs when Collibra Data Lineage associates multiple databases with the default system name that you provide in the <code>collibraSystemName</code> property. More information and example configuration Let's say you use the <code>collibraSystemName</code> property to specify the system name <code>SystemA</code>. This system name will be associated with all of the databases mentioned in your SQL statements in your SQL files. Now let's say that you have a SQL statement that selects data from a database <code>DB2</code> (which is in <code>SystemB</code>) and inserts the data in database <code>DB3</code> (which is part of <code>SystemC</code>). Both of these databases will be associated with <code>SystemA</code>, and both will appear under <code>SystemA</code> in the Browse tab pane. Therefore, stitching fails and duplicate databases might appear. To resolve this issue and obtain stitching, you can use this property to map <code>DB2</code> to its rightful <code>SystemB</code>, and <code>DB3</code> to its rightful <code>SystemC</code>. <pre> { "id": "sql_dir", "type": "SqlDirectory", "path": "/tmp/sql_dir", "dialect": "bigquery", "collibraSystemName": "SystemA", "database": "DB", "schema": "SCH", "databaseSystemMapping": { "DB2": "SystemB", "DB3": "SystemC" } }</pre>

Properties	Description
schema	The name of the default schema, if not specified in the data source itself. This corresponds to name of your Schema asset. Note You must use the same schema name as the name of the Schema asset that you create when you prepare the physical data layer in Data Catalog.
verbose	Indication whether you want to enable verbose logging. By default this is set to <code>True</code>. If you don't want to use verbose logging, set it to <code>False</code>.
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

Properties	Description
general	This section describes the connection between Collibra Data Lineage and Data Catalog.
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.
url	The URL of the Collibra Data Lineage service instance. Example “url”: “https://techlin-gov.collibra.com” Warning This section applies only to US government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.

Properties	Description
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.
useCollibraSystemName	Indicates whether you want to use the system or server name of a data source to match to the System asset you created when you prepared the Data Catalog physical data layer. The names are case-sensitive. This is useful when you have multiple databases with the same name.
sources	This configuration section contains the required information for a Google BigQuery database.
id	The unique ID of your data source. For example, <code>my_third_data_source</code> .
type	The kind of data source. In this case, the value has to be <code>DatabaseBigQuery</code> .

Properties	Description
projectIDs	The IDs of your Google BigQuery project. You can add multiple projects. For example, ["first-project", "second-project", "third-project"]. Note You have to use the same project ID as the name of the Database asset that you create when you prepare the physical data layer in Data Catalog.
region	The location of your BigQuery data. This is the region that you specified when you create a data set. If the region that you specify here doesn't match the region you specified when you created a data set, then: ◦ The metadata of that data set will not be harvested. ◦ Metadata of the data sets in the region you specify here will be harvested. If you don't specify a region, the region is defaulted to US, meaning that metadata (and lineage) will be harvested only for datasets located in the US region. You can only add one location as value. However, you can create separate BigQuery entries per location in the configuration file. As a result, you create a complete technical lineage with Google BigQuery data from different locations. Note This property is optional.

Properties	Description
auth	The path to a JSON file that contains authentication information. Tip For more information about setting up the authentication, see the Google Big Query user guide.
collibraSystemName	The name of the Google BigQuery system. This is also the name of your System asset in Data Catalog. Specify this property with the same name as the name of the System asset that you create when you prepare the physical data layer in Data Catalog. If you don't prepare the physical data layer, Collibra Data Lineage cannot stitch the data objects in your technical lineage to the assets in Data Catalog. Specify this property with the same name as the name of the System asset that you created when you registered the data source.

Properties	Description
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

2. Save the configuration file.

For complete information on creating custom technical lineage by using the lineage harvester, go to [Working with custom technical lineage](#).

Steps

1. Open the lineage-harvester.conf file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
general	This section describes the connection between Collibra Data Lineage and Data Catalog.

Properties	Description
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.
url	The URL of the Collibra Data Lineage service instance. Example “url”: “https://techlin-gov.collibra.com” Warning This section applies only to US government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.

Properties	Description
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.
useCollibraSystemName	The lineage harvester ignores this property for custom technical lineage. To use the system or server name of your data source to match the System asset in Data Catalog, specify the system data object in .
sources	Contains the required information to retrieve a custom lineage. Use this property to locate the JSON file that defines the custom technical lineage. If you want to create the technical lineage for multiple data sources, create a <code>sources</code> section for each data source.
type	The kind of data source. The value must be <code>ExternalDirectory</code> .
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable. For example, <code>MyCustomLineage</code> .
dirType	The type of external directory. The value is <code>custom-lineage</code> .

Properties	Description
<code>collibraSystemName</code>	The lineage harvester ignores this property for custom technical lineage. To use the system or server name of your data source to match the System asset in Data Catalog, specify the system data object in .
<code>path</code>	The full path to the folder of the custom technical lineage JSON file, for example <code>C:\path\to\custom-lineage\dir.</code> There must be only one JSON file that defines the lineage, and the JSON file must be named lineage.json. You can, however, add other files in the harvested directory and subdirectories and refer to those files from within the JSON file.
<code>deleteRawMetadataAfterProcessing</code>	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

2. Save the configuration file.

Steps

1. Open the lineage-harvester.conf file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
general	This section describes the connection between Collibra Data Lineage and Data Catalog.
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.
url	The URL of the Collibra Data Lineage service instance. Example "url": "https://techlin-gov.collibra.com" Warning This section applies only to US government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.

Properties	Description
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a JDBC data source to match the System asset that you created when you prepared the physical data layer. The names are case-sensitive. Specify one of the following values: <code>false</code> The lineage harvester ignores all system or server names that you specify on the <code>collibraSystemName</code> properties in the <source ID> file. This is the default value. <code>true</code> The lineage harvester reads the system and server names that you specify on the <code>collibraSystemName</code> properties in all sections of the <source ID> file. Note Specify this property with the value of <code>true</code> only when you have multiple databases with the same name.

Properties	Description
sources	This configuration section contains the required information to connect to IBM InfoSphere DataStage. Note Make sure that you have prepared a local folder with the DataStage files for which you want to create a technical lineage.
collibraSystemName (Deprecated)	This property is deprecated. If you specify a value for this property, it is ignored. To override the default Collibra System asset name, use the <code>useCollibraSystemName</code> property in the source id file.
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable. For example, <i>my_datastage</i>.
type	The kind of data source. In this case, the value has to be <i>ExternalDirectory</i>.
dirType	The type of external directory. The value has to be <i>datastage</i>.
path	The full path to the folder where you stored the data source, for example, C:\path\to\config\dir.
mask	The pattern of the file names in the directory. By default, this is <i>*</i>.
recursive	Indication whether you want to use recursive queries. By default, this is set to <i>False</i>. If you want to use recursive query, set it to <i>True</i>.

Properties	Description
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

2. Save the configuration file.

Steps

1. Open the lineage-harvester.conf file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description	Required?
general	This section describes the connection between Collibra lineage and Data Catalog.	Yes

Properties	Description	Required?
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.	Yes
url	The URL of the Collibra Data Lineage service instance. Example “url”: “https://techlin-gov.collibra.com” Warning This section applies only to US government customers.	Yes
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.	Yes
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.	Yes

Properties	Description	Required?
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.	Yes
username	The username that you use to sign in to Collibra.	Yes
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a JDBC data source to match the System asset that you created when you prepared the physical data layer. The names are case-sensitive. Specify one of the following values: false The lineage harvester ignores all system or server names that you specify on the <code>collibraSystemName</code> properties in the <source ID> file. This is the default value. true The lineage harvester reads the system and server names that you specify on the <code>collibraSystemName</code> properties in all sections of the <source ID> file. Note Specify this property with the value of <code>true</code> only when you have multiple databases with the same name.	No

Properties	Description	Required?
sources	This section describes the data sources for which you want to create the technical lineage. You have to create a configuration section for each data source. This configuration section contains the required information of one individual SQL directory with connection type "Folder". Note You can add multiple data sources to the same configuration file.	Yes
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable.	Yes
type	The kind of data source. The value must be <code>dbt</code> .	Yes
collibraSystemName (Deprecated)	This property is deprecated. If you specify a value for this property, it is ignored. To override the default Collibra System asset name, use the <code>useCollibraSystemName</code> property in the source id file .	No

Properties	Description	Required?
tokenName	The name of the service token. It can be any unique meaningful name. When you run the lineage harvester, you will be prompted for a token. Enter the token value for the service token. How to get a service token and token value. - Generate a Service token and ensure that you set the Read-Only permissions for CollibraData Lineage to work properly. - Copy the token value when you save the service token. For details, go to Generating service account tokens in dbt documentation.	Yes
adminUrl	The dbt Cloud Administrative API that Collibra Data Lineage uses to download job artifacts. The default value is <code>https://cloud.getdbt.com/api/v2</code>.	No
adminUrl	The dbt Cloud Administrative API that Collibra Data Lineage uses to download job artifacts. The default value is <code>https://cloud.getdbt.com/api/v2</code>. This property is used if you do not specify the <code>environmentIds</code> property. If you specify both the <code>adminUrl</code> and <code>environmentIds</code> properties, the <code>environmentIds</code> property takes precedence.	No

Properties	Description	Required?
metadataUrl	The dbt Cloud Discovery API. The default value is <code>https://metadata.cloud.getdbt.com/graphql</code>. For details, go to Query the Discovery API in dbt documentation.	No
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.	

2. Save the configuration file.

Steps

1. Open the `lineage-harvester.conf` file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
general	This section describes the connection between Collibra Data Lineage and Data Catalog.
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.
url	The URL of the Collibra Data Lineage service instance. Example “url”: “https://techlin-gov.collibra.com” Warning This section applies only to US government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.

Properties	Description
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a JDBC data source to match the System asset that you created when you prepared the physical data layer. The names are case-sensitive. Specify one of the following values: false The lineage harvester ignores all system or server names that you specify on the <code>collibraSystemName</code> properties in the <source ID> file. This is the default value. true The lineage harvester reads the system and server names that you specify on the <code>collibraSystemName</code> properties in all sections of the <source ID> file. Note Specify this property with the value of <code>true</code> only when you have multiple databases with the same name.

Properties	Description
sources	This configuration section contains the required information to enable the lineage harvester to collect and process Data Integration objects. You can create different Informatica Intelligent Cloud Services <source ID> configuration files for a large data source to avoid errors that might occur when the lineage harvester ingests metadata from one source with a large size. You can then decrease the size of the source by separating the projects to a different source with a different <source ID> configuration file name. Show me the example <pre> "sources" : [{ "type" : "IICS", "id" : "iics_source-1", "loginUrl" : "https://dm- us.informaticaintelligentcloud.com", "username" : "login-iics" "objects" : [{ "path" : "Default/Sales", "type" : "Project" }, { "path" : "My Project/Statistics", "type" : "Project" }] } { "type" : "IICS", "id" : "iics_source-2", "loginUrl" : "https://dm- us.informaticaintelligentcloud.com", "username" : "login-iics" "objects" : [{ "path" : "Finance/Task_Flows", "type" : "Folder" }, { "path" : "Common/Task_Flows/tf_ CalendarDimension", </pre>

Properties	Description
	<pre> "type" : "Taskflow" }] }] </pre> Tip Make sure you have READ permission on all data objects that you want to harvest.
type	The kind of data source. In this case, the value has to be IICS.
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable. For example, <code>my_data_integration</code> .
collibraSystemName (Deprecated)	This property is deprecated. If you specify a value for this property, it is ignored. To override the default Collibra System asset name, use the <code>useCollibraSystemName</code> property in the source id file .
loginURL	The URL of the Informatica Intelligent Cloud Services environment sign-in page. For example: <code>https://dm-us.informaticaintelligentcloud.com</code> .
username	The username you use to sign in to Informatica Intelligent Cloud Services.

Properties	Description
objects	The objects that you want to retrieve. Each object requires a path and a type, for example: Example The following example retrieves the Project, Folder, Taskflow, and Workflow objects. <pre>[{ "path" : "Sales", "type" : "Project" }, { "path" : "Finance/Task_Flows", "type" : "Folder" }, { "path" : "Common/Task_Flows/tf_CalendarDimension", "type" : "Taskflow" }, { "path" : "Common/Linear_Task_Flows/wf_StateProvinceDimension", "type" : "Workflow" }]</pre> Tip For more information about the objects that you can export and the required information, see the Informatica documentation.
path	The full path to the object, for example, C:\path\to\object-dir.
type	The type of the object, for example, Taskflow. IICS scanner's starting point is a Taskflow or Linear Taskflow (Workflow). Therefore the only meaningful types to retrieve are: Taskflow, Workflow, Project and Folder. The types are not case sensitive.

Properties	Description
paramFiles	The full path to the directory in which your parameter files are stored. This is an optional parameter that allows you to harvest parameter files in Informatica Intelligent Cloud Services data sources. Important The hierarchy of the files in the directory must be an exact match of the hierarchy of the files in your file system. Show me how to do this - Create a directory for your parameter files. For this example, let's name the directory <i>my-parameter-files</i>. - In your lineage harvester configuration file, the value of the <code>paramFiles</code> property needs to be the full path to your parameter files directory, for example <code>/full/path/<my-parameter-files>/</code>. - Copy your parameter files to your parameter files directory. Be sure to preserve the full path for each of your parameter files. For example, for parameter file <i>/root/child/child2/paramfile.txt</i>, run the following commands: <code>cd /full/path/<my-parameter-files>/</code> <code>mkdir -p root/child/child2/</code> <code>cp /root/child/child2/paramfile.txt root/child/child2/</code>

Properties	Description
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

2. Save the configuration file.

Steps

1. Open the lineage-harvester.conf file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
general	This section describes the connection between Collibra Data Lineage and Data Catalog.
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.

Properties	Description
url	The URL of the Collibra Data Lineage service instance. Example “url”: “https://techlin-gov.collibra.com” Warning This section applies only to US government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.

Properties	Description
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a JDBC data source to match the System asset that you created when you prepared the physical data layer. The names are case-sensitive. Specify one of the following values: false The lineage harvester ignores all system or server names that you specify on the <code>collibraSystemName</code> properties in the <source ID> file. This is the default value. true The lineage harvester reads the system and server names that you specify on the <code>collibraSystemName</code> properties in all sections of the <source ID> file. Note Specify this property with the value of <code>true</code> only when you have multiple databases with the same name.
sources	This configuration section contains the required information to connect to Informatica PowerCenter. Note Make sure that you have prepared a local folder with the Informatica objects for which you want to create a technical lineage.
collibraSystemName (Deprecated)	This property is deprecated. If you specify a value for this property, it is ignored. To override the default Collibra System asset name, use the <code>useCollibraSystemName</code> property in the source id file.

Properties	Description
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable. For example, <code>my_informatica</code> .
type	The kind of data source. In this case, the value has to be <i>ExternalDirectory</i> .
dirType	The type of external directory. The value has to be <i>infa</i> .
path	The full path to the folder where you stored the data source, for example, <code>C:\path\to\config\dir</code> .
mask	The pattern of the file names in the directory. By default, this is <code>*</code> .
recursive	Indication whether you want to use recursive queries. By default, this is set to <code>False</code>. If you want to use recursive query, set it to <code>True</code>.

Properties	Description
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

2. Save the configuration file.

Steps

1. Open the lineage-harvester.conf file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
general	This section describes the connection information between the lineage harvester and Data Catalog.

Properties	Description
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This applies only for Collibra Cloud for Government customers.
url	The URL of the Collibra Data Lineage service instance. "url": "https://techlin-gov.collibra.com" Warning This applies only for Collibra Cloud for Government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This applies only for Collibra Cloud for Government customers.
catalog	This section contains information that is necessary to connect to Data Catalog.
url	The URL of your Collibra Data Intelligence Cloud environment. Note You can only enter the public URL of your Collibra DGC environment. Other URLs will not be accepted.
username	The username that you use to sign in to Collibra.

Properties	Description
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a data source to match to the System asset in Data Catalog. Collibra Data Lineage uses the system names to match the structure of databases in Looker to assets in Data Catalog. This is useful when you have multiple databases with the same name. By default, the <code>useCollibraSystemName</code> property is set to <code>false</code>. If you want to use it, set it to <code>true</code>. Important ◦ If you set this property to <code>true</code>, the lineage harvester reads the value of the <code>collibraSystemName</code> property in your Looker <source-ID> configuration file. ◦ If you set the <code>useCollibraSystemName</code> property to <code>false</code>, the lineage harvester ignores the <code>collibraSystemName</code> property in the Looker <source-ID> configuration file.
sources	This section contains the Looker connection properties.

Properties	Description
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable. Warning In the <code>sources</code> section of your lineage harvester configuration file, you can only specify one <code>id</code> property per Looker instance. If you have multiple <code>id</code> properties for a single Looker instance, ingestion will fail. If you have multiple <code>id</code> properties in the configuration file, it means you intend to ingest from multiple unique Looker instances. Warning If you are switching between the lineage harvester and Edge, the value of this property must exactly match the value of the Source ID field in your Edge capacity.
type	The kind of data source. In this case, the value has to be <i>Looker</i>.

Properties	Description
lookerUrl	The URL to your Looker API. Tip There are two ways to find the Looker API URL: ◦ In the API Host URL field in the Looker Admin menu. If this field is empty, you can use the default Looker API URL which you can find in the interactive API documentation. ◦ In the interactive API documentation URL. It is the part of the URL before <code>/api-docs/</code>. Note Looker 3.1 APIs are deprecated; however, the API3 credentials for authorization and access control remain valid.
clientId	The username you use to access the Looker API.
domainId	The unique ID of the domain in Collibra Data Intelligence Cloud in which you want to ingest the Looker assets. This is the default domain. If you want to ingest the contents of specific Looker Folders into specific domains in Collibra, you specify the domain reference IDs in the filters section of the Looker <source ID> configuration file.

Properties	Description
pagingLimit	Optional property for customizing the Looker API pagination settings. The default value of 50 is sufficient in most cases; however, you can decrease it to help mitigate node limit errors, or increase it to speed up API calls. Example <code>"pagingLimit": 10</code>
concurrencyLevel	This optional property is intended to help if you are experiencing HTTP 401 Unauthorized errors due to too many concurrent HTTP calls, using the same token. It allows you to specify the internal sizing, meaning the amount of tasks that can be executed at the same time. The default value is "15", meaning as many as 15 HTTP requests can take place in parallel. Consider reducing the value if you are experiencing HTTP 401 Unauthorized errors. Setting the value to "1" effectively disables the concurrency level, so that HTTP requests will be run in a synchronous manner, instead of in parallel. Example <code>"concurrencyLevel": 5</code>

Properties	Description
connectionTimeoutSeconds	This optional property is intended to help avoid timeout errors, when the lineage harvester attempts to connect to your Looker instance. The default value is "30", meaning a timeout error is thrown if a connection is not established within 30 seconds. If timeout errors persist, try adding this property to you lineage harvester configuration file and setting the value to 60 or 90. Example <pre>"connectionTimeoutSeconds": 60</pre>
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

2. Save the configuration file.

Steps

1. Open the lineage-harvester.conf file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
general	This section describes the connection between Collibra Data Lineage and Data Catalog.
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.
url	The URL of the Collibra Data Lineage service instance. Example "url": "https://techlin-gov.collibra.com" Warning This section applies only to US government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.

Properties	Description
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a JDBC data source to match the System asset that you created when you prepared the physical data layer. The names are case-sensitive. Specify one of the following values: <code>false</code> The lineage harvester ignores all system or server names that you specify on the <code>collibraSystemName</code> properties in the <source ID> file. This is the default value. <code>true</code> The lineage harvester reads the system and server names that you specify on the <code>collibraSystemName</code> properties in all sections of the <source ID> file. Note Specify this property with the value of <code>true</code> only when you have multiple databases with the same name.

Properties	Description
sources	This section contains the required information for Matillion. Tip When you create a new project in Matillion, you define in which group you want to create the project, the project name and the environment name. This information is needed to enable the lineage harvester to access Matillion and scan your metadata. Important Currently, you can only create a technical lineage for Snowflake and Redshift projects in Matillion.
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable. For example, <code>my_matillion_data_integration</code>.
type	The kind of data source. In this case, the value has to be <code>Matillion</code>.
url	The URL of your Matillion environment. For example, <code>https://<domain name></code> or <code>https://<IP address></code>.
groupName	The name of your group in Matillion.
projectName	The name of your project in Matillion. You can only add the name of one project. If you want to create a technical lineage for other projects within the same group, create a new section in the lineage harvester configuration file.

Properties	Description
environmentName	The name of your environment in Matillion. You can only add the name of one environment. If you want to create a technical lineage for other environments within the same project, create a new section in the lineage harvester configuration file.
startTimestamp	The timestamp of tasks in Matillion. You can use this parameter to limit the amount of metadata that the lineage harvester scans. Specify this property with a UNIX timestamp in milliseconds. If this property remains empty or is deleted from the configuration file, all accessible tasks are scanned. Matillion provides seven days of history by default and automatically removes entries older than seven days.
collibraSystemName (Deprecated)	This property is deprecated. If you specify a value for this property, it is ignored. To override the default Collibra System asset name, use the <code>useCollibraSystemName</code> property in the source id file.
auth	The section contains the authentication details for signing in to Matillion.
type	The authentication method you want to use to sign in to Matillion. The value must be either: ◦ <code>Basic</code>, for username and password authentication. ◦ <code>Token</code>, for token-based authentication. Important These values are case-sensitive.

Properties	Description
username	The username that you use to sign in to Matillion. Important This property is only required if you are using the username and password authentication method. If you are using token-based authentication, do not include this property.
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

2. Save the configuration file.

Steps

1. Open the lineage-harvester.conf file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
general	This section describes the connection information between the lineage harvester and Data Catalog.
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This applies only for Collibra Cloud for Government customers.
url	The URL of the Collibra Data Lineage service instance. "url": "https://techlin-gov.collibra.com" Warning This applies only for Collibra Cloud for Government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This applies only for Collibra Cloud for Government customers.
catalog	This section contains information that is necessary to connect to Data Catalog.
url	The URL of your Collibra Data Intelligence Cloud environment. Note You can only enter the public URL of your Collibra DGC environment. Other URLs will not be accepted.
username	The username that you use to sign in to Collibra.

Properties	Description
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a data source to match to the System asset in Data Catalog during automatic stitching. This is useful when you have multiple databases with the same name. By default, the <code>useCollibraSystemName</code> property is set to <code>false</code>. If you want to use it, set it to <code>true</code>. Important ◦ If you set this property to <code>true</code>, the lineage harvester reads the value of the <code>collibraSystemName</code> property in your MicroStrategy <source ID> configuration file. ◦ If you set the <code>useCollibraSystemName</code> property to <code>false</code>, the lineage harvester ignores the <code>collibraSystemName</code> property in the <code><source-ID></code> configuration file.
sources	This section contains the MicroStrategy connection properties.

Properties	Description
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable. For example, <code>my_microstrategy</code>. Warning In the <code>sources</code> section of your lineage harvester configuration file, you can only specify one <code>id</code> property per MicroStrategy Intelligence Server. If you have multiple <code>id</code> properties for a single MicroStrategy Intelligence Server, ingestion will fail. If you have multiple <code>id</code> properties in the configuration file, it means you intend to ingest from multiple unique MicroStrategy Intelligence Servers. Warning If you are switching between the lineage harvester and Edge, the value of this property must exactly match the value of the Source ID field in your Edge capacity.
type	The kind of data source. In this case, the value has to be <code>MSTR_V2</code> .
url	The URL of your MicroStrategy account.
username	The username that you use to sign in to MicroStrategy.

Properties	Description
microStrategyLibraryUrl	If you are using a custom URL to connect to the MicroStrategy Library Server, use this property to specify the custom library URL. Important You only need to specify the URL if both of the following are true: - You are connecting to a proxy server. - You are not using the default, hardcoded URL to the MicroStrategy Library Server. Example If the URL to your MicroStrategy Library is <i>https://collibra.microstrategy.com/MicroStrategyLibrary/api</i>, you don't need to use this property, as that is the default, hardcoded URL. However, if the URL is something like <i>https://collibra.microstrategy.com/MicroStrategyLibraryProd/api</i>, then use this property and configure it as follows: <pre>"microStrategyLibraryUrl": "MicroStrategyLibraryProd"</pre>
maxParallelRequests	This optional property allows you to specify the internal sizing, meaning the amount of tasks that can be executed at the same time. The default value is "1", which means that HTTP requests are run in a synchronous manner, instead of in parallel. As value of "5", for example, means that as many as 5 HTTP requests can take place in parallel. A lower value reduces the chances of experiencing HTTP 401 Unauthorized errors.

Properties	Description
requestTimeoutMs	This optional property allows you to specify the maximum time, in milliseconds (ms), that the MicroStrategy Intelligence Server will wait for a request from the lineage harvester, before closing the connection. Tip A "connection timeout" refers to the amount of time that the lineage harvester will wait for a response from MicroStrategy. A "request timeout" is the converse of a connection timeout. The default value is "30000", or 30 seconds. A higher value reduces the chances of experiencing HTTP 408 Request Timeout errors.
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

Properties	Description
appUrlSuffix	This optional property ensures that the correct URL to data objects in MicroStrategy is included on the asset pages of corresponding MicroStrategy assets. The required value depends on which platform you run MicroStrategy: ◦ For J2EE, use: "appUrlSuffix": "MicroStrategy/servlet/mstrWeb" ◦ For .NET, use: "appUrlSuffix": "MicroStrategy/asp/Main.aspx"

2. Save the configuration file.

Steps

1. Open the lineage-harvester.conf file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
general	This section describes the connection between Colibra Data Lineage and Data Catalog.
techlin	This section contains information that is necessary to connect to the Colibra Data Lineage service instance. Warning This section applies only to US government customers.

Properties	Description
url	The URL of the Collibra Data Lineage service instance. Example “url”: “https://techlin-gov.collibra.com” Warning This section applies only to US government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.

Properties	Description
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a JDBC data source to match the System asset that you created when you prepared the physical data layer. The names are case-sensitive. Specify one of the following values: false The lineage harvester ignores all system or server names that you specify on the <code>collibraSystemName</code> properties in the configuration file. This is the default value. true The lineage harvester reads the system and server names that you specify on the <code>collibraSystemName</code> properties in all sections of the configuration file. Only specify this value when you have multiple databases with the same name. Note For SQL data sources, if this property is: ◦ <code>false</code>, system or server names in table references in analyzed SQL code are ignored. This means that a table that exists in two different systems or servers is identified (either correctly or incorrectly) as a single data object, with a single asset name. ◦ <code>true</code>, system or server names in table references are considered to be represented by different System assets in Data Catalog. The value of the <code>collibraSystemName</code> property is used as the default system or server name.

Properties	Description
sources	This section describes the data sources for which you want to create the technical lineage. You have to create a configuration section for each data source. This configuration section contains the required information of one individual SQL directory with connection type "Folder". Note You can add multiple data sources to the same configuration file.
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable.
type	The kind of data source. In this case, the value has to be <code>SqlDirectory</code> .
path	The full path to the folder where you added SQL files , for example, <code>C:\path\to\config\dir</code> .
mask	The pattern of the file names in the directory. By default, this is <code>*</code> .
recursive	Indication of the files you want to harvest: ◦ <code>false</code> (default): Only harvest the files in directly under the folder in the SQL directory path. ◦ <code>true</code>: Harvest all files under the folder in the SQL directory path and subdirectories.
dialect	The dialect of the database. For example, <i>oracle</i>. The value you put for this property has to match the dialect you provide with in the directory with your SQL files.

Properties	Description
database	The name of your database, which is the name of your Database asset. Note - You have to use the same database name as the name of the Database asset that you create when you prepare the physical data layer in Data Catalog. The names are case-sensitive. - The database and schema names in the SQL statements in your SQL files take precedence over the values that you provide for the <code>database</code> and <code>schema</code> properties in the lineage harvester configuration file. If your SQL statements contain database and schema names, Collibra Data Lineage uses them for stitching. If your SQL statements do not contain database and schema names, Collibra Data Lineage uses the values of the <code>database</code> and <code>schema</code> properties in the configuration file for stitching.. For more information, go to Prepare an SQL directory and Automatic stitching for technical lineage.
collibraSystemName	The name of the data source's system or server. This is also the name of your System asset in Data Catalog. Specify this property with the same name as the name of the System asset that you create when you prepare the physical data layer in Data Catalog. If you don't prepare the physical data layer, Collibra Data Lineage cannot stitch the data objects in your technical lineage to the assets in Data Catalog. Specify this property with the same name as the name of the System asset that you created when you registered the data source.

Properties	Description
databaseSystemMapping	This optional property allows you to map databases to their rightful systems, to obtain stitching. This resolves missing stitching, which occurs when Collibra Data Lineage associates multiple databases with the default system name that you provide in the <code>collibraSystemName</code> property. More information and example configuration Let's say you use the <code>collibraSystemName</code> property to specify the system name <code>SystemA</code>. This system name will be associated with all of the databases mentioned in your SQL statements in your SQL files. Now let's say that you have a SQL statement that selects data from a database <code>DB2</code> (which is in <code>SystemB</code>) and inserts the data in database <code>DB3</code> (which is part of <code>SystemC</code>). Both of these databases will be associated with <code>SystemA</code>, and both will appear under <code>SystemA</code> in the Browse tab pane. Therefore, stitching fails and duplicate databases might appear. To resolve this issue and obtain stitching, you can use this property to map <code>DB2</code> to its rightful <code>SystemB</code>, and <code>DB3</code> to its rightful <code>SystemC</code>. <pre> { "id": "sql_dir", "type": "SqlDirectory", "path": "/tmp/sql_dir", "dialect": "bigquery", "collibraSystemName": "SystemA", "database": "DB", "schema": "SCH", "databaseSystemMapping": { "DB2": "SystemB", "DB3": "SystemC" } }</pre>

Properties	Description
schema	The name of the default schema, if not specified in the data source itself. This corresponds to name of your Schema asset. Note You must use the same schema name as the name of the Schema asset that you create when you prepare the physical data layer in Data Catalog.
verbose	Indication whether you want to enable verbose logging. By default this is set to <code>True</code>. If you don't want to use verbose logging, set it to <code>False</code>.
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

Properties	Description
general	This section describes the connection between Collibra Data Lineage and Data Catalog.
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.
url	The URL of the Collibra Data Lineage service instance. Example "url": "https://techlin-gov.collibra.com" Warning This section applies only to US government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.

Properties	Description
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a JDBC data source to match the System asset that you created when you prepared the physical data layer . The names are case-sensitive. This is useful if you have multiple databases with the same name.
sources	This configuration section contains the required information for an Oracle database. Tip We recommend the <code>"type": "DatabaseOracle"</code> configuration described in this section, because it allows you to specify the Oracle database name and preserve stitching in cases where the database name is not the same as the SID or service name. You can, however, still use the legacy <code>"type": "Database"</code> configuration to ingest Oracle databases.
id	The unique ID of your Oracle database. For example, <code>my_oracle_db</code> .

Properties	Description
type	The kind of data source. In this case, the value has to be <code>DatabaseOracle</code> .
hostname	The name of your database host.
username	The username that you use to sign in to your Oracle database.
port	The port number.

Properties	Description
sids	One or more system identifiers (SID). An SID is a unique name for an Oracle database instance on a specific host. You can use this property in conjunction with the <code>databaseNames</code> property, to preserve stitching. Important You must specify either one or more SIDs via this property, or one or more service names via the <code>serviceNames</code> property. You cannot include both properties in the configuration file. Show me examples of how to configure the <code>sids</code> property, with and without the <code>databaseNames</code> property Example 1: You include the <code>sids</code> property, but not the <code>databaseNames</code> property: <pre> { "id": "oracle1", "type": "DatabaseOracle", "hostname": "host_url", "username": "user1", "collibraSystemName": "automation_csn", "port": 1521, "sids": ["sid1", "sid2"] }</pre> Result: The database names in the technical lineage will be "sid1" and "sid2". If these don't match with your Database assets in Collibra, then stitching won't work. Example 2: You include the <code>sids</code> property and the <code>databaseNames</code> property:

Properties	Description
	<pre>{ "id": "oracle2", "type": "DatabaseOracle", "hostname": "host_url", "username": "user1", "collibraSystemName": "automation_csn", "port": 1521, "sids": ["sid1", "sid2"], "databaseNames": ["db1", "db2"] }</pre> Result: The SID "sid1" corresponds to the Database asset name "db1" in Collibra, therefore stitching is preserved. The same is true for SID "sid2" and Database asset name "db2".

Properties	Description
serviceNames	One or more service names. A service name is the TNS alias that you give when you remotely connect to your database. You can use this property in conjunction with the databaseNames property, to preserve stitching. Important You must specify either one or more service names via this property, or one or more SIDs via the <code>sids</code> property. You cannot include both properties in the configuration file. Show me examples of how to configure the serviceNames property, with and without the databaseNames property Example 1: You include the <code>serviceNames</code> property, but not the <code>databaseNames</code> property: <pre>{ "id": "oracle3", "type": "DatabaseOracle", "hostname": "host_url", "username": "user1", "collibraSystemName": "automation_csn", "port": 1521, "serviceNames": ["sn1", "sn2"] }</pre> Result: The database names in the technical lineage will be "sn1" and "sn2". If these don't match with your Database assets in Collibra, then stitching won't work. Example 2: You include the <code>serviceNames</code> property and the <code>databaseNames</code> property:

Properties	Description
	<pre>{ "id": "oracle4", "type": "DatabaseOracle", "hostname": "host_url", "username": "user1", "collibraSystemName": "automation_csn", "port": 1521, "serviceNames": ["sn1", "sn2"], "databaseNames": ["db1", "db2"] }</pre> Result: The service name "sn1" corresponds to the Database asset name "db1" in Collibra, therefore stitching is preserved. The same is true for service name "sn2" and Database asset name "db2".

Properties	Description
databaseNames	The names of one or more Oracle databases. You can use this optional property in conjunction with the <code>sids</code> or <code>serviceNames</code> property, to preserve stitching. The value you specify has to match your Database asset (or assets) in Collibra. Enter the Oracle database names between double quotes (") and put everything between square brackets. If you want to include more than one database, separate them by a comma. For example, ["MyFirstDatabase", "MySecondDatabase"]. ◦ If you use this property, the database names that you specify have to correlate with the databases that you specify in the <code>sids</code> or <code>serviceNames</code> property. ◦ If you don't use this property, the database name in the technical lineage will be the value that you put for the <code>sids</code> or <code>serviceNames</code> property. Tip For examples of how to configure this property, see the <code>sids</code> or <code>serviceNames</code> property descriptions and examples.

Properties	Description
collibraSystemName	The name of the data source's system or server. This is also the name of your System asset in Data Catalog. Specify this property with the same name as the name of the System asset that you create when you prepare the physical data layer in Data Catalog. If you don't prepare the physical data layer, Collibra Data Lineage cannot stitch the data objects in your technical lineage to the assets in Data Catalog. Specify this property with the same name as the name of the System asset that you created when you registered the data source. If the <code>useCollibraSystemName</code> property is: ◦ <code>false</code> (default), system or server names in table references in analyzed SQL code are ignored. This means that a table that exists in two different systems or servers is identified (either correctly or incorrectly) as a single data object, with a single asset name. ◦ <code>true</code>, system or server names in table references are considered to be represented by different System assets in Data Catalog. The value of the <code>collibraSystemName</code> field is used as the default system or server name.

Properties	Description
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

2. Save the configuration file.

Steps

1. Open the lineage-harvester.conf file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
general	This section describes the necessary connection information.
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This applies only for Collibra Cloud for Government customers.

Properties	Description
url	The URL of the Collibra Data Lineage service instance. "url": "https://techlin-gov.collibra.com" Warning This applies only for Collibra Cloud for Government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This applies only for Collibra Cloud for Government customers.
catalog	This section contains information that is necessary to connect to Data Catalog.
url	The URL of your Collibra environment. Note You can only enter the public URL of your Collibra DGC environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.

Properties	Description
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a data source to match to the System asset in Data Catalog during automatic stitching. This is useful when you have multiple databases with the same name. By default, the <code>useCollibraSystemName</code> property is set to <code>false</code>. If you want to use it, set it to <code>true</code>. Important - If you set this property to <code>true</code>, the lineage harvester reads the value of the <code>collibraSystemName</code> property in your Power BI <source ID> configuration file. - If you set the <code>useCollibraSystemName</code> property to <code>false</code>, the lineage harvester ignores the <code>collibraSystemName</code> property in the Power BI <source-ID> configuration file.
sources	This section describes the data sources for which you want to create the technical lineage. You have to create a configuration section for each data source. Note You can add multiple data sources to the same configuration file, but you can't have multiple <code>sources</code> sections that refer to the same tenant.

Properties	Description
scope	Optional property that is intended only for customers with a different scope, such as Chinese tenants. Example “scope” : “https://analysis.chinacloudapi.cn/powerbi/api/default” Important If you are a US government or national cloud Power BI customer, you must include and specify values for both this property and the <code>apiUrl</code> property. For complete information, consult Microsoft's documentation on Power BI for US government customers.
apiUrl	The API URL of your Power BI service. The default value is <code>https://api.powerbi.com</code>. Important This property is only relevant for US government or national cloud Power BI customers, in which case you must include and specify values for both this property and the <code>scope</code> property. For complete information, consult Microsoft's documentation on Power BI for US government customers.
type	The kind of data source. In this case, the value has to be <i>PowerBI</i>.

Properties	Description
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable. For example, <code>my_power-bi</code>. Warning In the <code>sources</code> section of your lineage harvester configuration file, you can only specify one <code>id</code> property per Power BI service. If you have multiple <code>id</code> properties for a single Power BI service, ingestion will fail. If you have multiple <code>id</code> properties in the configuration file, it means you intend to ingest from multiple unique Power BI services. Warning If you are switching between the lineage harvester and Edge, the value of this property must exactly match the value of the Source ID field in your Edge capacity.
tenantDomain	The Power BI tenant domain is the domain associated with the Microsoft Azure tenant. It is either a default domain or a custom domain. You can specify this property with one of the following: ◦ The appropriate part of the URL, for example <code>collibrapowerbi.onmicrosoft.com</code>. Do not include the <code>http://</code> part of the URL. ◦ The tenant ID, for example <code>e**b****-****-****-****-1b**d****4663</code>. Tip Usually, you can find a list of Power BI tenant or server domains in your Azure Active Directory or in the upper-right menu.

Properties	Description
loginFlow	This section describes the authentication information for accessing your Power BI metadata. The lineage harvester supports two authentication methods: service principal, and username and password. For complete information on your authentication options, see Authentication.
type	This depends on the authentication method you use. ◦ Service principle: The value should be <code>ServicePrincipal</code>. ◦ Username and password: The value should be <code>ResourceOwnerPasswordCredentials</code>.
applicationId	The unique ID of the Microsoft Azure Application (client) ID .
username	The email address of your Azure Active Directory user. Tip This property only applies if you are using the username and password authentication method.
domainId	The reference ID of the domain in Collibra in which you want to ingest Power BI metadata.
useHttp1	Optional property to use HTTP/1.1 streams, in case file-size limitations are resulting in timeout errors when using the default HTTP/2 streams.

Properties	Description
daxParserEnabled (Beta)	Note This feature is not available on Collibra Cloud for Government. Optional property for enabling DAX parsing via Collibra AI. This feature: ◦ Allows you to create column-level lineage that includes your calculated columns and measures in Power BI. ◦ Enables stitching between calculated columns in the technical lineage and the corresponding Power BI Column assets in Data Catalog. The default value is <code>false</code>. To enable DAX parsing, set the value to <code>"daxParserEnabled": true</code>. For complete information on DAX parsing, go to DAX parsing via Collibra AI.
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

2. Save the configuration file.

Steps

1. Open the lineage-harvester.conf file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
general	This section describes the connection between Collibra Data Lineage and Data Catalog.
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.
url	The URL of the Collibra Data Lineage service instance. Example "url": "https://techlin-gov.collibra.com" Warning This section applies only to US government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.

Properties	Description
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a JDBC data source to match the System asset that you created when you prepared the physical data layer. The names are case-sensitive. Specify one of the following values: <code>false</code> The lineage harvester ignores all system or server names that you specify on the <code>collibraSystemName</code> properties in the <source ID> file. This is the default value. <code>true</code> The lineage harvester reads the system and server names that you specify on the <code>collibraSystemName</code> properties in all sections of the <source ID> file. Note Specify this property with the value of <code>true</code> only when you have multiple databases with the same name.

Properties	Description
sources	This configuration section contains the required information to connect to SQL Server Integration Services (SSIS). Note Make sure that you have prepared a local folder with the SSIS files for which you want to create a technical lineage.
collibraSystemName (Deprecated)	This property is deprecated. If you specify a value for this property, it is ignored. To override the default Collibra System asset name, use the <code>useCollibraSystemName</code> property in the source id file.
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable. For example, <i>my_ssis</i>.
type	The kind of data source. In this case, the value has to be <i>ExternalDirectory</i>.
dirType	The type of external directory. The value has to be <i>ssis</i>.
path	The full path to the folder where you stored the data source, for example, <code>C:\path\to\config\dir</code>.
mask	The pattern of the file names in the directory. By default, this is <code>*</code>.
recursive	Indication whether you want to use recursive queries. By default, this is set to <code>False</code>. If you want to use recursive query, set it to <code>True</code>.

Properties	Description
<code>deleteRawMetadataAfterProcessing</code>	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

2. Save the configuration file.

Steps

1. Open the `lineage-harvester.conf` file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
<code>general</code>	This section describes the connection between Collibra Data Lineage and Data Catalog.

Properties	Description
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.
url	The URL of the Collibra Data Lineage service instance. Example “url”: “https://techlin-gov.collibra.com” Warning This section applies only to US government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.

Properties	Description
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.

Properties	Description
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a JDBC data source to match the System asset that you created when you prepared the physical data layer. The names are case-sensitive. Specify one of the following values: false The lineage harvester ignores all system or server names that you specify on the <code>collibraSystemName</code> properties in the configuration file. This is the default value. true The lineage harvester reads the system and server names that you specify on the <code>collibraSystemName</code> properties in all sections of the configuration file. Only specify this value when you have multiple databases with the same name. Note For SQL data sources, if this property is: ◦ <code>false</code>, system or server names in table references in analyzed SQL code are ignored. This means that a table that exists in two different systems or servers is identified (either correctly or incorrectly) as a single data object, with a single asset name. ◦ <code>true</code>, system or server names in table references are considered to be represented by different System assets in Data Catalog. The value of the <code>collibraSystemName</code> property is used as the default system or server name.

Properties	Description
sources	This section describes the data sources for which you want to create the technical lineage. You have to create a configuration section for each data source. This configuration section contains the required information of one individual SQL directory with connection type "Folder". Note You can add multiple data sources to the same configuration file.
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable.
type	The kind of data source. In this case, the value has to be <code>SqlDirectory</code> .
path	The full path to the folder where you added SQL files , for example, <code>C:\path\to\config\dir</code> .
mask	The pattern of the file names in the directory. By default, this is <code>*</code> .
recursive	Indication of the files you want to harvest: ◦ <code>false</code> (default): Only harvest the files in directly under the folder in the SQL directory path. ◦ <code>true</code>: Harvest all files under the folder in the SQL directory path and subdirectories.
dialect	The dialect of the database. For example, <i>snowflake</i>. The value your put for this property has to match the dialect you provide with in the directory with your SQL files.

Properties	Description
database	The name of your database, which is the name of your Database asset. Note ◦ You have to use the same database name as the name of the Database asset that you create when you prepare the physical data layer in Data Catalog. The names are case-sensitive. ◦ The database and schema names in the SQL statements in your SQL files take precedence over the values that you provide for the <code>database</code> and <code>schema</code> properties in the lineage harvester configuration file. If your SQL statements contain database and schema names, Collibra Data Lineage uses them for stitching. If your SQL statements do not contain database and schema names, Collibra Data Lineage uses the values of the <code>database</code> and <code>schema</code> properties in the configuration file for stitching.. For more information, go to Prepare an SQL directory and Automatic stitching for technical lineage.
collibraSystemName	The name of the data source's system or server. This is also the name of your System asset in Data Catalog. Specify this property with the same name as the name of the System asset that you create when you prepare the physical data layer in Data Catalog. If you don't prepare the physical data layer, Collibra Data Lineage cannot stitch the data objects in your technical lineage to the assets in Data Catalog. Specify this property with the same name as the name of the System asset that you created when you registered the data source.

Properties	Description
databaseSystemMapping	This optional property allows you to map databases to their rightful systems, to obtain stitching. This resolves missing stitching, which occurs when Collibra Data Lineage associates multiple databases with the default system name that you provide in the <code>collibraSystemName</code> property. More information and example configuration Let's say you use the <code>collibraSystemName</code> property to specify the system name <code>SystemA</code>. This system name will be associated with all of the databases mentioned in your SQL statements in your SQL files. Now let's say that you have a SQL statement that selects data from a database <code>DB2</code> (which is in <code>SystemB</code>) and inserts the data in database <code>DB3</code> (which is part of <code>SystemC</code>). Both of these databases will be associated with <code>SystemA</code>, and both will appear under <code>SystemA</code> in the Browse tab pane. Therefore, stitching fails and duplicate databases might appear. To resolve this issue and obtain stitching, you can use this property to map <code>DB2</code> to its rightful <code>SystemB</code>, and <code>DB3</code> to its rightful <code>SystemC</code>. <pre> { "id": "sql_dir", "type": "SqlDirectory", "path": "/tmp/sql_dir", "dialect": "bigquery", "collibraSystemName": "SystemA", "database": "DB", "schema": "SCH", "databaseSystemMapping": { "DB2": "SystemB", "DB3": "SystemC" } }</pre>

Properties	Description
schema	The name of the default schema, if not specified in the data source itself. This corresponds to name of your Schema asset. Note You must use the same schema name as the name of the Schema asset that you create when you prepare the physical data layer in Data Catalog.
verbose	Indication whether you want to enable verbose logging. By default this is set to <code>True</code>. If you don't want to use verbose logging, set it to <code>False</code>.
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

Properties	Description
general	This section describes the connection between Collibra Data Lineage and Data Catalog.
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.
url	The URL of the Collibra Data Lineage service instance. Example "url": "https://techlin-gov.collibra.com" Warning This section applies only to US government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.

Properties	Description
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.
useCollibraSystemName	Indicates whether or not you want to use the system or server name of a JDBC data source to match the System asset that you created when you prepared the physical data layer . The names are case-sensitive. This is useful if you have multiple databases with the same name.
sources	This section contains the Snowflake connection properties. If you want to create the technical lineage for multiple data sources, create a <code>sources</code> section for each data source.
id	The unique ID that identifies the data source on a Collibra Data Lineage service instance, for example, <code>my_snowflake_2</code> .
type	The type of data source. The value must be <code>DatabaseSnowflake</code> .

Properties	Description
mode	The Snowflake ingestion methods that Collibra Data Lineage uses to ingest metadata from Snowflake data sources. Specify one of the following values: SQL The SQL Snowflake ingestion mode. Collibra Data Lineage creates a column-level technical lineage based on SQL statements. This is the default value. SQL-API The SQL-API Snowflake ingestion mode. Collibra Data Lineage creates a column-level technical lineage based on Snowflake schemas and the access history. For more information, go to Technical lineage for Snowflake ingestion methods.
collibraSystemName	Use this property with the <code>useCollibraSystemName</code> property in the configuration file to override the default Collibra System asset name for this data source. Specify this property with the same name as the name of the System asset that you create when you prepare the physical data layer in Data Catalog. If you don't prepare the physical data layer, Collibra Data Lineage cannot stitch the data objects in your technical lineage to the assets in Data Catalog. Note The <code>collibraSystemName</code> property in the source ID file takes precedence over the <code>collibraSystemName</code> property in the lineage harvester configuration file. If you specify the <code>collibraSystemName</code> property in the source ID file, you can omit the property in the configuration file.

Properties	Description
auth	This section indicates the authentication details to connect to the Snowflake database. Note The <code>username</code> and <code>auth</code> properties are mutually exclusive.
type	The authentication method. Specify one of the following values. The values are case-sensitive. Basic The username and password authentication method. Specify the <code>auth.username</code> property if you use this authentication method. KeyPair The key pair authentication method. Specify the <code>auth.username</code>, <code>auth.pathToPrivateKey</code>, and <code>auth.usePassword</code> properties if you use this authentication method.
username	The user name that you use to connect to the Snowflake database. This property is required for both the username and password authentication method and the key pair authentication method.
pathToPrivateKey	The path to your private key file. This property is required if you use the key pair authentication method. Ensure that the private key matches the public key; otherwise, an error occurs indicating that the JWT token is invalid. For more information about the error, go to Snowflake JDBC driver error at login: net.snowflake.client.jdbc.SnowflakeSQLException: JWT token is invalid in Collibra Support Portal.

Properties	Description
usePassword	The private key file password. This property is required if you use the key pair authentication method. Specify one of the following values: true The password is required. false The password is not required. This is the default value.
username	The username that you use to sign in to your Snowflake data source. Note This property is deprecated. Use the auth property instead. The property and the auth property are mutually exclusive.
hostname	The URL that you use to access Snowflake web console. When you enter the URL, do not include <code>https://</code> or the trailing slash (/). For example, specify <code><accountName>.snowflakecomputing.com</code>.
databaseNames	An array of database names. Ensure that the database names you specify match the Database asset names that you created when you prepared the physical data layer in Data Catalog. Enter the database names of your data source between double quotes (") and put everything between square brackets ([]). If you want to include more than one database, separate them by a comma, for example, <code>["MyFirstSnowflakeDatabase", "MySecondSnowflakeDatabase"]</code>.

Properties	Description
extraDatabaseDefinitions	An array of database names. Collibra Data Lineage collects metadata from the specified databases, but excludes these databases from the technical lineage that is created. This property is useful for stitching across databases. You can specify cross-referenced databases to ensure correct lineage across all databases that Collibra Data Lineage processes to create the technical lineage. This property is optional. To specify this property, enter the database names between double quotes (") and put everything between square brackets ([]). If you want to include more than one database, separate them by a comma, for example, ["MyFirstSnowflakeExternalDatabase", "MySecondSnowflakeExternalDatabase"].
schemaNames	An array of schema names of your data sources. This property takes effect only when you use the SQL-API Snowflake ingestion mode. You can use this property as a filter to include lineage for objects only in the specified schemas. Ensure that the schema names you specify match the Schema asset names that you created when you registered the data source in Data Catalog Enter the schema names between double quotes (") and put everything between square brackets ([]). If you want to include more than one schema, separate them by a comma, for example, ["MyFirstSnowflakeSchema", "MySecondSnowflakeSchema"].
warehouse	The name of your virtual warehouse. This property is optional.

Properties	Description
days	The number of days of the user access history that Collibra Data Lineage collects and processes. For example, if you set the value to 20, Collibra Data Lineage collects the last 20 days of user access history. You can use this property to limit reading from the ACCESS_HISTORY table. This property is optional and takes effect only when you use the SQL-API Snowflake ingestion mode. Specify a value in the range of 1 - 366. If you do not enter a value, all user access history is collected by default.
customConnectionProperties	An option to enable the lineage harvester to read additional connection parameters. This parameter is only required in very specific situations. If you don't need it, you can remove it from the configuration file. Example If you get an OSCP scan error, you can turn OSCP checking off by using the following value: <code>insecureMode=true</code>.

Properties	Description
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

2. Save the configuration file.

Steps

1. Open the lineage-harvester.conf file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
general	This section describes the connection information between the lineage harvester and Data Catalog.
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This applies only for Collibra Cloud for Government customers.

Properties	Description
url	The URL of the Collibra Data Lineage service instance. "url": "https://techlin-gov.collibra.com" Warning This applies only for Collibra Cloud for Government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This applies only for Collibra Cloud for Government customers.
catalog	This section contains information that is necessary to connect to Data Catalog.
url	The URL of your Collibra Data Intelligence Cloud environment. Note You can only enter the public URL of your Collibra Data Intelligence Cloud environment. Other URLs will not be accepted.
username	The username that you use to sign in to Collibra.

Properties	Description
useCollibraSystemName	Indication whether you want to use the system or server name of a data source to match to the System asset you created when you prepared the physical data layer. This is useful when you have multiple databases with the same name. By default, the <code>useCollibraSystemName</code> property is set to <code>false</code>. If you want to use it, set it to <code>true</code>. Important ◦ If you set this property to <code>true</code>, the lineage harvester reads the value of the <code>collibraSystemName</code> property in your SSRS-PBRS <source-ID> configuration file. ◦ If you set the <code>useCollibraSystemName</code> property to <code>false</code>, the lineage harvester ignores the <code>collibraSystemName</code> property in the <code><source-ID></code> configuration file.
sources	This section contains the SSRS connection properties.

Properties	Description
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable. Warning In the <code>sources</code> section of your lineage harvester configuration file, you can only specify one <code>id</code> property per SQL Server Reporting Service (SSRS) or Power BI Report Server (PBRS). If you have multiple <code>id</code> properties for a single SSRS or PBRS, ingestion will fail. If you have multiple <code>id</code> properties in the configuration file, it means you intend to ingest from multiple unique SSRS or PBRS. Warning If you are switching between the lineage harvester and Edge, the value of this property must exactly match the value of the Source ID field in your Edge capacity.
type	The kind of data source. In this case, the value has to be <i>SSRS</i> or <i>PBIRS</i>. Note There is no difference between type SSRS or PBIRS.
url	The URL to the server's web portal. By default, the URL is <code>http://<computer-name>/reports</code>. For example, "<code>http://1.23.45.678/PowerBIReports</code>".
username	The username you use to sign in to the web portal. Tip If you use NTLM authentication, your username also contains the NTLM domain name. For example <code>MyDomain\username</code>.

Properties	Description
domainId	The unique ID of the domain in Collibra Data Intelligence Cloud in which you want to ingest the assets. Finding the domain ID - Open the domain. - Copy the domain ID. Tip If you go to your domain, you can find the domain ID in the URL. The URL looks like: https://<yourcollibrainstance>/domain/ 22258f64-40b6-4b16-9c08- c95f8ec0da26?view=00000000-0000-0000- 0000-0000000040001. In this example, the domain ID is in bold.

Properties	Description						
folderFilter	An option to include only specific folders that contain reports or KPIs in the ingestion process. Important This property must be included in your configuration file and it cannot be empty. If you want to ingest all folders, use *, for example: "folderFilter": ["*"]. You can filter on multiple folders by: ◦ Specifying folder names. ◦ Specifying the full path to folders. ◦ Using a wildcard. ◦ Using a combination of these approaches. For example: ["folder1", "/database/folder2", /folder3/*"] Show me some examples <table><tr><th>Scenario</th><th>Configuration</th></tr><tr><td>Ingest all folders with the name Folder3, anywhere in the folder hierarchy.</td><td><pre>"folderFilter": ["Folder3"]</pre> Note Reports in child folders of Folder3 are not included in the ingestion. As such: ◦ Reports in /Folder1/Folder2/Folder3 are included in the ingestion. ◦ Reports in /Folder3/ChildFolder are not included in the ingestion. </td></tr><tr><td>Ingest Folder1 and Folder2.</td><td><pre>"folderFilter": ["Folder1", "Folder2"]</pre></td></tr></table>	Scenario	Configuration	Ingest all folders with the name Folder3, anywhere in the folder hierarchy.	<pre>"folderFilter": ["Folder3"]</pre> Note Reports in child folders of Folder3 are not included in the ingestion. As such: ◦ Reports in /Folder1/Folder2/Folder3 are included in the ingestion. ◦ Reports in /Folder3/ChildFolder are not included in the ingestion.	Ingest Folder1 and Folder2.	<pre>"folderFilter": ["Folder1", "Folder2"]</pre>
Scenario	Configuration						
Ingest all folders with the name Folder3, anywhere in the folder hierarchy.	<pre>"folderFilter": ["Folder3"]</pre> Note Reports in child folders of Folder3 are not included in the ingestion. As such: ◦ Reports in /Folder1/Folder2/Folder3 are included in the ingestion. ◦ Reports in /Folder3/ChildFolder are not included in the ingestion.						
Ingest Folder1 and Folder2.	<pre>"folderFilter": ["Folder1", "Folder2"]</pre>						

Properties	Description										
	<table><tr><th>Scenario</th><th>Configuration</th></tr><tr><td>Ingest two folders that are both named Folder1.</td><td>In this case, specify the full paths to the folders, for example: <pre>"folderFilter": ["/Database1/Folder1", "/Database2/Database3/Folder1"]</pre></td></tr><tr><td>Use a wildcard to ingest all child folders of Folder1.</td><td><pre>"folderFilter": ["/Folder1/*"]</pre> Note The reports in all child folders of Folder1 are ingested, but the reports in Folder1 itself are not ingested. </td></tr><tr><td>Ingest all reports from Folder1 and all of the reports in the child folders of Folder1.</td><td><pre>"folderFilter": ["/Folder1/*", "/Folder1"]</pre></td></tr><tr><td colspan="2"> Tip For more information about connecting to a SSRS or PBRS folder, see the Microsoft documentation. </td></tr></table>	Scenario	Configuration	Ingest two folders that are both named Folder1.	In this case, specify the full paths to the folders, for example: <pre>"folderFilter": ["/Database1/Folder1", "/Database2/Database3/Folder1"]</pre>	Use a wildcard to ingest all child folders of Folder1.	<pre>"folderFilter": ["/Folder1/*"]</pre> Note The reports in all child folders of Folder1 are ingested, but the reports in Folder1 itself are not ingested.	Ingest all reports from Folder1 and all of the reports in the child folders of Folder1.	<pre>"folderFilter": ["/Folder1/*", "/Folder1"]</pre>	Tip For more information about connecting to a SSRS or PBRS folder, see the Microsoft documentation.	
	Scenario	Configuration									
	Ingest two folders that are both named Folder1.	In this case, specify the full paths to the folders, for example: <pre>"folderFilter": ["/Database1/Folder1", "/Database2/Database3/Folder1"]</pre>									
	Use a wildcard to ingest all child folders of Folder1.	<pre>"folderFilter": ["/Folder1/*"]</pre> Note The reports in all child folders of Folder1 are ingested, but the reports in Folder1 itself are not ingested.									
	Ingest all reports from Folder1 and all of the reports in the child folders of Folder1.	<pre>"folderFilter": ["/Folder1/*", "/Folder1"]</pre>									
Tip For more information about connecting to a SSRS or PBRS folder, see the Microsoft documentation.											

Properties	Description
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

2. Save the configuration file.

Steps

1. Open the lineage-harvester.conf file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
general	This section describes the connection information between the lineage harvester and Data Catalog.
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This applies only for Collibra Cloud for Government customers.

Properties	Description
url	The URL of the Collibra Data Lineage service instance. “url”: “https://techlin-gov.collibra.com” Warning This applies only for Collibra Cloud for Government customers.
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This applies only for Collibra Cloud for Government customers.
catalog	This section contains information that is necessary to connect to Data Catalog.
url	The URL of your Collibra Data Intelligence Cloud environment. Note You can only enter the public URL of your Collibra DGC environment. Other URLs will not be accepted.
username	The username that you use to sign in to Collibra.

Properties	Description
useCollibraSystemName	Indication whether you want to use the system or server name of a data source to match to the System asset you created when you prepared the physical data layer. This is useful when you have multiple databases with the same name. By default, the <code>useCollibraSystemName</code> property is set to <code>false</code>. If you want to use it, set it to <code>true</code>. Important ◦ If you set this property to <code>true</code>, the lineage harvester reads the value of the <code>collibraSystemName</code> property in your Tableau <source-ID> configuration file. ◦ If you set the <code>useCollibraSystemName</code> property to <code>false</code>, the lineage harvester ignores the <code>collibraSystemName</code> property in the <code><source-ID></code> configuration file. Note If you set the <code>useCollibraSystemName</code> property to <code>true</code>, but you don't define the system name in the Tableau <code><source ID></code> configuration file, the system name in the technical lineage is <code>DEFAULT</code>.
type	The kind of data source. In this case, the value has to be <i>Tableau</i> .
sources	This section contains the Tableau connection properties.

Properties	Description
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable. For example, <code>my_tableau</code>. Warning In the <code>sources</code> section of your lineage harvester configuration file, you can only specify one <code>id</code> property per Tableau server or Tableau online account. If you have multiple <code>id</code> properties for a single Tableau server or Tableau online account, ingestion will fail. If you have multiple <code>id</code> properties in the configuration file, it means you intend to ingest from multiple unique Tableau servers or Tableau online accounts. Warning If you are switching between the lineage harvester and Edge, the value of this property must exactly match the value of the Source ID field in your Edge capacity.
url	The link to the data in Tableau.
username	The username you use to sign in to the Tableau server. Warning As of October 2022, Tableau is enforcing multi-factor authentication for Tableau Cloud Admin users. However, the lineage harvester doesn't support multi-factor authentication. Therefore, Tableau Cloud users with an Admin role must use token-based authentication. This does not affect Tableau Server users or Tableau Cloud users with an Explorer role. Important If you want to use token-based authentication, you need to replace <code>username</code> with <code>tokenName</code>. You must specify either <code>username</code> or <code>tokenName</code>; if both exist, then <code>tokenName</code> is used.

Properties	Description
tokenName	The lineage harvester authentication token. Note For token-based authentication, use this property in your lineage harvester configuration file, instead of the <code>username</code> property. If both properties are present, <code>tokenName</code> is used.

Properties	Description
sitelds	The site IDs of the Tableau sites that you want to include in the ingestion process. If you want to ingest the metadata in a Tableau site in a specific domain, specify the following properties: ◦ This property. ◦ The <code>site_name: domain_id</code> property in the <code>filters</code> section in the Tableau <source ID> configuration file. Important The site ID is the URL of the site to which you want to sign in. When you manually sign in to Tableau Server or Tableau Online, the site ID is the value that appears after <code>/site/</code> in the browser address bar. In the following example URLs, the site ID is <code>MarketingTeam</code>: ◦ Tableau Server: <code>http://MyServer/#!/site/MarketingTeam/projects</code> ◦ Tableau Online: <code>https://10ay.online.tableau.com/#!/site/MarketingTeam/workbooks</code> On Tableau Server, however, the URL of the Default site does not specify the site. For example, the URL for a view named <code>Profits</code>, on a site named <code>Sales</code>, is <code>http://localhost/#!/site/sales/views/profits</code>. The URL for this same view on the Default site is <code>http://localhost/#!/views/profits</code>. The site name <code>Sales</code> does not figure in the URL. If you can't see the site ID, leave this property empty: <code>"siteIds": [""]</code> Example If you want to ingest two Tableau sites "Site 1" and "Site 2", you can enter the following information in the <code>sitelds</code> property: <code>["site ID of Site 1", "site ID of Site 2"]</code>.

Properties	Description
siteNames	The site names of the corresponding site IDs. Important This property is: - Optional for Tableau Server - Mandatory for Tableau Online. Warning If you have Tableau Server and you don't use this property, you must delete it from your configuration file. Don't leave the property in the configuration file without a value.
restOnly	Indication whether or not you would like to use both the Tableau REST API and Tableau Metadata API to harvest Tableau metadata. <code>false</code> (default): The lineage harvester will use the REST API and Metadata API to harvest Tableau metadata. <code>true</code>: The lineage harvester will only use the REST API to harvest Tableau metadata. Note This property must be set to <code>false</code>, to: - Enable technical lineage and the automatic stitching of Column assets to Tableau Data Attribute assets. - Harvest owner information for Tableau projects, workbooks and data models.
domainId	The unique reference ID of the domain in Collibra Data Intelligence Cloud in which you want to ingest the Tableau assets. This property represents the default domain. How do I find a domain reference ID? Open the relevant domain in Collibra. The URL looks like: <code>https://<yourcollibrainstance>/domain/22258f64-40b6-4b16-9c08-c95f8ec0da26?view=00000000-0000-0000-0000-000000040001</code>. In this example, the reference ID is in bold.

Properties	Description
excludeImages	Optional property for excluding the downloading of images. To exclude the downloading of images, set this property to <code>true</code>. To indicate the projects that you want to ingest in different domains, specify the <code>filters</code> section in your Tableau <source ID> configuration file. Note The maximum number of images that can be uploaded to Collibra per day is determined by the configuration of the file upload service, in Collibra Console. For complete details, see the Upload configuration settings in DGC service configuration: options.
concurrencyLevel	This optional property is intended to help if you are experiencing HTTP 401 Unauthorized errors due to too many concurrent HTTP calls, using the same token. It allows you to specify the internal sizing, meaning the amount of tasks that can be executed at the same time. The default value is "10", meaning as many as 10 HTTP requests can take place in parallel. Consider reducing the value if you are experiencing HTTP 401 Unauthorized errors. Setting the value to "1" effectively disables the concurrency level, so that HTTP requests will be run in a synchronous manner, instead of in parallel. Example <code>"concurrencyLevel": 5</code>

Properties	Description
deleteRawMetadataAfter Processing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

Properties	Description						
paging	Optional property for customizing the Tableau API pagination settings. The default values are sufficient in most cases; however, you can decrease them to help mitigate node limit errors, or increase them to speed up API calls. Show me the complete list of pagination settings, descriptions and default values <pre>"paging": { "databasesPageSize": 100, "tablesPageSize": 100, "tablesColumnsPageSize": 100, "tableColumnsPageSize": 1000, "datasourcesPageSize": 50, "datasourcesFieldsPageSize": 50, "datasourceFieldsPageSize": 100, "worksheetsPageSize": 100, "worksheetsFieldsPageSize": 100, "worksheetFieldsPageSize": 1000, "parametersPageSize": 1000, "usersPageSize": 100, "dashboardsPageSize": 100, "columnsLimit": 20, "fieldsLimit": 20 }</pre> Settings per metadata type and descriptions <table><tr><th>Metadata type</th><th>Setting and description</th></tr><tr><td>Dashboard</td><td> ◦ <code>dashboardsPageSize</code>: The number of dashboards per page. </td></tr><tr><td>Worksheet</td><td> ◦ <code>worksheetsPageSize</code>: The number of worksheets per page. ◦ <code>worksheetsFieldsPageSize</code>: The number of worksheet fields per page. </td></tr></table>	Metadata type	Setting and description	Dashboard	◦ <code>dashboardsPageSize</code>: The number of dashboards per page.	Worksheet	◦ <code>worksheetsPageSize</code>: The number of worksheets per page. ◦ <code>worksheetsFieldsPageSize</code>: The number of worksheet fields per page.
Metadata type	Setting and description						
Dashboard	◦ <code>dashboardsPageSize</code>: The number of dashboards per page.						
Worksheet	◦ <code>worksheetsPageSize</code>: The number of worksheets per page. ◦ <code>worksheetsFieldsPageSize</code>: The number of worksheet fields per page.						

Properties	Description	
	Metadata type	Setting and description
	Database	databasesPageSize: The number of databases per page.
	Table	- tablesPageSize: The number of tables per page. - tablesColumnsPageSize: The number of table columns per page.
	Table columns	tableColumnsPageSize: The number of table columns per page.
	Parameter	parametersPageSize: The number of parameters per page.
	Users	usersPageSize: The number of users per page.
	Data source	- datasourcesPageSize: The number of data sources per page. - datasourcesFieldsPageSize: The number of data source fields per page. - columnsLimit: The number of data source field columns per page. - fieldsLimit: The number of referenced data source fields per page.
	Data source field	- datasourceFieldsPageSize: The number of data source fields per page. - columnsLimit: The number of data source field columns per page. - fieldsLimit: The number of referenced data source fields per page.

- 2. Save the configuration file.

Steps

- 1. Open the lineage-harvester.conf file that was created when you installed the lineage harvester, and enter the values for each property.

Properties	Description
general	This section describes the connection between Collibra Data Lineage and Data Catalog.
techlin	This section contains information that is necessary to connect to the Collibra Data Lineage service instance. Warning This section applies only to US government customers.
url	The URL of the Collibra Data Lineage service instance. Example “url”: “https://techlin-gov.collibra.com” Warning This section applies only to US government customers.

Properties	Description
userKey	The unique API key to connect to the Collibra Data Lineage service instance. A unique user key is needed for each Collibra environment. If you're not sure what your user key is, please contact your Collibra Customer Success Manager. Warning This section applies only to US government customers.
catalog	This section contains information that is necessary to connect to Data Catalog. Note Versions of the lineage harvester older than 1.1.2 show <code>collibra</code> instead of <code>catalog</code>.
url	The URL of your Collibra environment. Note Enter the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in to Collibra.
useCollibraSystemName	Indicates whether you want to use the system or server name of a data source to match to the System asset you created when you prepared the physical data layer . The names are case-sensitive. This is useful when you have multiple databases with the same name.
sources	This configuration section contains the required information for SQL files of a data source that were previously downloaded by the lineage harvester and is stored in the lineage harvester output folder.

Properties	Description
type	The kind of data source. In this case, the value has to be <code>LoadedSource</code> .
id	This property is used to identify the batch of harvested metadata on the Collibra Data Lineage service instance. The value can be anything as long as it is unique and human readable. For example, <code>my_loaded_snowflake_source</code> .
zipFile	The full path to the ZIP file that was created in the lineage harvester folder.
deleteRawMetadataAfterProcessing	The lineage harvester harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance, for processing. You can use this optional property to specify whether or not the raw metadata should be deleted from Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. The default value is <code>false</code>. If the property is set to <code>true</code>, the raw source metadata is deleted after processing. If set to <code>false</code>, it is stored in the Collibra infrastructure. Note Setting this property to <code>true</code> can negatively impact performance.

2. Save the configuration file.

What's next

[Run the lineage harvester](#). When you run the lineage harvester and encounter errors that are related to the lineage harvester configuration file, you can use the [technical lineage troubleshooting guide](#) or [Collibra Support Portal](#) to fix the errors.

The configuration file generator

The configuration file generator helps you create your lineage harvester configuration file by providing the structure of the file with the correct properties per data source.

The lineage harvester configuration file

The lineage harvester uses a configuration file to connect to JDBC data sources, BI tools and ETL tools. The configuration file contains references to the data sources for which you want to create a technical lineage. You have to [prepare the configuration file](#) if you want to create a technical lineage and add new relations of the type "Data Element targets / sources Data Element" between existing assets in Data Catalog, and "Column is target of / is source of Data Attribute" between assets from ingested BI sources and assets in Data Catalog.

Tip You have to save the configuration file in the **config** directory in the [lineage harvester](#) folder.

Empty configuration file

When you run the lineage harvester for the first time, it creates an empty configuration file. To create a technical lineage, you have to manually add properties and values, per data source, to this configuration file.

The following image shows an example of the empty configuration file created by the lineage harvester.

```
{
  "general" : {
    "catalog" : {
      "url" : "",
      "username" : "",
    },
    "useCollibraSystemName" : false
  },
  "sources" : [ {
    "type" : "Database",
    "id" : "MyDB",
    "hostname" : "",
    "username" : "",
  },
```

```
    "dialect" : "",
    "collibraSystemName" : "",
    "databaseNames" : [ ],
    "port" : 1521
  } ]
}
```

Configuration file generator

Note The configuration file generator is only available in the [online version](#) of this guide.

The configuration file generator creates an example configuration file with the data source properties of your choosing:

1. Scroll down to the configuration file example.
2. Paste the example in your empty configuration file in the lineage harvesterconfig folder.
3. Replace the values in the example to match your actual data source information.

Tip Make sure you [understand each property](#) and know which values you must use to access your data source information.

4. [Run](#) the lineage harvester.

Warning Some browser plug-ins may slow the configuration file generator down.

```
{
  "general": {
    "catalog" : {
      "url" : "https://companydomain.collibra.com",
      "username" : "my-Collibra-username"
    },
    "useCollibraSystemName" : false
  },
  "sources" : [
    {
      "id" : "adf_source",
      "type" : "AzureDataFactory",
      "tenantDomain": "tenant-domain",

```

```

    "loginFlow": {
      "type": "ServicePrincipal",
      "applicationId": "application-id"
    },
    "subscriptionId" : "subscription-id",
    "resourceGroupName" : "resource-group-name",
    "factories" : ["factoryname1","factoryname2"],
    "deleteRawMetadataAfterProcessing": false
  }
  {
    "type": "dbt",
    "id": "mydbt",
    "tokenName": "mytokenname",
    "adminUrl": "https://cloud.getdbt.com/api/v2",
    "metadataUrl": "https://metadata.cloud.getdbt.com/graphql"
    "deleteRawMetadataAfterProcessing": false
  }

  {
    "id" : "my_dbt_source",
    "type" : "ExternalDirectory",
    "dirType" : "dbt",
    "path" : "/opt/Colibra/techlin/dbt-core-files/",
    "recursive" : true,
    "deleteRawMetadataAfterProcessing" : false
  }

  {
    "id" : "datastage_source",
    "type" : "ExternalDirectory",
    "dirType" : "DATASTAGE",
    "path" : "/path/to/the/datastage/folder/",
    "mask" : "*",
    "recursive" : false,
    "deleteRawMetadataAfterProcessing": false
  }

  {
    "id" : "informatica_source",
    "type" : "ExternalDirectory",
    "dirType" : "INFA",
    "path" : "/path/to/the/informatica/folder/",
    "mask" : "*",
    "recursive" : false,
    "deleteRawMetadataAfterProcessing": false
  }

  {
    "id" : "datastage_source",
    "type" : "ExternalDirectory",
    "dirType" : "SSIS",

```

```

    "path" : "/path/to/the/ssis/folder/",
    "mask" : "*",
    "recursive" : false,
    "deleteRawMetadataAfterProcessing": false
  }
  {
    "type" : "IICS",
    "id" : "iics_source",
    "loginUrl" : "https://dm-us.informaticaintelligentcloud.com",
    "username" : "login-iics",
    "deleteRawMetadataAfterProcessing": false,
    "objects" : [
      {
        "path" : "Default/Sales",
        "type" : "Project"
      },
      {
        "path" : "My Project/Statistics",
        "type" : "Project"
      }
    ]
  }
  {
    "id" : "my-matillion-project",
    "type" : "Matillion",
    "url" : "https://my-domain",
    "groupName" : "my-matillion-group",
    "projectName" : "redshift-project",
    "environmentName" : "redshift-environment",
    "startTimestamp" : 1594080796911,
    "deleteRawMetadataAfterProcessing": false,
    "auth": {
      "type": "Basic",
      "username": "ec2-user"
    }
  }
  {
    "type": "Tableau",
    "id": "unique-ID",
    "url": "URL to Tableau server?",
    "username": "Admin",
    "siteIds": ["site ID of Tableau Site 1", "site ID of Tableau Site
2"],
    "siteNames": ["site name of Tableau Site 1", "site name of Tableau
Site 2"],
    "restOnly": false,
    "domainId": "Domain-resource-ID",
    "excludeImages": true,
    "deleteRawMetadataAfterProcessing": false,
    "paging": {

```

```

        "pagination-setting": 100,
        "pagination-setting-2": 100
    }
}
{
    "id" : "looker-source",
    "type" : "Looker",
    "lookerUrl" : "https://<instance-name.api.looker.com",
    "clientId" : "my-looker-api-user-name",
    "clientSecret": "looker-api-userkey",
    "domainId" : "22258f64-40b6-4b16-9c08-c95f8ec0da26",
    "deleteRawMetadataAfterProcessing": false
}
{
    "id": "<unique-id>",
    "type": "SSRS",
    "url": "http://<IP address or computer name>/Reports",
    "username": "<server-api-user-name>",
    "domainId": "<domain-resource-id>",
    "folderFilter": ["/Folder1/*", "Folder2"],
    "deleteRawMetadataAfterProcessing": false
}
{
    "collibraSystemName" : "custom-system-name",
    "id" : "MyCustomLineage",
    "type" : "ExternalDirectory",
    "dirType" : "custom-lineage",
    "path" : "/path/to/custom-lineage/dir",
    "deleteRawMetadataAfterProcessing": false
}
{
    "type" : "LoadedSource",
    "id" : "MySource",
    "zipFile" : "/path/to/source-MySource.zip",
    "deleteRawMetadataAfterProcessing": false
}
{
    "id" : "database_source",
    "type" : "Database",
    "username" : "MyUsername",
    "dialect" : "hive",
    "externalDbName": "CData"
    "databaseNames" : ["MyDefaultDbName"],
    "hostname" : "localhost",
    "collibraSystemName" : "apache-hive-system",
    "port" : 1521,
    "deleteRawMetadataAfterProcessing": false,
    "customConnectionProperties" : ""
}
}

```



```

        "id": "oracle-id",
        "type": "DatabaseOracle",
        "hostname": "host_url",
        "username": "user1",
        "collibraSystemName": "automation_csn",
        "port": 1521,
        "serviceNames": ["sn1", "sn2"],
        "databaseNames": ["db1", "db2"],
        "deleteRawMetadataAfterProcessing": false
    }
    {
        "id" : "bigquery_source",
        "type" : "DatabaseBigQuery",
        "projectIDs" : [ "bigquery_project1", "bigquery_project2" ],
        "region": "europe-west1"
        "auth" : "/path/to/the/authentication/file.json",
        "collibraSystemName" : "bigquery-system-name",
        "deleteRawMetadataAfterProcessing": false
    }
    {
        "id" : "snowflake_source",
        "type" : "DatabaseSnowflake",
        "mode" : "SQL|SQL-API",
        "collibraSystemName" : "snowflake-system-name",
        "auth": {
            "type": "KeyPair|Basic",
            "username": "some_username",
            "pathToPrivateKey": "path_to_your_private_key_file",
            "usePassword": "true|false"
        },
        "hostname" : "MyAccountName.snowflakecomputing.com",
        "databaseNames" : ["MyFirstDbName", "MySecondDbName"],
        "extraDatabaseDefinitions": :
["MyFirstExternalDbName", "MySecondExternalDbName"],
        "schemaNames" : ["MyFirstSchemaName", "MySecondSchemaName"],
        "warehouse" : "MySnowflakeWarehouseName",
        "days" : "1",
        "deleteRawMetadataAfterProcessing": false,
        "customConnectionProperties" : "role=MYROLE"
    }
    {
        "id": "mstr",
        "type": "MSTR_V2",
        "url": "http://ec2-18-216-24-232.us-east-
2.compute.amazonaws.com:8080",
        "username": "Administrator",
        "maxParallelRequests": 5,
        "appUrlSuffix": "MicroStrategy/servlet/mstrWeb",
        "deleteRawMetadataAfterProcessing": false
    }
}

```

```

{
  "type" : "PowerBI",
  "id" : "power-bi-1",
  "tenantDomain": "collibra3.onmicrosoft.com",
  "loginFlow": {
    "type": "ServicePrincipal",
    "applicationId": "be560fac-7545-4ce2-ad9f-cbce14c59af6"
  },
  "domainId": "domain-reference-ID"
  "daxParserEnabled": true,
  "deleteRawMetadataAfterProcessing": false
}
{
  "id" : "sqldirectory_source",
  "type" : "SqlDirectory",
  "path" : "/path/to/the/sql/folder/",
  "mask" : "*",
  "recursive" : false,
  "dialect" : "db2",
  "database" : "MyDefaultDbName",
  "collibraSystemName" : "data-source-system",
  "schema" : "MyDefaultDbSchema",
  "verbose" : true,
  "deleteRawMetadataAfterProcessing": false
} ]
}

```

Azure Data Factory

The following example shows an [Azure Data Factory <source ID> configuration file](#).

```

{
  "found_dbname=databasename1;found_hostname=server-
name.onmicrosoft.com;found_schema=schema1": {
    "dbname": "mssql-database-name",
    "schema": "mssql-schema-name",
    "dialect": "mssql",
    "collibraSystemName": "mssql-system-name"
  },
  "found_dbname=datafactory_linkedservice;found_hostname=*": {
    "dbname": "linkedservice-database",
    "schema": "linkedservice-schema",
    "collibraSystemName": "linkedservice-system-name"
  }
}

```

dbt Cloud

The following example shows a [dbt Cloud <source ID> configuration file](#).

```
{
  "collibraSystemNames": {
    "projects": [
      { "project_id": "654321", "collibraSystemName": "SystemName" }
    ],
    "filter": {
      "jobIds": [1234],
      "projectIds": [654321]
    },
    "materializedMapping": {
      "ELS_MATERIALIZE_MULTIPLE_EXTERNAL_TABLES": "VIEW"
    }
  }
}
```

Informatica PowerCenter

The following example shows an [Informatica PowerCenter <source ID> configuration file](#).

```
{
  "connectionDefinitions": {
    "oracle_source": {
      "dbname": "oracle-source-database-name1",
      "schema": "my Oracle source schema",
      "dialect": "oracle"
    },
    "oracle_target": {
      "dbname": "oracle-target-database-name2",
      "schema": "my other oracle target schema",
      "dialect": "oracle"
    }
  },
  "collibraSystemNames": {
    "databases": [
      {
        "dbname": "oracle-source-database-name1",
        "collibraSystemName": "oracle-system-name1"
      },
      {
        "dbname": "oracle-target-database-name2",
        "collibraSystemName": "oracle-system-name2"
      }
    ]
  }
}
```

```

    ],
    "connections": [
      {
        "connectionName": "oracle-connection-name1",
        "collibraSystemName": "oracle-system-name1"
      },
      {
        "connectionName": "oracle-connection-name2",
        "collibraSystemName": "oracle-system-name2"
      }
    ]
  }
}

```

SQL Server Integration Services

The following example shows an [SQL Server Integration Services connection definitions configuration file](#).

```

{
  "ConnStringRegExTranslation": {
    "Data Source=dhb-sql-prod;Initial Catalog=SFG_repl_staging;Provider=SQLNCLI11;Integrated Security=SSPI.*": {
      "dbname": "DATAHUB",
      "schema": "DBO",
      "dialect": "mssql",
      "collibraSystemName" : "WAREHOUSE"
    },
    "Server=sb-dhub;User ID=SYS_USER;Initial Catalog=STAGEDB;Port=6306.*": {
      "dbname": "STAGEDB",
      "schema": "STAGE_OWNER",
      "dialect": "sybase",
      "collibraSystemName" : ""
    }
  }
}

```

IBM InfoSphere DataStage

The following example shows a [DataStage connection definitions configuration file](#).

```

{
  "OdbcDataSources": {
    "oracle-data-source": {
      "dbname": "my-oracle-database",
      "schema": "my-oracle-schema",
      "dialect": "oracle",
      "collibraSystemName": "my-system"
    },
    "mssql-data-source": {
      "dbname": "my-mssql-database",
      "schema": "my-mssql-schema",
      "dialect": "mssql",
      "collibraSystemName": "my-system"
    }
  },
  "NonOdbcConnectors": {
    "admin@database-name": {
      "dbname": "my-netezza-database",
      "schema": "my-netezza-schema",
      "dialect": "netezza",
      "collibraSystemName": "my-system"
    },
    "admin@second-database-name": {
      "dbname": "my-second-netezza-database",
      "schema": "my-second-netezza-schema",
      "dialect": "netezza",
      "collibraSystemName": "my-system"
    }
  },
  "jobs": [
    "my_job_1",
    "my_job_2"
  ],
  "jobParameters": [
    {
      "name": "job_parameter_name_1",
      "value": "job_parameter_value_1"
    },
    {
      "name": "job_parameter_name_2",
      "value": "job_parameter_value_2"
    }
  ]
}

```

Informatica Intelligent Cloud Services

The following example shows an Informatica Intelligent Cloud Services [configuration file](#).

```
{
  "collibraSystemNames": {
    "connections": [
      {
        "connectionName": "DG_con_standby_cmdm_clientors",
        "collibraSystemName": "PUBLIC"
      },
      {
        "connectionName": "DG_con_dev_dg_dgiauser_su",
        "collibraSystemName": "PUBLIC"
      }
    ]
  },
  "connectionDefinitions": [
    {
      "connectionName": "DG_con_standby_cmdm_clientors",
      "databaseName": "main",
      "schemaName": "dbo",
      "dialect": "oracle"
    },
    {
      "connectionName": "DG_con_dev_dg_dgiauser_su",
      "databaseName": "main",
      "schemaName": "dbo",
      "dialect": "oracle"
    }
  ]
}
```

Tableau

The following example shows a Tableau [configuration file](#).

```
{
  "collibraSystemNames": {
    "files": [
      {
        "filePath": "C:\\ProgramData\\Tableau\\Tableau
Server\\data\\files\\sample.xls",
        "collibraSystemName": "sample-files"
      }
    ]
  },
}
```

```

    "connectors": [
      {
        "connectorUrl": "tableau-server-connector-url.com",
        "collibraSystemName": "Oracle-connector"
      }
    ],
    "cloudFiles": [
      {
        "name": "file-name",
        "collibraSystemName": "FILE"
      }
    ]
  },
  "hostnameMapping": {
    "found_dbname=*;found_hostname=abc.net;found_schema=DEFAULT": {
      "dbname": "CData",
      "schema": "Jan_1_2022",
      "dialect": "spark",
      "collibraSystemName": "TV_testing"
    }
  },
  "filters": {
    "sites": {
      "site_name": "domain_id"
    },
    "projects": {
      "site_name2 > project_name2": "domain-reference-id2",
      "site_name3 > project_name3 > subproject_name": "domain-reference-id2"
    }
  }
}

```

Looker

The following example shows a Looker [source ID](#) configuration file.

```

{
  "Connections": {
    "connection-object1": {
      "dialect": "mssql",
      "schema": "mssql-schema-name",
      "dbname": "mssql-database-name",
      "collibraSystemName": "mssql-system-name"
    },
    "connection-object2": {
      "dialect": "oracle",
      "schema": "oracle-schema-name",

```

```

        "dbname": "oracle-database-name",
        "collibraSystemName": "oracle-system-name"
    },
    "filters": [
        {
            "domainId": "605fa8ae-f8c6-4261-938b-8326e2806f3d",
            "description": "Databricks_folder",
            "folderIds": [
                "abc-123",
                "def-456"
            ]
        },
        {
            "domainId": "245bc5a4-4c30-44b5-8356-ddbe708b56d6",
            "description": "personal",
            "folderIds": [
                "hij-789*",
                "jkl-101112"
            ]
        }
    ]
}

```

SQL Server Reporting Services and Power BI Report Server

The following example shows a SQL Server Reporting Services and Power BI Report Server [source ID configuration file](#).

```

{
  "DataSources": {
    "Redshift": {
      "dbname": "redshift-database-name",
      "schema": "redshift-schema-name",
      "dialect": "redshift",
      "collibraSystemName": "redshift-system-name"
    },
    "Oracle": {
      "dbname": "oracle-database-name",
      "schema": "oracle-schema-name",
      "dialect": "oracle",
      "collibraSystemName": "oracle-system-name"
    }
  },
  "CustomDataSources": {

```



```

    "/path to report/custom data souce name": {
      "dbname": "mssql-database-name",
      "dialect": "mssql"
    }
  }
}

```

Power BI

The following example shows a Power BI [source ID](#) configuration file.

```

{
  "found_dbname=databasename1;found_hostname=*;found_schema=schema1":
  {
    "dbname": "mssql-database-name",
    "schema": "mssql-schema-name",
    "dialect": "mssql",
    "collibraSystemName": "mssql-system-name"
  },
  "found_dbname=databasename2;found_hostname=server-
name.onmicrosoft.com;found_schema=schema2": {
    "dbname": "oracle-database-name",
    "schema": "oracle-schema-name",
    "dialect": "oracle",
    "collibraSystemName": "oracle-system-name"
  },
  "filters": [
    {
      "domainId": "<domain-ref-id>",
      "description": "FirstFilter",
      "workspaceNames": "*",
      "excludeWorkspaceIds": ["workspaceC", "workspaceD"]
    },
    {
      "domainId": "<domain-ref-id>",
      "description": "SecondFilter",
      "workspaceNames": ["workspace3", "workspace4"],
      "capacityIds": ["id1", "id2"]
    }
  ]
}

```

Matillion

The following example shows a Matillion [source ID](#) configuration file.

```
{
  "found_dbname=dbtest;found_hostname=test": {
    "collibraSystemName": "mssql-system-name"
  },
  "found_dbname=testsid;found_hostname=*": {
    "dbname": "oracle-database-name",
    "schema": "oracle-schema-name",
    "collibraSystemName": "oracle-system-name"
  }
}
```

MicroStrategy

The following example shows a MicroStrategy [source ID](#) configuration file.

```
{
  "default_domain_id": "66ab7911-d84a-43d6-b9f7-40130jd84ds98",
  "filters": [
    {
      "domainId": "3ae946ec-9030-46b8-ad86-a217b8f99bab",
      "projectIDs": ["3FF7CF7D4C5A2AEBF14D8B9802698B27"]
    },
    {
      "domainId": "21cfb3ae-0616-478e-9060-114130540be5",
      "projectNames": ["Human Resources Analysis Module"]
    }
  ],
  "datasourceMapping": [
    {
      "found_datasource": "REDSHIFT",
      "found_project": "*",
      "mapping": {
        "dbname": "RD_pearl",
        "schema": "Default_North",
        "dialect": "spark",
        "collibraSystemName": "TV_dev"
      }
    }
  ]
}
```

The following is an example of a lineage harvester configuration file for US government customers using Power BI. The `techlin` section contains the information required to connect to the Collibra Data Lineage service instance.

```
{
  "general": {
    "techlin": {
```

```

    "url": "https://techlin-gov.collibra.com",
    "userKey": "<your-unique-api-key>"
  },
  "catalog": {
    "url": "https://catalog-instance.collibra.com",
    "username": "Admin"
  },
  "useCollibraSystemName": false
},
"sources": [ {
  "type": "PowerBI",
  "id": "power-bi-id",
  "tenantDomain": "collibrapowerbi.onmicrosoft.com",
  "loginFlow": {
    "type": "ServicePrincipal",
    "applicationId": "ab123cde-1234-1234-1234-abcd12e34fg5"
  },
  "domainId": "domain-reference-ID",
  "deleteRawMetadataAfterProcessing": true
} ]
}

```

Prepare an SQL directory

To create technical lineage for JDBC data sources by using the folder connection type, you must provide SQL files that include your SQL queries. Collibra Data Lineage processes the metadata based on your queries to create the technical lineage.

For more information about the connection types for different data sources, go to [Supported data sources for technical lineage](#).

Note For best technical lineage results, use the JDBC connection to ingest JDBC sources when possible, rather than using the folder connection type with the SQL files.

Steps

1. Create a local folder.
2. Create your SQL files. Ensure that the following requirements are met for the SQL files:
 - The SQL files must be UTF-8 encoded.
 - For better ingestion, include one SQL statement in one SQL file.

- Collibra Data Lineage processes the SQL files in alphabetical order. The SQL files that include the Data Definition Language (DDL) statements must be processed before the SQL files that include the Data Manipulation Language (DML) statements. To ensure this order, name the SQL files such that those containing DDL statements come before those containing DML statements alphabetically.
- The database and schema names in the SQL statements in your SQL files take precedence over the values that you provide for the `database` and `schema` properties in the lineage harvester configuration file. If your SQL statements contain database and schema names, Collibra Data Lineage uses them for stitching. If your SQL statements do not contain database and schema names, Collibra Data Lineage uses the values of the `database` and `schema` properties in the configuration file for stitching. For more information, go to [lineage harvester configuration file](#) and [Automatic stitching for technical lineage](#).

For more information, go to [Supported SQL syntax](#).

3. Store the SQL files in the local folder.

Example 1 SQL statements do not include schema and database names

This example shows the SQL files that include the queries on the `Persons` and `JobInformation` tables and the `JobTitleView` view. The SQL statements don't contain the database and schema values, so the lineage harvester uses the values of the `database` and `schema` properties that you specify in [the lineage harvester configuration file](#) for stitching. The SQL files are named in a way that ensures the DDL statements are processed before the DML statement.

- The `ddl-persons.sql` file

```
CREATE TABLE Persons (  
    PersonID int,  
    LastName varchar(255),  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255)  
);
```

- The **ddl-jobinformation.sql** file

```
CREATE TABLE JobInformation (  
    PersonID int,  
    Department varchar(255),  
    Title varchar(255)  
);
```

- The **view-jobtitle.sql** file

```
CREATE VIEW JobTitleView AS  
SELECT  
    Persons.PersonID,  
    Persons.FirstName,  
    Persons.LastName,  
    JobInformation.Title  
from  
    Persons  
    INNER JOIN JobInformation ON Persons.PersonID = JobIn-  
formation.PersonId
```

Example 2 SQL statements include schema and database names

This example shows SQL files that include the queries on the `Persons` and `JobInformation` tables and the `JobTitleView` view. The SQL statements contain the database and schema names for each table and view, and Collibra Data Lineage uses them for stitching. The SQL files are named in a way that ensures the DDL statements are processed before the DML statement.

- The **ddl-db1-schemaA-persons.sql** file

```
CREATE TABLE DB1.SchemaA.Persons (  
    PersonID int,  
    LastName varchar(255),  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255)  
);
```

- The **ddl-db2-schemaB-jobinformation.sql** file

```
CREATE TABLE DB2.SchemaB.JobInformation (  
    PersonID int,  
    Department varchar(255),
```

```
Title varchar(255)
);
```

- The **view-db2-schemaC-jobtitleview.sql** file

```
CREATE VIEW DB2.SchemaC.JobTitleView AS
SELECT
    Persons.PersonID,
    Persons.FirstName,
    Persons.LastName,
    JobInformation.Title
from
    DB1.SchemaA.Persons
    INNER JOIN DB2.SchemaB.JobInformation ON Persons.PersonID =
JobInformation.PersonId
```

What's next

Add your data source information in [the lineage harvester configuration file](#).

Download SQL files to the lineage harvester folder

You can download the SQL files of a data source that is stored locally and cannot be accessed via the network. The lineage harvester then stores the data source information in a ZIP file.

To create a technical lineage for these data sources, you only have to include the ID of the data source and the path to the ZIP file in the [configuration file](#).

Note Click [here](#) to see a list of all supported data sources.

Prerequisites

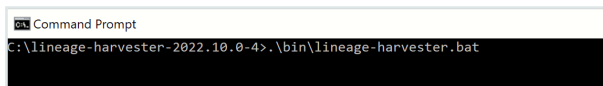
- You have [downloaded](#) the lineage harvester and you have the necessary [system requirements](#) to run it.
- You have the necessary permissions to all [database objects](#) that the lineage harvester accesses.
- You have the necessary [data source-specific permissions](#) to access the data objects of your data sources

Note For a detailed overview of the permissions that you need to access the data objects of your data sources, see the online user guide.

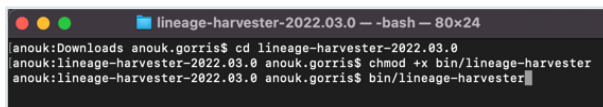
Steps

1. Start the lineage harvester to create an empty lineage harvester configuration file by entering the following command:

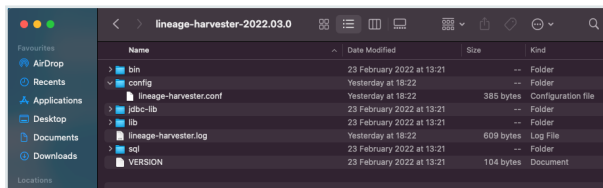
- **Windows:** `.\bin\lineage-harvester.bat`



- **For other operating systems:** `chmod +x bin/lineage-harvester` and then `bin/lineage-harvester`



» An empty configuration file is created in the config folder.



2. Save the configuration file in the **config** directory in the lineage harvester folder.
3. **Prepare the configuration file.**

Tip Use the configuration file generator to easily create a configuration file.

4. When prompted, enter the **passwords** to connect to Collibra and your data sources. Do one of the following:
 - Enter the passwords in the console.
 - » The passwords are encrypted and stored in `/config/pwd.conf`.
 - Provide the passwords via **command line**.
 - » The passwords are stored locally and not in your lineage harvester folder.
5. Start the lineage harvester again and do one of the following:
 - To download the SQL files of all data sources in the configuration file, run the following command:

```
./bin/lineage-harvester load-sources
```

- To download the SQL files of specific data sources in the configuration file, run the following command:

```
./bin/lineage-harvester load-sources -s "ID of the data source"
```

Tip This command allows you to download specific SQL files in the configuration file, without refreshing other SQL files. This reduces the time you need to download your SQL files, since you only download specific ones without affecting the others. If you want to download SQL files of multiple data sources, add `-s "ID of another data source"` per data source to the command.

- » The lineage harvester downloads the SQL files of the data sources and stores them in a ZIP file per data source in the lineage harvester output folder.

What's next?

You can now [prepare a configuration file](#) for the SQL files of data sources that you want to include in your technical lineage.

Prepare a <source ID> configuration file

Depending on your data source, you might have to, or want to, prepare a <source ID> configuration file. Select your data source below for data source-specific information.

The lineage harvester uses a [lineage harvester configuration file](#) to collect the Azure Data Factory data objects. It then sends the metadata to the Collibra Data Lineage service instance.

Example of the <source ID> configuration file

```
{
  "found_dbname=databasename1;found_hostname=server-
name.onmicrosoft.com;found_schema=schema1": {
    "dbname": "mssql-database-name",
    "schema": "mssql-schema-name",
    "dialect": "mssql",
    "collibraSystemName": "mssql-system-name"
  },
}
```



```
"found_dbname=datafactory_linkedservice;found_hostname=*": {  
  "dbname": "linkedservice-database",  
  "schema": "linkedservice-schema",  
  "collibraSystemName": "linkedservice-system-name"  
}  
}
```

Steps

1. Create a new JSON file in the lineage harvester **config** folder.
2. Name the JSON file as **<sourceId>.conf**, where **<sourceId>** is the same as the value of the `sourceId` property in the lineage harvester configuration file and the file extension must be `.conf`.

Example If the value of the `sourceId` property in the lineage harvester configuration file is `my-adf`, the name of your JSON file must be **my-adf.conf**.

3. For each database in Azure Data Factory, add the following content to the JSON file:

Property	Description	Mandatory?
found_dbname=<database name>;found_hostname=<server name>;found_schema=<schema name> found_dbname=<datafactory_name>_<linkedservice_name>;found_hostname=*	The information of the supported data sources in Azure Data Factory to be collected by Collibra Data Lineage. You can specify any of the following values for the found_dbname property: ◦ A database name. And then you can specify the following properties: ■ found_hostname=<server name>, where <server name> is the name of the server that the database is running on. ■ found_schema=<schema name>, where <schema name> is the name of the schema. This property is optional. ◦ The combination of <datafactory_name>_<linkedservice_name>, where <datafactory_name> is a data factory name and <linkedservice_name> is a linked service name. If you use this combination, specify * for the found_hostname property.	Yes

Tip
You can use wildcards to capture multiple connection string combinations:

Show me the supported wildcards

Pattern	Description
*	Matches everything.
?	Matches any single character.
[seq]	Matches any character in "seq".
[!seq]	Matches any character not in "seq".

Property	Description	Mandatory?
dbname	The name of the database asset in Data Catalog. Specify this property with the database name that you created when you prepared the Data Catalog physical data layer .	No
schema	The name of the schema asset in Data Catalog. Specify this property with the schema name that you created when you registered the data source. If the Collibra Data Lineage fails to find the schema that you specify, it uses the default schema.	No

Property	Description	Mandatory?
dialect	If you specify a database name for the <code>found_dbname</code> property, select one of the following dialects. If you specify a linked service name for the <code>found_dbname</code> property, ignore this property. Tip You can enter one of the following values: ◦ <i>azure</i>, for an Azure SQL Server data source. ◦ <i>bigquery</i>, for a Google BigQuery data source. ◦ <i>db2</i>, for a IBM Db2 data source. ◦ <i>generic</i>, for any data source. ◦ <i>greenplum</i>, for a Greenplum data source. ◦ <i>hana</i>, for a SAP HANA data source. ◦ <i>hive</i>, for a Hive data source. ◦ <i>impala</i>, for a Impala data source. ◦ <i>mssql</i>, for a Microsoft SQL Server data source. ◦ <i>mysql</i>, for a MySQL data source. ◦ <i>netezza</i>, for a Netezza data source. ◦ <i>oracle</i>, for an Oracle data source. ◦ <i>postgres</i>, for an PostgreSQL data source. ◦ <i>redshift</i>, for an Amazon Redshift data source. ◦ <i>snowflake</i>, for a Snowflake data source. ◦ <i>spark</i>, for a Spark SQL data source. ◦ <i>sybase</i>, for a Sybase data source. ◦ <i>teradata</i>, for a Teradata data source. ◦ <i>vertica</i>, for a Vertica data source.	No

Property	Description	Mandatory?
collibraSystemName	The system or server name of the data source. Use this property with the <code>useCollibraSystemName</code> property in the configuration file to override the default Collibra System asset name for this data source. Specify this property with the same name as the name of the System asset that you create when you prepare the physical data layer in Data Catalog. If you don't prepare the physical data layer, Collibra Data Lineage cannot stitch the data objects in your technical lineage to the assets in Data Catalog. If you don't specify a value for this property, <code>DEFAULT</code> is shown in the technical lineage. How do I configure this property if I have two databases with the same name? Let's assume you have two databases named Customers. When you prepare the physical data layer in Data Catalog, you create a System asset for each of these databases. Let's say you named them Customers-Europe and Customers-USA. You can then configure this property as follows. <pre> "found_ dbname=databasename1;found_ hostname=*;found_ schema=schema1":{ "dbname": "Customers", "schema": "mssql-schema- name", "dialect": "mssql", "collibraSystemName": "Customers-Europe" }, "found_ dbname=databasename2;found_ hostname=server- name.onmicrosoft.com;found_ </pre>	No

Property	Description	Mandatory?
	<pre>schema=schema2": { "dbname": "Customers", "schema": "oracle-schema- name", "dialect": "oracle", "collibraSystemName": "Customers-USA" },</pre> Warning The value of this property must exactly match (including for case-sensitivity) the name of your System asset in Collibra.	

4. Save the <source ID> configuration file.

The lineage harvester uses a [lineage harvester configuration file](#) to collect the DataStage data objects. It then sends the metadata to the Collibra Data Lineage service instance.

Example of the <source ID> configuration file

```
{
  "OdbcDataSources": {
    "oracle-data-source": {
      "dbname": "my-oracle-database",
      "schema": "my-oracle-schema",
      "dialect": "oracle",
      "collibraSystemName": "my-system"
    },
    "mssql-data-source": {
      "dbname": "my-mssql-database",
      "schema": "my-mssql-schema",
      "dialect": "mssql",
      "collibraSystemName": "my-system"
    }
  },
  "NonOdbcConnectors": {
    "admin@database-name": {
      "dbname": "my-netezza-database",
      "schema": "my-netezza-schema",
```

```

    "dialect": "netezza",
    "collibraSystemName": "my-system"
  },
  "admin@second-database-name": {
    "dbname": "my-second-netezza-database",
    "schema": "my-second-netezza-schema",
    "dialect": "netezza",
    "collibraSystemName": "my-system"
  }
},
"jobs": [
  "my_job_1",
  "my_job_2"
],
"jobParameters": [
  {
    "name": "job_parameter_name_1",
    "value": "job_parameter_value_1"
  },
  {
    "name": "job_parameter_name_2",
    "value": "job_parameter_value_2"
  }
]
}

```

Steps

1. Create a new JSON file in the lineage harvester **config** folder.
2. Name the JSON file as **<sourceId>.conf**, where **<sourceId>** is the same as the value of the `sourceId` property in the lineage harvester configuration file and the file extension must be `.conf`.

Example If the value of the `sourceId` property in the lineage harvester configuration file is `my-datastage`, the name of your JSON file must be **`my-datastage.conf`**.

3. For each database in DataStage, add the required content to the JSON file.

Property	Description
OdbcDataSources	Open Database Connectivity data sources in IBM InfoSphere DataStage for which you want to create a technical lineage.

Property	Description
<data-source-name>	The ODBC data source name that you use in your DataStage projects. This section contains the properties to translate the database, schema and dialect.
dbname	The name of your database, to which the ODBC data source connection refers.
schema	The name of your schema, to which the ODBC data source connection refers.

Property	Description
dialect	The dialect of the referenced database. Tip You can enter one of the following values: ◦ <code>azure</code>, for an Azure SQL Server data source. ◦ <code>bigquery</code>, for a Google BigQuery data source. ◦ <code>db2</code>, for an IBM DB2 data source. ◦ <code>hana</code>, for an SAP HANA data source. ◦ <code>hana-cviews</code>, for getting lineage from calculated views in an SAP HANA Classic on-premises data source. ◦ <code>hana-cviews-v2</code>, for getting lineage from calculated views in an SAP HANA Cloud/Advanced data source. Important To get technical lineage including calculated views, you must harvest SAP HANA by adding two Technical Lineage for SqlDirectory capabilities with the Shared Storage connections. In one capability, specify the <code>hana</code> dialect, and in the other, specify the <code>hana-cviews</code> or <code>hana-cviews-v2</code> dialect. ◦ <code>hive</code>, for a HiveQL data source. ◦ <code>greenplum</code>, for a Greenplum data source. ◦ <code>mssql</code>, for a Microsoft SQL Server data source. ◦ <code>mysql</code>, for a MySQL data source. ◦ <code>netezza</code>, for a Netezza data source. ◦ <code>oracle</code>, for an Oracle data source. ◦ <code>postgres</code>, for a PostgreSQL data source. ◦ <code>redshift</code>, for an Amazon Redshift data source. ◦ <code>snowflake</code>, for a Snowflake data source. ◦ <code>spark</code>, for a Spark SQL data source. ◦ <code>sybase</code>, for a Sybase data source. ◦ <code>teradata</code>, for a Teradata data source.

Property	Description
collibraSystemName	The system or server name of the data source. Use this property with the <code>useCollibraSystemName</code> property in the configuration file to override the default Collibra System asset name for this data source. Specify this property with the same name as the name of the System asset that you create when you prepare the physical data layer in Data Catalog. If you don't prepare the physical data layer, Collibra Data Lineage cannot stitch the data objects in your technical lineage to the assets in Data Catalog. This property is optional.
NonOdbcConnectors	Other data source connectors in IBM InfoSphere DataStage for which you want to create a technical lineage. For example, DB2, Oracle or Netezza. Note This section is optional.
<data-source-connector-ID>	The data source username and database of the connector that you use in your DataStage projects. This usually looks like for example <i>admin@database-name</i>. The combination of the username and database name should be unique. The following section contains the properties to translate the database, schema and dialect.
dbname	The name of your database, to which the data source connection refers.
schema	The name of your schema, to which the data source connection refers.

Property	Description
dialect	The dialect of the referenced database. Tip You can enter one of the following values: ◦ <code>azure</code>, for an Azure SQL Server data source. ◦ <code>bigquery</code>, for a Google BigQuery data source. ◦ <code>db2</code>, for an IBM DB2 data source. ◦ <code>hana</code>, for an SAP HANA data source. ◦ <code>hana-cviews</code>, for getting lineage from calculated views in an SAP HANA Classic on-premises data source. ◦ <code>hana-cviews-v2</code>, for getting lineage from calculated views in an SAP HANA Cloud/Advanced data source. Important To get technical lineage including calculated views, you must harvest SAP HANA by adding two Technical Lineage for SqlDirectory capabilities with the Shared Storage connections. In one capability, specify the <code>hana</code> dialect, and in the other, specify the <code>hana-cviews</code> or <code>hana-cviews-v2</code> dialect. ◦ <code>hive</code>, for a HiveQL data source. ◦ <code>greenplum</code>, for a Greenplum data source. ◦ <code>mssql</code>, for a Microsoft SQL Server data source. ◦ <code>mysql</code>, for a MySQL data source. ◦ <code>netezza</code>, for a Netezza data source. ◦ <code>oracle</code>, for an Oracle data source. ◦ <code>postgres</code>, for a PostgreSQL data source. ◦ <code>redshift</code>, for an Amazon Redshift data source. ◦ <code>snowflake</code>, for a Snowflake data source. ◦ <code>spark</code>, for a Spark SQL data source. ◦ <code>sybase</code>, for a Sybase data source. ◦ <code>teradata</code>, for a Teradata data source.

Property	Description
collibraSystemName	The system or server name of the data source. Use this property with the <code>useCollibraSystemName</code> property in the configuration file to override the default Collibra System asset name for this data source. Specify this property with the same name as the name of the System asset that you create when you prepare the physical data layer in Data Catalog. If you don't prepare the physical data layer, Collibra Data Lineage cannot stitch the data objects in your technical lineage to the assets in Data Catalog. This property is optional.
Jobs	The jobs that you want the lineage harvester to collect and process to create the technical lineage. This section is optional. The following rules apply when you specify this section: - Specify jobs that are executed so that the technical lineage graph does not include any job parameters with undefined values. - Specify only the first and parent jobs in a sequence of executed jobs. The lineage harvester automatically collects all jobs that are called by the parent jobs. For example, if you have the a sequence of jobs that include job1, job2, job3, job4, and job5, where job1 calls job2, job2 calls job3, job3 calls job5, and job4 calls job3. Specify only job1 and job4, and the lineage harvester collects and processes all five jobs based on the sequence. If you do not specify this section, the lineage harvester collects all jobs, but without proper sequencing. Therefore, some inherited parameters might not be parsed.
JobParameters	The runtime parameters that are not in the DSX and ENV files. You can specify multiple job parameters.
name	The name of the job parameter.
value	The value of the job parameter.

4. Save the <source ID> configuration file.

The lineage harvester uses a [lineage harvester configuration file](#) to collect the dbt data objects. It then sends the metadata to Collibra Data Lineage service for processing. By default, the lineage harvester downloads all accounts that are accessible with the API token that you provided in the lineage harvester configuration file. For each account, the lineage harvester downloads all jobs and the resulting dbt models for each job. You can use this <source ID> configuration file to reduce the amount of data objects to be downloaded and enhance the lineage harvester performance in the following ways:

- Filter the projects and jobs to be downloaded. Include projects and jobs to be downloaded by specifying the `filter` property.
- Specify different Collibra system names for different projects by specifying the `collibraSystemNames` property.
- Map a materialization as a view instead of a table by specifying the `materializedMapping` property.

Example of the <source ID> configuration file

```
{
  "collibraSystemNames":{
    "projects":[
      {"project_id":"654321","collibraSystemName":"SystemName"}
    ]
  },
  "filter":{
    "jobIds":[1234],
    "projectIds":[654321]
  },
  "materializedMapping":{
    "ELS_MATERIALIZE_MULTIPLE_EXTERNAL_TABLES":"VIEW"
  }
}
```

Steps

1. Create a new JSON file in the lineage harvester **config** folder.
2. Name the JSON file as **<sourceId>.conf**, where <sourceId> is the same as the value of the `sourceId` property in the lineage harvester configuration file and the file extension must be `.conf`.

Example If the value of the `sourceId` property in the lineage harvester configuration file is `my-dbt`, the name of your JSON file must be `my-dbt.conf`.

3. For each database in dbt, add the following content to the JSON file:

Property	Description	Required?
<code>collibraSystemNames</code>	You can use this section to specify the Collibra System Name for each project.	No
<code>projects</code>	This section contains the project names and the Collibra system names.	No
<code>project_id</code>	Your project ID. You can find the project ID in the dbt URL right after <code>projects</code> . For example, if your dbt URL is <code>https://cloud.getdbt.com/develop/54321/projects/12345</code> , your <code>project_id</code> is 12345.	No

Property	Description	Required?
collibraSystemName	The system or server name of the data source. Use this property with the <code>useCollibraSystemName</code> property in the configuration file to override the default Collibra System asset name for this data source. Specify this property with the same name as the name of the System asset that you create when you prepare the physical data layer in Data Catalog. If you don't prepare the physical data layer, Collibra Data Lineage cannot stitch the data objects in your technical lineage to the assets in Data Catalog. See an example. In this code example, the project with the 12345 project ID is stitched to the <code>systemname1</code> System asset in Data Catalog. <pre> { "collibraSystemNames": { "projects": [{ "project_id": "12345", "collibraSystemName": "systemname1" }] }, }</pre>	No

Property	Description	Required?
filter	You can use this section to include projects and jobs to be downloaded. Collibra Data Lineage downloads and processes only the specified jobs and projects. See an example. In this code example, the job with the 1234 job ID and the projects with the 98 and 5678 project IDs are downloaded. <pre> { "filter": { "jobIds": [1234], "projectIds": [98, 5678] } } </pre>	No
jobIds	The job IDs of the jobs that you want to include. Specify an integer. Do not specify a string. To get your job ID, in your dbt, select Deploy and then Jobs. Select a job and you can find your job ID in the URL. For example, if your URL is <code>cloud.getdbt.com/deploy/65432/projects/23456/jobs/123456</code>, 123456 is your job ID.	No
projectIds	The account IDs of the accounts that you want to include. Specify an integer. Do not specify a string. To get your account ID, in dbt, click the gear icon in the upper right, select Account Settings and find your account ID in the URL. For example, if your URL is <code>cloud.getdbt.com/settings/accounts/65432</code>, 65432 is your account ID.	No

Property	Description	Required?
materializedMapping	Indicates how materializations in dbt are mapped. If you do not specify this property, CollibraData Lineage maps materializations to tables by default. You can change the mapping of a materialization to view. In the following example, the ELS_MATERIALIZE_MULTIPLE_EXTERNAL_TABLES materialization is mapped to a view. <pre> "materializedMapping": { "ELS_MATERIALIZE_MULTIPLE_EXTERNAL_TABLES": "VIEW" } </pre>	No

4. Save the <source ID> configuration file.

The lineage harvester uses a [lineage harvester configuration file](#) to collect the Informatica PowerCenter data objects. It then sends the metadata to the Collibra Data Lineage service instance.

Example of the <source ID> configuration file

```

{
  "connectionDefinitions": {
    "oracle_source": {
      "dbname": "oracle-source-database-name1",
      "schema": "my Oracle source schema",
      "dialect": "oracle"
    },
    "oracle_target": {
      "dbname": "oracle-target-database-name2",
      "schema": "my other oracle target schema",
      "dialect": "oracle"
    }
  },
  "collibraSystemNames": {
    "databases": [
      {
        "dbname": "oracle-source-database-name1",
        "collibraSystemName": "oracle-system-name1"
      },
      {
        "dbname": "oracle-target-database-name2",
        "collibraSystemName": "oracle-system-name2"
      }
    ]
  }
}

```

```

    }
  ],
  "connections": [
    {
      "connectionName": "oracle-connection-name1",
      "collibraSystemName": "oracle-system-name1"
    },
    {
      "connectionName": "oracle-connection-name2",
      "collibraSystemName": "oracle-system-name2"
    }
  ]
}

```

Steps

1. Create a new JSON file in the lineage harvester **config** folder.
2. Name the JSON file as **<sourceId>.conf**, where **<sourceId>** is the same as the value of the `sourceId` property in the lineage harvester configuration file and the file extension must be `.conf`.

Example If the value of the `sourceId` property in the lineage harvester configuration file is `my-infa`, the name of your JSON file must be **my-infa.conf**.

3. For each database, add the required content to the JSON file.
If certain properties are not specified in the source ID file, an analyze error called **CONFIGURATION** is displayed in the transformations table on the [Sources tab page](#) when the technical lineage is created. The unspecified properties are marked as **UNDEFINED** in the analyze error. For more information about the analyze errors, go to [Analyze errors and possible solutions](#) in Technical lineage Sources tab page.

Property	Description
connectionDefinitions	This section contains the connection properties to a source in Informatica PowerCenter.

Property	Description
<connectionName>	The type of your source or target data source. This section contains the connection properties to a source or target in Informatica PowerCenter. Note Define a connection in the JSON file only once; specifically, define a data source with the <connectionName> property specified only once in the JSON file. If you define a connection multiple times, unexpected lineage and stitching issues might occur.
dbname	The name of your source or target database.
schema	The name of your source or target schema.

Property	Description
dialect	The dialect of the referenced database. Tip You can enter one of the following values: ◦ <code>azure</code>, for an Azure SQL Server data source. ◦ <code>bigquery</code>, for a Google BigQuery data source. ◦ <code>db2</code>, for an IBM DB2 data source. ◦ <code>hana</code>, for an SAP HANA data source. ◦ <code>hana-cviews</code>, for getting lineage from calculated views in an SAP HANA Classic on-premises data source. ◦ <code>hana-cviews-v2</code>, for getting lineage from calculated views in an SAP HANA Cloud/Advanced data source. Important To get technical lineage including calculated views, you must harvest SAP HANA by adding two Technical Lineage for SqlDirectory capabilities with the Shared Storage connections. In one capability, specify the <code>hana</code> dialect, and in the other, specify the <code>hana-cviews</code> or <code>hana-cviews-v2</code> dialect. ◦ <code>hive</code>, for a HiveQL data source. ◦ <code>greenplum</code>, for a Greenplum data source. ◦ <code>mssql</code>, for a Microsoft SQL Server data source. ◦ <code>mysql</code>, for a MySQL data source. ◦ <code>netezza</code>, for a Netezza data source. ◦ <code>oracle</code>, for an Oracle data source. ◦ <code>postgres</code>, for a PostgreSQL data source. ◦ <code>redshift</code>, for an Amazon Redshift data source. ◦ <code>snowflake</code>, for a Snowflake data source. ◦ <code>spark</code>, for a Spark SQL data source. ◦ <code>sybase</code>, for a Sybase data source. ◦ <code>teradata</code>, for a Teradata data source.

Property	Description
collibraSystemNames	This section contains the system or server name that is specified in your database and referenced in your connection. Note This section is only required when you set the Collibra system name flag setting to True. Note This section is only required when the useCollibraSystemName flag in the lineage harvester configuration file is set to <code>true</code>.
databases	This section contains the database information. This is required to connect directly to the system or server of the database.
dbname	The name of the database. The database name is the same as the name you entered in the <connectionName> section.
collibraSystemName	The system or server name of the database.
connections	This section contains the connection information. This is required to reference to the system or server of the connection.
connectionName	The name of the connection.
collibraSystemName	The system or server name of the connection.

Important If you are using variables in Informatica PowerCenter, add the value of the variable instead of the name in the connection definitionsJSON file. For example, if the parameter file contains `$DBConnection_dwh=DWH_EXPORT`, add the following connection definitions to the JSON file:

```
{
  "DWH_EXPORT":
    { "dbname": "DWH", "schema": "DBO" }
}
```

4. Save the <source ID> configuration file.

You use the [lineage harvester configuration file](#) to access Informatica Intelligent Cloud Services Data Integration data objects. The [lineage harvester](#) processes the data objects to create a technical lineage. You also have to prepare a specific <source ID> configuration file that defines the Intelligent Cloud Services system name.

Important You must prepare a <source ID> configuration file regardless of whether the `useCollibraSystemName` property in your lineage harvester configuration files is set to *true* or *false*.

Prerequisites

You have Admin permission on all objects that you want to harvest.

Example of the <source-ID>.conf file

```
{
  "collibraSystemNames": {
    "connections": [
      {
        "connectionName": "DG_con_standby_cmdm_clientors",
        "collibraSystemName": "PUBLIC"
      },
      {
        "connectionName": "DG_con_dev_dg_dgiauser_su",
        "collibraSystemName": "PUBLIC"
      }
    ]
  },
  "connectionDefinitions": [
    {
      "connectionName": "DG_con_standby_cmdm_clientors",
      "databaseName": "main",
      "schemaName": "dbo",
      "dialect": "oracle"
    },
    {
      "connectionName": "DG_con_dev_dg_dgiauser_su",
      "databaseName": "main",
      "schemaName": "dbo",
      "dialect": "oracle"
    }
  ]
}
```

Steps

1. Create a new JSON configuration file in the lineage harvester`config` folder.
If you have a data source with a large size for an Informatica Intelligent Cloud Services connection, consider creating more than one JSON file for the data source. Each JSON file must have a unique name. The contents in the JSON files are the same. In this way, you can avoid errors that might occur when the lineage harvester ingests metadata from one source with a large size.
2. Give the JSON file the same name as the value of the `Id` property in the lineage harvester configuration file.

Example If the value of the `Id` property in your lineage harvester configuration file is `iics-source-1`, then the name of your JSON file should be `iics-source-1.conf`.

Important Your JSON file must have the file extension `.conf`.

3. For each Informatica Intelligent Cloud Services connection, you can add the following content to the JSON file:

Property	Description	Required?
<code>collibraSystemNames</code>	This section contains the system information for Informatica Intelligent Cloud Services.	
<code>connections</code>	This section contains the system connection information. This is required to reference to the system or server of the connection.	
<code>connectionName</code>	The name of the connection. The name must match the System asset name in Data Catalog for stitching.	Yes

Property	Description	Required?
collibraSystemName	The system or server name of the data source. Use this property with the <code>useCollibraSystemName</code> property in the configuration file to override the default Collibra System asset name for this data source. Specify this property with the same name as the name of the System asset that you create when you prepare the physical data layer in Data Catalog. If you don't prepare the physical data layer, Collibra Data Lineage cannot stitch the data objects in your technical lineage to the assets in Data Catalog.	No
connectionDefinitions	This section contains the database, schema and dialect information for each connection in Informatica Intelligent Cloud Services. Note You can add connection information for each connection in the <code>connections</code> section.	
connectionName	The name of the connection. The name must match with the name in a connection name in the <code>connections</code> section. This property is required.	Yes
databaseName	The name of your database. The name must match the Database asset name in Data Catalog for stitching.	Yes
schemaName	The name of your schema. The name must match the Schema asset name in Data Catalog for stitching.	Yes

Property	Description	Required?
dialect	The dialect of the connection. Specify this property for Collibra Data Lineage to properly extract and parse queries that are related to this connection. You can enter one of the following values: ◦ bigquery ◦ db2 ◦ hana ◦ hive ◦ greenplum ◦ mssql ◦ mysql ◦ netezza ◦ oracle ◦ postgres ◦ redshift ◦ snowflake ◦ spark ◦ teradata	No

4. Save the configuration file.

The lineage harvester uses the [lineage harvester configuration file](#) to collect the Looker data objects and send them to the [Collibra Data Lineage service instance](#).

The <source ID> configuration file allows you to:

- Filter on the Looker folders from which you want to ingest metadata.
- If `useCollibraSystemName` in the lineage harvester configuration file is set to `true`, use the `collibraSystemName` property to specify the system name of databases in Looker.

Collibra Data Lineage uses the system names to match the structure of databases in Looker to assets in Data Catalog.

Example of <source ID> configuration file

```
{
  "Connections":{
    "connection-object1":{
      "dialect":"mssql",
      "schema":"mssql-schema-name",
      "dbname":"mssql-database-name",
      "collibraSystemName":"mssql-system-name"
    },
    "connection-object2":{
      "dialect":"oracle",
      "schema":"oracle-schema-name",
      "dbname":"oracle-database-name",
      "collibraSystemName":"oracle-system-name"
    }
  },
  "filters":[
    {
      "domainId":"605fa8ae-f8c6-4261-938b-8326e2806f3d",
      "description":"Databricks_folder",
      "folderIds":[
        "abc-123",
        "def-456"
      ]
    },
    {
      "domainId":"245bc5a4-4c30-44b5-8356-ddbe708b56d6",
      "description":"personal",
      "folderIds":[
        "hij-789*",
        "jkl-101112"
      ]
    }
  ]
}
```

Steps

1. Create a new JSON file in the lineage harvester **config** folder.
2. Give the JSON file the same name as the value of the `Id` property in the lineage harvester configuration file.

Example The value of the `Id` property in the lineage harvester configuration file is `looker-source-1`. As a result, the name of your JSON file should be *looker-source-1.conf*.

Important Your JSON file must have the file extension *.conf*.

3. For each database in Looker, add the following content to the JSON file:

Property	Description	Mandatory?
Connections	This section contains all Looker connections for which you want to create a technical lineage.	Yes
<connection name>	The name of a connection object in Looker.	Yes
dialect	The dialect of the supported data source in Looker.	No
schema	The name of the default schema of a supported data source in Looker. If the lineage harvester fails to find a specific schema, it uses the default schema.	No
dbname	The name of the database of a supported data source in Looker.	No

Property	Description	Mandatory?
collibraSystemName	The system or server name of a database. If you set the <code>useCollibraSystemName</code> property to <code>true</code> in your lineage harvester configuration file, but you either don't create a <source ID> configuration file, or don't specify a value for the <code>collibraSystemName</code> property in your <source ID> configuration file, the system name in the technical lineage is "DEFAULT". How do I configure this property if I have two databases with the same name? Let's assume you have two databases named Customers. When you prepare the physical data layer in Data Catalog, you create a System asset for each of these databases. Let's say you named them Customers-Europe and Customers-USA. You can then configure this property as follows. <pre> "connection-object1": { "dialect": "mssql", "schema": "mssql-schema-name", "dbname": "Customers", "collibraSystemName": "Customers-Europe" }, "connection-object2": { "dialect": "oracle", "schema": "oracle-schema-name", "dbname": "Customers", "collibraSystemName": "Customers-USA" } </pre>	Yes

Property	Description	Mandatory?
filters	Optionally, use this section to specify the Looker folders from which you want to ingest metadata. Note You can filter on Looker folders, but not on Looker data sets. That's because Looker data sets are linked directly to the server, instead of a folder, as shown in the Looker metadata overview. Looker data sets are ingested in the default domain, regardless of any filtering. Let's say, for example, you filter on folder B. A Looker Folder asset is created in the specified domain in Collibra, and all of the metadata in folder B is ingested. If folder B has a parent folder A, then a Looker Folder asset is created (in the domain specified for folder B) to preserve the hierarchy, but no metadata from folder A is ingested. You can specify more than one Looker folder for ingestion into a single domain in Collibra. Warning If you don't want to filter on Looker Folders, you must completely remove this filters section. Tip There are significant benefits to filtering by folder ID. For information, see the <code>filters > folderIds</code> property description.	No

Property	Description	Mandatory?										
	Tip You can use wildcards to capture multiple connection string combinations: Show me the supported wildcards <table><thead><tr><th>Pattern</th><th>Description</th></tr></thead><tbody><tr><td>*</td><td>Matches everything.</td></tr><tr><td>?</td><td>Matches any single character.</td></tr><tr><td>[seq]</td><td>Matches any character in "seq".</td></tr><tr><td>[!seq]</td><td>Matches any character not in "seq".</td></tr></tbody></table>	Pattern	Description	*	Matches everything.	?	Matches any single character.	[seq]	Matches any character in "seq".	[!seq]	Matches any character not in "seq".	
Pattern	Description											
*	Matches everything.											
?	Matches any single character.											
[seq]	Matches any character in "seq".											
[!seq]	Matches any character not in "seq".											
domainId	The unique resource ID of the domain (or domains), in Collibra, in which you want to ingest data objects from one or more Looker Folders. Tip You can find the domain ID by clicking the domain type. Then look in the URL of your browser to find the ID. The URL looks like https://<yourcollibrainstance>/domain/<domain ID>?<view>.											
description	Any description, as you see fit.											
folderNames	The name (or names) of the Looker Folders from which you want to ingest. Note You must specify either a folder name, a folder ID, or both.											

Property	Description	Mandatory?
folderIds	The ID (or IDs) of the Looker Folder you want to ingest. Note You must specify either a folder ID, a folder name, or both. Tip If you filter by folder ID, filtering is carried out via the API, instead of on the Collibra Data Lineage service instances. When you filter by folder ID, the lineage harvester accesses only the folders you specify via this property, and sends only that metadata to the Collibra Data Lineage service instance for processing and ingestion in Data Catalog. Conversely, if you filter by folder name (via the <code>folderNames</code> property), metadata from all Looker folders is sent to the Collibra Data Lineage service instance. Only then is filtering applied. What are the advantages? ◦ Faster integration testing, as you can filter on a single folder. ◦ Enhanced data security and privacy by not harvesting folders that contain sensitive information. ◦ Improved processing times by not including folders dedicated to, for example, development and testing. This is especially beneficial for organizations with a lot of data in Looker.	

4. Save the <source ID> configuration file.

The lineage harvester uses a lineage harvester [configuration file](#) to collect the Matillion data objects. It then sends the metadata to the Collibra Data Lineage service instance.

Example of the <source ID> configuration file

```
{
  "found_dbname=dbtest;found_hostname=test": {
    "collibraSystemName": "mssql-system-name"
  },
  "found_dbname=testsid;found_hostname=*": {
    "dbname": "oracle-database-name",
    "schema": "oracle-schema-name",
    "collibraSystemName": "oracle-system-name"
  }
}
```

Steps

1. Create a new JSON file in the lineage harvester **config** folder.
2. Name the JSON file as **<sourceId>.conf**, where <sourceId> is the same as the value of the `sourceId` property in the lineage harvester configuration file and the file extension must be `.conf`.

Example If the value of the `sourceId` property in the lineage harvester configuration file is `my-matillion`, the name of your JSON file must be **`my-matillion.conf`**.

3. Add the required content to the JSON file.

Property	Description	Mandatory?										
found_ dbname=<database name>;found_ hostname=<server name>	The information of the supported data sources in Matillion to be collected by Collibra Data Lineage. <database name> The database name in Matillion. <server name> The name of the server that the database is running on. You can specify found_ hostname=* to include all servers. Note Define a connection in the connection definitions only once; specifically, define a data source with the found_dbname and found_hostname properties specified only once in the connection definitions. If you define a connection multiple times, unexpected lineage and stitching issues might occur. Tip You can use wildcards to capture multiple connection string combinations: Show me the supported wildcards <table><tr><th>Pattern</th><th>Description</th></tr><tr><td>*</td><td>Matches everything.</td></tr><tr><td>?</td><td>Matches any single character.</td></tr><tr><td>[seq]</td><td>Matches any character in "seq".</td></tr><tr><td>[!seq]</td><td>Matches any character not in "seq".</td></tr></table>	Pattern	Description	*	Matches everything.	?	Matches any single character.	[seq]	Matches any character in "seq".	[!seq]	Matches any character not in "seq".	Yes
Pattern	Description											
*	Matches everything.											
?	Matches any single character.											
[seq]	Matches any character in "seq".											
[!seq]	Matches any character not in "seq".											

Property	Description	Mandatory?
dbname	The name of the database asset in Data Catalog. Specify this property with the database name that you created when you prepared the Data Catalog physical data layer. If you leave this property blank, the database is stitched to the database of DEFAULT in Data Catalog.	No
schema	The name of the schema asset in Data Catalog. Specify this property with the schema name that you created when you registered the data source. If you leave this property blank, the schema is stitched to the schema of DEFAULT in Data Catalog.	No

Property	Description	Mandatory?
collibraSystemName	The system or server name of the data source. Use this property with the <code>useCollibraSystemName</code> property in the configuration file to override the default Collibra System asset name for this data source. Specify this property with the same name as the name of the System asset that you create when you prepare the physical data layer in Data Catalog. If you don't prepare the physical data layer, Collibra Data Lineage cannot stitch the data objects in your technical lineage to the assets in Data Catalog. If you leave this property blank, the system is stitched to the system of DEFAULT in Data Catalog. If you are missing lineage or your lineage objects aren't stitching to Catalog assets in Data Catalog as you expect, ensure this property is specified properly. How do I configure this property if I have two databases with the same name? For example, if you have two databases named Customers. When you prepare the physical data layer in Data Catalog, you create a System asset for each of these databases. Let's say you named them Customers-Europe and Customers-USA. You can then configure this property as follows. <pre> { "found_dbname=dbtest;found_ hostname=test": { "dbname": "Customers", "collibraSystemName": "Customers-Europe" }, "found_dbname=testsid;found_ hostname=*": { "dbname": "Customers", "schema": "oracle- schema-name", </pre>	No

Property	Description	Mandatory?
	<pre> "collibraSystemName": "Customers-USA" } </pre> Warning The value of this property must exactly match (including for case-sensitivity) the name of your System asset in Collibra.	

4. Save the <source ID> configuration file.

The lineage harvester uses the [configuration file](#) to connect to MicroStrategy. You must also prepare a MicroStrategy <source ID> configuration file to:

- Specify the default domain, meaning the domain in Collibra in which the corresponding assets of MicroStrategy metadata will be ingested if domain mapping is not configured.

Note If you do configure domain mapping, the default domain is still the destination domain of the MicroStrategy Server asset.

- Optionally, specify from which MicroStrategy projects you want to ingest metadata, and into which domains you want to ingest the corresponding assets.
- Optionally, configure data source mapping, to map the name of a data source returned by the lineage harvester to the true name of the data source.

Note Mapping doesn't work for custom SQL.

Tip "<source ID>" refers to the value of the `Id` property in the lineage harvester configuration file.

Example

```
{
  "default_domain_id": "66ab7911-d84a-43d6-b9f7-40130jd84ds98",
  "filters": [
    {
      "domainId": "3ae946ec-9030-46b8-ad86-a217b8f99bab",
      "projectIDs": ["3FF7CF7D4C5A2AEBF14D8B9802698B27"]
    },
    {
      "domainId": "21cfb3ae-0616-478e-9060-114130540be5",
      "projectNames": ["Human Resources Analysis Module"]
    }
  ],
  "datasourceMapping": [
    {
      "found_datasource": "REDSHIFT",
      "found_project": "*",
      "mapping": {
        "dbname": "RD_pearl",
        "schema": "Default_North",
        "dialect": "spark",
        "colibraSystemName": "TV_dev"
      }
    }
  ]
}
```

Steps

1. Create a new JSON file in the lineage harvester`config` folder.
2. Give the JSON file the same name as the value of the `Id` property in the lineage harvester`configuration` file.

Example If the value of the `Id` property in the lineage harvester configuration file is `mstr-source-1`, then the name of your JSON file should be *mstr-source-1.conf*.

Important Your JSON file must have the file extension `.conf`.

Property	Description	Mandatory
default_domain_id	The domain in which you want the corresponding assets of MicroStrategy metadata to be ingested. Note If you configure filtering, only the MicroStrategy Server asset is ingested into this default domain.	Yes
filters	This section allows you to specify: • From which MicroStrategy projects you want to harvest metadata. • Into which domains in Collibra you want to ingest the corresponding assets. If you don't want to filter on projects, don't include this section in your <source ID> configuration file.	No

Property	Description	Mandatory
domainId	The unique resource ID of the domain (or domains) in Collibra in which you want to ingest the MicroStrategy assets. Tip If you use a <code>filters</code> section, you must include the <code>domainId</code> property in the section. If, by chance, you want to filter on certain projects, but you want to ingest all assets into the default domain, then the value of the <code>domainId</code> property must match the value of the <code>default_domain_id</code> property. Show me an example <pre> "default_domain_id": "1234567890", "filters": [{ "domainId": "1234567890", "projectNames": ["MicroStrategy Tutorial", "Testing_ MSTR"] }, </pre> How do I find a domain reference ID? Open the relevant domain in Collibra. The URL looks like: https://<yourcollibrainstance>/domain/22258f64-40b6-4b16-9c08-c95f8ec0da26?view=00000000-0000-0000-0000-0000000040001. In this example, the reference ID is in bold.	No
projectIds	The IDs of the MicroStrategy projects from which you want to ingest metadata.	No

Property	Description	Mandatory
projectNames	The project names of the MicroStrategy projects from which you want to ingest metadata.	No
datasourceMapping	This optional section allows you to configure data source mapping. Include this section only if you need to differentiate between multiple data sources that have the same name. Note Mapping doesn't work for custom SQL.	No
found_datasource	The name of the data source that was returned by the lineage harvester, as shown in the technical lineage. Note The data source name is case-sensitive.	Yes
found_project	The name of the project in which the data source information resides. You can specify an asterisk (*) to search for data source information across all projects.	Yes

Property	Description	Mandatory
mapping	Use this section to map the data source name that was returned by the lineage harvester to the true name of the data source. Example You have a Redshift data source named "RD_pearl", but the lineage harvester has returned the name "Redshift_connection". You can configure the <code>datasourceMapping</code> section as follows: <pre> { "datasourceMapping": [{ "found_ datasource": "REDSHIFT", "found_ project": "*", "mapping": { "dbname": "RD_ pearl", "collibraSystemName": "TV_dev" } }] }</pre>	Yes
dbname	The name of the database to which you want to map the found data source.	Yes
schema	The name of the schema in MicroStrategy.	No
dialect	The dialect of the data source in MicroStrategy.	No

Property	Description	Mandatory
collibraSystemName	The system or server name of a database. If you set the <code>useCollibraSystemName</code> property to <code>true</code> in your lineage harvester configuration file, but you either don't create a <code><source ID></code> configuration file, or don't specify a value for the <code>collibraSystemName</code> property in your <code><source ID></code> configuration file, the system name in the technical lineage is "DEFAULT". If you set the <code>useCollibraSystemName</code> property to <code>false</code> in your lineage harvester configuration file, leave this property empty as follows: <code>"collibraSystemName": ""</code>. How do I configure this property if I have two databases with the same name? Let's assume that you have a data source named Customers. You use this data source connection in two different projects, Project_A and Project_B, but they are actually two different databases. When you prepare the physical data layer in Data Catalog, you create a System asset for each of these databases. Let's say you named them Customers-North and Customers-South. You can then configure this property as follows. <pre> "datasourceMapping": [{ "found_ datasource": "Customers", "found_ project": "Project_A", "mapping": { "dbname": "Customers", </pre>	Yes

Property	Description	Mandatory
	<pre> "collibraSystemName": "Customers_North" }, { "found_ datasource": "Customers", "found_ project": "Project_B", "mapping": { "dbname": "Customers", "collibraSystemName": "Customers_South" } }] </pre> Warning The values of this property must exactly match the name of your System asset in Collibra.	

3. Save the <source ID> configuration file.

The lineage harvester uses a [lineage harvester configuration file](#) to collect the Power BI data objects. It then sends the metadata to the [Collibra Data Lineage service instances](#).

The <source ID> configuration file allows you to:

- Map the names of the server, database and schema that were collected by the lineage harvester to their true names.

Note Mapping doesn't work for custom SQL.

- Configure workspace filtering.

Tip We highly recommend that you read through [Filtering Power BI workspaces](#) for important information and guidance before configuring your filters.

- If `useCollibraSystemName` in the lineage harvester configuration file is set to `true`, use the `collibraSystemName` property to specify the system name of databases in Power BI. Collibra Data Lineage uses the system names to match the structure of databases in Power BI to assets in Data Catalog.

Example of the <source ID> configuration file

```
{
  "found_dbname=databasename1;found_hostname=*;found_schema=schema1":
  {
    "dbname": "mssql-database-name",
    "schema": "mssql-schema-name",
    "dialect": "mssql",
    "collibraSystemName": "mssql-system-name"
  },
  "found_dbname=databasename2;found_hostname=server-
name.onmicrosoft.com;found_schema=schema2": {
    "dbname": "oracle-database-name",
    "schema": "oracle-schema-name",
    "dialect": "oracle",
    "collibraSystemName": "oracle-system-name"
  },
  "filters":[
    {
      "domainId": "<domain-ref-id>",
      "description": "FirstFilter",
      "workspaceNames": "*",
      "excludeWorkspaceIds": ["workspaceC", "workspaceD"]
    },
    {
      "domainId": "<domain-ref-id>",
      "description": "SecondFilter",
      "workspaceNames": ["workspace3", "workspace4"],
      "capacityIds": ["id1","id2"]
    }
  ]
}
```

Steps

1. Create a new JSON file in the lineage harvester **config** folder.
2. Give the JSON file the same name as the value of the `sourceId` property in the lineage harvester configuration file.

Example The value of the `sourceId` property in the lineage harvester configuration file is `power-bi-source-1`. Therefore, the name of your JSON file should be *power-bi-source-1.conf*.

Important Your JSON file must have the file extension *.conf*.

3. For each database in Power BI, add the following content to the JSON file:

Property	Description	Mandatory?
found_dbname=<database name>;found_hostname=<server name>;found_schema=<schema name>	The database information of supported data sources in Power BI that is typically collected by the lineage harvester. Specify the name of the database (found_dbname), on which server a database is running (found_hostname), and optionally, the name of the schema (found_schema). You then use the child properties to map the names collected by the lineage harvester to the true names. Important Schema mapping is available for schemas that come from Power Query connections. It is not available, however, if a Power Query connection is created with SQL (or MDX) statements and the schema is specified in those statements. Important The keys that you specify must be unique. Show me an example of a duplicate key that would result in an error The following configuration would result in an error, because the key found_dbname=databasename1; found_hostname=*; found_schema=schema1 is specified twice. <pre> { "found_dbname=databasename1; found_hostname=*; found_schema=schema1": { "dbname": "mssql-database-name", "schema": "mssql-</pre>	Yes

Property	Description	Mandatory?
	<pre> schema-name", "dialect": "mssql", "collibraSystemName": "mssql-system-name" }, "found_ dbname=databasename1;fo und_hostname=*;found_ schema=schema1": { "dbname": "oracle- database-name", "schema": "oracle- schema-name", "dialect": "oracle", "collibraSystemName": "oracle-system-name" }, "filters":[{ "domainId": "<domain- ref-id>", "description": "FirstFilter", "workspaceNames": "*", "excludeWorkspaceIds": ["workspaceC", "workspaceD"] }, { "domainId": "<domain- ref-id>", "description": "SecondFilter", "workspaceNames": ["workspace3", "workspace4"], "capacityIds": ["id1","id2"] }] } </pre>	

Property	Description	Mandatory?										
	Tip You can use wildcards to capture multiple connection string combinations: Show me the supported wildcards <table><tr><th>Pattern</th><th>Description</th></tr><tr><td>*</td><td>Matches everything.</td></tr><tr><td>?</td><td>Matches any single character.</td></tr><tr><td>[seq]</td><td>Matches any character in "seq".</td></tr><tr><td>[!seq]</td><td>Matches any character not in "seq".</td></tr></table>	Pattern	Description	*	Matches everything.	?	Matches any single character.	[seq]	Matches any character in "seq".	[!seq]	Matches any character not in "seq".	
Pattern	Description											
*	Matches everything.											
?	Matches any single character.											
[seq]	Matches any character in "seq".											
[!seq]	Matches any character not in "seq".											
dbname	The name of the database of a supported data source in Power BI.	No										
schema	The name of the schema of a supported data source in Power BI. If the lineage harvester fails to find a specific schema, it uses the schema you specify in this property. Important Schema mapping is available for schemas that come from Power Query connections. It is not available, however, if a Power Query connection is created with SQL (or MDX) statements and the schema is specified in those statements.	No										

Property	Description	Mandatory?
dialect	The dialect of the supported data source in Power BI. Tip You can enter one of the following values: • <i>azure</i>, for an Azure SQL Server data source. • <i>bigquery</i>, for a Google BigQuery data source. • <i>mssql</i>, for a Microsoft SQL Server data source. • <i>oracle</i>, for an Oracle data source. • <i>redshift</i>, for an Amazon Redshift data source. • <i>snowflake</i>, for a Snowflake data source. • <i>sybase</i>, for a Sybase data source.	No

Property	Description	Mandatory?
collibraSystemName	The system or server name of a database. If you set the <code>useCollibraSystemName</code> property to <code>true</code> in your lineage harvester configuration file, but you either don't create a <code><source ID></code> configuration file, or don't specify a value for the <code>collibraSystemName</code> property in your <code><source ID></code> configuration file, the system name in the technical lineage is "DEFAULT". How do I configure this property if I have two databases with the same name? Let's assume you have two databases named Customers. When you prepare the physical data layer in Data Catalog, you create a System asset for each of these databases. Let's say you named them Customers-Europe and Customers-USA. You can then configure this property as follows. <pre> "found_ dbname=databasename1;found_ hostname=*;found_ schema=schema1":{ "dbname": "Customers", "schema": "mssql-schema- name", "dialect": "mssql", "collibraSystemName": "Customers-Europe" }, "found_ dbname=databasename2;found_ hostname=server- name.onmicrosoft.com;found_ schema=schema2": { "dbname": "Customers", "schema": "oracle-schema- name", "dialect": "oracle", "collibraSystemName": "Customers-USA" }, </pre>	Yes (unless you are using the <code><source ID></code> file to provide the true system names of ODBC databases in Power BI.)

Property	Description	Mandatory?
	Warning The value of this property must exactly match (including for case-sensitivity) the name of your System asset in Collibra. Important If you are using a <source ID> configuration file for the purpose of providing the true system name of an ODBC database in Power BI, you are not required to: - Set the <code>useCollibraSystemName</code> property in the lineage harvester configuration file to <code>true</code>. - Specify a Collibra system name in the <source ID> configuration file. However, if the <code>useCollibraSystemName</code> property is set to <code>true</code> in the lineage harvester configuration file, then you must specify a Collibra system name in the <source ID> configuration file.	

Property	Description	Mandatory?										
filters	This section allows you to specify the Power BI workspaces from which you want to ingest metadata. The filters work as "workspace AND workspace AND capacity AND capacity", meaning that if you specify a capacity, all of the workspaces in that capacity are also ingested. Warning If you don't want to specify the Power BI workspaces from which to ingest, you must completely remove this filters section. Tip You can use wildcards to capture multiple connection string combinations: Show me the supported wildcards <table><tr><th>Pattern</th><th>Description</th></tr><tr><td>*</td><td>Matches everything.</td></tr><tr><td>?</td><td>Matches any single character.</td></tr><tr><td>[seq]</td><td>Matches any character in "seq".</td></tr><tr><td>[!seq]</td><td>Matches any character not in "seq".</td></tr></table>	Pattern	Description	*	Matches everything.	?	Matches any single character.	[seq]	Matches any character in "seq".	[!seq]	Matches any character not in "seq".	No
Pattern	Description											
*	Matches everything.											
?	Matches any single character.											
[seq]	Matches any character in "seq".											
[!seq]	Matches any character not in "seq".											

Property	Description	Mandatory?
domainId	The unique resource ID of the domain (or domains), in Collibra Data Intelligence Cloud, in which you want to ingest the Power BI assets. Tip You can find the domain ID by clicking the domain type. Then look in the URL of your browser to find the ID. The URL looks like <code>https://<yourcollibrainstance>/domain/<domain ID>?<view></code>.	Yes
description	Any description, as you see fit.	Yes
workspaceNames	The names of Power BI workspaces from which you want to ingest metadata. Important Any meta-characters in the name of a workspace must be enclosed in square brackets "[]". For example, a workspace with the name "Sale and Marketing [automobiles]" should be formatted as follows: <pre>Sale and Marketing [[]automobiles[]]</pre>	No
workspaceIds	The IDs of Power BI workspaces from which you want to ingest metadata. Tip We highly recommend that you read through Filtering Power BI workspaces for important information and guidance before configuring your filters.	No
capacityNames	The names of capacities on which you want to filter.	No

Property	Description	Mandatory?
capacityIds	The IDs of capacities on which you want to filter. Warning Any letters in a capacity ID must be in upper case.	No
excludeWorkspaceNames	The names of Power BI workspaces that you want to exclude from the ingestion job. This is useful if you want to exclude, for example, dedicated development and testing workspaces. Note The metadata of inactive and personal workspaces is not harvested or uploaded to the Collibra Data Lineage service instance. An inactive workspace is one for which no reports or dashboards have been viewed in the past 60 days. My workspace is the personal workspace for any Power BI customer to work with their own, personal content. For complete details on the advantages, limitations and configuration considerations of this property, see Filtering Power BI workspaces.	No
excludeWorkspaceIds	The IDs of Power BI workspaces that you want to exclude from the ingestion job. This is useful if you want to exclude, for example, dedicated development and testing workspaces. For complete details on the advantages, limitations and configuration considerations of this property, see Filtering Power BI workspaces.	No

4. Save the <source ID> configuration file.

When you create technical lineage for Snowflake by using the SQL-API ingestion method, you can create a <source ID> configuration file to configure the metadata that Collibra Data Lineage collects

Example of the <source ID> configuration file

```
{
  "displaySampleQueries": true|false
  "analyzeTemporaryTables": true|false
}
```

Steps

1. Create a new JSON file in the lineage harvester **config** folder.
2. Name the JSON file as **<sourceId>.conf**, where <sourceId> is the same as the value of the `sourceId` property in the lineage harvester configuration file and the file extension must be `.conf`.

Example If the value of the `sourceId` property in the lineage harvester configuration file is `my-adf`, the name of your JSON file must be **my-adf.conf**.

3. For each database in Snowflake, add the following content to the JSON file:

Property	Description	Required?
displaySampleQueries	Indicates whether to display transformations with a question mark (?) or with actual values from queries in the Source code pane in the technical lineage graph. For example, you can choose to display <code>WHERE amount < 100</code> or <code>WHERE amount < ?</code>. Specify one of the following values: <code>true</code> Actual values from queries are displayed. <code>false</code> A question mark (?) is displayed. This is the default value.	No
analyzeTemporaryTables	Indicates whether to parse the CREATE TEMPORARY TABLE statement in the ingested queries. Specify one of the following values: <code>true</code> Collibra Data Lineage examines the queries and parses the CREATE TEMPORARY TABLE statement when the following conditions are met: ◦ The query starts with the CREATE TEMPORARY TABLE statement. ◦ Collibra Data Lineage did not encounter the CREATE TEMPORARY TABLE statement before this query. <code>false</code> Collibra Data Lineage does not examine or parse the CREATE TEMPORARY TABLE statement in the ingested queries. This is the default value.	No

4. Save the <source ID> configuration file.

The [lineage harvester](#) uses the lineage harvester [configuration file](#) to collect the SQL Server Reporting Services (SSRS) and Power BI Report Server (PBRs) data objects

and send them to the [Collibra Data Lineage service](#).

The <source ID> configuration file allows you to:

- If `useCollibraSystemName` in the lineage harvester configuration file is set to `true`, use the `collibraSystemName` property to specify the system name of databases in SSRS and PBRS.
- Provide additional information about databases in SSRS and PBRS, which is necessary if the databases do not contain all information to process the SQL source code correctly.

Example <source ID> configuration file

```
{
  "DataSources": {
    "Redshift": {
      "dbname": "redshift-database-name",
      "schema": "redshift-schema-name",
      "dialect": "redshift",
      "collibraSystemName": "redshift-system-name"
    },
    "Oracle": {
      "dbname": "oracle-database-name",
      "schema": "oracle-schema-name",
      "dialect": "oracle",
      "collibraSystemName": "oracle-system-name"
    }
  },
  "CustomDataSources": {
    "/path to report/custom data source name": {
      "dbname": "mssql-database-name",
      "dialect": "mssql"
    }
  }
}
```

Steps

1. Create a new JSON file in the lineage harvester **config** folder.
2. Give the JSON file the same name as the value of the `Id` property in the lineage harvester configuration file.

Example The value of the `Id` property in the lineage harvester configuration file is `ssrs-source-1`. As a result, the name of your JSON file should be `ssrs-source-1.conf`.

Important Your JSON file must have the file extension `.conf`.

3. For each database in SSRS and PBRs, add the following content to the JSON file:

Property	Description	Required?
DataSources	This section contains all connections for which you want to create a technical lineage. The <code>DataSources</code> section refers to shared data sources in SSRS and PBRs. For more information about shared data sources, see the Microsoft documentation .	Yes
<data source type>	The path of a connection object in SSRS and PBRs.	Yes
dbname	The name of the database of a supported data source in SSRS and PBRs.	No
schema	The name of the default schema of a supported data source in SSRS and PBRs.	No
dialect	The dialect of the supported data source in SSRS and PBRs.	No

Property	Description	Required?
collibraSystemName	The system or server name of the database. If you set the <code>useCollibraSystemName</code> property to <code>true</code> in your lineage harvester configuration file, but you either don't create a <source ID> configuration file, or don't specify a value for the <code>collibraSystemName</code> property in your <source ID> configuration file, the system name in the technical lineage is "DEFAULT". How do I configure this property if I have two databases with the same name? Let's assume you have two databases named Customers. When you prepare the physical data layer in Data Catalog, you create a System asset for each of these databases. Let's say you named them Customers-Europe and Customers-USA. You can then configure this property as follows. <pre> "Redshift": { "dbname": "Customer", "schema": "redshift- schema-name", "dialect": "redshift", "collibraSystemName": "Customers-Europe" }, "Oracle": { "dbname": "Customer", "schema": "oracle-schema- name", "dialect": "oracle", "collibraSystemName": "Customers-USA" } </pre>	Yes

Property	Description	Required?
CustomDataSources	You can use custom data processing extensions that are used to support embedded data sources of which the data source definition is specified locally in a report or embedded data set. The <code>CustomDataSources</code> section refers to embedded data sources in SSRS and PBRs. For more information about embedded data sources, see the Microsoft documentation.	No
<path to report>/<custom data source name>	The full path to the report and the custom data source name. You can use wildcards to match multiple folders, reports or data sets. The connection information in this section is used to add missing information or to overwrite parsed information.	No
dbname	The name of the database of a custom data source in SSRS and PBRs..	No
schema	The name of the schema of a custom data source in power. If you don't provide the schema name, the default schema is used.	No

Property	Description	Required?
dialect	The dialect of the custom data source in SSRS and PBRs.. Tip You can enter one of the following values: ◦ <i>azure</i>, for an Azure SQL Server data source. ◦ <i>bigquery</i>, for a Google BigQuery data source. ◦ <i>db2</i>, for an IBM DB2 data source. ◦ <i>hana</i>, for a SAP Hana data source. ◦ <i>hive</i>, for a HiveQL data source. ◦ <i>greenplum</i>, for a Greenplum data source. ◦ <i>mssql</i>, for a Microsoft SQL Server data source. ◦ <i>mysql</i>, for a MySQL data source. ◦ <i>netezza</i>, for a Netezza data source. ◦ <i>oracle</i>, for an Oracle data source. ◦ <i>postgres</i>, for a PostgreSQL data source. ◦ <i>redshift</i>, for an Amazon Redshift data source. ◦ <i>snowflake</i>, for a Snowflake data source. ◦ <i>spark</i>, for a Spark SQL data source. ◦ <i>sybase</i>, for a Sybase data source. ◦ <i>teradata</i>, for a Teradata data source.	No

4. Save the <source ID> configuration file.

The lineage harvester uses a lineage harvester [configuration file](#) to collect the SQL Server Integration Services data objects. It then sends the metadata to the Collibra Data Lineage service instance.

Example of the <source ID> configuration file

```
{
  "ConnectionStringRegexTranslation": {
```

```

    "Data Source=dhb-sql-prod;Initial Catalog=SFG_repl
staging;Provider=SQLNCLI11;Integrated Security=SSPI.*": {
      "dbname": "DATAHUB",
      "schema": "DBO",
      "dialect": "mssql",
      "collibraSystemName" : "WAREHOUSE"
    },
    "Server=sb-dhub;User ID=SYS_USER;Initial
Catalog=STAGEDB;Port=6306.*": {
      "dbname": "STAGEDB",
      "schema": "STAGE_OWNER",
      "dialect": "sybase",
      "collibraSystemName" : ""
    }
  }
}

```

Steps

1. Create a new JSON file in the lineage harvester **config** folder.
2. Name the JSON file as **<sourceId>.conf**, where **<sourceId>** is the same as the value of the `sourceId` property in the lineage harvester configuration file and the file extension must be `.conf`.

Example If the value of the `sourceId` property in the lineage harvester configuration file is `my-ssis`, the name of your JSON file must be **my-ssis.conf**.

3. For each database, add the required content to the JSON file.

Property	Description
ConnStringRegexTranslation	The parent element that opens the connection definitions.

Property	Description
<regular expression>	A regular expression that must match one or more connection strings. Note Important considerations: - By default, the regular expression is not case sensitive. As a consequence, a regular expression can match with connection strings containing uppercase characters or lowercase characters. - The connection string is part of the SSIS connection manager. - SSIS connection managers are included in an SSIS package files (DTSX) or in connection manager files (CONMGR). Example Regular expression: <code>Server=sb-dhub;User ID=SYB_USER2;Initial Catalog=STAGEDB;Port=6306.*</code> Explanation: The first section, up to <code>.*</code>, is a literal, but not case-sensitive, match of the characters. The dot (<code>.</code>) can match any single character. The asterisk (<code>*</code>) means zero or more of the previous, in this case any character. Match: Any connection string that starts with <code>Server=sb-dhub;User ID=SYB_USER2;Initial Catalog=STAGEDB;Port=6306.</code> Example: <code>Server=sb-dhub;User ID=SYB_USER2;Initial Catalog=STAGEDB;Port=6306;Persist Security Info=True;Auto Translate=False;.</code>
dbname	The name of your database, to which the data source connection refers.
schema	The name of your schema, to which the regular expression refers.

Property	Description
dialect	The dialect of the referenced database. Tip You can enter one of the following values: ◦ <code>azure</code>, for an Azure SQL Server data source. ◦ <code>bigquery</code>, for a Google BigQuery data source. ◦ <code>db2</code>, for an IBM DB2 data source. ◦ <code>hana</code>, for an SAP HANA data source. ◦ <code>hana-cviews</code>, for getting lineage from calculated views in an SAP HANA Classic on-premises data source. ◦ <code>hana-cviews-v2</code>, for getting lineage from calculated views in an SAP HANA Cloud/Advanced data source. Important To get technical lineage including calculated views, you must harvest SAP HANA by adding two Technical Lineage for SqlDirectory capabilities with the Shared Storage connections. In one capability, specify the <code>hana</code> dialect, and in the other, specify the <code>hana-cviews</code> or <code>hana-cviews-v2</code> dialect. ◦ <code>hive</code>, for a HiveQL data source. ◦ <code>greenplum</code>, for a Greenplum data source. ◦ <code>mssql</code>, for a Microsoft SQL Server data source. ◦ <code>mysql</code>, for a MySQL data source. ◦ <code>netezza</code>, for a Netezza data source. ◦ <code>oracle</code>, for an Oracle data source. ◦ <code>postgres</code>, for a PostgreSQL data source. ◦ <code>redshift</code>, for an Amazon Redshift data source. ◦ <code>snowflake</code>, for a Snowflake data source. ◦ <code>spark</code>, for a Spark SQL data source. ◦ <code>sybase</code>, for a Sybase data source. ◦ <code>teradata</code>, for a Teradata data source.

Property	Description
collibraSystemName	The system or server name of the data source. This property is optional. Use this property with the <code>useCollibraSystemName</code> property in the configuration file to override the default Collibra System asset name for this data source. Specify this property with the same name as the name of the System asset that you create when you prepare the physical data layer in Data Catalog. If you don't prepare the physical data layer, Collibra Data Lineage cannot stitch the data objects in your technical lineage to the assets in Data Catalog.

4. Save the <source ID> configuration file.

The lineage harvester uses the [configuration file](#) to connect to Tableau. You are not required to create a <source ID> configuration file, but you need one if you want to:

- Define your Tableau operating model.
- Provide additional information about databases and files in Tableau. For example, you can define the system name of files and connectors in Tableau.
- Use the [hostnameMapping](#) property to map the database, schema or system names that were returned by the Tableau APIs to the actual names of the assets in Data Catalog. For complete information, go to [Tableau hostname, schema, and system name mapping](#).

Note Mapping doesn't work for custom SQL.

- Define in which domains in Collibra you want to ingest assets from your Tableau sites and projects. See the [domainMapping](#) and [filters](#) properties.

Tip "<source ID>" refers to the value of the `Id` property in the lineage harvester configuration file.

Example <source ID> configuration file

```
{
  "collibraSystemNames": {
    "files": [
      {
        "filePath": "C:\\\\ProgramData\\\\Tableau\\\\Tableau
Server\\\\data\\\\files\\\\sample.xls",
        "collibraSystemName": "sample-files"
      }
    ],
    "connectors": [
      {
        "connectorUrl": "tableau-server-connector-url.com",
        "collibraSystemName": "Oracle-connector"
      }
    ],
    "cloudFiles": [
      {
        "name": "file-name",
        "collibraSystemName": "FILE"
      }
    ]
  },
  "hostnameMapping": {
    "found_dbname=*;found_hostname=abc.net;found_schema=DEFAULT": {
      "dbname": "CData",
      "schema": "Jan_1_2022",
      "dialect": "spark",
      "collibraSystemName": "TV_testing"
    }
  },
  "filters": {
    "sites": {
      "site_name": "domain_id"
    },
    "projects": {
      "site_name2 > project_name2": "domain-reference-id2",
      "site_name3 > project_name3 > subproject_name": "domain-
reference-id2"
    }
  }
}
```

Steps

1. Create a new JSON file in the lineage harvester`config` folder.
2. Give the JSON file the same name as the value of the `Id` property in the lineage harvester`configuration` file.

Example If the value of the `Id` property in the lineage harvester configuration file is `tableau-source-1`, then the name of your JSON file should be `tableau-source-1.conf`.

Important Your JSON file must have the file extension `.conf`.

3. For each database in Tableau, add the following content to the JSON file:

Tip You can use wildcards to capture multiple string combinations for any of these properties.

Show me the supported wildcards

Pattern	Description
*	Matches everything.
?	Matches any single character.
[seq]	Matches any character in "seq".
[!seq]	Matches any character not in "seq".

Property	Description
collibraSystemNames	This section contains the system information for different Tableau data sources. Depending on the kind of data source or connection, you have to specify how to connect to this data source. Tip For more information, see the Tableau documentation. We also recommend to check the list of supported connectors in Tableau.
files	This section contains connection information to one or more files in Tableau. Tip If you do not have files in Tableau, you can remove this section.

Property	Description
filePath	The full path to the file. For example, the path to a JSON file.
colibraSystemName	The system name of the file.
connectors	This section contains connection information to one or more connectors in Tableau. Tip ◦ If you do not have connectors in Tableau, you can remove this section. ◦ The values that you specify for this property are not case-sensitive.
connectorUrl	The URL of the connector. For example, the URL to Google Analytics.
colibraSystemName	The system name of the connector.
cloudFiles	This section contains connection information to one or more cloud files in Tableau's input data. Tip If you do not have cloud files in Tableau, you can remove this section.
name	The name of the file. For example, the name of a Zendesk file.
colibraSystemName	The system name of the cloud file.

Property	Description
hostnameMapping	This section allows you to map Tableau technical database, server and schema names to the respective real names, to preserve stitching. Warning ◦ <code>hostnameMapping</code> replaces the following deprecated properties, which have been removed from this topic: ■ The <code>databaseMapping</code> property. ■ The <code>databases</code> sub-section of the <code>collibraSystemNames</code> section. <code>hostnameMapping</code> must not be used in combination with either of these properties. If you use the <code>hostnameMapping</code> section, you can still use the <code>collibraSystemName</code> property in conjunction with the <code>files</code>, <code>connectors</code> or <code>cloudfiles</code> sub-sections.
found_ dbname=<database name>;found_ hostname=<server name>;found_ schema=<schema name>	The database information of supported data sources in Tableau that is typically collected by the lineage harvester. It allows you to specify the name of the database (<code>found_dbname</code>), on which server a database is running (<code>found_hostname</code>), and optionally, the name of the schema (<code>found_schema</code>).
dbname	The name of the database of a supported data source in Tableau.
schema	The name of the default schema of a supported data source in Tableau. If the lineage harvester fails to find a specific schema, it uses the default schema.

Property	Description
dialect	The dialect of the supported data source in Tableau. You don't have to specify a dialect; it will automatically be detected. If, however, you are using a dialect that is not supported, you can use this property to specify a supported dialect that is a close comparison. That way, most of your queries will be detected and processed. Tip You can enter one of the following values: ◦ <i>redshift</i>, for an Amazon Redshift data source. ◦ <i>azure</i>, for an Azure SQL Server data source. ◦ <i>bigquery</i>, for a Google BigQuery data source. ◦ <i>greenplum</i>, for a Greenplum data source. ◦ <i>hive</i>, for a HiveQL data source. ◦ <i>oracle</i>, for an Oracle data source. ◦ <i>postgres</i>, for a PostgreSQL data source. ◦ <i>mssql</i>, for a Microsoft SQL Server data source. ◦ <i>mysql</i>, for a MySQL data source. ◦ <i>netezza</i>, for a Netezza data source. ◦ <i>hana</i>, for a SAP HANA data source. ◦ <i>spark</i>, for a Spark SQL data source. ◦ <i>sybase</i>, for a Sybase data source. ◦ <i>teradata</i>, for a Teradata data source.

Property	Description
filters	This section defines: ◦ From which Tableau sites and projects you want to harvest metadata. ◦ Into which domains in Collibra you want to ingest the corresponding assets. Filtering is transitive, which means that all resources in a specified project, such as Tableau workbooks and all sub-projects, are ingested. Tableau assets that are not mapped to the specified domains, for example the Tableau Server assets and the parent projects (if you specify their sub-projects), are ingested in the default domain. Important ◦ Filtering does not affect the amount of raw metadata that is harvested from Tableau and sent to the Collibra Data Lineage service instance. Rather, it determines which metadata is ingested as assets in Data Catalog. ◦ The <code>domainMapping</code> and <code>filters</code> sections are mutually exclusive. Do not include both <code>domainMapping</code> and <code>filters</code> sections in your JSON file.

Property	Description
	Tip ◦ If you want to ingest all of the projects in a Tableau site into multiple domains in Collibra, use the <code>domainMapping</code> section. ◦ If you want to ingest all of the projects in a Tableau site into the default domain, use only the <code>domainID</code> property in the lineage harvester configuration file. The <code>domainID</code> property represents the default domain. ◦ If you want to ingest all of the projects in a Tableau site into a single domain in Collibra, use site filtering. ◦ If you want to ingest metadata from only some of the projects in a Tableau site, use project filtering. ◦ You can use site filtering and project filtering together: ◦ If filtering on the same site, this "filtering" is actually domain mapping, because nothing is filtered out. The contents of the projects are ingested in the specified domains, and the rest of the contents of the site are ingested in a different, specified domain. ◦ If you are site filtering on a specific site and project filtering a different site, then site filtering is again a form of domain mapping, and the filtered projects are ingested in their specified domains. ◦ If your lineage harvester configuration file includes sites that are not mentioned in the <code>filters</code> section of your <code><source ID></code> configuration file, those sites are ingested in the default domain.

Property	Description
sites	The Tableau sites to be ingested and the domain in which you want to ingest metadata from the Tableau sites. Tip If you have only one Tableau site, do not include a <code>sites</code> section in your <source ID> file. Instead, use a <code>projects</code> section, to filter on Tableau projects. Include a <code>sites</code> section only if all of the following are true: ◦ You have more than one Tableau site. ◦ You want to ingest all of the metadata from only one Tableau site into a single domain in Collibra. ◦ The domain into which you want to ingest is not the default domain, meaning the domain specified in the <code>domainId</code> property in your lineage harvester configuration file.

Property	Description
site_name: domain_id	site_name The name of the site to be ingested. The site name is case-sensitive. domain_id The unique reference ID of the domain in Collibra in which you want to ingest metadata. The domain ID is case-sensitive. To ingest all metadata from a Tableau site in the specified domain, specify the site name and a separate domain ID for each site that you list on the <code>siteIds</code> property in the lineage harvester configuration file for Tableau. If the <code>site_name</code> or <code>domain_id</code> property is not specified for a site, the metadata from the site is ingested in the default domain. How do I find a domain reference ID? Open the relevant domain in Collibra. The URL looks like: <code>https://<yourcollibrainstance>/domain/22258f64-40b6-4b16-9c08-c95f8ec0da26?view=00000000-0000-0000-0000-0000000040001</code>. In this example, the reference ID is in bold. Show me the example <pre>{ "filters":{ "sites":{ "Training":"ca60b822-781b-4b3a-b44d-f65bd107ff92" }, "projects":{ "Testing > Databricks":"e8f4e4a8-4062-4a33-9b44-3ce3e18e4e22", "Product Demo > Customer Insights":"a305e6f7-7a49-49aa-aa85-41b1e689121b" } } }</pre>

Property	Description
projects	The Tableau projects to be ingested and the domain in which you want to ingest metadata from the Tableau projects or sub-projects. Tip Project filtering is not relevant for those who have an Explorer role in Tableau, because Explorers need to configure permissions for each data object in Tableau that they want to ingest. As the Administrator role has access to all data objects, project filtering allows Administrators to specify which projects to ingest.
site_name > project_name : domain_id	The <code>site_name</code> should be the Tableau site name. The <code>project_name</code> should be the Tableau project name. The <code>domain_id</code> should be the unique reference ID of the domain in Collibra in which you want to ingest metadata. When you specify the site and project names, the following rules apply: ◦ Add spaces before and after <code>></code>. The spaces are separators between the site and project. ◦ Specify the full exact site and project names. ◦ The values are case-sensitive. When you specify a Tableau project, all assets in the project are ingested in the specified domain. If you want to ingest assets from different Tableau projects in one domain, you can specify the same value for <code>domain id</code> for different projects. Example <pre>"Collibra_tab_partner_site > JB_Test_2812": "d224a1a5-43b4-43b2-8df0-ddf8f2726b82"</pre>

Property	Description
<code>site_name > project_name > sub-project_name : domain_id</code>	The <code>site_name</code> should be the Tableau site name. The <code>project_name</code> should be the Tableau project name. Optionally, use <code>sub-project_name</code> to specify the Tableau sub-project name. The <code>domain_id</code> property should be the unique reference ID of the domain in Collibra in which you want to ingest metadata. When you specify the site, project and sub-project names, the following rules apply: ◦ Add spaces before and after <code>></code>. The spaces are separators between the site and project. ◦ Specify the full exact site and project names. ◦ The values are case-sensitive. Example <pre>"Collibra_tab_partner_site > JB_Test_2812 > ProjectJJ2": "d224a1a5-43b4-43b2-8df0- ddf8f2726b82"</pre>

Property	Description
domainMapping	This section defines in which domains in Collibra you want to ingest assets from your Tableau sites and Tableau projects. Domain mapping is transitive, meaning that all resources, such as Tableau workbooks and data attributes in a parent Tableau site, project or sub-project, are ingested in the same domain as the parent. Important The <code>domainMapping</code> and <code>filters</code> sections are mutually exclusive. Do not include both <code>domainMapping</code> and <code>filters</code> sections in your JSON file. Tip ◦ If you want to ingest all of the projects in a Tableau site into multiple domains in Collibra, use this <code>domainMapping</code> section. ◦ If you want to ingest all of the projects in a Tableau site into the default domain, use only the <code>domainID</code> property in the lineage harvester configuration file. The <code>domainID</code> property represents the default domain. Note Tableau assets that are not mapped to specific domains via this <code>domainMapping</code> section, for example Tableau Server assets, are ingested in that default domain. ◦ If you want to ingest all of the projects in a Tableau site into a single domain in Collibra, use site filtering. ◦ If you want to ingest metadata from only some of the projects in a Tableau site, use project filtering. Show me an example Let's say that you have a Tableau site named "Site-1". You want to ingest all Tableau projects in "Site-1" in a domain named "Domain-1" in Collibra, with the exception of one

Property	Description
	Tableau project named "Project-Default", which you want to ingest in "Domain-2". You should configure the <code>domainMapping</code> section as follows. <pre>"domainMapping": { "<Site-1>": "reference-id-of-Domain-1", "<Site-1> > <Project-Default>": "reference-id-of-Domain-2" }</pre> If you want to specify a domain for a sub-project of "Project-Default", use the <code><site name> > <project name> > <sub-project name></code> property, as described below. Tip For the properties in this <code>domainMapping</code> section, ensure that you maintain the spaces before and after ">", for example "Site-1 > Project-Default". The spaces serve as a separator between the site and the projects.
site name	The Tableau site name, followed by the unique reference ID of the domain in Collibra in which you want to ingest resources from the Tableau site. Important In the configuration file, use the actual site name, along with the domain reference ID, for example: "Collibra_tab_partner_site": "afc8cfb0-91f1-4075-a3e5-7ce6d1f9bcc9"

Property	Description
site name > project name	The Tableau project name, preceded by the name of the Tableau site to which it belongs, and followed by the unique reference ID of the domain in Collibra in which you want to ingest resources from the Tableau project. Important In the configuration file, use the actual site and project names, along with the domain reference ID, for example: "Collibra_tab_partner_site > JB_Test_2812": "d224a1a5-43b4-43b2-8df0-ddf8f2726b82"
site name > project name > sub-project name	The Tableau sub-project name, preceded by the name of the Tableau site and project to which it belongs, and followed by the unique reference ID of the domain in Collibra in which you want to ingest resources from the Tableau sub-project. Important In the configuration file, use the actual site, project and sub-project names, along with the domain reference ID, for example: "Collibra_tab_partner_site > JB_Test_2812 > ProjectJJ2": "d224a1a5-43b4-43b2-8df0-ddf8f2726b82"

4. Save the <source ID> configuration file.

Harvesting materialized views that were generated via an external script

The lineage harvester can harvest materialized views that are native to a data source—meaning the data flow is performed by SQL code stored in the data source. If, however, an external script is used to materialize views into tables, so to speak, they cannot be harvested by the lineage harvester. In this case, you could create a [custom technical lineage](#), which requires a user-defined JSON file.

Tip We recommend creating a script to generate a list of SQL queries to be harvested by the lineage harvester.

For each pair of source (view) and target (materialized view table), create a script as follows:

```
INSERT INTO 'dhw.sales.mv_customers'
  SELECT * FROM 'dhw.sales.v_customers';
```

The generated SQL queries then need to be harvested by the lineage harvester. There are two options for this, depending on where you choose to store the generated SQL code:

- If you store the SQL code in text files, it is harvested using an additional `SqlDirectory` type source.
- If you store the SQL code in a table in the data source, you need to modify the harvesting query, to harvest the table.

In this case, actually, the generated SQL queries don't have to be stored anywhere; rather, they are generated on the fly by a harvesting query. Modify the harvesting query as follows:

```
SELECT
  t.table_name,
  t.ddl as sourceCode,
  CONCAT(t.table_schema, '.', t.table_name) as groupName,
  t.table_schema as schemaName
FROM `##PROJECT_ID##`.`##DSNAME##`.`INFORMATION_
SCHEMA.TABLES` t
WHERE t.table_type IN ('MATERIALIZED VIEW','VIEW')

UNION ALL

SELECT
  CONCAT('m', t.table_name),
  CONCAT('INSERT INTO`m', t.table_name, '` SELECT * FROM
`, t.table_name, ``) as sourceCode,
  CONCAT('Generated m', t.table_schema, '.', t.table_name)
as groupName,
  t.table_schema as schemaName
FROM `##PROJECT_ID##`.`##DSNAME##`.`INFORMATION_
SCHEMA.TABLES` t
WHERE
```

```
t.table_type IN ('VIEW')
AND STARTS_WITH(t.table_name, 'v_')
```

The second SELECT generates the necessary INSERT INTOs for all views in your data source that have a name starting with `v_`.

Manage technical lineage ingestion

You can create a customized SQL file to manage which data objects, for example columns and tables, are ingested in the technical lineage. In the SQL file, you can exclude data objects or change queries that are used to extract data from the database. You specify:

- Which data objects you want to visualize in the technical lineage.
- Between which columns you want to create new relations of the type "Data Element targets / sources Data Element" in Data Catalog.

Note

- If you change queries, you can only use [supported SQL syntax](#).
- Collibra Support does not provide support for customized SQL files.

Steps

1. Open the [lineage harvester](#) folder.
2. Go to the **sql** folder and open the folder of the data source type of which you want to exclude tables or schemas or change queries.
3. Create a copy of the file you want to edit.
4. Rename the copy to *[original name]-custom.sql*.

Example You want to change the file `columns.sql`, so you name the copy of this file and rename it to *columns-custom.sql*.

5. Delete or edit the content of the new SQL file to include or exclude specific tables or schemas or change specific queries in the file.
6. Save the new SQL file.
 - » The lineage harvester uses the new file and ignores the old one.

Run the lineage harvester

After you have prepared the lineage harvester configuration file, run the lineage harvester to create the technical lineage.

Before you begin

If you use a proxy server, connect to the proxy server. For more information, go to [Connecting to a proxy server](#).

Requirements and permissions

- Collibra Data Intelligence Cloud.
- You have purchased Collibra Data Lineage.
- A [global role](#) with the following [global permissions](#):
 - Catalog, for example Catalog Author
 - Data Stewardship Manager
 - Manage all resources
 - System administration
 - Technical lineage
- A [resource role](#) with the following [resource permissions](#) on the community level in which you created the domain:
 - Asset: add
 - Attribute: add
 - Domain: add
 - Attachment: add
- Necessary permissions to all [database objects](#) that the lineage harvester accesses.

Steps

1. Start the lineage harvester by entering the `full-sync` command.
 - To process data from all data sources in the configuration file:
For windows:

```
.\bin\lineage-harvester.bat full-sync
```

For other operating systems:

```
./bin/lineage-harvester full-sync
```

- To process data from specific data sources in the configuration file:

For windows:

```
.\bin\lineage-harvester.bat full-sync -s "ID of the data  
source"
```

For other operating systems:

```
./bin/lineage-harvester full-sync -s "ID of the data  
source"
```

Tip For more information and command options, go to [Lineage harvesting app command options and arguments](#).

Note

If you have Snowflake data sources in your lineage harvester configuration file, set the `JAVA_OPTS` environment variable first. For example, to process data from all data sources including the Snowflake data sources, take the following steps:

On Windows

a. Enter one of the following commands:

- If you use OpenJDK 16:

```
set JAVA_OPTS="-  
Djdk.moduleillegalAccess=permit"
```

- If you use OpenJDK 17 or higher:

```
set JAVA_OPTS="--add-  
opens=java.base/java.nio=ALL-UNNAMED"
```

b. In the same command line, enter the following command:

```
.\bin\lineage-harvester.bat full-sync
```

Note The `set` command is specific to the Windows Command Shell. The command is different if you are using PowerShell.

On Linux

Enter the following command:

- If you use OpenJDK 16:

```
JAVA_OPTS="-Djdk.moduleillegalAccess=permit"  
./bin/lineage-harvester full-sync
```

- If you use OpenJDK 17 or higher:

```
JAVA_OPTS="--add-opens=java.base/java.nio=ALL-UNNAMED" ./bin/lineage-harvester full-sync
```

2. When prompted, enter the [passwords](#) to connect to Collibra and your data sources. Do one of the following:
 - Enter the passwords in the console.
 - » The passwords are encrypted and stored in `/config/pwd.conf`.
 - Provide the passwords via [command line](#).
 - » The passwords are stored locally and not in your lineage harvester folder.
3. If you are creating technical lineage for dbt Cloud and prompted to enter your API token, enter the token value for the service token that you specified for the `tokenName` property in [the lineage harvester configuration file for dbt Cloud](#).

What's next

The lineage harvester sends the data source information to the [Collibra Data Lineage service](#) by using Collibra REST API, where it is parsed and analyzed. As a result, the technical lineage is created and shown in Data Catalog. You can view the technical lineage. For more information, go to [Technical lineage viewer](#).

You can check the progress of the technical lineage creation in [Activities](#) in your Collibra Data Intelligence Cloud environment. The **Results** field indicates how many relations were imported into Data Catalog. Go to the [status page](#) to see the log files of the SQL analysis.

If the lineage harvester log shows an error message or the [harvesting process](#) fails, you can use the technical lineage common errors and issues in [Collibra Support Portal](#) to fix the error.

If you want to synchronize the data sources on fixed times, you can use [scheduled jobs](#).

Schedule jobs

You can use [Task Scheduler](#) on Windows or [Crontab](#) on Mac and Linux to make the [lineage harvester](#) run scheduled jobs at specific times, dates or intervals. In a scheduled

job, the lineage harvester uploads data source information to the Collibra Data Intelligence Cloud and Data Catalog automatically creates new relations of the type "Data Element sources / targets Data Element"

- Between data objects in your data source and their corresponding assets in Data Catalog, including the asset that you create when [preparing the Data Catalog physical data layer](#).
- If you are integrating a BI tool, between ingested assets from BI sources and Data Catalog assets from registered data sources.

You can run one scheduled job for each data source that is listed in the same [configuration file](#).

Note If you provide the [passwords](#) to your Collibra environment and/or to your individual data sources via stdin, you have to use the correct [command](#).

Example You created a configuration file with two data sources. Data source A can run a scheduled job each day at 11 pm, while data source B can run a scheduled job every two days at 6 am.

Upgrade the lineage harvester

Each new lineage harvester adds features and enhancements to the previous version. We highly recommend that you always use the newest lineage harvester available.

If you have created a technical lineage using an older [lineage harvester](#), you can easily upgrade to the newest lineage harvester and reuse your configuration file.

Tip For a list of differences between lineage harvester versions, see the lineage harvester [change log](#).

Steps

1. Download the newest lineage harvester from the [Collibra Downloads page](#). To log in to the Collibra Downloads page, use your Collibra.com username and password.
2. [Install](#) the lineage harvester and a new lineage harvester folder is created.

3. Copy all files from your **config** folder in the old lineage harvester folder to the **config** folder in the new lineage harvester folder.
 - » All files, including the **pwd.conf** and **lineage-harvester.conf** files, are in the **config** folder in the new lineage harvester folder.
4. In the **config** folder, open the **lineage-harvester.conf** file to check if there are other auxiliary files to be moved to the new lineage harvester folder. If needed, copy those files from the old lineage harvester folder to the new lineage harvester folder. Those files can be [the custom technical lineage JSON file](#), the Informatica Intelligent Cloud Services [<source ID> configuration file](#), the Matillion [<source ID>](#) configuration file, and so on.
5. If you have customized SQL files that end with **-custom.sql** in the **sql** folder in the old lineage harvester folder, complete the following steps:
 - a. Compare the original SQL files before customization with the SQL files in the new lineage harvester folder. For example, if you have a customized SQL file named **access_history-custom.sql**, compare the **access_history.sql** file in the old lineage harvester folder with the **access_history.sql** file in the new lineage harvester folder.
 - b. Take any of the following actions:
 - If the SQL files are identical, copy the customized SQL files from the old lineage harvester folder to the new lineage harvester folder.
 - If the SQL files are not the same, complete the following steps:
 - i. Create new SQL files that end with **-custom.sql** in the new lineage harvester folder based on the SQL files in the new lineage harvester folder.
 - ii. Review the customizations in the customized SQL files in the old lineage harvester folder, and make the same customizations to the newly created customized SQL files in the new lineage harvester folder.

Example Take the `access_history-custom.sql` file as an example, and the customization in the `access_history-custom.sql` file was to change the `database.schema` from `SNOWFLAKE.ACCOUNT_USAGE` to `MYDB.ACCOUNT_USAGE`.

- a. Compare the following files:
 - `lineage-harvester-OLD/sql/snowflake/access_history.sql`
 - `lineage-harvester-NEW/sql/snowflake/access_history.sql`
- b. Take any of the following steps:
 - If the `access_history.sql` files are identical, copy the `access_history-custom.sql` file from `lineage-harvester-OLD/sql/snowflake` to the `lineage-harvester-NEW/sql/snowflake` directory.
 - If the `access_history.sql` files are not the same, complete the following steps:
 - i. Create an `access_history-custom.sql` file in the `lineage-harvester-NEW/sql/snowflake` directory by copying the content of the `lineage-harvester-NEW/sql/snowflake/access_history.sql` file to the new `access_history-custom.sql` file.
 - ii. Customize the new `access_history-custom.sql` file by changing the `database.schema` from `SNOWFLAKE.ACCOUNT_USAGE` to `MYDB.ACCOUNT_USAGE`.

Note Beginning with the lineage harvester version 2023.02, the SQL file that was named `access_history_lineage_query_text.sql` has been renamed to `access_history.sql`.

6. Use the `full-sync` [command](#) to synchronize all data sources in your [configuration file](#).
 - » The lineage harvester synchronizes your data sources on the Collibra Data Lineage service and refreshes your [technical lineage](#).

What's next

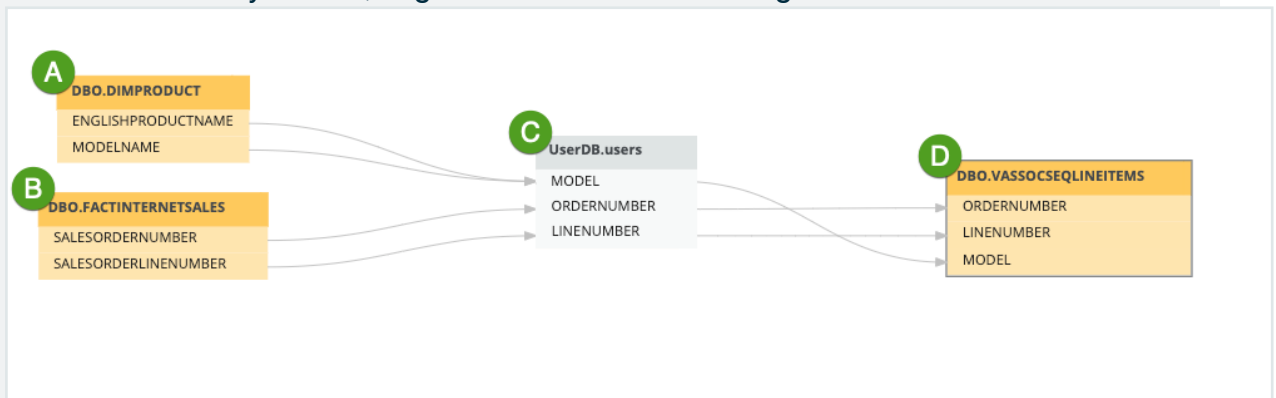
You can check the progress of the technical lineage creation in [Activities](#) in your Collibra Data Intelligence Cloud environment. The **Results** field indicates how many relations were imported into Data Catalog. Go to the [status page](#) to see the log files of the SQL analysis.

If the lineage harvester log shows an error message or the [harvesting process](#) fails, you can use the technical lineage common errors and issues in [Collibra Support Portal](#) to fix the error.

Custom technical lineage

You can create a technical lineage for a particular data source, even if Collibra Data Lineage does not currently support integration of that data source. We refer to this as custom technical lineage.

Example As shown in the following image, you want to create a technical lineage that shows relations between tables and columns from system A and system B, to system C, to system D (A and B -> C -> D). System A, B and D are [supported data sources](#), but system C is a custom application. Depending on the instruction formatting option you choose, you can create one or more JSON files that defines the metadata of system C, to generate a technical lineage.



You can use your custom technical lineage JSON files as the only lineage source, or you can harvest metadata from supported data sources, such as Oracle and Tableau, to visualize lineage alongside the custom technical lineage defined by your JSON files.

In this topic

- [How to create a custom technical lineage](#)
- [JSON file formatting options: Single-file or batch definition](#)
- [Limitations](#)

How to create a custom technical lineage

You can create a custom technical lineage via the lineage harvester or via Edge.

- Go to [Create a custom technical lineage via lineage harvester](#)
- Go to [Create a technical lineage via Edge](#)

JSON file formatting options: Single-file or batch definition

There are two formatting options for defining your custom technical lineage.

Important The new batch definition option, which introduces several benefits, is in beta. After a successful beta period, we will announce the deprecation of the single-file definition option.

Batch definition (Beta)

To create a custom technical lineage via the batch definition option, you define the custom technical lineage in any number of JSON files, and refer to the directory that contains the files in the [lineage harvester configuration file](#). Collibra Data Lineage generates a technical lineage based on the definitions in your JSON files and, optionally, any source code files you've added in the same directory.

Note With the batch definition option, we no longer draw a distinction between simple and advanced custom technical lineage. You can still choose to include source code files to provide additional details on where the lineage comes from, but we no longer differentiate between a custom technical lineage that includes source code and one that does not.

Benefits

- You can define any number of JSON files, which allows for more efficient processing.

- Multiple organizational teams can contribute to the lineage, without having to merge their definitions into a single JSON file.
- You can show table-level lineage, column-level lineage, and indirect lineage.
- Stitching is natively available for any asset type.

Batch file contents

In a CUSTOM-LINEAGE folder, you need:

- Exactly one metadata file.
The name of the file must be **metadata.json**.
- Optionally, one or more assets files. These are required if you want to achieve [stitching](#).
File names must follow the format: **assets<something unique>.json**.
- One or more lineage files, describing table-level, column-level, and indirect lineage.
File names must follow the format: **lineage<something unique>.json**.
- Optionally, a source codes subdirectory with source code files. You refer to the source code files from within your lineage files.

Note Collibra Data Lineage doesn't parse the files in the source codes directory, meaning it does not extract lineage from the SQL and PY files.

For more information about these JSON files, go to [Custom technical lineage JSON file details](#).

Example Directory tree:

```
CUSTOM-LINEAGE
├── metadata.json
├── assets_domain1.json
├── assets1.json
├── lineage.abc.json
├── lineage-extra.json
├── source_codes
│   ├── sc1.sql
│   └── sc2.py
```

Single-file definition

To create a custom technical lineage via the single-file definition option, you have to define the custom technical lineage in a single JSON file and refer to the JSON file in the [lineage harvester configuration file](#). Collibra Data Lineage generates the technical lineage based on your JSON file and, optionally, any source code files to which you refer.

Limitations

- The custom technical lineage must be defined in a single JSON file. This can be challenging if, for example, you have several organizational teams contributing to the lineage definition.
- You can only show column-level lineage, meaning the target of your lineage relationship always has to be a column.
- Stitching is only available natively for Column assets. Stitching is available for other asset types, but only if those assets are discovered by another scanner.

Simple and advanced custom technical lineage

If you opt for the single-file definition option, there are two types of custom technical lineages that you can create:

- Simple custom technical lineage, which defines a basic object hierarchy and creates a lineage between two or more data objects. To create a simple custom technical lineage, you need to include `assets` and `lineages` sections in your JSON file. You can add the transformation code in the `lineages` section.
- Advanced custom technical lineage, which contains a simple custom technical lineage and uses separate source code files that include transformation details, to create the lineage. To create an advanced custom technical lineage, you need to include `assets`, `lineages` and `codebase_files` sections in your JSON file. You add references to the transformation code in source code files in the `codebase_files` section.

Transformation code in both simple and advanced custom technical lineages is shown in the [source code pane](#) at the bottom part of the technical lineage graph.

The screenshot displays the Tableau interface with a data model lineage diagram. The diagram shows a central node 'collibra_rpt [eTDM] Collapsed attributes [menu] Total Data Model Assets' connected to several other nodes: 'Data Model', 'Community Details [TWSH]', 'Data Structure Metrics [TWSH]', 'Community [TWSH]', and 'Data Model Metrics [TWSH]'. The left sidebar shows a list of attributes, and the bottom pane displays the underlying Tableau code for the worksheet 'Data Model [TWSH]'.

```

1 Tableau Worksheet: Data Model [TWSH] [afa2c619-add2-4e43-8077-bd5359ddc64d]
2
3 CONNECTOR: 'collibra_rpt [eTDM].Data Model-Fill => Data Model [TWSH].Data Model-Fill'
4 CALCULATION RULE: '1-[Data Model]'
5
6 CONNECTOR: 'collibra_rpt [eTDM].Data Model => Data Model [TWSH].Data Model-Fill'
7 CALCULATION RULE: '1-[Data Model]'
8
9 CONNECTOR: 'collibra_rpt [eTDM].Data Model => Data Model [TWSH].Data Model'
10 CALCULATION RULE: '(((Approved & Accepted Data Model Assets))/(Total Data Model Assets) )'
11
12 CONNECTOR: 'collibra_rpt [eTDM].Total Data Model Assets => Data Model [TWSH].Data Model'
13 CALCULATION RULE: '(((Approved & Accepted Data Model Assets))/(Total Data Model Assets) )'
14
15 CONNECTOR: 'collibra_rpt [eTDM].Approved & Accepted Data Model Assets => Data Model [TWSH].Data Model'
16 CALCULATION RULE: '(((Approved & Accepted Data Model Assets))/(Total Data Model Assets) )'
17
18 CONNECTOR: 'collibra_rpt [eTDM].Inner => Data Model [TWSH].Inner'
19 CALCULATION RULE: '0'

```

Note With the batch definition option, we no longer draw a distinction between simple and advanced custom technical lineage. You can still include, or not include, source code files to create the lineage, but we no longer differentiate between a custom technical lineage that includes source code and one that does not.

Limitations

Issue	Description									
The "Data Element targets / sources Data Element" relations aren't created for some asset types.	As part of the technical lineage process, Collibra Data Lineage automatically creates relations of the type "Data Element targets / sources Data Element" between assets in Collibra that represent the data objects in your external data sources. However, the "Data Element targets / sources Data Element" relation can only be created between Data Element assets or assets of child asset types, for example: • Column assets. • Data Attribute assets. • Any custom asset types that are a child of the Data Element asset type. <table><tr><td>Data Element</td><td>A unit of data that is defined for processing. It usually documents an aspect of something abstract. Ex...</td><td>00000000-0000-0000-0000-000000031026</td></tr><tr><td>Column</td><td>An atomic unit of data that can be stored in a database table. Examples: FST_NM, EMPID</td><td>00000000-0000-0000-0000-000000031008</td></tr><tr><td>Data Attribute</td><td>A specification that defines a property of a data entity. Examples: CustomerBirthDate, EmployeeFirstN...</td><td>00000000-0000-0000-0000-000000031005</td></tr></table> To confirm, the "Data Element targets / sources Data Element" relation can't be created between other asset types; therefore that lineage can't be shown in a business lineage for other asset types.	Data Element	A unit of data that is defined for processing. It usually documents an aspect of something abstract. Ex...	00000000-0000-0000-0000-000000031026	Column	An atomic unit of data that can be stored in a database table. Examples: FST_NM, EMPID	00000000-0000-0000-0000-000000031008	Data Attribute	A specification that defines a property of a data entity. Examples: CustomerBirthDate, EmployeeFirstN...	00000000-0000-0000-0000-000000031005
Data Element	A unit of data that is defined for processing. It usually documents an aspect of something abstract. Ex...	00000000-0000-0000-0000-000000031026								
Column	An atomic unit of data that can be stored in a database table. Examples: FST_NM, EMPID	00000000-0000-0000-0000-000000031008								
Data Attribute	A specification that defines a property of a data entity. Examples: CustomerBirthDate, EmployeeFirstN...	00000000-0000-0000-0000-000000031005								
The Technical Lineage tab is not included on an asset page.	The Technical Lineage tab, by which you access the technical lineage viewer, is not available on the asset pages of all asset types. If you want to view the technical lineage for an asset of a type for which the Technical Lineage tab is not included on the asset page, you need to access the lineage via an asset of a type for which the tab is included on the asset page. Keep in mind that the Technical Lineage tab is only visible if you have the following permissions: • You have a global role with the Catalog global permission, for example, Catalog Author. • You have a global role with the Technical lineageglobal permission.									

Custom technical lineage JSON file details

This topic describes the properties that you need to include in your JSON files, for both the single-file and batch definition options.

Batch definition (Beta)

If you opt for the batch definition option, you need to create a folder with all of your JSON files and specify the folder in your lineage harvester configuration file. The harvester then accesses the folder, zips the content and ingests it for processing.

Which files do you need in your batch folder?

Let's say that you create a folder and name it **custom-lineage**. In this folder, you need the following:

- Exactly one [metadata file](#), to provide the JSON architecture version, the data source type, and asset type UUIDs of the assets you want to include in the technical lineage.
- Optionally, one or more [asset files](#), to provide a list of data objects you want to include in the technical lineage and define the data object hierarchy to achieve [stitching](#).
- One or more [lineage files](#), to define the lineage relation between two or more data objects.
- Optionally, a [subfolder of source code files](#) that contain the transformation code.

Example

```
__CUSTOM-LINEAGE__
├── assets-domain1.json
├── assets1.json
├── lineage.json
├── lineage-extra.json
├── metadata.json
└── source_codes
    ├── sc1.sql
    └── sc2.py
```

Metadata file

Your metadata file has to be named **metadata.json**. Format the file as shown in the following image:

Example

```
{
  "version": 3,
  "application_name": "databricks",
  "asset_types": {
    "Column": {"uuid": "00000000-0000-0000-0000-000000031008"},
    "Table": {"uuid": "00000000-0000-0000-0000-000000031007"},
    "Database": {"uuid": "00000000-0000-0000-0000-000000031006"},
    "Schema": {"uuid": "00000000-0000-0000-0001-000400000002"}
  }
}
```

Tip

View the JSON schema for the metadata file

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "properties": {
    "version": {
      "anyOf": [
        {
          "type": "integer"
        },
        {
          "type": "string",
          "pattern": "^\\d+$"
        }
      ]
    },
    "application_name": {
      "type": "string"
    },
    "asset_types": {
      "type": "object",
      "additionalProperties": {
        "type": "object",
        "properties": {
          "uuid": {
            "type": "string"
          }
        }
      }
    },
    "required": [
      "application_name",
      "version"
    ]
  }
}
```

Section	Description
version	The version of the JSON architecture. For batch-file instruction, the value must be 3.

Section	Description
application_name	The type of data source for which you are creating a technical lineage. This helps us to better understand your needs and make more informed decisions concerning future integrations.
asset_types	The asset types and UUIDs of the asset types you want to include in the technical lineage. Important If you choose to include asset files in your batch definition, the values (meaning the asset types) that you specify in this property must match the values that you specify in the <code>type</code> properties in your asset files. Likewise, the values that you specify in this property must match the asset types that you mention in your lineage files.

Assets files

Optionally, you can include one or more assets files. You use asset files to provide the list of data objects you want to include in the technical lineage and define the data object hierarchy. The `props` property allows you to specify the full names and domain IDs of the assets.

Tip

Don't use asset files in the following scenarios:

- Your data source consists of the traditional **(System) > Database > Schema > Table > Column** asset types and hierarchy. In that case, full names are automatically, correctly constructed.
- You are working with assets that are not part of that traditional asset hierarchy (in which case, you need to use the `props` property to achieve stitching) and you define `props` in one or more lineage files.

The names of your assets files have to follow the format **assets<something-unique>.json**.

Asset files can consist of `nodes`, `parent`, and `leaf` kinds of assets. In the following example code, we used the `nodes` property to specify the highest levels of the data object hierarchy that we want to view in the technical lineage: Database and Schema. We then

used the `parent` and `leaf` properties to build out the lower levels of the data object hierarchy: Table and Column, respectively.

`parent` assets represent what we traditionally refer to as the table-level lineage. `leaf` assets represents what we traditionally refer to as the column-level lineage.

Keep in mind that the property names `nodes`, `parent`, and `leaf` are designed to be non-restrictive, so you can define a hierarchy to reflect the hierarchy of any asset types (similar to the database > schema > table > column hierarchy), including your custom asset types.

Tip For examples of how to configure the `props` property, as shown in the following code examples, see [Using the props property](#).

Tip

View the JSON schema for assets files

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$defs": {
    "assetData": {
      "type": "object",
      "properties": {
        "name": {
          "type": "string"
        },
        "type": {
          "type": "string"
        }
      },
      "required": [
        "name",
        "type"
      ]
    },
    "props": {
      "type": ["object", "null"],
      "properties": {
        "fullname": {
          "type": "string"
        },
        "domain_id": {
          "type": "string"
        }
      },
      "required": [
        "fullname"
      ]
    }
  },
  "anyOf": [
    {
      "type": "object",
      "properties": {
        "nodes": {
          "type": "array",
          "items": {
            "$ref": "#/$defs/assetData"
          }
        }
      }
    }
  ]
}
```

```

        },
        "props": {
            "$ref": "#/$defs/props"
        },
        "parent": {
            "$ref": "#/$defs/assetData"
        },
        "leaf": {
            "$ref": "#/$defs/assetData"
        }
    },
    "required": [
        "nodes",
        "parent",
        "leaf"
    ]
},
{
    "type": "object",
    "properties": {
        "nodes": {
            "type": "array",
            "items": {
                "$ref": "#/$defs/assetData"
            }
        },
        "props": {
            "$ref": "#/$defs/props"
        },
        "parent": {
            "$ref": "#/$defs/assetData"
        }
    },
    "required": [
        "nodes",
        "parent"
    ]
},
{
    "type": "object",
    "properties": {
        "nodes": {
            "type": "array",
            "items": {
                "$ref": "#/$defs/assetData"
            }
        }
    }
}

```

```

        }
      },
      "props": {
        "$ref": "#/$defs/props"
      }
    },
    "required": [
      "nodes"
    ]
  }
]
}

```

Property	Description
nodes	A JSON element in which you specify the highest levels of the hierarchy. In the example code, the nodes specify the hierarchy of GCS File System > GCS Bucket. Example <pre> { "nodes": [{ "name": "GCS1", "type": "GCS File System" }, { "name": "GCS-B1", "type": "GCS Bucket" }], "props": { "fullname": "<full name of the GCS Bucket asset>", "domain_id": "<domain of the GCS Bucket asset>" } } </pre>

Property	Description
name	The name of the node data object. The value is case-sensitive. Case-sensitivity exception The value of the <code>name</code> property is not case-sensitive for Database, Schema, Table and Column assets. For those assets, any capitalization discrepancies are rectified during processing, and the names always appear in uppercase in the technical lineage. However, assets files are not needed or recommended for these asset types.
type	The type of data object of the specified node, for example: <code>System</code>, <code>Database</code>, <code>Dashboard</code>, or <code>Report</code>. The value is case-sensitive. Important The values (meaning the asset types) that you specify for this property must match the values that you specify in the <code>asset_types</code> property in your metadata file.

Property	Description
parent	A lower-level data object in a hierarchy for which the highest levels are specified in the <code>nodes</code> section. The <code>parent</code> property represents what we traditionally refer to as the table-level lineage. When specifying parent data objects, you also have to include the nodes information, as shown in the following example code. Example <pre> { "nodes": [{ "name": "GCS1", "type": "GCS File System" }, { "name": "GCS-B1", "type": "GCS Bucket" }], "parent": { "name": "DIR1", "type": "Directory" }, "props": { "fullname": "<full name of the Directory asset>", "domain_id": "<domain of the Directory asset>" } }</pre> Tip Each parent object can contain <code>leaf</code> data objects. For example, you can use the <code>parent</code> property to specify a table, and use the <code>leaf</code> properties to specify the columns in the table.

Property	Description
name	The name of the parent data object. The value is case-sensitive. Case-sensitivity exception The value of the <code>name</code> property is not case-sensitive for Database, Schema, Table and Column assets. For those assets, any capitalization discrepancies are rectified during processing, and the names always appear in uppercase in the technical lineage. However, assets files are not needed or recommended for these asset types.
type	The asset type of the parent data object, for example: <code>Table</code>, <code>Directory</code>, <code>Dashboard</code>, or <code>Report</code>. The value is case-sensitive. Important The values (meaning the asset types) that you specify for this property must match the values that you specify in the <code>asset_types</code> property in your metadata file.

Property	Description
leaf	The lowest level data object in your hierarchy. The <code>leaf</code> property represents what we traditionally refer to as the column-level lineage. When specifying leaf data objects, you also have to include the nodes and parent information, as shown in the following example code. The names of parents and leaf data objects can be identical if the data objects with the same names are sub-objects of different <code>nodes</code> data objects. Example <pre> { "nodes": [{ "name": "GCS1", "type": "GCS File System" }, { "name": "GCS-B1", "type": "GCS Bucket" }], "parent": { "name": "DIR1", "type": "Directory" }, "leaf": { "name": "data.xls", "type": "File" }, "props": { "fullname": "<full name of the File asset>", "domain_id": "<domain of the File asset>" } }</pre>

Property	Description
name	The name of the leaf data object. The value is case-sensitive. Case-sensitivity exception The value of the <code>name</code> property is not case-sensitive for Database, Schema, Table and Column assets. For those assets, any capitalization discrepancies are rectified during processing, and the names always appear in uppercase in the technical lineage. However, assets files are not needed or recommended for these asset types.
type	The asset type of the leaf data object, for example: <code>Column</code>, <code>Dashboard</code>, or <code>Report</code>. The value is case-sensitive. Important The values (meaning the asset types) that you specify for this property must match the values that you specify in the <code>asset_types</code> property in your metadata file.

Property	Description
props	This property allows you to specify the full name and domain ID of an asset for the purpose of stitching, regardless of asset type hierarchy. When you add the props property to define the full name of an asset, it applies to the last asset in the array. Tip For examples of how to configure the <code>props</code> property and how to use it for a custom hierarchy, see Using the props property. Important considerations - You don't need to use this property for the traditional (<i>System</i>) > <i>Database</i> > <i>Schema</i> > <i>Table</i> > <i>Column</i> asset types and hierarchy. In fact, assets files are not needed or recommended for those asset types, as the full name is automatically, correctly constructed for that hierarchy. Instead, use this property to specify the full names of assets that are not part of that traditional asset hierarchy. - You must specify in your metadata file the asset types and UUIDs of all the assets types used. - If the <code>useCollibraSystemName</code> property in your lineage harvester configuration file is set to <code>true</code>, the system data object is used to stitch to the System asset in Data Catalog. If the <code>useCollibraSystemName</code> property is set to <code>false</code> in your lineage harvester configuration file, do not specify the system data object in this section, or else stitching will fail. A word about file processing order and inadvertently specifying the same asset more than once Assets files and lineage files are processed in the following order: first, all assets files in alphabetical order, followed by all lineage files in alphabetical order. If you choose to specify <code>props</code> for an asset, we recommend that you do so in either an assets file or a lineage file; not both. For any asset that is inadvertently defined more than once, the first occurrence, with respect to the processing order, is the occurrence that is used. In other words:

Property	Description
	- If you inadvertently define a single asset, with <code>props</code>, in both an assets file and a lineage file, the <code>props</code> values in the assets file are used. - If you inadvertently define a single asset, with <code>props</code>, more than once in a single assets file, or in multiple assets files, the first occurrence of the asset, with respect to the processing order, is used along with the <code>props</code> values defined for that occurrence of the asset.
<code>fullname</code>	The full name of the asset in Collibra. Names are case-sensitive.
<code>domainId</code>	The reference ID of the domain in which the asset exists in Collibra.

Using the props property

The following examples offer some guidance as to when to use the `props` property and how to configure it.

Show me a typical scenario using the props property for a custom hierarchy

First off, let's examine why you don't need to use the `props` property for the traditional database > schema > table > column hierarchy. Let's say you have an assets file in which you define a leaf kind of asset:

```
{
  "nodes": [{
    "name": "Snowflake",
    "type": "System"
  }, {
    "name": "DB1",
    "type": "Database"
  }, {
    "name": "PUBLIC",
    "type": "Schema"
  }],
  "parent": {
    "name": "T1",
    "type": "Table"
  },
  "leaf": {
    "name": "COL1",
```

```
    "type": "Column"
  }
}
```

In this case, the full name of the leaf asset (in this case a Column asset) is automatically and correctly constructed as: "snowflake>DB1>PUBLIC>T1>COL1". In fact, for the traditional database type of hierarchy, you don't even need to use asset files, much less the `props` property.

However, for the following custom hierarchy, you can use the `props` property to specify the correct full name of the leaf asset, in this case a File asset.

```
{
  "nodes": [{
    "name": "gcs",
    "type": "GCS File System"
  }, {
    "name": "bucket1",
    "type": "GCS Bucket"
  }, {
    "name": "/",
    "type": "Directory"
  }],
  "parent": {
    "name": "examples",
    "type": "Directory"
  },
  "leaf": {
    "name": "data.xls",
    "type": "File"
  },
  "props": {
    "fullname": "gcs > bucket1/examples/data.xls",
    "domain_id": "<domain in which the file asset resides>"
  }
}
```

If you don't provide the full name of the leaf asset, it will be constructed using the default traditional formatting (*system*) > *database* > *schema* > *table* > *column*. The result would be the full name: "gcs > bucket1 > / > examples > data.xls". However, this is not the correct construction for File assets. The full name provided in the example above ensures the correct construction, so that stitching is achieved.

Show me example configurations for the props property

- A `props` property in a `nodes` section describes the last asset in the array. In this example, you are specifying the full name and domain ID of the Schema asset "SCH1", not the Database asset "DB1".

```
{
  "nodes": [{"name": "DB1", "type": "Database"}, {"name": "SCH1", "type": "Schema"}],
  "props": {
    "fullname": "<full name of the Schema asset>",
    "domain_id": "<domain of the Schema asset>"
  }
}
```

- If you also want to specify the full name and domain ID of the Database asset, you need two entries, as follows:

```
{
  "nodes": [{"name": "DB1", "type": "Database"}],
  "props": {
    "fullname": "<full name of the Database asset>",
    "domain_id": "<domain of the Database asset>"
  }
}
```

```

  "nodes": [{"name": "DB1", "type": "Database"}, {"name": "SCH1", "type": "Schema"}],
  "props": {
    "fullname": "<full name of the Schema asset>",
    "domain_id": "<domain of the Schema asset>"
  }
}
```

- To specify the full name and domain ID of a `parent` asset, use the `props` property as follows:


```
{
  "nodes": [{"name": "DB1", "type": "Database"}, {"name": "SCH1", "type": "Schema"}],
  "parent": {"name": "TB1", "type": "Table"},
  "props": {
    "fullname": "<full name of the Table asset>",
    "domain_id": "<domain of the Table asset>"
  }
}
```

- To specify the full name and domain ID of a leaf asset, use the `props` property as follows:

```
{
  "nodes": [{"name": "DB1", "type": "Database"}, {"name": "SCH1", "type": "Schema"}],
  "parent": {"name": "TB1", "type": "Table"},
  "leaf": {"name": "COL1", "type": "Column"},
  "props": {
    "fullname": "<full name of the Column asset>",
    "domain_id": "<domain of the Column asset>"
  }
}
```

Show me how to find the full name and domain ID of an asset

There are two ways to find this information.

Via API

You can use the [Find assets API](#), as documented in the Collibra Developer Portal. In the body of the response, look for the following details in the results array:

- `name`: This is the full name of the asset.
- `domain > id`: This is the unique reference ID of the domain in which the asset is located.

Via Data Catalog

- To find the full name, open the relevant asset page, and then click **Actions > Edit**. The **Edit** dialog box appears. The full name is shown in the **Name** field.
- To find the domain ID, open the relevant domain in Collibra. In the following example URL, the reference ID is in bold:
`https://<yourcollibrainstance>/domain/22258f64-40b6-4b16-9c08-c95f8ec0da26?view=00000000-0000-0000-0000-0000000040001`.

Lineage files

You can have one or more lineage files in the folder. The names of your lineage files have to follow the format **lineage<something-unique>.json**.

You use the lineage file to define the lineage relation between two or more data objects. The lineage relations are shown as edges in the technical lineage graph. The edges represent the data flow from a source to a target.

This section contains the path from a source to a target and defines the transformation code or transformation references to be processed by the Collibra Data Lineage service.

Note If the `useCollibraSystemName` property in your lineage harvester configuration file is set to `true`, the system data object is used to stitch to the **System** asset in Data Catalog. If the `useCollibraSystemName` property is set to `false` in your lineage harvester configuration file, do not specify the system data object in these files, or else stitching will fail.

Tip

View the JSON schema for lineage files

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$defs": {
    "assetData": {
      "type": "object",
      "properties": {
        "name": {
          "type": "string"
        },
        "type": {
          "type": "string"
        }
      },
      "required": [
        "name",
        "type"
      ]
    },
    "props": {
      "type": ["object", "null"],
      "properties": {
        "fullname": {
          "type": "string"
        },
        "domain_id": {
          "type": "string"
        }
      },
      "required": [
        "fullname"
      ]
    }
  },
  "type": "object",
  "properties": {
    "src": {
      "anyOf": [
        {
          "type": "object",
          "properties": {
            "nodes": {
              "type": "array",
```

```

        "items": {
            "$ref": "#/$defs/assetData"
        },
        "parent": {
            "$ref": "#/$defs/assetData"
        },
        "leaf": {
            "$ref": "#/$defs/assetData"
        },
        "props": {
            "$ref": "#/$defs/props"
        }
    },
    "required": [
        "nodes",
        "parent",
        "leaf"
    ]
},
{
    "type": "object",
    "properties": {
        "nodes": {
            "type": "array",
            "items": {
                "$ref": "#/$defs/assetData"
            }
        },
        "parent": {
            "$ref": "#/$defs/assetData"
        },
        "props": {
            "$ref": "#/$defs/props"
        }
    },
    "required": [
        "nodes",
        "parent"
    ]
}
],
"trg": {
    "anyOf": [

```

```

{
  "type": "object",
  "properties": {
    "nodes": {
      "type": "array",
      "items": {
        "$ref": "#/$defs/assetData"
      }
    },
    "parent": {
      "$ref": "#/$defs/assetData"
    },
    "leaf": {
      "$ref": "#/$defs/assetData"
    },
    "props": {
      "$ref": "#/$defs/props"
    }
  },
  "required": [
    "nodes",
    "parent",
    "leaf"
  ]
},
{
  "type": "object",
  "properties": {
    "nodes": {
      "type": "array",
      "items": {
        "$ref": "#/$defs/assetData"
      }
    },
    "parent": {
      "$ref": "#/$defs/assetData"
    },
    "props": {
      "$ref": "#/$defs/props"
    }
  },
  "required": [
    "nodes",
    "parent"
  ]
}

```

```

    }
  ]
},
"source_code": {
  "type": "object",
  "properties": {
    "path": {
      "type": "string"
    },
    "highlights": {
      "type": ["array", "null"],
      "items": {
        "type": "object",
        "properties": {
          "start": {
            "type": "integer"
          },
          "len": {
            "type": "integer"
          }
        },
        "required": [
          "len",
          "start"
        ]
      }
    },
    "required": [
      "path"
    ]
  }
},
"required": [
  "src",
  "trg"
]
}

```

Example

```
[
  {
    "src": {
      "nodes": [{"name": "DB1", "type": "Database"},
{"name": "SCH1", "type": "Schema"}],
      "parent": {"name": "TB1", "type": "Table"},
      "leaf": {"name": "COL1", "type": "Column"},
      "props": {
        "fullname": "<full name of the leaf asset>",
        "domain_id": "<domain of the leaf asset>"
      },
    },
    "trg": {
      "nodes": [{"name": "DB1", "type": "Database"},
{"name": "SCH1", "type": "Schema"}],
      "parent": {"name": "TB2", "type": "Table"},
      "props": {
        "fullname": "<full name of the parent asset>",
        "domain_id": "<domain of the parent asset>"
      },
    },
    "source_code" : {
      "path": "<folder name>/sc1.sql",
      "highlights": [{"start": 71, "len": 69 }, ...],
      "transformation_display_name": "middle bubble"
    }
  }
]
```

Properties	Description
src	The hierarchical path to the source data object. This property represents where the data comes from for a transformation. Important The source of a lineage can only be a parent or a leaf. Example <pre>{ "src": { "nodes": [{"name": "DB1", "type": "Database"}, {"name": "SCH1", "type": "Schema"}], "parent": {"name": "TB1", "type": "Table"}, "leaf": {"name": "COL1", "type": "Column"} } }</pre>

Properties	Description
trg	The hierarchical path to the target data object. This property represents where the data flows to. Important The target can be a parent or a leaf; however, if the source is a parent, the target must be a parent. Tip If the target asset is a parent asset and the source asset is a leaf asset, we refer to the lineage as "indirect lineage". If the target asset is a parent asset and the source asset is a parent asset, we refer to the lineage as "table-level lineage". Example <pre>{ "trg": { "nodes": [{"name": "DB1", "type": "Database"}, {"name": "SCH1", "type": "Schema"}], "parent": {"name": "TB2", "type": "Table"} } }</pre>

Properties	Description
props	An optional property that allows you to specify the full name and domain of an asset, for the purpose of stitching. This property is not required for Database, Schema, Table and Column asset types. A word about file processing order and inadvertently specifying the same asset more than once Assets files and lineage files are processed in the following order: first, all assets files in alphabetical order, followed by all lineage files in alphabetical order. If you choose to specify <code>props</code> for an asset, we recommend that you do so in either an assets file or a lineage file; not both. For any asset that is inadvertently defined more than once, the first occurrence, with respect to the processing order, is the occurrence that is used. In other words: • If you inadvertently define a single asset, with <code>props</code>, in both an assets file and a lineage file, the <code>props</code> values in the assets file are used. • If you inadvertently define a single asset, with <code>props</code>, more than once in a single assets file, or in multiple assets files, the first occurrence of the asset, with respect to the processing order, is used along with the <code>props</code> values defined for that occurrence of the asset.
source_code	The transformation code that determines how the technical lineage is constructed. This can be a descriptive string or a SQL statement that manipulates data. This section is optional.
path	The path and name of the source code file that contains the transformation code. The path relative to the <code>source_codes</code> folder, which is in the same folder as the lineage JSON files.

Properties	Description
highlights	This optional property identifies a string of transformation code in a source code file to be highlighted in the source code pane at the bottom part of the technical lineage graph. The entire lines that include the transformation code are highlighted. The string must be a subset of the string of transformation code that is defined by the <code>start</code> and <code>len</code> properties.
start	The start position of the string of the transformation code to be highlighted. The start position is in characters, not bytes.
len	The length of the string of the transformation code to be highlighted. The length is in characters, not bytes.
transformation_ display_name	The name of the transformation when looking at the transformations view in the technical lineage viewer .

Source codes subfolder and files

You can provide a subfolder of source code files that define the transformation details. The source code folder and your JSON files must be in the *CUSTOM_LINEAGE* folder, along with the JSON files. If it's not, an error occurs indicating that the lineage harvester cannot find the source code files.

The source code paths are relative to the *CUSTOM_LINEAGE* folder.

Example

- `source_codes/sc1.sql`
- `source_codes/another-subfolder/sc2.sql`

What happens if I choose not to provide source code files?

If you are using the lineage harvester and there are no source code files to analyze, the batch stats are empty, as shown below. The lineage relations are still created, but because

batch stats are directly linked to the source codes, if source code files are not provided, this is expected.

```
Batch stats:
  Parsing errors: 0
  Analysis errors: 0
  Done: 1
```

The `Done: 1` result is a dummy entry, so that the source appears in the [Sources tab page](#).

Example JSON files

For some example JSON files, go to [Custom technical lineage JSON file examples](#).

Single-file definition

If you opt for the single-file definition option, you use a **lineage.json** file to define the lineage between two or more data objects, and optionally include transformations details to create the custom technical lineage.

The following sections in the JSON file define different parts in the resulting Collibra technical lineage graph:

- `tree`, which defines the data object hierarchy. The data objects are shown as nodes in the technical lineage graph.
- `lineages`, which defines the lineage relation. The lineage relations are shown as edges in the technical lineage graph. The edges represent the data flow from a source to a target.
- `codebase_files`, which points to the source code files that include transformation details.

To create a simple custom technical lineage, you need to include `assets` and `lineages` sections in your JSON file. You can add the transformation code in the `lineages` section.

To create an advanced custom technical lineage, you need to include `assets`, `lineages` and `codebase_files` sections in your JSON file. You add references to the transformation code in source code files in the `codebase_files` section.

Transformation code in both simple and advanced custom technical lineages is shown in the [source code pane](#) at the bottom part of the technical lineage graph.

Requirements and restrictions

The source code files must be in the same directory as the **lineage.json** file. Otherwise, an error occurs indicating that the lineage harvester cannot find the source code files.

Sections

Sections	Description
version	The version of the JSON architecture. Specify the value of 1 . 0, which is the only supported version.

Sections	Description
tree	This section contains tree definitions of data objects between which lineages can be defined. The data objects are systems, databases, schemas, tables, views, columns, dashboards and reports. Each node of a tree contains the name, type and optionally children or leaves properties which form a hierarchy of data objects. You must define a node only once in this section. With the nested tree format, you can reuse the properties of one node for multiple children. For example, you can define a database once and use the <code>children</code> array to define multiple tables in the database. Tip Usually, the structure you map is the following: system > database > schema > table > column. The system is optional, unless the <code>useCollibraSystemName</code> property is set to <code>true</code> in your lineage harvester configuration file. Collibra Data Lineage can stitch these data objects to assets in Data Catalog. However, you can also map custom objects, for example dashboards and reports. Custom objects cannot be stitched to assets in Data Catalog. Important If the <code>useCollibraSystemName</code> property is set to <code>false</code> in your lineage harvester configuration file, do not specify the system data object in this section, or else stitching will fail.
lineages	This section contains the path from a source to a target and defines the transformation code or transformation references to be processed by the Collibra Data Lineage service. Important If the <code>useCollibraSystemName</code> property is set to <code>false</code> in your lineage harvester configuration file, do not specify the system data object in this section, or else stitching will fail.
codebase_files	This optional section defines the reference to source code files. Store the source code files that contain the transformation code in the same directory as the <code>lineage.json</code> file. Include this section only when you create an advanced custom technical lineage.

tree section properties

Properties	Description
name	The name of your data object. Specify this property with the system name, database name, schema name, table name, view name or column name. The following rules apply when you specify this property: • The names are case sensitive. • The names of children and leaves can be identical if the children and leaves with the same names are in different parent nodes.
type	The type of your data object. You can specify one of the following options: <code>system</code>, <code>database</code>, <code>schema</code>, <code>table</code>, <code>view</code>, <code>column</code>, <code>dashboard</code> or <code>report</code>. If the <code>useCollibraSystemName</code> property in your lineage harvester configuration file is set to <code>true</code>, the system data object is used to stitch to the System asset in Data Catalog. If the <code>useCollibraSystemName</code> property is set to <code>false</code> in your lineage harvester configuration file, do not specify the system data object in this section, or else stitching will fail.
children	The sub-objects that have a hierarchical relation to the defined data object. Each child can contain <code>children</code> properties, except for the penultimate child. The penultimate <code>children</code> property must contain the <code>leaves</code> property. The <code>leaves</code> property cannot contain a <code>children</code> property. For example, you can use the <code>children</code> property to define a table and use the <code>leaves</code> properties to define columns that have a relation to the table node. Each child and leave have the <code>name</code> and <code>type</code> properties and the optional <code>catalog_fullname</code>, <code>catalog_domain_id</code>, <code>catalog_asset_type_name</code> and <code>catalog_asset_type_uuid</code> properties.

Properties	Description
leaves	The sub-objects of an object that is defined in a <code>children</code> property, but cannot have sub-objects of their own. A technical lineage is defined as relations between leaf nodes of the tree. The value of the <code>type</code> property of the <code>leaves</code> property must be <code>column</code> or <code>report</code>. Indirect and table-level technical lineages are not supported. For the workarounds to create a table level or indirect technical lineage, see Programming considerations.

lineage section properties

Properties	Required	Description
src_path	Yes	The hierarchical path to the source data object. This data object is defined as a leaf in the <code>tree</code> section. This property represents where the data comes from for a transformation.
trg_path	Yes	The hierarchical path to the target data object. This data object is defined as a leaf in the <code>tree</code> section. This property represents where the data flows to.

Properties	Required	Description
<data objects>	Yes	An ordered array of data object names. This array is required to define the sub-objects of the <code>src_path</code> and <code>trg_path</code> properties. Specify the array with the data object names that start from the top of the <code>tree</code> section and finish at a leaf node. This example shows data objects that can be stitched: <code>system > database > schema > table > column</code>. This example shows data objects that cannot be stitched: <code>dashboard > report > column</code>. If the <code>useCollibraSystemName</code> property in your lineage harvester configuration file is set to <code>true</code>, the system data object is used to stitch to the System asset in Data Catalog. If the <code>useCollibraSystemName</code> property is set to <code>false</code> in your lineage harvester configuration file, do not specify the system data object in this section, or else stitching will fail.
mapping	Yes Simple custom technical lineage only	The mapping name. This property specifies a name for the transformation code.
source_code	Yes Simple custom technical lineage only	The transformation code, which determines how the technical lineage is constructed. The transformation code can be a descriptive string or a SQL statement that manipulates data.
mapping_ref	No Advanced custom technical lineage only	This property contains the name of the mapping reference to the transformation code in source code files. This property also contains the position and length of the transformation code to be highlighted in the technical lineage graph.

Properties	Required	Description
source_code	No Advanced custom technical lineage only	The name of the source code file that contains the transformation code. The transformation code can be a SQL statement, code that manipulates data or a descriptive string. The source code file must be in the same folder as the lineage.json file.
mapping	No Advanced custom technical lineage only	The unique descriptor of a part of transformation code in a source code file that is in the same directory as the lineage.json file. A source code file can contain different parts of transformation code that represent different data flows. This property indicates the referenced data flow. The value of this property is the same as the value of the mapping_refs property in the <code>codebase_files</code> section.
codebase_pos	No Advanced custom technical lineage only	The positions indicate a string of the transformation code in a source code file to be highlighted in the bottom part of the Collibra technical lineage graph. The whole lines that include the transformation code are highlighted. The string must be a subset of the string of the transformation code that is defined by the <code>pos_start</code> and <code>pos_len</code> properties of the mapping_refs property in the <code>codebase_files</code> section.
pos_start	No Advanced custom technical lineage only	The start position of the string of the transformation code to be highlighted. The start position is in characters, not bytes. The value must be equal to or greater than the value of the <code>pos_start</code> property of the mapping_refs property in the <code>codebase_files</code> section.

Properties	Required	Description
pos_len	No Advanced custom technical lineage only	The length of the string of the transformation code to be highlighted. The length is in characters, not bytes. Specify a value in the following range: • Equal to or greater than 1. • Less than or equal to the length of the string that is defined by the pos_len property of the mapping_refs property in the the codebase_files section. For example, if you specify "pos_start": 10 and "pos_len": 160 in the codebase_files section, specify a value for this property in the range of 0 - 149.

codebase_files section properties

Properties	Description
<source code path>	The file path to source code files that contain the transformation code. The transformation code can be a SQL statement or code that manipulates data. The source code file must be in the same directory as the lineage.json file.
mapping_refs	The mapping of the transformation code and the position of the transformation code that is shown in the bottom part of the technical lineage graph. This property defines a string of the transformation code in the source code file to be shown in the technical lineage graph. The string must include the string that is defined by the pos_start and pos_len properties of the mapping property in the lineage section.

Properties	Description
<mapping>	The unique descriptor of a part of transformation code in a source code file that is in the same directory as the lineage.json file. A source code file can contain different parts of transformation code that represent different data flows. This property indicates the referenced data flow. The value must match the value of the mapping property in the lineage section.
pos_start	The start position of the string of the transformation code. The start position is in characters, not bytes. Specify a value in the following range: • Equal to or greater than 0. • Less than or equal to the value of the <code>pos_start</code> property in the mapping property in the lineage section.
pos_len	The length of the string of the transformation code. The length is in characters, not bytes. Specify a value in the following range: • Greater than or equal to 1. • Less than or equal to the length of the source code file minus the start position. For example, if you specify "<code>pos_start</code>": 10 and the file length is 160 characters, specify a value for this property in the range of 1 - 150.

Programming considerations

Currently, there is no native support for indirect and table-level lineages. As a workaround, you can specify "`type`": "column" and "`name`": "*" for the `leaves` property to create a table level or indirect technical lineage. With this specification, the indirect technical lineage is shown as a solid line instead of a dashed line in the Collibra technical lineage graph, and is always shown, regardless of whether or not the [Show indirect dependencies](#) option is enable or disabled.

Example

For some example JSON files, go to [Custom technical lineage JSON file examples](#).

Custom technical lineage JSON file examples

Batch definition (Beta)

In this section, we provide three detailed examples of how to configure your **metadata**, **assets** and **lineage** JSON files for the batch definition option.

Example 1 is the least complex. It shows a configuration for working with data sources that conform to the traditional **(System) > Database > Schema > Table > Column** hierarchy. Examples 2 and 3 are more complex examples.

Helpful considerations to keep in mind

- Don't use asset files in the following scenarios:
 - Your data source consists of the traditional **(System) > Database > Schema > Table > Column** asset types and hierarchy. In that case, full names are automatically, correctly constructed.
 - You are working with assets that are not part of that traditional asset hierarchy (in which case, you need to use the `props` property to achieve stitching) and you define `props` in one or more lineage files.
- Don't use the `props` property for the traditional **(System) > Database > Schema > Table > Column** hierarchy.

Typical scenario in which you need to use the `props` property

First off, let's examine why you don't need to use the `props` property for the traditional database > schema > table > column hierarchy. Let's say you have an assets file in which you define a leaf kind of asset:

```

{
  "nodes": [{
    "name": "Snowflake",
    "type": "System"
  }, {
    "name": "DB1",
    "type": "Database"
  }, {
    "name": "PUBLIC",
    "type": "Schema"
  }],
  "parent": {
    "name": "T1",
    "type": "Table"
  },
  "leaf": {
    "name": "COL1",
    "type": "Column"
  }
}

```

In this case, the full name of the leaf asset (in this case a Column asset) is automatically and correctly constructed as: "snowflake>DB1>PUBLIC>T1>COL1".

However, for the following custom hierarchy, you can use the `props` property to specify the correct full name of the leaf asset, in this case a File asset.

```

{
  "nodes": [{
    "name": "gcs",
    "type": "GCS File System"
  }, {
    "name": "bucket1",
    "type": "GCS Bucket"
  }, {
    "name": "/",
    "type": "Directory"
  }],
  "parent": {
    "name": "examples",
    "type": "Directory"
  },
  "leaf": {
    "name": "data.xls",

```

```
    "type": "File"
  },
  "props": {
    "fullname": "gcs > bucket1/examples/data.xls",
    "domain_id": "<domain in which the file asset resides>"
  }
}
```

If you don't provide the full name of the leaf asset, it will be constructed using the default traditional formatting (*system*) > *database* > *schema* > *table* > *column*. The result would be the full name: "gcs > bucket1 > / > examples > data.xls". However, this is not the correct construction for File assets. The full name provided in the example above ensures the correct construction, so that stitching is achieved.

Examples

Example 1

Scenario

In this example, we are working with a single data source with the traditional (**System**) > **Database** > **Schema** > **Table** > **Column** asset types and hierarchy. Therefore, we don't need to include an asset file or use the `props` property.

Which files to include?

We are including the following files:

```
— CUSTOM-LINEAGE
  |
  |— lineage.json
  |— metadata.json
  |— source_codes
  |   |— source_code_view_1.txt
  |   |— source_code_view_2.txt
```

- A metadata JSON file.

Tip You always need exactly one metadata JSON file.

- One lineage JSON file, to define the lineage relationships and to show:
 - Column-level lineage.
 - Column-level lineage from the same source (`src`), which is achieved by adding another entry with the same source.
 - Indirect lineage, in other words the target (`trg`) is a parent asset.
 - Table-level lineage, in other words both the source and the target are a parent asset.
- Two source code files in a `source_code` directory.

Tip Source code files are always optional.

The metadata JSON file

```
{
  "version": 3,
  "application_name": "custom lineage batch example 1",
  "asset_types": {
    "System": {
      "uuid": "00000000-0000-0000-0000-0000000031302"
    },
    "Column": {
      "uuid": "00000000-0000-0000-0000-0000000031008"
    },
    "Table": {
      "uuid": "00000000-0000-0000-0000-0000000031007"
    },
    "Database": {
      "uuid": "00000000-0000-0000-0000-0000000031006"
    },
    "Schema": {
      "uuid": "00000000-0000-0000-0001-000400000002"
    }
  }
}
```


The lineage JSON file

```
[{
  "src": {
    "nodes": [{
      "name": "snowflake",
      "type": "System"
    }, {
      "name": "DB1",
      "type": "Database"
    }, {
      "name": "PUBLIC",
      "type": "Schema"
    }],
    "parent": {
      "name": "T1",
      "type": "Table"
    },
    "leaf": {
      "name": "col1",
      "type": "Column"
    }
  },
  "trg": {
    "nodes": [{
      "name": "snowflake",
      "type": "System"
    }, {
      "name": "DB1",
      "type": "Database"
    }, {
      "name": "PUBLIC",
      "type": "Schema"
    }],
    "parent": {
      "name": "VIEW_1",
      "type": "Table"
    },
    "leaf": {
      "name": "col1",
      "type": "Column"
    }
  }
}, {
  "src": {
    "nodes": [{
      "name": "snowflake",
```

```

        "type": "System"
    }, {
        "name": "DB1",
        "type": "Database"
    }, {
        "name": "PUBLIC",
        "type": "Schema"
    }],
    "parent": {
        "name": "T1",
        "type": "Table"
    },
    "leaf": {
        "name": "col1",
        "type": "Column"
    }
},
"trg": {
    "nodes": [{
        "name": "snowflake",
        "type": "System"
    }, {
        "name": "DB1",
        "type": "Database"
    }, {
        "name": "PUBLIC",
        "type": "Schema"
    }],
    "parent": {
        "name": "VIEW_1",
        "type": "Table"
    },
    "leaf": {
        "name": "col2",
        "type": "Column"
    }
},
"source_code": {
    "path": "source_codes/source_code_view_1.txt",
    "highlights": [{
        "start": 0,
        "len": 43
    }],
    "transformation_display_name": "view_1 creation"
},
{
    "src": {
        "nodes": [{

```

```

        "name": "snowflake",
        "type": "System"
    }, {
        "name": "DB1",
        "type": "Database"
    }, {
        "name": "PUBLIC",
        "type": "Schema"
    }],
    "parent": {
        "name": "T1",
        "type": "Table"
    },
    "leaf": {
        "name": "col2",
        "type": "Column"
    }
},
"trg": {
    "nodes": [{
        "name": "snowflake",
        "type": "System"
    }, {
        "name": "DB1",
        "type": "Database"
    }, {
        "name": "PUBLIC",
        "type": "Schema"
    }],
    "parent": {
        "name": "VIEW_1",
        "type": "Table"
    }
},
{
    "src": {
        "nodes": [{
            "name": "snowflake",
            "type": "System"
        }, {
            "name": "DB1",
            "type": "Database"
        }, {
            "name": "PUBLIC",
            "type": "Schema"
        }],
        "parent": {

```

```

        "name": "T2",
        "type": "Table"
    },
    "trg": {
        "nodes": [{
            "name": "snowflake",
            "type": "System"
        }, {
            "name": "DB1",
            "type": "Database"
        }, {
            "name": "PUBLIC",
            "type": "Schema"
        }],
        "parent": {
            "name": "VIEW_2",
            "type": "Table"
        }
    },
    "source_code": {
        "path": "source_codes/source_code_view_2.txt",
        "highlights": [{
            "start": 0,
            "len": 39
        }],
        "transformation_display_name": "view_2 creation"
    }
}
]

```

Example 2

Scenario

In this example, we are working with two data sources:

- A source system with the following asset types and hierarchy: **GCS File System > GCS Bucket > Directory > File**.
- A target system with the traditional **(System) > Database > Schema > Table > Column** asset types and hierarchy.

To achieve stitching, we need to use the `props` property to define the correct full-name construction for the File asset in the source system. We can define `props` in a lineage file. By doing so, we don't need to include any asset files.

Which files to include?

We are including the following files:

```
—CUSTOM-LINEAGE—  
  └─ lineage.json  
  └─ metadata.json
```

- A metadata JSON file.

Tip You always need exactly one metadata JSON file.

- One lineage JSON file, with the `props` property, to define the correct full-name construction for the File asset in the source system and achieve stitching.

Tip By defining the `props` in the lineage file, we don't need to include an assets file. If, for example, we had several lineage files, it would be easier to include an assets file and define the `props` there once, rather than define the `props` in each lineage file.

We are not including source code files. They are always optional.

The metadata JSON file

```
{  
  "version": 3,  
  "application_name": "custom lineage batch example 2",  
  "asset_types": {  
    "System": {  
      "uuid": "00000000-0000-0000-0000-000000031302"  
    },  
    "Column": {
```

```

    "uuid": "00000000-0000-0000-0000-000000031008"
  },
  "Table": {
    "uuid": "00000000-0000-0000-0000-000000031007"
  },
  "Database": {
    "uuid": "00000000-0000-0000-0000-000000031006"
  },
  "Schema": {
    "uuid": "00000000-0000-0000-0001-000400000002"
  },
  "File": {
    "uuid": "00000000-0000-0000-0000-000000031304"
  },
  "Directory": {
    "uuid": "00000000-0000-0000-0000-000000031303"
  },
  "GCS Bucket": {
    "uuid": "00000000-0000-0000-0001-002700000002"
  },
  "GCS File System": {
    "uuid": "00000000-0000-0000-0001-002700000001"
  }
}

```

The lineage JSON file

```

[ {
  "src": {
    "nodes": [ {
      "name": "gcs",
      "type": "GCS File System"
    }, {
      "name": "catingestiontest",
      "type": "GCS Bucket"
    }, {
      "name": "/",
      "type": "Directory"
    }, {
      "name": "ingestion-test",
      "type": "Directory"
    } ],
    "parent": {

```

```

      "name": "ingestion copy",
      "type": "Directory"
    },
    "leaf": {
      "name": "mytest.csv",
      "type": "File"
    },
    "props": {
      "fullname": "f13bf705-13a4-44c9-843e-f341feccfb6e >
catingestiontest/ingestion-test/ingestion
copy/mytest.csv/1611609340099809",
      "domain_id": "fealb0b0-705f-4e0d-b5eb-1f21132cc718"
    }
  },
  "trg": {
    "nodes": [{
      "name": "snowflake",
      "type": "System"
    }, {
      "name": "DB1",
      "type": "Database"
    }, {
      "name": "PUBLIC",
      "type": "Schema"
    }],
    "parent": {
      "name": "T1",
      "type": "Table"
    },
    "leaf": {
      "name": "col1",
      "type": "Column"
    }
  }
}
}]

```

Example 3

Scenario

In this example, we are working with the same two data sources as in Example 2:

- A source system with the following asset types and hierarchy: **GCS File System > GCS Bucket > Directory > File**.

- A target system with the traditional **(System) > Database > Schema > Table > Column** asset types and hierarchy.

To achieve stitching, we need to use the `props` property to define the correct full-name construction for the File asset in the source system. However, this time we have multiple lineage files. Instead of defining `props` in each lineage file, let's include an asset file. That way we only have to define `props` there once.

Which files to include?

We are including the following files:

```
CUSTOM-LINEAGE
├── assets.json
├── lineage-1.json
├── lineage-2.json
└── metadata.json
```

- A metadata JSON file.

Tip You always need exactly one metadata JSON file.

- An assets JSON file, in which provide the list of data objects we want to include in the technical lineage, and define `props`.
- Two lineage JSON files, to define the lineage relationships.

We are not including source code files. They are always optional.

Metadata JSON file

```
{
  "version": 3,
  "application_name": "custom lineage batch example 2",
  "asset_types": {
    "System": {
      "uuid": "00000000-0000-0000-0000-000000031302"
    }
  }
}
```



```

    },
    "Column": {
      "uuid": "00000000-0000-0000-0000-0000000031008"
    },
    "Table": {
      "uuid": "00000000-0000-0000-0000-0000000031007"
    },
    "Database": {
      "uuid": "00000000-0000-0000-0000-0000000031006"
    },
    "Schema": {
      "uuid": "00000000-0000-0000-0001-000400000002"
    },
    "File": {
      "uuid": "00000000-0000-0000-0000-0000000031304"
    },
    "Directory": {
      "uuid": "00000000-0000-0000-0000-0000000031303"
    },
    "GCS Bucket": {
      "uuid": "00000000-0000-0000-0001-002700000002"
    },
    "GCS File System": {
      "uuid": "00000000-0000-0000-0001-002700000001"
    }
  }
}

```

Assets JSON file

```

[ {
  "nodes": [ {
    "name": "gcs",
    "type": "GCS File System"
  }, {
    "name": "catingestiontest",
    "type": "GCS Bucket"
  }, {
    "name": "/",
    "type": "Directory"
  }, {
    "name": "ingestion-test",
    "type": "Directory"
  } ]
} ]

```

```

    "parent": {
      "name": "ingestion copy",
      "type": "Directory"
    },
    "leaf": {
      "name": "mytest.csv",
      "type": "File"
    },
    "props": {
      "fullname": "f13bf705-13a4-44c9-843e-f341feccfb6e >
catingestiontest/ingestion-test/ingestion
copy/mytest.csv/1611609340099809",
      "domain_id": "fe1b0b0-705f-4e0d-b5eb-1f21132cc718"
    }
  }, {
    "nodes": [{
      "name": "snowflake",
      "type": "System"
    }, {
      "name": "DB1",
      "type": "Database"
    }, {
      "name": "PUBLIC",
      "type": "Schema"
    }
  ],
    "parent": {
      "name": "T1",
      "type": "Table"
    },
    "leaf": {
      "name": "col3",
      "type": "Column"
    }
  }
}]

```

Lineage JSON file (1 of 2)

```

[ {
  "src": {
    "nodes": [{
      "name": "gcs",
      "type": "GCS File System"
    }, {
      "name": "catingestiontest",

```

```

        "type": "GCS Bucket"
      }, {
        "name": "/",
        "type": "Directory"
      }, {
        "name": "ingestion-test",
        "type": "Directory"
      }
    ],
    "parent": {
      "name": "ingestion copy",
      "type": "Directory"
    },
    "leaf": {
      "name": "mytest.csv",
      "type": "File"
    }
  },
  "trg": {
    "nodes": [{
      "name": "snowflake",
      "type": "System"
    }, {
      "name": "DB1",
      "type": "Database"
    }, {
      "name": "PUBLIC",
      "type": "Schema"
    }
  ],
  "parent": {
    "name": "T1",
    "type": "Table"
  },
  "leaf": {
    "name": "col1",
    "type": "Column"
  }
}
}]

```

Lineage JSON file (2 of 2)

```

[ {
  "src": {

```

```

    "nodes": [{
      "name": "gcs",
      "type": "GCS File System"
    }, {
      "name": "catingestiontest",
      "type": "GCS Bucket"
    }, {
      "name": "/",
      "type": "Directory"
    }, {
      "name": "ingestion-test",
      "type": "Directory"
    }
  ],
  "parent": {
    "name": "ingestion copy",
    "type": "Directory"
  },
  "leaf": {
    "name": "mytest.csv",
    "type": "File"
  }
},
"trg": {
  "nodes": [{
    "name": "snowflake",
    "type": "System"
  }, {
    "name": "DB1",
    "type": "Database"
  }, {
    "name": "PUBLIC",
    "type": "Schema"
  }
  ],
  "parent": {
    "name": "T1",
    "type": "Table"
  },
  "leaf": {
    "name": "col2",
    "type": "Column"
  }
}
}]

```

Single-file definition

This section shows some example **lineage.json** files for simple custom technical lineage and advanced custom technical lineage.

Each example can be used to generate technical lineage graphs in Collibra to represent the IOT_JSON and IOT_DEVICES_PER_COUNTRY tables with the following columns:

IOT_JSON	IOT_DEVICES_PER_COUNTRY
CCA3	COUNTRY
DEVICE_ID	NUMBER_DEVICES

Example JSON file for a simple custom technical lineage

In the following example, the `tree` section defines the IOT_JSON and IOT_DEVICES_PER_COUNTRY tables and columns. The tables are in a schema named COLLIBRA. The COLLIBRA schema is in a database named COLLIBRA and a system named Databricks.

Important If you define the System asset in your lineage.json file, the `useCollibraSystemName` property in your lineage harvester configuration file must be set to `true`; otherwise, relations will not be created between the relevant assets in Collibra and stitching will fail.

To show the transformation code at the bottom of the technical lineage graph, specify the `mapping` and `source_code` properties in the `lineages` section.

```
{
  "version": "1.0",
  "tree": [
    {
      "name": "Databricks",
      "type": "system",
      "children": [
        {
          "name": "COLLIBRA",
          "type": "database",
          "children": [
            {
              "name": "COLLIBRA",
```

```

        "type": "schema",
        "children": [
          {
            "name": "IOT_JSON",
            "type": "table",
            "leaves": [
              {
                "name": "CCA3",
                "type": "column"
              },
              {
                "name": "DEVICE_ID",
                "type": "column"
              }
            ]
          },
          {
            "name": "IOT_DEVICES_PER_COUNTRY",
            "type": "table",
            "leaves": [
              {
                "name": "COUNTRY",
                "type": "column"
              },
              {
                "name": "NUMBER_DEVICES",
                "type": "column"
              }
            ]
          }
        ]
      }
    ]
  },
  "lineages": [
    {
      "src_path": [
        {
          "system": "Databricks"
        },
        {
          "database": "COLLIBRA"
        },
        {
          "schema": "COLLIBRA"
        },
        {
          "table": "IOT_JSON"
        }
      ]
    }
  ]
}

```

```

        },
        {
            "column": "CCA3"
        }
    ],
    "trg_path": [
        {
            "system": "Databricks"
        },
        {
            "database": "COLLIBRA"
        },
        {
            "schema": "COLLIBRA"
        },
        {
            "table": "IOT_DEVICES_PER_COUNTRY"
        },
        {
            "column": "COUNTRY"
        }
    ],
    "mapping": "dev_no_bat_per_country_view",
    "source_code": "INSERT INTO ... SELECT CCA3 AS COUNTRY...FROM
IOT_JSON"
    }
]
}

```

Example JSON file for an advanced custom technical lineage

In the following example, the `tree` section defines the `IOT_JSON` and `IOT_DEVICES_PER_COUNTRY` tables and columns. The tables are in a schema named `COLLIBRA`. The `COLLIBRA` schema is in a database named `COLLIBRA` and a system named `Databricks`. If you define the `System` asset in your `lineage.json` file, the `useCollibraSystemName` property in your lineage harvester configuration file must be set to `true`; otherwise, relations will not be created between the relevant assets in Collibra and stitching will fail.

```

{
  "version": "1.0",
  "tree": [
    {
      "name": "Databricks",
      "type": "system",
      "children": [
        {
          "name": "COLLIBRA",

```

```

        "type": "database",
        "children": [
            {
                "name": "COLLIBRA",
                "type": "schema",
                "children": [
                    {
                        "name": "IOT_JSON",
                        "type": "table",
                        "leaves": [
                            {
                                "name": "CCA3",
                                "type": "column"
                            },
                            {
                                "name": "DEVICE_ID",
                                "type": "column"
                            }
                        ]
                    }
                ]
            },
            {
                "name": "IOT_DEVICES_PER_COUNTRY",
                "type": "table",
                "leaves": [
                    {
                        "name": "COUNTRY",
                        "type": "column"
                    },
                    {
                        "name": "NUMBER_DEVICES",
                        "type": "column"
                    }
                ]
            }
        ]
    },
    ],
    "lineages": [
        {
            "src_path": [
                {
                    "system": "Databricks"
                },
                {
                    "database": "COLLIBRA"
                }
            ]
        }
    ]
}

```



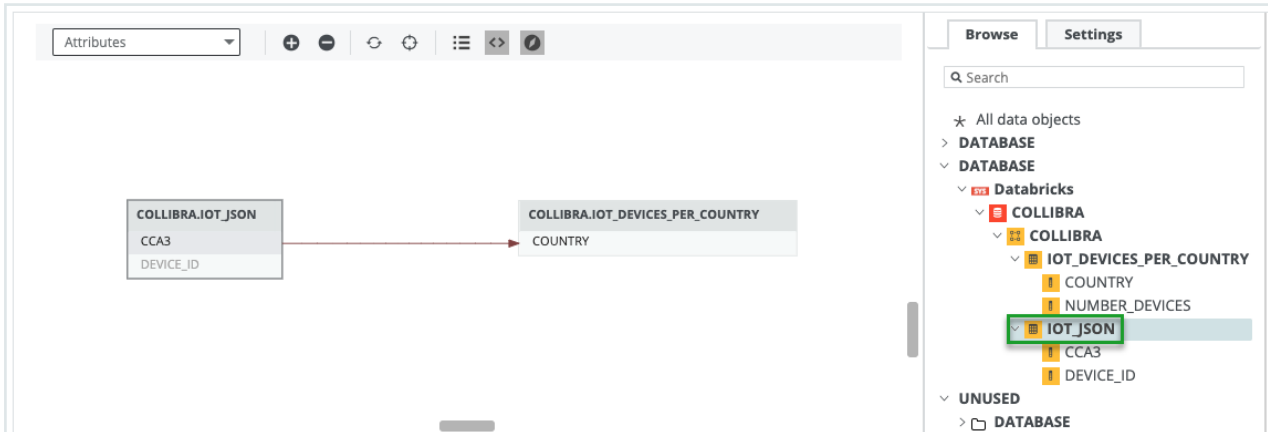
```

        "schema": "COLLIBRA"
    },
    {
        "table": "IOT_JSON"
    },
    {
        "column": "CCA3"
    }
],
"trg_path": [
    {
        "system": "Databricks"
    },
    {
        "database": "COLLIBRA"
    },
    {
        "schema": "COLLIBRA"
    },
    {
        "table": "IOT_DEVICES_PER_COUNTRY"
    },
    {
        "column": "COUNTRY"
    }
],
"mapping_ref":
{
    "source_code": "transforms.sql",
    "mapping": "dev_no_bat_per_country_view",
    "codebase_pos": [
        {
            "pos_start": 71, "pos_len": 69
        }
    ]
}
],
"codebase_files":
{
    "transforms.sql":
    {
        "mapping_refs":
        {
            "dev_no_bat_per_country_view":
            {
                "pos_start": 0,
                "pos_len": 246
            }
        }
    }
}

```

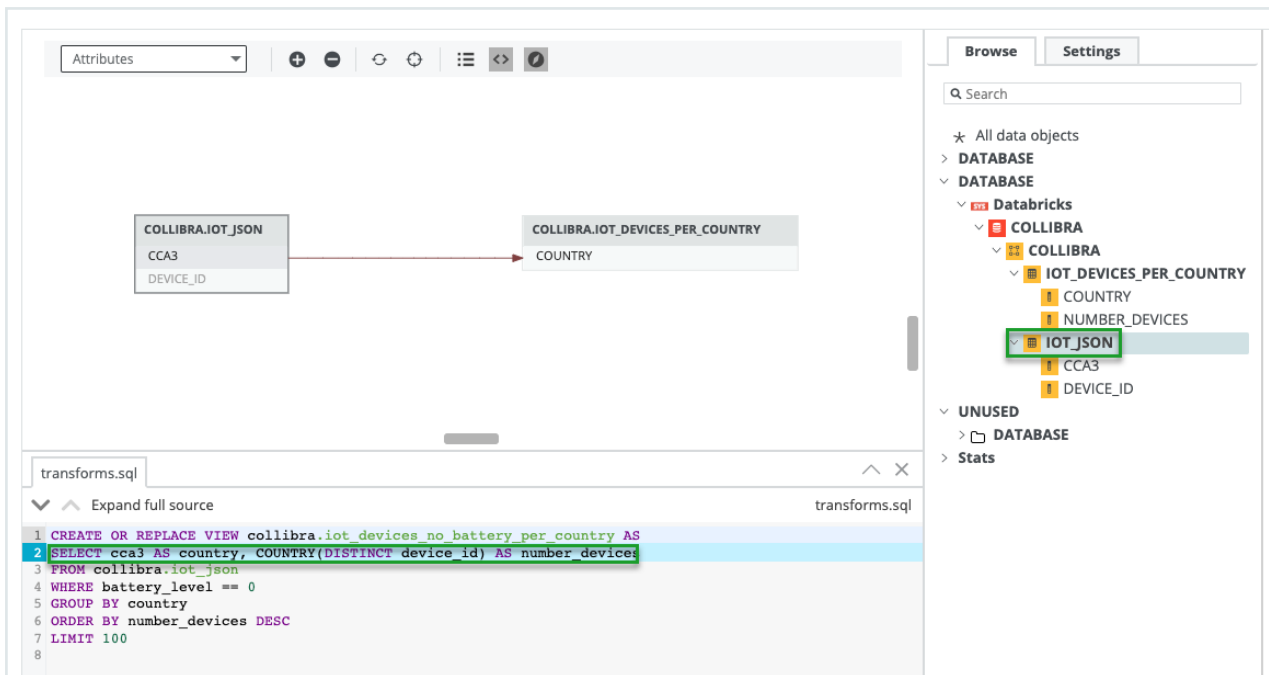
Example technical lineage graphs

Both example **lineage.json** files generate the following technical lineage graph, which contains 2 nodes and 1 edge.



The following technical lineage graph is generated by using the example **lineage.json** file for an advanced custom technical lineage. The bottom part shows the transformation code that generated the data flow.

In the `lineages` section, the `pos_start` property is specified with 71 and the `pos_len` property is specified with 69. The specifications indicate that the transformation code that starts at position 71 and the following 69 characters are highlighted in blue. Line 2 in the technical lineage graph contains the highlighted transformation code.



Delete the technical lineage of a data source

You can delete the technical lineage of a data source if you no longer want to see it in the [technical lineage graph](#). To delete the technical lineage of the data source, you must remove the configuration of the data source from the lineage harvester configuration file and use the `ignore-source` command to exclude the data source when you synchronize the technical lineage again.

Note You always need at least one source in your lineage harvester configuration file.

Before you begin

Install the lineage harvester 2023.04 or newer.

Steps

1. Optional: To determine the data source that you want to exclude from the Technical lineage, enter the `list-sources` command:

- For Windows: `.\bin\lineage-harvester.bat list-sources`
 - For other operating systems: `./bin/lineage-harvester list-sources`
 - » All data sources that were used to create the technical lineage are listed. The list also includes the source ID of each data source. You can use the list to identify the data source to be excluded.
2. In the lineage harvester folder, open your [lineage harvester configuration file](#).
 3. Delete the section with connection properties of the data source.
 4. Save the configuration file.
 5. Start the lineage harvester in the console and run the following command to ignore the data source:
 - For Windows: `.\bin\lineage-harvester.bat ignore-source <source_ID>`, where `<source_id>` is the ID of the data source that you want to ignore.
 - For other operating systems: `./bin/lineage-harvester ignore-source <source_ID>`, where `<source_id>` is the ID of the data source that you want to ignore.
 - » The data source is excluded from the list of data sources that are used to create the technical lineage.
 6. Synchronize the technical lineage by running any of the following commands:
 - The `sync` command:
 - For Windows: `.\bin\lineage-harvester.bat sync`
 - For other operating systems: `./bin/lineage-harvester sync`
 - The `full-sync` command:
 - For Windows: `.\bin\lineage-harvester.bat full-sync`
 - For other operating systems: `./bin/lineage-harvester full-sync`
- For more information, go to [Lineage harvesting app command options and arguments](#).
7. When prompted, enter the password to connect to your Collibra Data Intelligence Cloud and data sources in the configuration file.
 - » The lineage harvester uploads the metadata of the remaining data sources in the configuration file to the Collibra Data Lineage service.
 - » The Collibra Data Lineage service synchronizes the technical lineage and removes the deleted data source from the technical lineage graph.

What's next

You can view the technical lineage. For more information, go to [Technical lineage viewer](#).

You can check the progress of the technical lineage creation in [Activities](#) in your Collibra Data Intelligence Cloud environment. The **Results** field indicates how many relations were imported into Data Catalog. Go to the [status page](#) to see the log files of the SQL analysis.

If the lineage harvester log shows an error message or the [harvesting process](#) fails, you can use the technical lineage common errors and issues in [Collibra Support Portal](#) to fix the error.

Working with BI tools

This section addresses BI tool-specific integration concepts and tasks for Collibra Admins.

BI tool operating models

This section shows the BI tool operating models and related information.

Note Keep in mind, it is possible that your organization has renamed the out-of-the-box asset types and characteristics.

Steps

- [Overview and diagram view](#)
- [Harvested metadata per asset type](#)
- [Example of ingested Tableau metadata](#)
- [Recommended hierarchy within a domain](#)
- [Create a Tableau operating model diagram view](#)

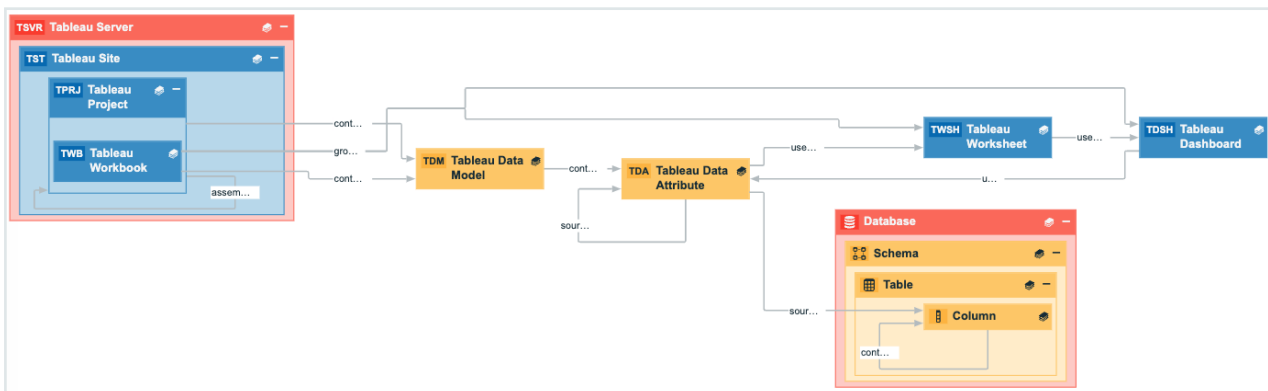
Overview and diagram view

Synchronizing means refreshing the assets that are currently in Data Catalog as a result of a previous ingestion or synchronization job. After synchronizing Tableau, the assets in Data Catalog accurately reflect the metadata as it exists at the time of synchronization.

Note

- The assets have the same names as their counterparts in Tableau.
- Some asset types are only created if the Tableau user has specific permissions.
- Relations that were created between Tableau assets and other assets via a relation type in the Tableau operating model, are deleted upon synchronization. The same is true of any attribute types in the operating model that you add to Tableau assets. To ensure that the characteristics you add to Tableau assets are not deleted upon synchronization, be sure to use characteristics that are not part of the Tableau operating model.

The following image shows the relations between Tableau asset types.



You can easily recreate this diagram view in your Collibra environment. See [Create a Tableau operating model diagram view](#).

Harvested metadata per asset type

This table shows the metadata for each Tableau asset type and the resource ID for each asset type and metadata.

Asset type	Synchronized metadata	Resource ID
Tableau Server Resource ID: 00000000-0000-0000-0000-110000000005	Description	00000000-0000-0000-0000-000000003114
	Server hosts / is hosted in Business Dimension	00000000-0000-0000-0000-120000000000
	URL: The link to the data in Tableau	00000000-0000-0000-0000-000000000258
Tableau Site Resource ID: 00000000-0000-0000-0000-110000000000	Description	00000000-0000-0000-0000-000000003114
	BI Folder assemblies / Is assembled in BI Folder	00000000-0000-0000-0000-120000000001
	Server hosts / is hosted in Business Dimension	00000000-0000-0000-0000-120000000000
	URL: The link to the data in Tableau	00000000-0000-0000-0000-000000000258

Asset type	Synchronized metadata	Resource ID
Tableau Project Resource ID: 00000000-0000-0000-0000-110000000001	Description	00000000-0000-0000-0000-000000003114
	Owner in source The only harvested metadata are email addresses. To harvest this metadata, you need to enable the Metadata API by setting the <code>restOnly</code> property in your lineage harvester configuration file to <code>false</code> . For additional rules and limitations, see Additional information , further down in this topic.	00000000-0000-0000-0000-200000000001
	BI Folder assembles / is assembled in BI Folder	00000000-0000-0000-0000-120000000001
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	BI Folder contains / contained in Data Asset	00000000-0000-0000-0000-120000000014

Asset type	Synchronized metadata	Resource ID
Tableau Workbook Resource ID: 00000000-0000-0000-0000-110000000002	Description	00000000-0000-0000-0000-000000003114
	Document creation date	00000000-0000-0000-0000-000000000260
	Document modification date	00000000-0000-0000-0000-000000000261
	Document size	00000000-0000-0000-0000-000000000259
	Owner in source The only harvested metadata are email addresses. To harvest this metadata, you need to enable the Metadata API by setting the <code>restOnly</code> property in your lineage harvester configuration file to <code>false</code> . For additional rules and limitations, see Additional information , further down in this topic.	00000000-0000-0000-0000-200000000001
	URL: The link to the data in Tableau	00000000-0000-0000-0000-000000000258
	Visits count This is the amount of times the workbook was viewed in Tableau.	00000000-0000-0000-0000-000000000264
	Report groups / is grouped into Report	00000000-0000-0000-0000-120000000004
	Tableau Workbook contains / contained in Tableau Data Model	00000000-0000-0000-0000-120000000020
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002

Asset type	Synchronized metadata	Resource ID
Tableau Dashboard Resource ID: 00000000-0000-0000-0001-110000000301 Assets of this type are only created if the Tableau user has the Download/Save As permission on the workbook.		

Asset type	Synchronized metadata	Resource ID
	Document creation date	00000000-0000-0000-0000-000000000260
	Document modification date	00000000-0000-0000-0000-000000000261
	Owner in source The only harvested metadata are email addresses. To harvest this metadata, you need to enable the Metadata API by setting the <code>restOnly</code> property in your lineage harvester configuration file to <code>false</code>. For additional rules and limitations, see Additional information, further down in this topic.	00000000-0000-0000-0000-200000000001
	Report image: The image of the report. Images are downloaded and stored in Data Catalog. You can configure the maximum file size and content types of the Tableau images in the Collibra DGC service settings.	00000000-0000-0000-0000-000000000262
	URL: The link to the data in Tableau	00000000-0000-0000-0000-000000000258
	Visible on server	00000000-0000-0000-0000-000000000265
	Visits count This is the amount of times the dashboard was viewed in Tableau.	00000000-0000-0000-0000-000000000264

Asset type	Synchronized metadata	Resource ID
	Report groups / is grouped into Report	00000000-0000-0000-0000-120000000004
	Report uses / used in Data Attribute	00000000-0000-0000-0000-120000000021
	Report uses / used in Report	00000000-0000-0000-0000-120000000007

Asset type	Synchronized metadata	Resource ID
Tableau Worksheet Resource ID: 00000000-0000-0000-0001-110000000300 Assets of this type are only created if the Tableau user has the Download/Save As permission on the workbook.	Document creation date	00000000-0000-0000-0000-000000000260
	Document modification date	00000000-0000-0000-0000-000000000261
	Report image: The image of the report. Images are downloaded and stored in Data Catalog. You can configure the maximum file size and content types of the Tableau images in the Colibra DGC service settings.	00000000-0000-0000-0000-000000000262
	URL: The link to the data in Tableau	00000000-0000-0000-0000-000000000258
	Visible on server	00000000-0000-0000-0000-000000000265
	Visits count This is the amount of times the worksheet was viewed in Tableau.	00000000-0000-0000-0000-000000000264
	Report groups / is grouped into Report	00000000-0000-0000-0000-120000000004
	Report uses / used in Data Attribute	00000000-0000-0000-0000-120000000021
	Report uses / used in Report	00000000-0000-0000-0000-120000000007

Asset type	Synchronized metadata	Resource ID
Tableau Data Attribute Resource ID: 00000000-0000-0000-0000-110000000010 Assets of this type are only created if the Tableau user has the Download/Save As permission on the data source.	Description	00000000-0000-0000-0000-000000003114
	Calculation Rule	00000000-0000-0000-0000-000000003117
	Data Type: The data type of a data asset, as it is declared by the data source.	00000000-0000-0000-0001-000500000005
	Role in Report	00000000-0000-0000-0000-000000000266
	BI Data Model contains / is part of BI Data Attribute	00000000-0000-0000-0000-000000007196
	Data Element targets / sources Data Element	00000000-0000-0000-0000-000000007069
	Report uses / used in Data Attribute	00000000-0000-0000-0000-120000000021

Asset type	Synchronized metadata	Resource ID
Tableau Data Model Resource ID: 00000000-0000-0000-0000-110000000008 Assets of this type are only created if the Tableau user has the Download/Save As permission on the data source.	Description	00000000-0000-0000-0000-000000003114
	Certified Note Certification is only possible for published Tableau data sources.	00000000-0000-0000-0001-000500000001
	Document creation date	00000000-0000-0000-0000-000000000260
	Document modification date	00000000-0000-0000-0000-000000000261
	Original Name: The name of the data source in Tableau	00000000-0000-0000-0001-000500000032
	Owner in source The only harvested metadata are email addresses. To harvest this metadata, you need to enable the Metadata API by setting the <code>restOnly</code> property in your lineage harvester configuration file to <code>false</code> . For additional rules and limitations, see Additional information , further down in this topic.	00000000-0000-0000-0000-200000000001
	BI Data Model contains / is part of BI Data Attribute	00000000-0000-0000-0000-000000007196
	Business Dimension source / is source of System	00000000-0000-0000-0000-120000000003
	Tableau Workbook contains / contained in Tableau Data Model	00000000-0000-0000-0000-120000000020

Asset type	Synchronized metadata	Resource ID
	Data Asset contained in / contains BI Folder	00000000-0000-0000-0000-120000000014

Additional information

For the Owner in source attribute, the following rules apply:

- If the system creates a Tableau data object and the Tableau data object does not have a user ID, the Owner in source attribute is shown as System on the asset page.
- If the user who created a Tableau data object no longer exists, the Owner in source attribute is shown as empty on the asset page.

Note The Owner in source metadata (email address) is only returned if the user is the owner of the data inside the project. If the user is the owner of the project, but not the data in the project, the metadata is not returned. This is a known issue that is specific to the Tableau Metadata API.

Example of ingested Tableau metadata

The following image shows an example structure after synchronizing Tableau.

Name	Description	Domain Type
Business Analysts Community		
New Tableau		
New Tableau		BI Catalog
Schemas	Community containing all inge...	
Tableau		
Tableau > Annual reporting		
Annual financial reporting		BI Catalog
Default		BI Catalog
Tableau > Management reporting		
Tableau > Wholesale reporting		
Tableau		BI Catalog

Recommended hierarchy within a domain

You can enable [hierarchies](#) for the domain (or domains) in which your Tableau assets were ingested. Doing so makes it easier to understand the relation between your Tableau assets, when viewing the assets on the domain page.

Follow these steps to enable and configure the recommended hierarchy.

Steps

1. Open the domain page of the relevant BI Catalog domain.
2. On the content toolbar, click .
- » The **Configure Hierarchy** dialog box appears.
3. Select **Enable Hierarchy**.
4. Select **Multipath**.
5. Start typing and select each of the following relation types:
 - Server **hosts** Business Dimension
 - BI Folder **assembles** BI Folder
 - Business Dimension **groups** Report
 - Report **groups** Report
 - Report **uses** Report
 - Report **uses** Data Attribute
 - BI Folder **contains** Data Asset
 - BI Data Model **contains** BI Data Attribute
 - Tableau Workbook **contains** Tableau Data Model
6. Click **Apply**.

Note

- In an asset view, if any asset is deleted, for example via synchronization or manual deletion, the view is recreated and the hierarchy is lost. In this case, you can again enable and configure the recommended hierarchy.
- When viewing the hierarchy for a community or domain, if the parent of a node that is in the community or domain belongs to a different community or domain, that node is not shown in the hierarchy.

Create a Tableau operating model diagram view

You can create a Tableau-specific diagram view, to visualize the operating model. The following procedure provides instruction on how to quickly create a new diagram view by copying and pasting the JSON code in the diagram view text editor.

Steps

1. Open an asset page.
2. Click the **Diagram** tab.
 - » The diagram is shown in the default [diagram view](#).
3. Click **+** to add a new view.
4. Click the **Text** tab, to switch to the diagram view text editor.
5. Click **Show me the JSON code** below this procedure, to expand the code.
6. Paste the code in diagram view text editor.
7. Click **Save**.
8. [Edit](#) the name and description of the diagram view, to suit your needs.

Show me the JSON code

```
{
  "nodes": [
    {
      "id": "Tableau Workbook",
      "type": {
        "id": "00000000-0000-0000-0000-110000000002"
      },
      "layoutRegion": "context"
    },
    {
      "id": "Tableau Dashboard",
      "type": {
        "id": "00000000-0000-0000-0001-110000000301"
      },
      "layoutRegion": "context"
    },
    {
      "id": "Tableau Worksheet",
      "type": {
        "id": "00000000-0000-0000-0001-110000000300"
      },
      "layoutRegion": "context"
    }
  ]
}
```

```

    },
    {
      "id": "Tableau Data Model",
      "type": {
        "id": "00000000-0000-0000-0000-110000000008"
      },
      "layoutRegion": "context",
      "editorSettings": {
        "nodePropsExpanded": false
      }
    },
    {
      "id": "Tableau Project",
      "type": {
        "id": "00000000-0000-0000-0000-110000000001"
      },
      "layoutRegion": "context"
    },
    {
      "id": "Tableau Site",
      "type": {
        "id": "00000000-0000-0000-0000-110000000000"
      },
      "layoutRegion": "context"
    },
    {
      "id": "Tableau Server",
      "type": {
        "id": "00000000-0000-0000-0000-110000000005"
      },
      "layoutRegion": "context"
    },
    {
      "id": "Tableau Data Attribute",
      "type": {
        "id": "00000000-0000-0000-0000-110000000010"
      },
      "layoutRegion": "context"
    },
    {
      "id": "Column",
      "type": {
        "id": "00000000-0000-0000-0000-000000031008"
      },
      "layoutRegion": "context"
    },
    {
      "id": "Table",
      "type": {
        "id": "00000000-0000-0000-0000-000000031007"
      }
    }
  ],
  {
    "id": "Table",
    "type": {
      "id": "00000000-0000-0000-0000-000000031007"
    }
  }
]

```

```

    },
    "layoutRegion": "context"
  },
  {
    "id": "Schema",
    "type": {
      "id": "000000000-0000-0000-0001-000400000002"
    },
    "layoutRegion": "context"
  },
  {
    "id": "Database",
    "type": {
      "id": "000000000-0000-0000-0000-0000000031006"
    },
    "layoutRegion": "context"
  }
],
"edges": [
  {
    "from": "Tableau Project",
    "to": "Tableau Workbook",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-120000000002"
    },
    "roleDirection": true
  },
  {
    "from": "Tableau Site",
    "to": "Tableau Project",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-120000000001"
    },
    "roleDirection": true
  },
  {
    "from": "Tableau Server",
    "to": "Tableau Site",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-120000000000"
    },
    "roleDirection": true
  },
  {

```

```

    "from": "Tableau Data Model",
    "to": "Tableau Data Attribute",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "00000000-0000-0000-0000-000000007196"
    },
    "roleDirection": true
  },
  {
    "from": "Tableau Data Attribute",
    "to": "Tableau Data Attribute",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "00000000-0000-0000-0000-000000007069"
    },
    "roleDirection": false
  },
  {
    "from": "Tableau Workbook",
    "to": "Tableau Data Model",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "00000000-0000-0000-0000-120000000020"
    },
    "roleDirection": true
  },
  {
    "from": "Tableau Project",
    "to": "Tableau Data Model",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "00000000-0000-0000-0000-120000000014"
    },
    "roleDirection": true
  },
  {
    "from": "Column",
    "to": "Column",
    "label": "",
    "style": "boxing",
    "type": {
      "id": "00000000-0000-0000-0000-000000007042"
    },
    "roleDirection": false
  },
  {

```

```

        "from": "Column",
        "to": "Table",
        "label": "",
        "style": "boxed",
        "type": {
            "id": "00000000-0000-0000-0000-000000007042"
        },
        "roleDirection": true
    },
    {
        "from": "Table",
        "to": "Schema",
        "label": "",
        "style": "boxed",
        "type": {
            "id": "00000000-0000-0000-0000-000000007043"
        },
        "roleDirection": false
    },
    {
        "from": "Schema",
        "to": "Database",
        "label": "",
        "style": "boxed",
        "type": {
            "id": "00000000-0000-0000-0000-000000007024"
        },
        "roleDirection": false
    },
    {
        "from": "Tableau Data Attribute",
        "to": "Tableau Worksheet",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000021"
        },
        "roleDirection": false
    },
    {
        "from": "Tableau Workbook",
        "to": "Tableau Worksheet",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000004"
        },
        "roleDirection": true
    },
    {

```

```

        "from": "Tableau Workbook",
        "to": "Tableau Dashboard",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000004"
        },
        "roleDirection": true
    },
    {
        "from": "Tableau Worksheet",
        "to": "Tableau Dashboard",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000007"
        },
        "roleDirection": false
    },
    {
        "from": "Tableau Data Attribute",
        "to": "Column",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-000000007069"
        },
        "roleDirection": false
    },
    {
        "from": "Tableau Project",
        "to": "Tableau Project",
        "label": "",
        "style": "boxed",
        "type": {
            "id": "00000000-0000-0000-0000-120000000001"
        },
        "roleDirection": true
    },
    {
        "from": "Tableau Dashboard",
        "to": "Tableau Data Attribute",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000012"
        },
        "roleDirection": true
    }
],

```

```

"showOverview": false,
"enableFilters": true,
"showLabels": true,
"showFields": true,
"showLegend": true,
"showPreview": true,
"visitStrategy": "directed",
"layout": "HierarchyTopBottom",
"maxNodeLabelLength": 50,
"maxEdgeLabelLength": 30,
"layoutOptions": {
  "compactGroups": false,
  "componentArrangementPolicy": "topmost",
  "edgeBends": true,
  "edgeBundling": true,
  "edgeToEdgeDistance": 5,
  "minimumLayerDistance": "auto",
  "nodeToEdgeDistance": 5,
  "orthogonalRouting": true,
  "preciseNodeHeightCalculation": true,
  "recursiveGroupLayering": true,
  "separateLayers": true,
  "webWorkers": true,
  "nodePlacer": {
    "barycenterMode": true,
    "breakLongSegments": true,
    "groupCompactionStrategy": "none",
    "nodeCompaction": false,
    "straightenEdges": true
  }
}
}

```

- [Overview and diagram view](#)
- [Harvested metadata per asset type](#)
- [Example of ingested Power BI metadata](#)
- [Recommended hierarchy within a domain](#)
- [Create a Power BI operating model diagram view](#)

Overview and diagram view

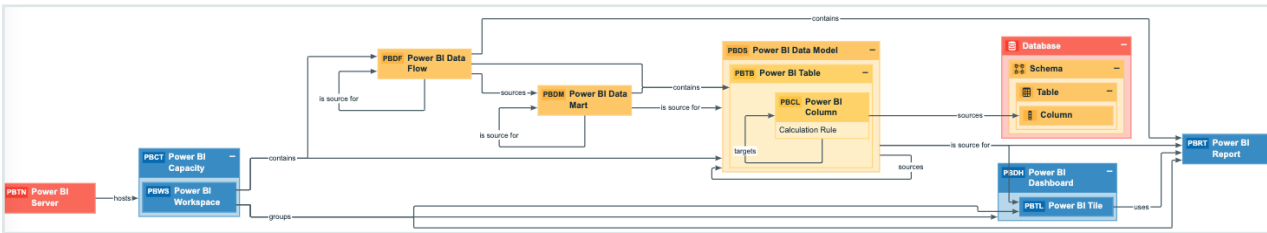
The lineage harvester collects Power BI metadata and sends it to the [Collibra Data Lineage service instances](#). Collibra processes the metadata and creates new Power BI

assets and relations in Data Catalog. You can see them on the asset page overview or visualize them in a [diagram](#) or in a technical lineage.

Note

- The assets have the same names as their counterparts in Power BI. Full names and Display names cannot be changed in Data Catalog.
- Asset types are only created if you have all specific Power BI and Data Catalog permissions.
- The Power BI assets are created in the domain (or domains) that you specify in the lineage harvester configuration file.
- Relations that were created between Power BI assets and other assets via a relation type in the Power BI operating model, are deleted upon synchronization. The same is true of any attribute types in the operating model that you add to Power BI assets. To ensure that the characteristics you add to Power BI assets are not deleted upon synchronization, be sure to use characteristics that are not part of the Power BI operating model.

The following image shows the relations between Power BI asset types.



Harvested metadata per asset type

This table shows the harvested Power BI metadata for each Power BI asset type. This table also shows the resource ID for each asset type, attribute, and relation.

Asset type	Synchronized metadata	Resource ID
Power BI Capacity Resource ID: 00000000-0000-0000-0000-100000000002	Full name	
	Display name	
	Server hosts / is hosted in Business Dimension	00000000-0000-0000-0000-120000000000
	BI Folder assemblies / is assembled in BI Folder	00000000-0000-0000-0000-120000000001
Power BI Column Resource ID: 00000000-0000-0000-0000-100000000008	Full name	
	Display name	
	Description	00000000-0000-0000-0000-000000003114
	Calculation Rule	00000000-0000-0000-0000-000000003117
	Role in Report	00000000-0000-0000-0000-000000000266
	Technical Data Type	00000000-0000-0000-0000-000000000219
	BI Data Model contains / is part of BI Data Attribute	00000000-0000-0000-0000-000000007196
	Data Element targets / sources Data Element	00000000-0000-0000-0000-000000007069
	Data Entity contains / is part of Data Attribute	00000000-0000-0000-0000-000000007047

Asset type	Synchronized metadata	Resource ID
Power BI Dashboard Resource ID: 00000000-0000-0000-0000-100000000004	Full name	
	Display name	
	URL Note If the dashboard is part of an app in Power BI, the URL on the asset page links to the dashboard in the Power BI app.	00000000-0000-0000-0000-000000000258
	Data Asset is source for / sources for BI Report	00000000-0000-0000-0000-120000000013
	Report uses / used in Report	00000000-0000-0000-0000-120000000007
	Report related to / is impacted by Business Asset	00000000-0000-0000-0000-120000000006
Power BI Data Flow Resource ID: 00000000-0000-0000-0000-100000000010	Full name	
	Display name	
	Description	00000000-0000-0000-0000-000000003114
	Owner in source The only harvested metadata are the email addresses of all Admins and one Owner.	00000000-0000-0000-0000-200000000001
	BI Folder contains / contained in Data Asset	00000000-0000-0000-0000-120000000014
	Data Entity is part of / contains Data Model	00000000-0000-0000-0000-000000007046
	BI Data Model is source for / sources BI Data Model	00000000-0000-0000-0000-120000000022

Asset type	Synchronized metadata	Resource ID
Power BI Data Mart Resource ID: 00000000-0000-0000-0000-100000000052	Full name	
	Display name	
	Certified	00000000-0000-0000-0001-000500000001
	Document modification date	00000000-0000-0000-0000-000000000261
	URL	00000000-0000-0000-0000-000000000258
	Owner in source The only harvested metadata are email addresses.	00000000-0000-0000-0000-200000000001
	Data Asset is source for / sources BI Report	00000000-0000-0000-0000-120000000013
	BI Folder contains / contained in Data Asset	00000000-0000-0000-0000-120000000014
	Data Entity is part of / contains Data Model	00000000-0000-0000-0000-000000007046
	BI Data Model is source for / sources BI Data Model	00000000-0000-0000-0000-120000000022

Asset type	Synchronized metadata	Resource ID
Power BI Data Model Resource ID: 00000000-0000-0000-0000-100000000007	Full name	
	Display name	
	Description	00000000-0000-0000-0000-000000003114
	Owner in source The only harvested metadata are email addresses.	00000000-0000-0000-0000-200000000001
	Source type	00000000-0000-0000-0000-000000000230
	BI Data Model contains / is part of BI Data Attribute	00000000-0000-0000-0000-000000007196
	BI Folder contains / contained in Data Asset	00000000-0000-0000-0000-120000000014
	Data Asset is source for / sources BI report	00000000-0000-0000-0000-120000000013
	Data Entity is part of / contains Data Model	00000000-0000-0000-0000-000000007046
	BI Data Model is source for / sources BI Data Model	00000000-0000-0000-0000-120000000022

Asset type	Synchronized metadata	Resource ID
Power BI Report Resource ID: 00000000-0000-0000-0000-100000000006	Full name	
	Display name	
	Description	00000000-0000-0000-0000-000000003114
	Owner in source The only harvested metadata are the email addresses of all Admins and one Owner.	00000000-0000-0000-0000-200000000001
	Source type	00000000-0000-0000-0000-000000000230
	URL Note If the report is part of an app in Power BI, the URL on the asset page links to the report in the Power BI app.	00000000-0000-0000-0000-000000000258
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	Data Asset is source for / sources BI Report	00000000-0000-0000-0000-120000000013
	Report related to / is impacted by Business Asset	00000000-0000-0000-0000-120000000006
	Report uses / used in Report	00000000-0000-0000-0000-120000000007
Power BI Server Resource ID: 00000000-0000-0000-0000-100000000001	Full name	
	Display name	
	Server hosts / is hosted in Business Dimension	00000000-0000-0000-0000-120000000000

Asset type	Synchronized metadata	Resource ID
Power BI Table Resource ID: 00000000-0000-0000-0000-100000000009	Full name	
	Display name	
	Description	00000000-0000-0000-0000-000000003114
	Calculation Rule	00000000-0000-0000-0000-000000003117
	Data Entity contains / is part of Data Attribute	00000000-0000-0000-0000-000000007047
	Data Entity is part of / contains Data Model	00000000-0000-0000-0000-000000007046
Power BI Tile Resource ID: 00000000-0000-0000-0000-100000000005	Full name	
	Display name	
	URL	00000000-0000-0000-0000-000000000258
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	Data Asset is source for / sources BI Report	00000000-0000-0000-0000-120000000013
	Report related to / is impacted by Business Asset	00000000-0000-0000-0000-120000000006
	Report uses / used in Report	00000000-0000-0000-0000-120000000007

Asset type	Synchronized metadata	Resource ID
Power BI Workspace Resource ID: 00000000-0000-0000-0000-100000000003	Full name	
	Display name	
	Description	00000000-0000-0000-0000-000000003114
	State	00000000-0000-0000-0000-000000000227
	Owner in source The only harvested metadata are email addresses.	00000000-0000-0000-0000-200000000001
	BI Folder assemblies / is assembled in BI Folder	00000000-0000-0000-0000-120000000001
	BI Folder contains / contained in Data Asset	00000000-0000-0000-0000-120000000014
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002

Note The metadata that is shown on the assets' pages depends on the asset type's assignment. As a result, you might not see all harvested metadata on the asset's page by default.

Additional information

For the Owner in source attribute, the following rules apply:

- If the system creates a Power BI data object and the Power BI data object does not have a user ID, the Owner in source attribute is shown as System on the asset page.
- If the user who created a Power BI data object no longer exists, the Owner in source attribute is shown as empty on the asset page.

URLs to reports and dashboards that can't be found in Power BI

When you add a report or dashboard to an app in Power BI, what happens is that copies of the original report or dashboard is created in the app. The URL on the corresponding asset page in Collibra links directly to the copied report or dashboard in the app. However, if the name of the original report or dashboard changes, or if it has been deleted in Power BI, the copies in the app remain unchanged. Therefore, to remedy what would otherwise be links to outdated copies of reports or dashboards in Power BI, the URLs on the asset pages instead link to the Power BI app.

Example of ingested Power BI metadata

The following image shows an example structure after Power BI ingestion.

☐ Name ↑	Asset Type
☐ Sales Engineer Demo Server	Power BI Server
☐ presalespowerbiresource	Power BI Capacity
☐ Power BI Demo	Power BI Workspace
☐ Call Center Performance	Power BI Report
☐ Call Center Performance	Power BI Data Model
☐ CallCenterAggregates	Power BI Table
☐ CallCenterId	Power BI Column
☐ CallsReceived	Power BI Column
☐ OrdersReceived	Power BI Column
☐ Performance %	Power BI Column
☐ Customer Sales Report	Power BI Report
☐ Product Cost Statistics Report	Power BI Report

Recommended hierarchy within a domain

You can enable [hierarchies](#) for the domain (or domains) in which your Power BI assets are ingested. Doing so makes it easier to understand the relation between your Power BI assets, when viewing the assets on the domain page.

Follow these steps to enable and configure the recommended hierarchy.

Steps

1. Open the domain page of the relevant BI Catalog domain.
2. On the content toolbar, click .
- » The **Configure Hierarchy** dialog box appears.
3. Select **Enable Hierarchy**.
4. Select **Single path**.
5. Start typing and select each of the following relation types:
 - Server **hosts** Business Dimension
 - BI Folder **assembles** BI Folder
 - Business Dimension **groups** Report
 - BI Report **source** Data Asset
 - Data Model **contains** Data Entity
 - Data Entity **contains** Data Attribute
6. Click **Apply**.

Note

- In an asset view, if any asset is deleted, for example via synchronization or manual deletion, the view is recreated and the hierarchy is lost. In this case, you can again enable and configure the recommended hierarchy.
- When viewing the hierarchy for a community or domain, if the parent of a node that is in the community or domain belongs to a different community or domain, that node is not shown in the hierarchy.

Create a Power BI operating model diagram view

You can create a Power BI-specific diagram view, to visualize the operating model. The following procedure provides instruction on how to quickly create a new diagram view by copying and pasting the JSON code in the diagram view text editor.

Steps

1. Open an asset page.
2. Click the **Diagram** tab.- » The diagram is shown in the default [diagram view](#).
3. Click **+** to add a new view.

4. Click the **Text** tab, to switch to the diagram view text editor.
5. Click **Show me the JSON code** below this procedure, to expand the code.
6. Paste the code in diagram view text editor.
7. Click **Save**.
8. [Edit](#) the name and description of the diagram view, to suit your needs.

Show me the JSON code

```
{
  "nodes": [
    {
      "id": "Power BI Server",
      "type": {
        "id": "00000000-0000-0000-0000-100000000001"
      },
      "fields": []
    },
    {
      "id": "Power BI Capacity",
      "type": {
        "id": "00000000-0000-0000-0000-100000000002"
      }
    },
    {
      "id": "Power BI Workspace",
      "type": {
        "id": "00000000-0000-0000-0000-100000000003"
      }
    },
    {
      "id": "Power BI Dashboard",
      "type": {
        "id": "00000000-0000-0000-0000-100000000004"
      }
    },
    {
      "id": "Power BI Report",
      "type": {
        "id": "00000000-0000-0000-0000-100000000006"
      }
    },
    {
      "id": "Power BI Tile",
      "type": {
        "id": "00000000-0000-0000-0000-100000000005"
      }
    },
    {
      "id": "Power BI Data Model",
```

```

        "type": {
            "id": "00000000-0000-0000-0000-100000000007"
        }
    },
    {
        "id": "Power BI Data Flow",
        "type": {
            "id": "00000000-0000-0000-0000-100000000010"
        }
    },
    {
        "id": "Power BI Table",
        "type": {
            "id": "00000000-0000-0000-0000-100000000009"
        }
    },
    {
        "id": "Power BI Column",
        "type": {
            "id": "00000000-0000-0000-0000-100000000008"
        },
        "fields": [
            {
                "key": "stringAttribute_00000000-0000-0000-0000-0000000003117",
                "label": "Calculation Rule"
            }
        ]
    },
    {
        "id": "Column",
        "type": {
            "id": "00000000-0000-0000-0000-000000031008"
        }
    },
    {
        "id": "Table",
        "type": {
            "id": "00000000-0000-0000-0000-000000031007"
        }
    },
    {
        "id": "Schema",
        "type": {
            "id": "00000000-0000-0000-0001-000400000002"
        }
    },
    {
        "id": "Database",
        "type": {

```

```

        "id": "00000000-0000-0000-0000-000000031006"
    },
    {
        "id": "Power BI Data Mart",
        "type": {
            "id": "00000000-0000-0000-0000-100000000052"
        }
    }
],
"edges": [
    {
        "from": "Power BI Server",
        "to": "Power BI Capacity",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000000"
        },
        "roleDirection": true
    },
    {
        "from": "Power BI Capacity",
        "to": "Power BI Workspace",
        "label": "",
        "style": "boxing",
        "type": {
            "id": "00000000-0000-0000-0000-120000000001"
        },
        "roleDirection": true
    },
    {
        "from": "Power BI Workspace",
        "to": "Power BI Dashboard",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000002"
        },
        "roleDirection": true
    },
    {
        "from": "Power BI Workspace",
        "to": "Power BI Report",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000002"
        },
        "roleDirection": true
    }
]

```

```

    },
    {
      "from": "Power BI Workspace",
      "to": "Power BI Tile",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "00000000-0000-0000-0000-120000000002"
      },
      "roleDirection": true
    },
    {
      "from": "Power BI Workspace",
      "to": "Power BI Data Model",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "00000000-0000-0000-0000-120000000014"
      },
      "roleDirection": true
    },
    {
      "from": "Power BI Workspace",
      "to": "Power BI Data Flow",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "00000000-0000-0000-0000-120000000014"
      },
      "roleDirection": true
    },
    {
      "from": "Power BI Dashboard",
      "to": "Power BI Tile",
      "label": "",
      "style": "boxing",
      "type": {
        "id": "00000000-0000-0000-0000-120000000007"
      },
      "roleDirection": true
    },
    {
      "from": "Power BI Data Model",
      "to": "Power BI Tile",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "00000000-0000-0000-0000-120000000013"
      },
      "roleDirection": true
    }
  ],

```

```

    },
    {
      "from": "Power BI Data Model",
      "to": "Power BI Report",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "00000000-0000-0000-0000-120000000013"
      },
      "roleDirection": true
    },
    {
      "from": "Power BI Data Flow",
      "to": "Power BI Report",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "00000000-0000-0000-0000-000000007196"
      },
      "roleDirection": true
    },
    {
      "from": "Power BI Data Flow",
      "to": "Power BI Table",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "00000000-0000-0000-0000-120000000014"
      },
      "roleDirection": true
    },
    {
      "from": "Power BI Tile",
      "to": "Power BI Report",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "00000000-0000-0000-0000-120000000007"
      },
      "roleDirection": true
    },
    {
      "from": "Power BI Data Flow",
      "to": "Power BI Data Flow",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "00000000-0000-0000-0000-120000000022"
      },
      "roleDirection": true
    }
  ],

```

```

    },
    {
      "from": "Power BI Column",
      "to": "Column",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "00000000-0000-0000-0000-000000007069"
      },
      "roleDirection": false
    },
    {
      "from": "Column",
      "to": "Table",
      "label": "",
      "style": "boxed",
      "type": {
        "id": "00000000-0000-0000-0000-000000007042"
      },
      "roleDirection": true
    },
    {
      "from": "Table",
      "to": "Schema",
      "label": "",
      "style": "boxed",
      "type": {
        "id": "00000000-0000-0000-0000-000000007043"
      },
      "roleDirection": false
    },
    {
      "from": "Schema",
      "to": "Database",
      "label": "",
      "style": "boxed",
      "type": {
        "id": "00000000-0000-0000-0000-000000007024"
      },
      "roleDirection": false
    },
    {
      "from": "Power BI Data Flow",
      "to": "Power BI Data Mart",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "00000000-0000-0000-0000-120000000022"
      },
      "roleDirection": false
    }
  ]

```



```

    },
    {
      "from": "Power BI Data Mart",
      "to": "Power BI Data Mart",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "00000000-0000-0000-0000-120000000022"
      },
      "roleDirection": true
    },
    {
      "from": "Power BI Data Mart",
      "to": "Power BI Table",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "00000000-0000-0000-0000-120000000014"
      },
      "roleDirection": true
    },
    {
      "from": "Power BI Data Mart",
      "to": "Power BI Data Model",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "00000000-0000-0000-0000-120000000022"
      },
      "roleDirection": true
    },
    {
      "from": "Power BI Data Model",
      "to": "Power BI Data Model",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "00000000-0000-0000-0000-120000000022"
      },
      "roleDirection": false
    },
    {
      "from": "Power BI Table",
      "to": "Power BI Data Model",
      "label": "",
      "style": "boxed",
      "type": {
        "id": "00000000-0000-0000-0000-000000007046"
      },
      "roleDirection": true
    }
  ],

```

```

    },
    {
      "from": "Power BI Column",
      "to": "Power BI Table",
      "label": "",
      "style": "boxed",
      "type": {
        "id": "00000000-0000-0000-0000-000000007047"
      },
      "roleDirection": false
    },
    {
      "from": "Power BI Column",
      "to": "Power BI Column",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "00000000-0000-0000-0000-000000007069"
      },
      "roleDirection": true
    }
  ],
  "showOverview": false,
  "enableFilters": true,
  "showLabels": true,
  "showFields": true,
  "showLegend": true,
  "showPreview": true,
  "visitStrategy": "directed",
  "layout": "HierarchyTopBottom",
  "maxNodeLabelLength": 50,
  "maxEdgeLabelLength": 30,
  "layoutOptions": {
    "compactGroups": false,
    "componentArrangementPolicy": "topmost",
    "edgeBends": true,
    "edgeBundling": true,
    "edgeToEdgeDistance": 5,
    "minimumLayerDistance": "auto",
    "nodeToEdgeDistance": 5,
    "orthogonalRouting": true,
    "preciseNodeHeightCalculation": true,
    "recursiveGroupLayering": true,
    "separateLayers": true,
    "webWorkers": true,
    "nodePlacer": {
      "barycenterMode": true,
      "breakLongSegments": true,
      "groupCompactionStrategy": "none",
      "nodeCompaction": false,
    }
  }
}

```

```

    "straightenEdges": true
  }
}

```

- Overview and diagram view
- Harvested metadata per asset type
- Recommended hierarchy within a domain
- Create a MicroStrategy operating model diagram view

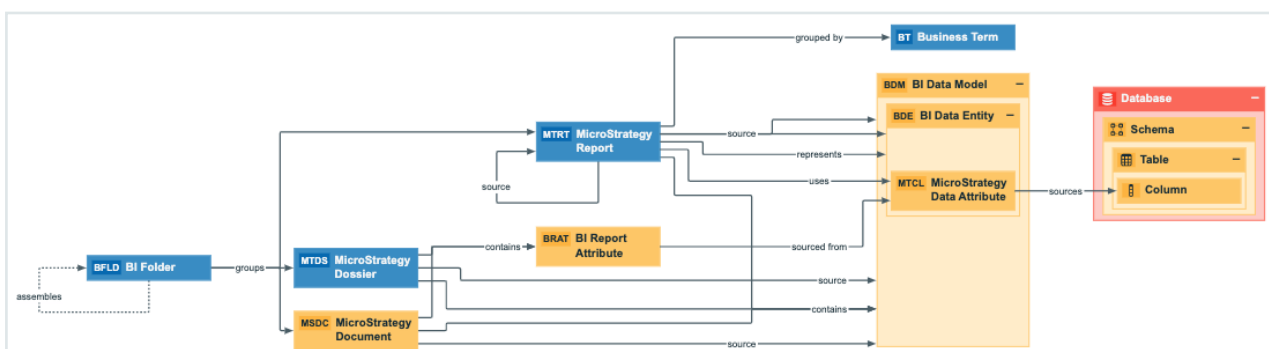
Overview and diagram view

When you integrate MicroStrategy, Collibra Data Lineage creates new MicroStrategy assets and relations in Data Catalog. You can see them on the asset page overview or visualize them in a [diagram](#).

Note

- The assets have the same names as their [corresponding data objects in MicroStrategy](#).
- Asset types are only created if you have all specific MicroStrategy and Data Catalog permissions.
- All MicroStrategy assets are created in the same domain.
- Relations that were manually created between MicroStrategy assets and other assets via a relation type in the MicroStrategy operating model, are deleted after synchronizing the MicroStrategy metadata.

The following image shows the relations between MicroStrategy asset types.



Note Currently, relations between MicroStrategy Documents, MicroStrategy Dossiers, and MicroStrategy Reports cannot be shown.

Harvested metadata per asset type

This table shows the harvested MicroStrategy metadata for assets of each MicroStrategy asset type, assuming you have the necessary subscriptions and configurations for a full ingestion.

Important To access MicroStrategy data, you have to use the In-memory Dataset connection method in MicroStrategy, not the Live Connect connection method. If the data is not stored in memory, the MicroStrategy APIs can't access it.

Note The following folders in MicroStrategy are not included in the ingestion:

- Object Templates
- System Objects
- Version Update History

Asset type	Harvested MicroStrategy metadata in Data Catalog	Resource ID
MicroStrategy Data Attribute Resource ID: 00000000-0000-0000-0000-100000000047	Description	00000000-0000-0000-0000-0000000003114
	Source Type	00000000-0000-0000-0000-000000000230
	Technical Data Type	00000000-0000-0000-0000-000000000219
	Data Element targets / sources Data Element	00000000-0000-0000-0000-0000000007069
	Data Entity contains / is part of Data Attribute	00000000-0000-0000-0000-0000000007047
	Data Attribute used in / uses Report	00000000-0000-0000-0000-1200000000021
MicroStrategy Data Entity Resource ID: 00000000-0000-0000-0000-100000000048	Description	00000000-0000-0000-0000-0000000003114
	Source Type	00000000-0000-0000-0000-000000000230
	Data Entity contains / is part of Data Attribute	00000000-0000-0000-0000-0000000007047
	Data Entity is part of / contains Data Model	00000000-0000-0000-0000-0000000007046
	Data Asset is source for / sources BI Report	00000000-0000-0000-0000-1200000000013

Asset type	Harvested MicroStrategy metadata in Data Catalog	Resource ID
MicroStrategy Data Model Resource ID: 00000000-0000-0000-0000-100000000046 Note If the data model is embedded in the project, Collibra Data Lineage automatically creates a dummy data model.	Description	00000000-0000-0000-0000-000000003114
	Certified	00000000-0000-0000-0001-000500000001
	Owner in source The only harvested metadata are email addresses.	00000000-0000-0000-0000-200000000001
	Source Type	00000000-0000-0000-0000-000000000230
	BI Folder contains / contained in Data Asset	00000000-0000-0000-0000-120000000014
	Data Asset is source for / sources BI Report	00000000-0000-0000-0000-120000000013
	Data Model contains / is part of Data Entity	00000000-0000-0000-0000-000000007046

Asset type	Harvested MicroStrategy metadata in Data Catalog	Resource ID
MicroStrategy Document Resource ID: 00000000-0000-0000-0000-100000000049	Description	00000000-0000-0000-0000-0000000003114
	Certified	00000000-0000-0000-0001-000500000001
	Owner in source The only harvested metadata are email addresses.	00000000-0000-0000-0000-200000000001
	URL Note You can specify the platform on which you run MicroStrategy, in your lineage harvester configuration file, to ensure the correct URL.	00000000-0000-0000-0000-000000000258
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	BI Report contains / contained in BI Data Model	00000000-0000-0000-0000-120000000015

Asset type	Harvested MicroStrategy metadata in Data Catalog	Resource ID
MicroStrategy Dossier Resource ID: 00000000-0000-0000-0000-100000000043	Description	00000000-0000-0000-0000-000000003114
	Certified	00000000-0000-0000-0001-000500000001
	Owner in source The only harvested metadata are email addresses.	00000000-0000-0000-0000-200000000001
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	Report uses / used in Report	00000000-0000-0000-0000-120000000007
MicroStrategy Folder Resource ID: 00000000-0000-0000-0000-100000000042	Description	00000000-0000-0000-0000-000000003114
	BI Folder assembles / is assembled in BI Folder	00000000-0000-0000-0000-120000000001
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	BI Folder contains / contained in Data Asset	00000000-0000-0000-0000-120000000014
MicroStrategy Project Resource ID: 00000000-0000-0000-0000-100000000041	Description	00000000-0000-0000-0000-000000003114
	BI Folder assembles / is assembled in BI Folder	00000000-0000-0000-0000-120000000001
	Server hosts / is hosted in Business Dimension	00000000-0000-0000-0000-120000000000

Asset type	Harvested MicroStrategy metadata in Data Catalog	Resource ID
MicroStrategy Report Resource ID: 00000000-0000-0000-0000-100000000044	Description	00000000-0000-0000-0000-000000003114
	Certified	00000000-0000-0000-0001-000500000001
	URL	00000000-0000-0000-0000-000000000258
	Owner in source The only harvested metadata are email addresses.	00000000-0000-0000-0000-200000000001
	Report groups / is grouped into Report	00000000-0000-0000-0000-120000000004
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	Data Asset is source for / sources BI Report	00000000-0000-0000-0000-120000000013
MicroStrategy Server Resource ID: 00000000-0000-0000-0000-100000000040	Server hosts / is hosted in Business Dimension	00000000-0000-0000-0000-120000000000

Recommended hierarchy within a domain

You can enable [hierarchies](#) for the domain in which your MicroStrategy assets are ingested. Doing so makes it easier to understand the relation between your MicroStrategy assets, when viewing the assets on the domain page.

Follow these steps to enable and configure the recommended hierarchy.

Steps

1. Open the domain page of the relevant BI Catalog domain.
2. On the content toolbar, click .
- » The **Configure Hierarchy** dialog box appears.
3. Select **Enable Hierarchy**.
4. Select **Single path**.
5. Start typing and select each of the following relation types:
 - Server **hosts** Business Dimension
 - BI Folder **assembles** BI Folder
 - Business Dimension **groups** Report
 - BI Report **source** Data Asset
 - BI Folder **contains** Data Asset
 - Data Model **contains** Data Entity
 - Data Entity **contains** Data Attribute
6. Click **Apply**.

Note

- In an asset view, if any asset is deleted, for example via synchronization or manual deletion, the view is recreated and the hierarchy is lost. In this case, you can again enable and configure the recommended hierarchy.
- When viewing the hierarchy for a community or domain, if the parent of a node that is in the community or domain belongs to a different community or domain, that node is not shown in the hierarchy.

Create a MicroStrategy operating model diagram view

You can create a MicroStrategy-specific diagram view, to visualize the operating model. The following procedure provides instruction on how to quickly create a new diagram view by copying and pasting the JSON code in the diagram view text editor.

Steps

1. Open an asset page.
2. Click the **Diagram** tab.
- » The diagram is shown in the default [diagram view](#).

3. Click **+** to add a new view.
4. Click the **Text** tab, to switch to the diagram view text editor.
5. Click **Show me the JSON code** below this procedure, to expand the code.
6. Paste the code in diagram view text editor.
7. Click **Save**.
8. [Edit](#) the name and description of the diagram view, to suit your needs.

Show me the JSON code

```
{
  "nodes": [
    {
      "id": "MicroStrategy Report",
      "type": {
        "id": "00000000-0000-0000-0000-100000000044"
      }
    },
    {
      "id": "MicroStrategy Data Attribute",
      "type": {
        "id": "00000000-0000-0000-0000-100000000047"
      },
      "editorSettings": {
        "edgePropsExpanded": true
      }
    },
    {
      "id": "BI Folder",
      "type": {
        "id": "00000000-0000-0000-0000-090000000002"
      },
      "display": "expanded"
    },
    {
      "id": "MicroStrategy Dossier",
      "type": {
        "id": "00000000-0000-0000-0000-100000000043"
      }
    },
    {
      "id": "MicroStrategy Document",
      "type": {
        "id": "00000000-0000-0000-0000-100000000049"
      }
    },
    {
      "id": "BI Report Attribute",
      "type": {
```

```

        "id": "00000000-0000-0000-0000-090000000004"
    },
    {
        "id": "BI Data Entity",
        "type": {
            "id": "00000000-0000-0000-0000-090000000007"
        }
    },
    {
        "id": "BI Data Model",
        "type": {
            "id": "00000000-0000-0000-0000-090000000008"
        }
    },
    {
        "id": "Column",
        "type": {
            "id": "00000000-0000-0000-0000-000000031008"
        }
    },
    {
        "id": "Table",
        "type": {
            "id": "00000000-0000-0000-0000-000000031007"
        }
    },
    {
        "id": "Schema",
        "type": {
            "id": "00000000-0000-0000-0001-000400000002"
        }
    },
    {
        "id": "Database",
        "type": {
            "id": "00000000-0000-0000-0000-000000031006"
        }
    },
    {
        "id": "Business Term",
        "type": {
            "id": "00000000-0000-0000-0000-000000011001"
        }
    }
],
"edges": [
    {
        "from": "MicroStrategy Report",
        "to": "MicroStrategy Data Attribute",

```

```

        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000021"
        },
        "roleDirection": true
    },
    {
        "from": "BI Folder",
        "to": "MicroStrategy Report",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000002"
        },
        "roleDirection": true
    },
    {
        "from": "BI Folder",
        "to": "BI Folder",
        "label": "",
        "style": "boxing",
        "type": {
            "id": "00000000-0000-0000-0000-120000000001"
        },
        "roleDirection": true
    },
    {
        "from": "BI Folder",
        "to": "MicroStrategy Dossier",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000002"
        },
        "roleDirection": true
    },
    {
        "from": "BI Folder",
        "to": "MicroStrategy Document",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000002"
        },
        "roleDirection": true
    },
    {
        "from": "MicroStrategy Dossier",
        "to": "BI Report Attribute",

```

```

        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-000000007058"
        },
        "roleDirection": false
    },
    {
        "from": "MicroStrategy Document",
        "to": "BI Report Attribute",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-000000007058"
        },
        "roleDirection": false
    },
    {
        "from": "BI Report Attribute",
        "to": "MicroStrategy Data Attribute",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000010"
        },
        "roleDirection": true
    },
    {
        "from": "BI Data Entity",
        "to": "MicroStrategy Data Attribute",
        "label": "",
        "style": "boxing",
        "type": {
            "id": "00000000-0000-0000-0000-000000007047"
        },
        "roleDirection": true
    },
    {
        "from": "BI Data Model",
        "to": "BI Data Entity",
        "label": "",
        "style": "boxing",
        "type": {
            "id": "00000000-0000-0000-0000-000000007046"
        },
        "roleDirection": false
    },
    {
        "from": "MicroStrategy Report",
        "to": "MicroStrategy Report",

```

```

        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000013"
        },
        "roleDirection": false
    },
    {
        "from": "MicroStrategy Report",
        "to": "BI Data Model",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000013"
        },
        "roleDirection": false
    },
    {
        "from": "MicroStrategy Document",
        "to": "BI Data Model",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000013"
        },
        "roleDirection": false
    },
    {
        "from": "MicroStrategy Dossier",
        "to": "BI Data Model",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000013"
        },
        "roleDirection": false
    },
    {
        "from": "MicroStrategy Data Attribute",
        "to": "Column",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-000000007069"
        },
        "roleDirection": false
    },
    {
        "from": "MicroStrategy Dossier",
        "to": "BI Data Model",

```

```

        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000015"
        },
        "roleDirection": true
    },
    {
        "from": "MicroStrategy Document",
        "to": "BI Data Model",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000015"
        },
        "roleDirection": true
    },
    {
        "from": "MicroStrategy Report",
        "to": "BI Data Model",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000015"
        },
        "roleDirection": true
    },
    {
        "from": "MicroStrategy Report",
        "to": "BI Data Entity",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-000000007038"
        },
        "roleDirection": true
    },
    {
        "from": "Column",
        "to": "Table",
        "label": "",
        "style": "boxed",
        "type": {
            "id": "00000000-0000-0000-0000-000000007042"
        },
        "roleDirection": true
    },
    {
        "from": "Table",
        "to": "Schema",

```



```

        "label": "",
        "style": "boxed",
        "type": {
            "id": "00000000-0000-0000-0000-000000007043"
        },
        "roleDirection": false
    },
    {
        "from": "Schema",
        "to": "Database",
        "label": "",
        "style": "boxed",
        "type": {
            "id": "00000000-0000-0000-0000-000000007024"
        },
        "roleDirection": false
    },
    {
        "from": "MicroStrategy Report",
        "to": "Business Term",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-000000007021"
        },
        "roleDirection": false
    },
    {
        "from": "MicroStrategy Report",
        "to": "BI Data Entity",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000013"
        },
        "roleDirection": false
    }
},
"showOverview": false,
"enableFilters": true,
"showLabels": false,
"showFields": true,
"showLegend": true,
"showPreview": true,
"visitStrategy": "directed",
"layout": "HierarchyLeftRight",
"maxNodeLabelLength": 50,
"maxEdgeLabelLength": 30,
"layoutOptions": {
    "compactGroups": false,

```

```

    "componentArrangementPolicy": "topmost",
    "edgeBends": true,
    "edgeBundling": true,
    "edgeToEdgeDistance": 5,
    "minimumLayerDistance": "auto",
    "nodeToEdgeDistance": 5,
    "orthogonalRouting": true,
    "preciseNodeHeightCalculation": true,
    "recursiveGroupLayering": true,
    "separateLayers": true,
    "webWorkers": true,
    "nodePlacer": {
      "barycenterMode": true,
      "breakLongSegments": true,
      "groupCompactionStrategy": "none",
      "nodeCompaction": false,
      "straightenEdges": true
    }
  }
}

```

- [Overview and diagram view](#)
- [Harvested metadata per asset type](#)
- [Example of ingested Looker metadata](#)
- [Recommended hierarchy within a domain](#)
- [Create a Looker operating model diagram view](#)

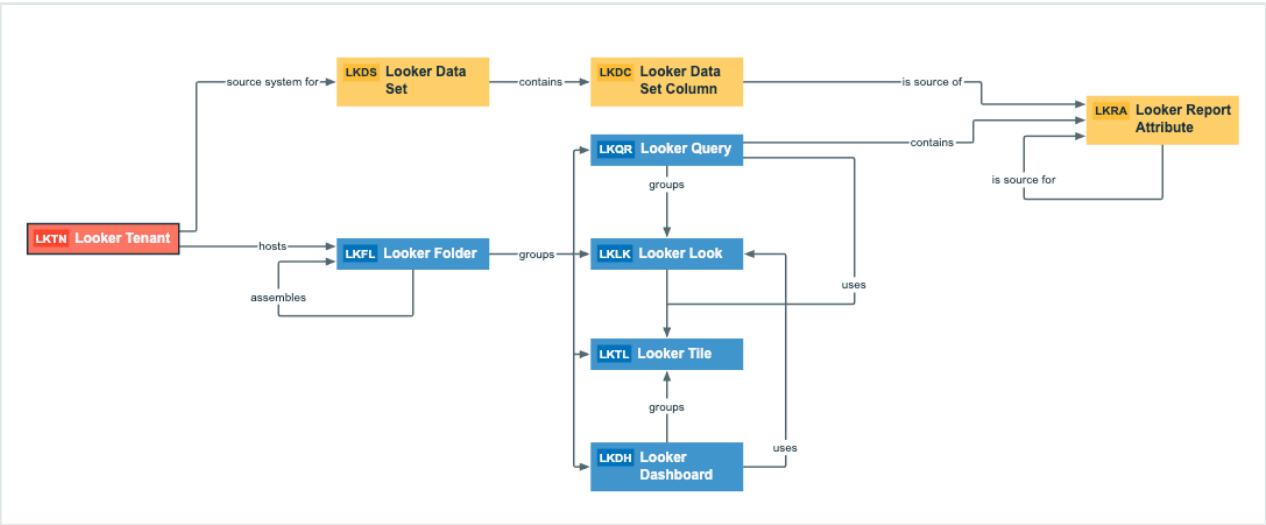
Overview and diagram view

The Looker scanner collects Looker metadata and sends it to the Collibra Data Lineage service. Collibra processes the metadata and creates new Looker assets and relations in Data Catalog. You can see them on the asset page overview or visualize them in a [diagram](#) or in a [technical lineage](#).

Note

- The assets have the same names as their counterparts in Looker. Full names and Display names cannot be changed in Data Catalog.
- Asset types are only created if you have all specific Looker and Data Catalog permissions.
- All Looker asset types are created in the same domain.
- Relations that were manually created between Looker assets and other assets via a relation type in the Looker operating model are deleted after a refresh of the Looker metadata.

The following image shows the relations between Looker asset types.



Harvested metadata per asset type

The following table shows the harvested Looker metadata for each Looker asset type. This table also shows the resource ID for each asset type and metadata.

Asset type	Synchronized metadata	Resource ID
Looker Dashboard Resource ID: 00000000-0000-0000-0000-100000000013	Full name	
	Display name	
	Description	00000000-0000-0000-0000-0000000003114
	Document creation date	00000000-0000-0000-0000-000000000260
	Document last accessed date	00000000-0000-0000-0000-000000000268
	Favorites count	00000000-0000-0000-0000-000000000269
	Owner in source The only harvested metadata are email addresses.	00000000-0000-0000-0000-200000000001
	Technical Data Type	00000000-0000-0000-0000-000000000219
	URL	00000000-0000-0000-0000-000000000258
	Visit count	00000000-0000-0000-0000-000000000264
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	Report groups / is grouped into Report	00000000-0000-0000-0000-120000000004
	Report related to / is impacted by Business Asset	00000000-0000-0000-0000-120000000006
	Report uses / used in Report	00000000-0000-0000-0000-120000000007

Asset type	Synchronized metadata	Resource ID
Looker Data Set Resource ID: 00000000-0000-0000-0000-100000000017	Full name	
	Display name	
	Description	00000000-0000-0000-0000-000000003114
	Data Set contains / is part of Data Element	00000000-0000-0000-0000-000000007062
	Technology Asset source system for / source system Data Asset	00000000-0000-0000-0000-000000007050
Looker Data Set Column Resource ID: 00000000-0000-0000-0000-100000000018	Full name	
	Display name	
	Description	00000000-0000-0000-0000-000000003114
	Data Set contains / is part of Data Element	00000000-0000-0000-0000-000000007062
	Report Attribute sourced from / is source of Data Attribute	00000000-0000-0000-0000-120000000010

Asset type	Synchronized metadata	Resource ID
Looker Folder Resource ID: 00000000-0000-0000-0000-100000000012	Full name	
	Display name	
	Document creation date	00000000-0000-0000-0000-000000000260
	Owner in source The only harvested metadata are email addresses.	00000000-0000-0000-0000-200000000001
	BI Folder assembles / is assembled in BI Folder	00000000-0000-0000-0000-120000000001
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	Server hosts / is hosted in Business Dimension	00000000-0000-0000-0000-120000000000

Asset type	Synchronized metadata	Resource ID
Looker Look Resource ID: 00000000-0000-0000-0000-100000000014	Full name	
	Display name	
	Description	00000000-0000-0000-0000-0000000003114
	Document creation date	00000000-0000-0000-0000-000000000260
	Document last accessed date	00000000-0000-0000-0000-000000000268
	Document modification date	00000000-0000-0000-0000-000000000261
	Favorites count	00000000-0000-0000-0000-000000000269
	Owner in source The only harvested metadata are email addresses.	00000000-0000-0000-0000-200000000001
	Report image	00000000-0000-0000-0000-000000000262
	URL	00000000-0000-0000-0000-000000000258
	Visit count	00000000-0000-0000-0000-000000000264
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	Report groups / is grouped into Report	00000000-0000-0000-0000-120000000004
	Report uses / used in Report	00000000-0000-0000-0000-120000000007

Asset type	Synchronized metadata	Resource ID
Looker Report Attribute Resource ID: 00000000-0000-0000-0000-100000000019	Full name	
	Display name	
	Report Attribute contained in / contains Report	00000000-0000-0000-0000-000000007058
	Report Attribute sourced from / is source of Data Attribute	00000000-0000-0000-0000-120000000010
Looker Query Resource ID: 00000000-0000-0000-0000-100000000016	Full name	
	Display name	
	URL	00000000-0000-0000-0000-000000000258
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	Report Attribute contained in / contains Report	00000000-0000-0000-0000-000000007058
	Report uses / used in Report	00000000-0000-0000-0000-120000000007
Looker Tenant Resource ID: 00000000-0000-0000-0000-100000000011	Full name	
	Display name	
	Description	00000000-0000-0000-0000-000000003114
	Server hosts / is hosted in Business Dimension	00000000-0000-0000-0000-120000000000
	Technology Asset source system for / source system Data Asset	00000000-0000-0000-0000-000000007050

Asset type	Synchronized metadata	Resource ID
Looker Tile Resource ID: 00000000-0000-0000-0000-100000000015	Full name	
	Display name	
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	Report uses / used in Report	00000000-0000-0000-0000-120000000007

Note The metadata that is shown on the assets' pages depends on the asset type's assignment. As a result, you might not see all harvested metadata on the asset's page by default.

Additional information

For the Owner in source attribute, the following rules apply:

- If the system creates a Looker data object and the Looker data object does not have a user ID, the Owner in source attribute is shown as System on the asset page.
- If the user who created a Looker data object no longer exists, the Owner in source attribute is shown as empty on the asset page.

Example of ingested Looker metadata

The following image shows an example structure after Looker ingestion.

 Name ↑	Status	Asset Type
 1 explore	Candidate	Looker Tile
 2 different explores	Candidate	Looker Tile
 30 Day Repeat Purchase Rate	Candidate	Looker Tile
 actor	Candidate	Looker Data Set
 actor.actor_id	Candidate	Looker Report Attribute
 actor.actor_id	Candidate	Looker Report Attribute
 actor.actor_id	Candidate	Looker Report Attribute
 Actor Actor ID	Candidate	Looker Data Set Column
 actor.count	Candidate	Looker Report Attribute
 actor.count	Candidate	Looker Report Attribute
 Actor Count	Candidate	Looker Data Set Column
 actor.first_name	Candidate	Looker Report Attribute
 actor.first_name	Candidate	Looker Report Attribute
 actor.first_name	Candidate	Looker Report Attribute
 Actor First Name	Candidate	Looker Data Set Column
 actor.last_name	Candidate	Looker Report Attribute

Recommended hierarchy within a domain

You can enable [hierarchies](#) for the domain in which your Looker assets were ingested. Doing so makes it easier to understand the relation between your Looker assets, when viewing the assets on the domain page.

Follow these steps to enable and configure the recommended hierarchy.

Steps

1. Open the domain page of the relevant BI Catalog domain.
2. On the content toolbar, click .
- » The **Configure Hierarchy** dialog box appears.
3. Select **Enable Hierarchy**.
4. Select **Multipath**.
5. Start typing and select each of the following relation types:
 - Server **hosts** Business Dimension
 - Business Dimension **groups** Report
 - Report **contains** Report Attribute
 - Technology Asset **source system** for Data Asset
 - Data Set **contains** Data Element
 - Data Attribute **is source of** Report Attribute
6. Click **Apply**.

Note

- In an asset view, if any asset is deleted, for example via synchronization or manual deletion, the view is recreated and the hierarchy is lost. In this case, you can again enable and configure the recommended hierarchy.
- When viewing the hierarchy for a community or domain, if the parent of a node that is in the community or domain belongs to a different community or domain, that node is not shown in the hierarchy.

Create a Looker operating model diagram view

You can create a Looker-specific diagram view, to visualize the operating model. The following procedure provides instruction on how to quickly create a new diagram view by copying and pasting the JSON code in the diagram view text editor.

Steps

1. Open an asset page.
2. Click the **Diagram** tab.- » The diagram is shown in the default [diagram view](#).
3. Click **+** to add a new view.

4. Click the **Text** tab, to switch to the diagram view text editor.
5. Click **Show me the JSON code** below this procedure, to expand the code.
6. Paste the code in diagram view text editor.
7. Click **Save**.
8. [Edit](#) the name and description of the diagram view, to suit your needs.

Show me the JSON code

```
{
  "nodes": [
    {
      "id": "Looker Tenant",
      "type": {
        "id": "00000000-0000-0000-0000-100000000011"
      }
    },
    {
      "id": "Looker Folder",
      "type": {
        "id": "00000000-0000-0000-0000-100000000012"
      }
    },
    {
      "id": "Looker Project",
      "type": {
        "id": "750ee74c-84dc-494f-84c0-7ab14105432a"
      }
    },
    {
      "id": "Looker Board",
      "type": {
        "id": "1118d30f-846b-4bae-93d2-97488a0d9796"
      }
    },
    {
      "id": "Looker Look",
      "type": {
        "id": "00000000-0000-0000-0000-100000000014"
      }
    },
    {
      "id": "Looker Dashboard",
      "type": {
        "id": "00000000-0000-0000-0000-100000000013"
      }
    },
    {
      "id": "Looker Data Model",
      "type": {
```

```

        "id": "3ed176fa-78c8-4116-a771-9dd100ad1129"
      },
      {
        "id": "Looker Data Attribute",
        "type": {
          "id": "232fecbc-7f20-45c2-bbcf-7329cd0b17df"
        }
      },
      {
        "id": "Looker Query",
        "type": {
          "id": "00000000-0000-0000-0000-100000000016"
        },
        "layoutRegion": "flow"
      },
      {
        "id": "Looker Tile",
        "type": {
          "id": "00000000-0000-0000-0000-100000000015"
        }
      }
    ],
    "edges": [
      {
        "from": "Looker Tenant",
        "to": "Looker Folder",
        "label": "",
        "style": "arrow",
        "type": {
          "id": "00000000-0000-0000-0000-120000000000"
        },
        "roleDirection": true
      },
      {
        "from": "Looker Tenant",
        "to": "Looker Project",
        "label": "",
        "style": "arrow",
        "type": {
          "id": "00000000-0000-0000-0000-120000000000"
        },
        "roleDirection": true
      },
      {
        "from": "Looker Tenant",
        "to": "Looker Board",
        "label": "",
        "style": "arrow",
        "type": {

```

```

        "id": "f953c3da-6923-4301-b467-2f7066232b47"
    },
    "roleDirection": false
},
{
    "from": "Looker Board",
    "to": "Looker Look",
    "label": "",
    "style": "arrow",
    "type": {
        "id": "00000000-0000-0000-0000-120000000004"
    },
    "roleDirection": true
},
{
    "from": "Looker Board",
    "to": "Looker Dashboard",
    "label": "",
    "style": "arrow",
    "type": {
        "id": "00000000-0000-0000-0000-120000000004"
    },
    "roleDirection": true
},
{
    "from": "Looker Project",
    "to": "Looker Dashboard",
    "label": "",
    "style": "arrow",
    "type": {
        "id": "00000000-0000-0000-0000-120000000002"
    },
    "roleDirection": true
},
{
    "from": "Looker Project",
    "to": "Looker Data Model",
    "label": "",
    "style": "arrow",
    "type": {
        "id": "00000000-0000-0000-0000-120000000014"
    },
    "roleDirection": true
},
{
    "from": "Looker Data Model",
    "to": "Looker Data Attribute",
    "label": "",
    "style": "boxing",
    "type": {

```

```

        "id": "00000000-0000-0000-0000-000000007196"
    },
    "roleDirection": true
},
{
    "from": "Looker Dashboard",
    "to": "Looker Data Attribute",
    "label": "",
    "style": "arrow",
    "type": {
        "id": "00000000-0000-0000-0000-120000000021"
    },
    "roleDirection": true
},
{
    "from": "Looker Look",
    "to": "Looker Data Attribute",
    "label": "",
    "style": "arrow",
    "type": {
        "id": "00000000-0000-0000-0000-120000000021"
    },
    "roleDirection": true
},
{
    "from": "Looker Query",
    "to": "Looker Data Attribute",
    "label": "",
    "style": "arrow",
    "type": {
        "id": "00000000-0000-0000-0000-120000000021"
    },
    "roleDirection": true
},
{
    "from": "Looker Folder",
    "to": "Looker Query",
    "label": "",
    "type": {
        "id": "00000000-0000-0000-0000-120000000002"
    },
    "roleDirection": true
},
{
    "from": "Looker Folder",
    "to": "Looker Look",
    "label": "",
    "style": "arrow",
    "type": {
        "id": "00000000-0000-0000-0000-120000000002"
    }
}

```

```

    },
    "roleDirection": true
  },
  {
    "from": "Looker Folder",
    "to": "Looker Dashboard",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "00000000-0000-0000-0000-120000000002"
    },
    "roleDirection": true
  },
  {
    "from": "Looker Folder",
    "to": "Looker Tile",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "00000000-0000-0000-0000-120000000002"
    },
    "roleDirection": true
  },
  {
    "from": "Looker Query",
    "to": "Looker Look",
    "label": "",
    "style": "boxing",
    "type": {
      "id": "00000000-0000-0000-0000-120000000004"
    },
    "roleDirection": true
  },
  {
    "from": "Looker Dashboard",
    "to": "Looker Tile",
    "label": "",
    "style": "boxing",
    "type": {
      "id": "00000000-0000-0000-0000-120000000004"
    },
    "roleDirection": true
  },
  {
    "from": "Looker Dashboard",
    "to": "Looker Look",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "00000000-0000-0000-0000-120000000007"
    }
  }

```



```

    },
    "roleDirection": true
  },
  {
    "from": "Looker Query",
    "to": "Looker Tile",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "00000000-0000-0000-0000-120000000007"
    },
    "roleDirection": true
  },
  {
    "from": "Looker Folder",
    "to": "Looker Folder",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "00000000-0000-0000-0000-120000000001"
    },
    "roleDirection": true
  }
],
"showOverview": false,
"enableFilters": true,
"showLabels": true,
"showFields": true,
"showLegend": true,
"showPreview": true,
"visitStrategy": "directed",
"layout": "HierarchyLeftRight",
"maxNodeLabelLength": 50,
"maxEdgeLabelLength": 30,
"layoutOptions": {
  "compactGroups": false,
  "componentArrangementPolicy": "topmost",
  "edgeBends": true,
  "edgeBundling": true,
  "edgeToEdgeDistance": 5,
  "minimumLayerDistance": "auto",
  "nodeToEdgeDistance": 5,
  "orthogonalRouting": true,
  "preciseNodeHeightCalculation": true,
  "recursiveGroupLayering": true,
  "separateLayers": true,
  "webWorkers": true,
  "nodePlacer": {
    "barycenterMode": true,
    "breakLongSegments": true,

```

```
        "groupCompactionStrategy": "none",
        "nodeCompaction": false,
        "straightenEdges": true
    }
}
```

- [Overview and diagram view](#)
- [Harvested metadata per asset type](#)
- [Example of ingested SSRS and PBRS metadata](#)
- [Recommended hierarchy within a domain](#)
- [Create a SSRS and PBRS operating model diagram view](#)

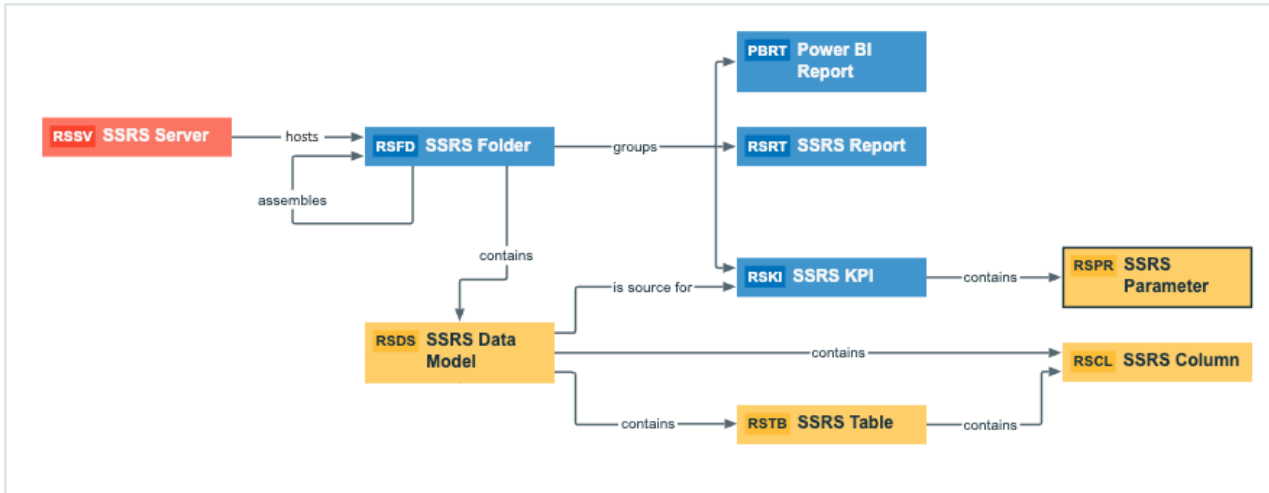
Overview and diagram view

The lineage harvester collects SQL Server Reporting Services (SSRS) metadata and sends it to the Collibra Data Lineage service. Collibra processes the metadata and creates new SSRS assets and relations in Data Catalog. You can see them on the asset page overview or visualize them in a [diagram](#) or in a technical lineage.

Note

- The assets have the same names as their counterparts in SSRS and Power BI Report Server (PBRS). Full names and Names cannot be changed in Data Catalog.
- Assets ingested from SSRS and PBRS are called SSRS assets in Data Catalog, except for PBRS reports which are called Power BI Report assets.
- Asset types are only created if you have all specific Data Catalog permissions.
- All SSRS and PBRS assets are created in the same domain.
- Relations that were manually created between SSRS assets or PBRS assets and other assets via a relation type in the SSRS and PBRS operating model, are deleted after synchronizing the metadata.

The following image shows the relations between SSRS asset types and the Power BI Report asset type.



Harvested metadata per asset type

This table shows the harvested SSRS and PBRS metadata for each SSRS asset type and Power BI Report asset type, assuming you have the necessary subscriptions and configurations for a full ingestion. This table also shows the resource ID for each asset type and metadata.

Asset type	Synchronized metadata	Resource ID
SSRS Column Resource ID: 00000000-0000-0000-0000-100000000029	Full name	
	Display name	
	Description	00000000-0000-0000-0000-0000000003114
	Technical Data Type	00000000-0000-0000-0000-000000000219
	BI Data Model contains / is part of BI Data Attribute	00000000-0000-0000-0000-0000000007196
	Data Element targets / sources Data Element	00000000-0000-0000-0000-0000000007069
	Data Entity contains / is part of Data Attribute	00000000-0000-0000-0000-0000000007047

Asset type	Synchronized metadata	Resource ID
SSRS Data Model Resource ID: 00000000-0000-0000-0000-100000000028	Full name	
	Display name	
	Certified	00000000-0000-0000-0001-000500000001
	Description	00000000-0000-0000-0000-000000003114
	Document creation date	00000000-0000-0000-0000-000000000260
	Document modification date	00000000-0000-0000-0000-000000000261
	Document size	00000000-0000-0000-0000-000000000259
	Location	00000000-0000-0000-0000-000000000203
	URL	00000000-0000-0000-0000-000000000258
	Visible on server	00000000-0000-0000-0000-000000000265
	BI Data Model contains / is part of BI Data Attribute	00000000-0000-0000-0000-000000007196
	BI Folder contains / contained in Data Asset	00000000-0000-0000-0000-120000000014
	Data Asset is source for / sources BI report	00000000-0000-0000-0000-120000000013
	Data Entity is part of / contains Data Model	00000000-0000-0000-0000-000000007046

Asset type	Synchronized metadata	Resource ID
SSRS Folder Resource ID: 00000000-0000-0000-0000-100000000024	Full name	
	Display name	
	Description	00000000-0000-0000-0000-000000003114
	Document creation date	00000000-0000-0000-0000-000000000260
	Document modification date	00000000-0000-0000-0000-000000000261
	Location	00000000-0000-0000-0000-000000000203
	URL	00000000-0000-0000-0000-000000000258
	Visible on server	00000000-0000-0000-0000-000000000265
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	BI Folder assembles / is assembled in BI Folder	00000000-0000-0000-0000-120000000001
	BI Folder contains / contained in Data Asset	00000000-0000-0000-0000-120000000014
	Server hosts / is hosted in Business Dimension	00000000-0000-0000-0000-120000000000

Asset type	Synchronized metadata	Resource ID
SSRS KPI Resource ID: 00000000-0000-0000-0000-100000000026	Full name	
	Display name	
	Certified	00000000-0000-0000-0001-000500000001
	Description	00000000-0000-0000-0000-000000003114
	Document creation date	00000000-0000-0000-0000-000000000260
	Document modification date	00000000-0000-0000-0000-000000000261
	Document size	00000000-0000-0000-0000-000000000259
	Location	00000000-0000-0000-0000-000000000203
	URL	00000000-0000-0000-0000-000000000258
	Visible on server	00000000-0000-0000-0000-000000000265
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	Data Asset is source for / sources BI Report	00000000-0000-0000-0000-120000000013
	Report Attribute contained in / contains Report	00000000-0000-0000-0000-000000007058
	Report related to / impacted by Business Asset	00000000-0000-0000-0000-120000000006

Asset type	Synchronized metadata	Resource ID
SSRS Parameter Resource ID: 00000000-0000-0000-0000-100000000027	Full name	
	Display name	
	Description	00000000-0000-0000-0000-000000003114
	Business Asset represents / represented by Data Asset	00000000-0000-0000-0000-000000007038
	Report Attribute contained in / contains Report	00000000-0000-0000-0000-000000007058
	Report Attribute sourced from / is source of Data Attribute	00000000-0000-0000-0000-120000000010

Asset type	Synchronized metadata	Resource ID
SSRS Report Resource ID: 00000000-0000-0000-0000-100000000025	Full name	
	Display name	
	Certified	00000000-0000-0000-0001-000500000001
	Description	00000000-0000-0000-0000-000000003114
	Document creation date	00000000-0000-0000-0000-000000000260
	Document modification date	00000000-0000-0000-0000-000000000261
	Document size	00000000-0000-0000-0000-000000000259
	Location	00000000-0000-0000-0000-000000000203
	URL	00000000-0000-0000-0000-000000000258
	Visible on server	00000000-0000-0000-0000-000000000265
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	Data Asset is source for / sources BI Report	00000000-0000-0000-0000-120000000013
	Report related to / impacted by Business Asset	00000000-0000-0000-0000-120000000006
	Report uses / used in Report	00000000-0000-0000-0000-120000000007

Asset type	Synchronized metadata	Resource ID
SSRS Server Resource ID: 00000000-0000-0000-0000-100000000023	Full name	
	Display name	
	Description	00000000-0000-0000-0000-000000003114
	Server hosts / is hosted in Business Dimension	00000000-0000-0000-0000-120000000000
SSRS Table Resource ID: 00000000-0000-0000-0000-100000000030	Full name	
	Display name	
	Description	00000000-0000-0000-0000-000000003114
	Data Entity contains / is part of Data Attribute	00000000-0000-0000-0000-000000007047
	Data Entity is part of / contains Data Model	00000000-0000-0000-0000-000000007046

Example of ingested SSRS and PBRS metadata

The following image shows an example structure after SSRS and PBRS ingestion.

>	Delete	Move	Validate
	☐ Name	Status	Asset Type ↑
☐	testFolder	Candidate	SSRS Folder
☐	first	Candidate	Power BI Report
☐	MSA KPI	Candidate	SSRS KPI
☐	Status	Candidate	SSRS Parameter
☐	Value	Candidate	SSRS Parameter
☐	Goal	Candidate	SSRS Parameter
☐	Oracle KPI	Candidate	SSRS KPI
☐	Redshift KPI	Candidate	SSRS KPI
☐	MSA Mobile Report	Candidate	SSRS Report
☐	Oracle-paginated report	Candidate	SSRS Report
☐	Redshift paginated repo...	Candidate	SSRS Report
☐	Redshift Mobile Report	Candidate	SSRS Report
☐	Oracle Mobile Report	Candidate	SSRS Report
☐	MSA paginated report	Candidate	SSRS Report

Recommended hierarchy within a domain

You can enable [hierarchies](#) for the domain in which your SSRS assets were ingested. Doing so makes it easier to understand the relation between your SSRS assets, when viewing the assets on the domain page.

Follow these steps to enable and configure the recommended hierarchy.

Steps

1. Open the domain page of the relevant BI Catalog domain.
2. On the content toolbar, click .
- » The **Configure Hierarchy** dialog box appears.
3. Select **Enable Hierarchy**.
4. Select **Multipath**.
5. Start typing and select each of the following relation types:
 - Server **hosts** Business Dimension
 - Business Dimension **groups** Report
 - BI Folder **contains** Data Asset
 - Data Asset **is source for** BI Report
 - Report **contains** Report Attribute
 - BI Folder **contains** Data Asset
 - BI Data Model **contains** BI Data Attribute
 - Data Entity **contains** Data Attribute
6. Click **Apply**.

Note

- In an asset view, if any asset is deleted, for example via synchronization or manual deletion, the view is recreated and the hierarchy is lost. In this case, you can again enable and configure the recommended hierarchy.
- When viewing the hierarchy for a community or domain, if the parent of a node that is in the community or domain belongs to a different community or domain, that node is not shown in the hierarchy.

Create an SSRS and PBRs operating model diagram view

You can create a diagram view for SSRS and PBRs to visualize the operating model. Complete the following steps to create a new diagram view by copying and pasting the JSON code in the diagram view text editor.

Steps

1. Open an asset page.
2. Click the **Diagram** tab.
 - » The diagram is shown in the default [diagram view](#).
3. Click + to add a new view.
4. Click the **Text** tab, to switch to the diagram view text editor.
5. Click **Show me the JSON code** below this procedure, to expand the code.
6. Paste the code in diagram view text editor.
7. Click **Save**.
8. [Edit](#) the name and description of the diagram view, to suit your needs.

Show me the JSON code

```
{
  "nodes": [
    {
      "id": "SSRS Column",
      "type": {
        "id": "00000000-0000-0000-0000-100000000029"
      }
    },
    {
      "id": "SSRS Data Model",
      "type": {
        "id": "00000000-0000-0000-0000-100000000028"
      }
    },
    {
      "id": "SSRS Table",
      "type": {
        "id": "00000000-0000-0000-0000-100000000030"
      }
    },
    {
      "id": "SSRS KPI",
      "type": {
        "id": "00000000-0000-0000-0000-100000000026"
      }
    },
    {
      "id": "SSRS Parameter",
      "type": {
        "id": "00000000-0000-0000-0000-100000000027"
      }
    }
  ],
}
```

```

{
  "id": "SSRS Folder",
  "type": {
    "id": "00000000-0000-0000-0000-100000000024"
  }
},
{
  "id": "Power BI Report",
  "type": {
    "id": "00000000-0000-0000-0000-100000000006"
  }
},
{
  "id": "SSRS Report",
  "type": {
    "id": "00000000-0000-0000-0000-100000000025"
  }
},
{
  "id": "SSRS Folder 2",
  "type": {
    "id": "00000000-0000-0000-0000-100000000024"
  }
},
{
  "id": "SSRS Server",
  "type": {
    "id": "00000000-0000-0000-0000-100000000023"
  }
},
{
  "id": "Column",
  "type": {
    "id": "00000000-0000-0000-0000-000000031008"
  }
},
{
  "id": "Table",
  "type": {
    "id": "00000000-0000-0000-0000-000000031007"
  }
},
{
  "id": "Schema",
  "type": {
    "id": "00000000-0000-0000-0001-000400000002"
  }
},
{
  "id": "Database",

```

```

        "type": {
          "id": "00000000-0000-0000-0000-0000000031006"
        }
      }
    ],
    "edges": [
      {
        "from": "SSRS Data Model",
        "to": "SSRS Column",
        "label": "",
        "style": "arrow",
        "type": {
          "id": "00000000-0000-0000-0000-0000000007196"
        },
        "roleDirection": true
      },
      {
        "from": "SSRS Table",
        "to": "SSRS Column",
        "label": "",
        "style": "arrow",
        "type": {
          "id": "00000000-0000-0000-0000-0000000007047"
        },
        "roleDirection": true
      },
      {
        "from": "SSRS Data Model",
        "to": "SSRS Table",
        "label": "",
        "style": "arrow",
        "type": {
          "id": "00000000-0000-0000-0000-0000000007046"
        },
        "roleDirection": true
      },
      {
        "from": "SSRS Data Model",
        "to": "SSRS KPI",
        "label": "",
        "style": "arrow",
        "type": {
          "id": "00000000-0000-0000-0000-1200000000013"
        },
        "roleDirection": true
      },
      {
        "from": "SSRS KPI",
        "to": "SSRS Parameter",
        "label": "",

```

```

        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000014"
        },
        "roleDirection": true
    },
    {
        "from": "SSRS Folder",
        "to": "SSRS Data Model",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-120000000014"
        },
        "roleDirection": true
    },
    {
        "from": "SSRS Folder",
        "to": "Power BI Report",
        "label": "",
        "style": "boxing",
        "type": {
            "id": "00000000-0000-0000-0000-120000000002"
        },
        "roleDirection": true
    },
    {
        "from": "SSRS Folder",
        "to": "SSRS Report",
        "label": "",
        "style": "boxing",
        "type": {
            "id": "00000000-0000-0000-0000-120000000002"
        },
        "roleDirection": true
    },
    {
        "from": "SSRS Folder",
        "to": "SSRS KPI",
        "label": "",
        "style": "boxing",
        "type": {
            "id": "00000000-0000-0000-0000-120000000004"
        },
        "roleDirection": true
    },
    {
        "from": "SSRS Server",
        "to": "SSRS Folder 2",
        "label": "",

```

```

    "style": "boxing",
    "type": {
      "id": "00000000-0000-0000-0000-120000000000"
    },
    "roleDirection": false
  },
  {
    "from": "SSRS Folder 2",
    "to": "SSRS Folder",
    "label": "",
    "style": "boxing",
    "type": {
      "id": "00000000-0000-0000-0000-120000000001"
    },
    "roleDirection": true
  },
  {
    "from": "SSRS Folder",
    "to": "SSRS Server",
    "label": "",
    "style": "boxed",
    "type": {
      "id": "00000000-0000-0000-0000-120000000000"
    },
    "roleDirection": false
  },
  {
    "from": "SSRS Report",
    "to": "SSRS Data Model",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "00000000-0000-0000-0000-120000000013"
    },
    "roleDirection": false
  },
  {
    "from": "SSRS Column",
    "to": "Column",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "00000000-0000-0000-0000-000000007069"
    },
    "roleDirection": false
  },
  {
    "from": "Column",
    "to": "Table",
    "label": "",

```



```

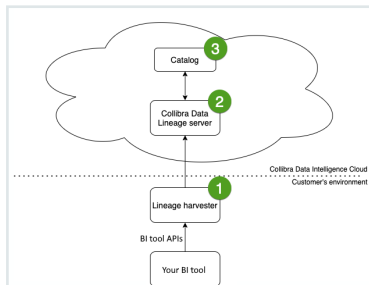
    "style": "boxed",
    "type": {
      "id": "00000000-0000-0000-0000-000000007042"
    },
    "roleDirection": true
  },
  {
    "from": "Table",
    "to": "Schema",
    "label": "",
    "style": "boxed",
    "type": {
      "id": "00000000-0000-0000-0000-000000007043"
    },
    "roleDirection": false
  },
  {
    "from": "Schema",
    "to": "Database",
    "label": "",
    "style": "boxed",
    "type": {
      "id": "00000000-0000-0000-0000-000000007024"
    },
    "roleDirection": false
  }
],
"showOverview": false,
"enableFilters": true,
"showLabels": false,
"showFields": true,
"showLegend": true,
"showPreview": true,
"visitStrategy": "directed",
"layout": "HierarchyLeftRight",
"maxNodeLabelLength": 50,
"maxEdgeLabelLength": 30,
"layoutOptions": {
  "compactGroups": false,
  "componentArrangementPolicy": "topmost",
  "edgeBends": true,
  "edgeBundling": true,
  "edgeToEdgeDistance": 5,
  "minimumLayerDistance": "auto",
  "nodeToEdgeDistance": 5,
  "orthogonalRouting": true,
  "preciseNodeHeightCalculation": true,
  "recursiveGroupLayering": true,
  "separateLayers": true,
  "webWorkers": true,

```

```
"nodePlacer": {  
  "barycenterMode": true,  
  "breakLongSegments": true,  
  "groupCompactionStrategy": "none",  
  "nodeCompaction": false,  
  "straightenEdges": true  
}  
}  
}
```

BI tool ingestion workflow

You run the lineage harvester to start the BI tool ingestion workflow. When you initiate the integration, each workflow component performs the following actions:



Prepare a domain for BI asset ingestion

Before you can ingest BI metadata, you have to designate a domain for storing the new BI assets. You can choose an existing domain or create one or more new domains. You then have to include the domain reference ID (or IDs) in the appropriate configuration file.

Important The amount of domains into which you ingest assets differs according to your BI tool:

- **Looker:** You can designate one domain in the lineage harvester configuration file. However, you can also set up a filter in the [<source ID> configuration file](#), to ingest into different domains.
- **MicroStrategy:** You can ingest into one domain.
- **Power BI:** You can ingest into one or more domains.
- **SSRS-PBRS:** You can ingest into one domain.
- **Tableau:** You can ingest into one or more domains.

Prerequisites

- You have a resource role with the Domain > Add resource permission.

Steps

1. On the main toolbar, click **+**.
 - » The **Create** dialog box appears.
2. Click the **Organization** tab.
3. Click **BI Catalog**.

If you clicked the wrong domain type here, you can change it in the **Type** field in the next screen.

» The **Create Domain** dialog box appears.

4. Enter the required information.

Field	Description
Domain Type	The domain type of the domain you are creating. In this case, you need to select <i>BI Catalog</i> .

Field	Description
Community	The community under which the domain will be located.
Domain Name	The name of the new domain or domains. Tip You can create multiple domains in one go. To do this, press <code>Enter</code> after typing a value and then type the next. Domain names have to be unique in their parent community. If you type a name that already exists, it will appear in strike-through style.

5. Click **Create**.
6. Open your domain. If you created multiple domains, open each of them in turn.
7. Copy the reference ID of each domain you created.

Tip If you go to your domain, you can find the domain ID in the URL. The URL looks like: `https://<yourcollibrainstance>/domain/22258f64-40b6-4b16-9c08-c95f8ec0da26?view=00000000-0000-0000-0000-000000040001`. In this example, the domain ID is in bold.

8. Paste the domain reference ID (or IDs) in the appropriate configuration file, depending on whether you want to ingest Tableau assets in a single domain or multiple domains.

For complete information on which properties and which configuration files to use, see the `domainId` property description in the [Prepare the lineage harvester configuration file](#) topic for the relevant BI tool.

What's next?

[Prepare](#) the Data Catalog physical data layer.

Looker supported data sources

The following table shows the supported data sources in Looker that have been tested, and whether or not technical lineage and stitching is supported for the data source. Looker might support other data sources, but only those mentioned in the table have been tested.

Data source	Version	Support for technical lineage	Support for stitching
Amazon Redshift	1.2.34.1058 and newer	Yes	No
Apache Impala	3.1 and newer	Yes	No
Azure SQL Server	Newest version	Yes	No
Google BigQuery	Newest version	Yes	No
Greenplum	6.10 and newer	Yes	No
HiveQL (SQL-like statements)	2 and newer	Yes	No
IBM DB2	11.5 and newer	Yes	No
MySQL	5.7, 8 and newer	Yes	No
Netezza	7.2.1.0 and newer	Yes	No
Oracle	11g, 12c and newer	Yes	No
PostgreSQL	9.5 and newer	Yes	No
SAP HANA	2.00.40 and newer	Yes	No
Snowflake	Newest version	Yes	No
Spark SQL	1.5 and newer	Yes	No
Teradata	15.0, 16.20.07.01 and newer	Yes	No
Vertica	5 and newer	Yes	No

Working with Power BI

This section addresses tasks and concepts that can be of interest to Collibra Admins who are working with Power BI.

Supported data sources in Power BI

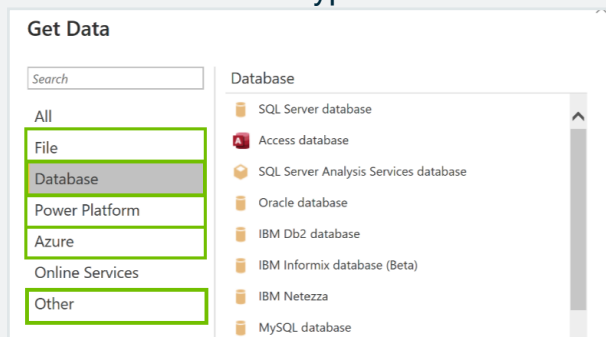
Power BI is business intelligence software that can integrate with various data sources. When you ingest Power BI metadata, Collibra Data Lineage tries to automatically stitch this metadata to data sources registered in Data Catalog. It also creates a technical lineage that shows where metadata is used and how it transforms.

The following table shows the supported data source types in Power BI that have been tested. If a data source is identified as certified, it means that the data source:

- Is ingested in Data Catalog as a Power BI Data Model asset.
- Is shown in the technical lineage and stitching is possible.

Tip When selecting your data sources in Power BI, the data source types under the following tabs require the specified connection types:

- The **Files** tab refers to Parquet files. Connect via the URL to the files.
- All data source types under **Database** require a JDBC connection.
- For **Power Platform**, you need a live connection.
- The three data source types under the **Azure** tab require either a JDBC or ODBC connection.
- All data source types under **Other** require an ODBC connection.



The connections types are mentioned in the following table, for each supported data source type.

Important

- Although the following data sources have been tested extensively, there still may be some issues caused by unsupported elements within the data source or [limitations](#) in the Power BI integration process.

- Collibra Data Lineage can connect only to datasets that are hosted by Power BI. It cannot connect to externally hosted datasets or models. For complete information, consult Microsoft's [Power BI documentation](#).
- Real-time datasets, semantic models with object-level security, semantic models with a live connection to AS-Azure and AS on-premises, and Excel full fidelity datasets aren't supported for subartifact metadata. For complete information, consult Microsoft's [Real-time streaming in Power BI documentation](#).

Power BI data source	Connection type	Certified ?
Amazon Redshift	JDBC	Yes
Apache Hive	ODBC	Yes
Azure Analysis Services	ODBC	Yes
	JDBC	No • Via import, technical lineage is possible only from the data set to the report. Stitching is not supported. • Via direct connection, technical lineage and stitching are not supported.

Power BI data source	Connection type	Certified ?
Azure Databricks	JDBC, ODBC	Yes Collibra Data Lineage supports the following functions: • Databricks.Catalogs • Databricks.Contents • Databricks.Query Note For Databricks.Query, the ingestion success rate is high, but it's not 100%.
Azure Synapse Analytics	JDBC	Yes
Dremio	JDBC	No Dremio data sources are ingested as Power BI Data Model assets and shown in the technical lineage, but stitching is not possible.
Google BigQuery	JDBC	Yes
Impala	ODBC	Yes
MySQL	JDBC	No MySQL data sources are ingested as Power BI Data Model assets, but not shown in the technical lineage and stitching is not possible.
Netezza	ODBC	Yes

Power BI data source	Connection type	Certified ?
ODBC	ODBC	Yes Important You need to use a Power BI <source ID> configuration file to provide the true system names of the ODBC databases in Power BI. For more information, see Providing ODBC database names in Power BI.
Oracle	JDBC, ODBC	Yes Note If you connect via ODBC: • Oracle Views are supported. • In most cases, you need to use a Power BI <source ID> configuration file for database mapping, as the database name returned by the API differs from the true database name.
Parquet file	URL to the files	No
SAP HANA	ODBC	Yes
Snowflake	JDBC	Yes

Power BI data source	Connection type	Certified ?
SQL Server	JDBC	Yes
Sybase	JDBC	Yes
Power BI with measures only, no columns	JDBC	Yes
Power Platform > Power BI data set	Live connection	Yes Important Supported only if the data set is from one of these supported data sources.

Note We cannot guarantee that other data sources in Power BI can be stitched successfully.

Power BI workspaces

Power BI workspaces represent the most used metadata in Power BI. They contain, for example, reports and data sets. If you want a full ingestion, you have to make sure that the lineage harvester can access all metadata in your Power BI workspaces.

Important considerations

- Depending on the [authentication](#) type, you must have specific roles and permissions to access the metadata in the Power BI workspaces.
- You can only fully ingest [new Power BI workspaces](#). This means that classic workspaces and My Workspace in Power BI are not supported.
- To ingest Power BI dataflows:
 - You need access to the Power BI environment in which the data flow is stored.
 - The data set in the data flow must exist in a premium workspace.

Filtering Power BI workspaces

By default, the lineage harvester accesses the metadata of all Power BI workspaces. If you don't use filtering, the metadata of all workspaces is uploaded to the Collibra Data Lineage service instance and ingested in Data Catalog. Filtering allows you to process and ingest only the metadata that matters most to you.

You use the [Power BI <source ID>_filter configuration file](#) to configure filtering for Power BI workspaces.

Inclusion and exclusion filters

You can use the following inclusion filters to ingest only the Power BI capacities and workspaces you specify:

- `capacityNames`
- `capacityIds`
- `workspaceNames`
- `workspaceIds`

You can use the following exclusion filters to ingest all workspaces except for those you specify:

- `excludeWorkspaceNames`
- `excludeWorkspaceIds`

Tip

- Wildcards are supported for the `capacityNames`, `workspaceNames` and `excludeWorkspaceNames` properties.
- You can combine inclusion and exclusion filters in the same <source ID> configuration file.

Show me an example

In this example, the metadata from all workspaces is uploaded to the Collibra Data Lineage service instance. Then, the metadata in all of the workspaces in CapacityABC, except for Workspace1, is ingested in Data Catalog.

```
{
  "filters": [
    {
      "domainId": "07d5d441-b9f8-4add-982f-d7a5d6ba06cc",
      "description": "Domain for BICatalogJBTest1",
      "capacityNames": ["CapacityABC"],
      "excludeWorkspaceNames": ["Workspace1"]
    }
  ]
}
```

- In the Power BI <source ID> configuration file, you can also specify the domain (or domains) in which you want to ingest, to help structure your Power BI assets in Collibra.

Two filtering methods

The filter properties that you use in your Power BI <source ID> configuration file determine whether filtering is done by the lineage harvester or done on the Collibra Data Lineage service instance. The following table highlights the advantages, limitations and configuration considerations of the two methods.

Filtering method	Description
By the lineage harvester	The lineage harvester accesses only the workspaces specified in your <source ID> configuration file, and sends metadata from only those workspaces to the Collibra Data Lineage service instance for processing and ingestion in Data Catalog. Advantages • Faster integration testing, as you can filter on a single workspace. • Enhanced data security and privacy by excluding workspaces that contain sensitive information. Metadata from workspaces that are filtered out by the lineage harvester is not sent to the Collibra Data Lineage service instance for processing. • Improve processing times by excluding workspaces dedicated to, for example, development and testing. This is especially beneficial for organizations with more than 50,000 workspaces. Limitations • For this to work as described, you can only use the <code>workspaceIds</code> property. None of the following properties can be included anywhere in your <source ID> configuration file: ◦ <code>capacityNames</code> ◦ <code>capacityIds</code> ◦ <code>workspaceNames</code> ◦ <code>excludeWorkspaceNames</code> ◦ <code>excludeWorkspaceIds</code> • You cannot use wildcards with the <code>workspaceIds</code> property. Show me an example setup for the <source ID> configuration file <pre> { "filters": [{ "domainId": "b5d02896-8a79-49a3-bab0-12a7b37f45c6", "description": "Any description, for your internal use", "workspaceIds": ["ee23f25b-0ed9-490a-9cca-8a0e8886173e", "8e86429d-f985-4a81-818d-8e05ac256a74"] }] }</pre>

Filtering method	Description
On the Collibra Data Lineage service instance	The lineage harvester accesses all workspaces and filtering is carried out only after knowing the names and IDs of all workspaces and capacities. As a result, the raw metadata is accessed by the lineage harvester, but only the filtered metadata is processed on the Collibra Data Lineage service instance and ingested in Data Catalog. Advantages • Greater choice of filtering options. You can use any of the following properties: ◦ capacityNames ◦ capacityIds ◦ workspaceNames ◦ excludeWorkspaceNames ◦ excludeWorkspaceIds • You can use wildcards with the following properties: ◦ capacityNames ◦ workspaceNames ◦ excludeWorkspaceNames Limitations • Longer processing times, especially if you have tens of thousands of workspaces. • Although you can limit which workspaces are processed and ingested, you can't limit which workspaces are uploaded to the Collibra Data Lineage service instance. The raw metadata from all workspaces is uploaded. Tip You can use the <code>deleteRawMetadataAfterProcessing</code> property in your lineage harvester configuration file, to automatically delete the uploaded raw metadata that you don't want to ingest in Data Catalog. Show me an example setup for the <source ID> configuration file <pre> { "filters": [{ "domainId": "07d5d441-b9f8-4add-982f-d7a5d6ba06cc", "decription": "Domain for BICatalogJBTest1", </pre>

Filtering method	Description
	<pre> "capacityNames": ["CapacityABC"], "excludeWorkspaceNames": ["Workspace1"] }] } </pre>

Note The metadata of inactive and personal workspaces is not harvested or uploaded to the Collibra Data Lineage service instance. An inactive workspace is one for which no reports or dashboards have been viewed in the past 60 days. My workspace is the personal workspace for any Power BI customer to work with their own, personal content.

Best practices

Filter on a capacity

You can filter on a capacity to ingest the metadata from all workspaces in that capacity. Let's say, for example, that you have 50,000 workspaces but you only want to ingest metadata from the workspaces related to a specific department in your organization. You could specify each of the relevant workspaces in the configuration file, but that could be tedious if there are lots of workspaces. Furthermore, if someone in your organization creates a new workspace, it will have to be added to your configuration file. Instead, you can filter on a capacity. Then, when a new workspace is created, ensure that it is added to the department's capacity and metadata from that workspace will be automatically ingested, without having to update the configuration file.

Filter on workspaces with domain mapping

Let's say that you have three workspaces, each dedicated to a different department in your organization: HR, Finance, and Marketing. You want to ingest the metadata from these three workspaces into three different domains and set the permissions in Collibra so that people can access only the domain for the department to which they belong.

In this case, you can use the `workspaceIds` property in your filter configuration, to specify which workspaces are ingested into which domains.

Example

```
{
  "filters": [
    {
      "domainId": "b5d02896-8a79-49a3-bab0-12a7b37f45c6",
      "description": "<Any description, for the HR
department>",
      "workspaceIds": "<HR's workspace ID>"
    },
    {
      "domainId": "b5d3h796-wg79-86a3-hab3-e7cdj48r9fk49",
      "description": "<Any description, for the Finance
department>",
      "workspaceIds": "<Finance's workspace ID>"
    },
    {
      "domainId": "8sue2896-0s79-93d3-ki90-2fr459jht4s4",
      "description": "<Any description, for the Marketing
department>",
      "workspaceIds": "<Marketing's workspace ID>"
    }
  ]
}
```

Use only the `workspaceIds` property, to enhance performance and security

If you include only the `workspaceIds` property in your filter configuration:

- The lineage harvester (via CLI or Edge) connects only to those workspaces.
- Collibra bypasses the endpoint: `GetModifiedWorkspaces`, which lists all of the workspaces in your tenant.

By specifying only the IDs of the workspaces you want to ingest and bypassing the `GetModifiedWorkspaces` endpoint, Collibra is unaware of any other workspaces that exist in your tenant. This makes the harvesting job a lot faster and more secure.

If this method is not preferred in your organization, you can instead use another filter property and include the `deleteMetadataAfterProcessing` property in your configuration file (or select the option in your Edge capacity) and set the value to `true`. In this case, filtering is done as described in the table above: [Filtering on the Collibra Data Lineage service instance](#):

1. All of the raw metadata in the capacity is harvested and sent to the Collibra Data Lineage service instance.
2. Only the metadata from the workspaces you want to ingest are ingested in Collibra.
3. All metadata is permanently deleted from the Collibra Data Lineage service instance.

This approach, however, does not enhance performance.

Workspace states

On Power BI Workspace asset pages, you can include the attribute type State, to show the state of ingested Power BI workspaces, for example Active, Orphaned or Deleted. To do so, you have to [edit](#) the global assignment of the Power BI Workspace asset type and assign the attribute type State.

For complete information on Power BI workspaces and possible states, see the [Microsoft Power BI documentation](#).

Tip If you only want to see Power BI workspaces that have the state Active:

1. Ensure that the attribute type State is assigned to the Power BI Workspace asset type via the global assignment.
2. Go to the [Global view](#), and then [create](#) an advance filter and filter by the following clauses:
 - a. Asset type equals Power BI Workspace
 - b. Characteristic State equals Active.

Deleted workspaces

If you delete a Power BI workspace, the workspace is maintained for a 90-day grace period, during which a Power BI administrator can restore the workspace. During the grace period, the workspace has the state Deleted. When you ingest Power BI metadata in Data Catalog, this deleted workspace is ingested.

When the grace period elapses, the state of the workspace becomes Removing, for a short time, while it is being permanently removed. The state then becomes Not found. At this point, as the workspace no longer exists in Power BI, the Power BI Workspace asset in Collibra will also be deleted upon the next synchronization.

If a workspace becomes inactive, meaning no reports or dashboards have been viewed in the past 60 days, it is excluded from the ingestion.

Why are deleted workspaces ingested?

Let's imagine that you ingest a Power BI workspace with the Active state and that over time, you add comments, tags and characteristics to the asset in Collibra. Now let's imagine that the workspace is deleted in Power BI and we do not ingest the deleted workspace. In this case, the Power BI Workspace asset in Collibra is deleted upon the next synchronization. But what if the Power BI administrator decides, during the 90-day grace period, to restore the workspace in Power BI? Upon the next synchronization, a new Power BI Workspace asset is created in Collibra, but all of the comments, tags and characteristics that were part of the deleted asset are lost.

By ingesting deleted Power BI workspaces, we safeguard against losing any of the additional information on the Power BI Workspace asset, in case a Power BI administrator decides to restore a workspace during the grace period.

Note This feature is not available on Collibra Cloud for Government.

DAX parsing via Collibra AI (Beta)

Data Analysis Expressions (DAX) parsing via Collibra AI:

- Allows you to create column-level lineage that includes your calculated columns and measures in Power BI.
- Enables stitching between calculated columns in the technical lineage and the corresponding Power BI Column assets in Data Catalog.

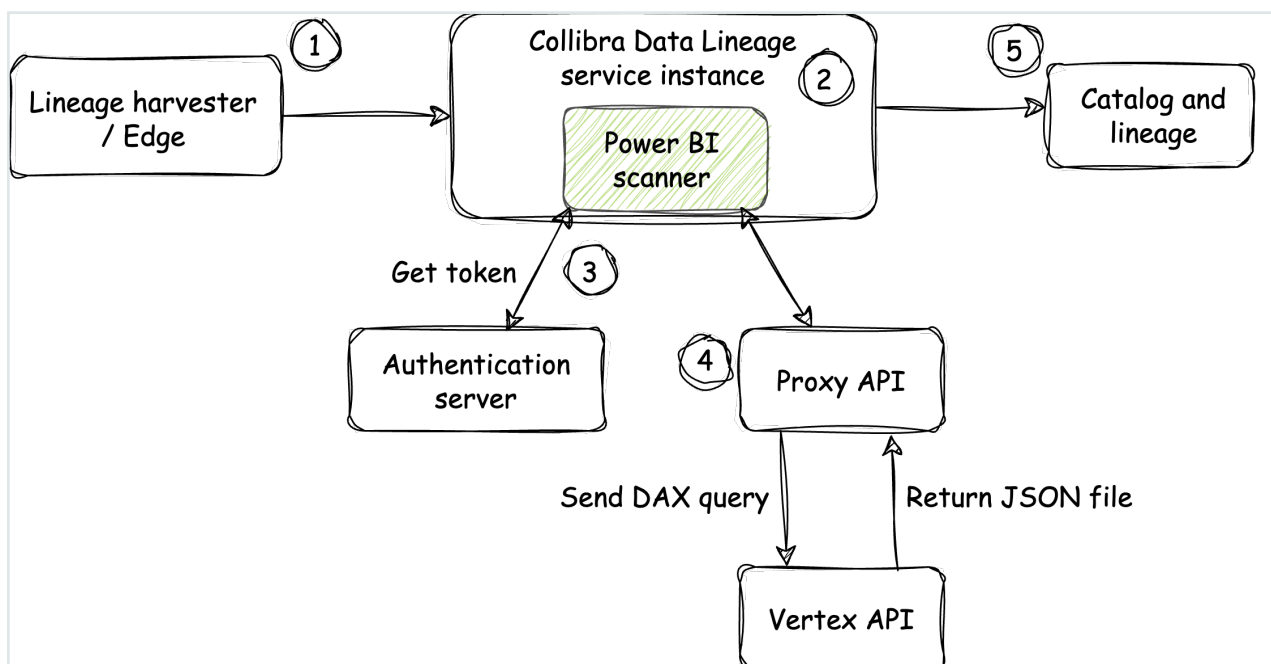
Note This feature is in Beta testing and is powered by Collibra AI.

DAX is a programming language that is used in Power BI for creating calculated columns, measures and custom tables. Calculated columns and measures in Power BI are ingested in Data Catalog as Power BI Column assets.

Prior to this feature, Power BI columns that are derived from DAX were shown in the technical lineage, but because the Collibra Data Lineage service instances are unable to parse DAX, stitching between calculated columns in the technical lineage and the corresponding Power BI Column assets in Data Catalog was not possible.

Tip For information on how we leverage AI in our products, visit the [Collibra Trust Site](#).

How does DAX parsing via Collibra AI work?



1. During Power BI integration, the lineage harvester or Edge sends a request to the Collibra Data Lineage service instance, to parse the Power BI DAX queries.
2. The Collibra Data Lineage service instance processes the data as usual, but the DAX queries are collected separately.
3. The Power BI scanner connects to the authentication server, to request a token.

4. The token is used to make a request of the proxy API.
 - a. The DAX queries are sent to the Vertex AI model.

Note Vertex AI has no context about the DAX queries. Vertex AI only parses the DAX queries and sends them to the Power BI scanner, via the proxy API, in a JSON file.

- b. The Power BI scanner receives a JSON file with the parsed DAX queries.
 - c. The Power BI scanner continues processing the tables and columns, just as it does for Power Query M functions, and creates the technical lineage.
5. The technical lineage is ingested in Collibra and new relations are created between Power BI Column assets.

Attributes on Power BI Column asset pages

On Power BI Column asset pages, there are two attribute types that are relevant for this feature: the Role in Report and Calculation Rule attributes.

Role in Report

Name	Status	Asset Type	Role in Report	Calculation Rule
Column_abs	Candidate	Power BI Column		abs(AMAZINGMARTEU2GEO2[Profit])
Column_abs	Candidate	Power BI Column		abs(AMAZINGMARTEU2GEO2[Profit])
Column_abs	Candidate	Power BI Column		abs(AMAZINGMARTEU2GEO2[Profit])
Column_ACCRINT_Ex1	Candidate	Power BI Column		ACCRINT(FINANCE_H51[issueDate_col], FINANCE_H51[first_interestDate
Column_ACCRINT_Ex2	Candidate	Power BI Column		ACCRINT(FINANCE_H51[issueDate_col], FINANCE_H51[first_interestDate
Column_ACCRINTM	Candidate	Power BI Column		ACCRINTM(FINANCE_H51[issueDate_col], FINANCE_H51[maturityDate_c
Measure_ACOS	Candidate	Power BI Column	Measure	ACOS(-0.5)
Measure_ACOS	Candidate	Power BI Column	Measure	ACOS(-0.5)
Measure_ACOS	Candidate	Power BI Column	Measure	ACOS(-0.5)

1 For measures, the Power BI API returns sufficient information for Collibra Data Lineage to identify the columns as such. Therefore, the value of the Role in Report attribute is appropriately shown as Measure.

2 For calculated columns, the Power BI API can't distinguish between native columns (those read from the data source or evaluated by a query written in Power Query M or Power BI) and calculated columns. Therefore, no value is shown for the Role in Report attribute.

Calculation Rule

The Calculation Rule attribute type shows the DAX query of the calculated column or measure.

If the Calculation Rule attribute type is not showing on the asset page, ensure that the attribute type is part of the global assignment of the Power BI Column asset type. By default, it is not included.

For more information, go to [Show DAX calculations on Power BI Column asset pages](#).

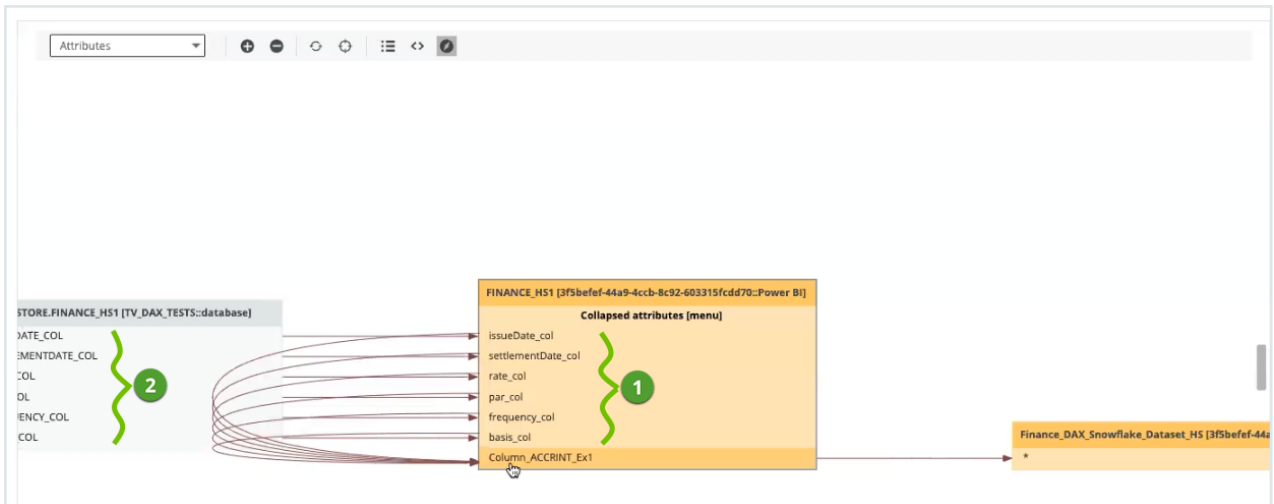
Results in the technical lineage graph

Prior to DAX parsing via Collibra AI, column-level lineage showed only the calculated column and how it related to upstream and downstream data sources. It was not possible to know which columns are aggregated to derive the calculated column. Now, much more information is now shown.

With DAX parsing enabled, all of the columns that are aggregated to derive the calculated column are shown in the technical lineage graph.

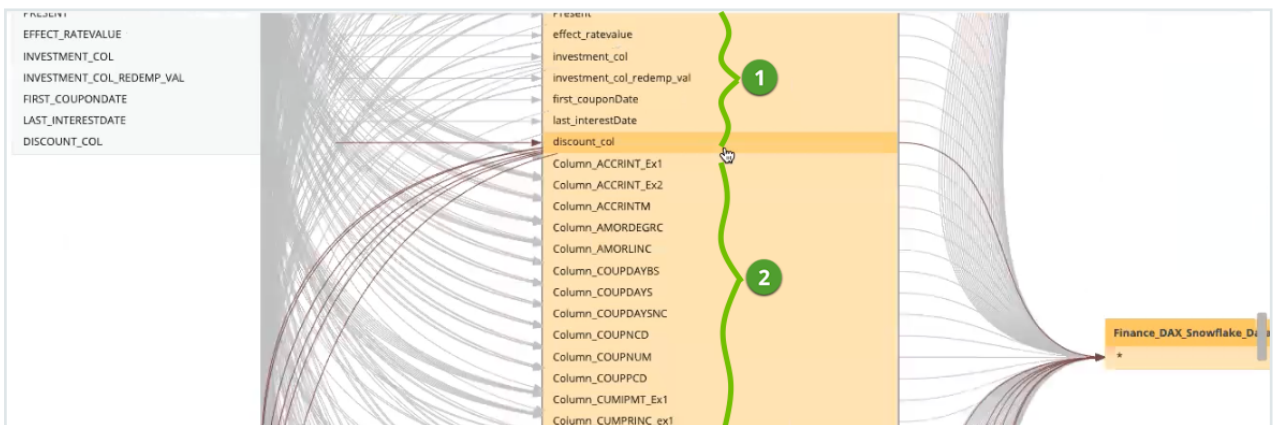
In the following example image:

- The six native columns (1) that are aggregated to derive the column Column_ACCRINT_Ex1 are shown in the graph.
- Outbound edges from the data source (2) show the relation between the columns in the data source and the native columns in the downstream data set (1).



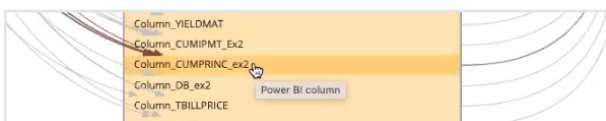
If you zoom out to the table-level lineage:

- The columns that are aggregated (1) to derive the calculated columns are shown in the graph.
- The calculated columns and measures (2) are shown.

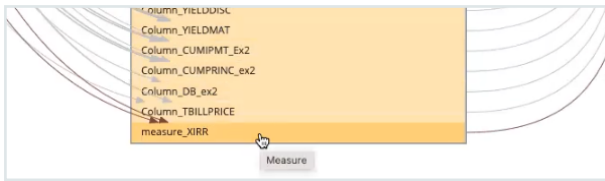


Tooltips in the technical lineage graph

As the information returned by the Power BI API doesn't distinguish between native Power BI columns and calculated columns, if you hover over a calculated column, the tooltip identifies it only as a Power BI column.



If the column is a measure, however, the tooltip confirms that it is indeed a measure.



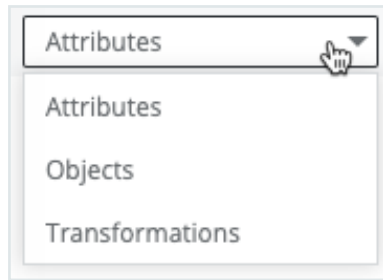
Note The inherent difference between calculated columns and measures is reflected in how their values are calculated. As shown in the previous image, due to the nature of measures and the basis upon which they are calculated, they have no incoming edges from other columns.

Review the results of DAX parsing

1. Open the asset page of a Power BI Report, Power BI Column or Power BI Table, and then click **Technical lineage**.
2. In the technical lineage toolbar, click **<>**.
 - » The source code pane opens. Here you can see the DAX query and compare it to the lineage.

```
Source Code 5728
Expand full source
76
77 CONNECTOR: 'FACTS INTERNETSALES.REVISIONNUMBER=>Facts internetsales.REVISIONNUMBER'
78 TRANSFORMATION: 'None'
79
80
81 CONNECTOR: 'Dim salesterritory.SALESTERRITORYALTERNATEKEY=>Facts internetsales.Column_LOOKUPVALUE_Ex2'
82 RELATED('Dim salesterritory' [SALESTERRITORYALTERNATEKEY])
83
84 CONNECTOR: 'Dim salesterritory.SALESTERRITORYKEY=>Facts internetsales.Measure_FILTER_Ex1'
85 SUMX(FILTER('Facts internetsales', RELATED('Dim salesterritory' [SALESTERRITORYKEY])=4), 'Facts internetsales' [SALESAMOUNT])
86
87 CONNECTOR: 'Dim salesterritory.SALESTERRITORYALTERNATEKEY=>Facts internetsales.Measure_COUNTX_Ex3'
88 COUNTX(FILTER('Facts internetsales', RELATED('Dim salesterritory' [SALESTERRITORYALTERNATEKEY])<=4), 'Facts internetsales' [SALESAMOUNT])
89
90
91 CONNECTOR: 'Facts internetsales.SALESAMOUNT=>Facts internetsales.Measure_FILTER_Ex1'
92 SUMX(FILTER('Facts internetsales', RELATED('Dim salesterritory' [SALESTERRITORYKEY])=4), 'Facts internetsales' [SALESAMOUNT])
```


3. In the technical lineage toolbar, in the drop-down list, select **Objects** and **Transformations**, to see a high-level view.



Show DAX calculations on Power BI Column asset pages

Data Analysis Expressions (DAX) is a programming language that is used in Power BI for creating calculated columns, measures and custom tables.

Power BI columns that are derived from DAX are shown in the technical lineage. However, the Collibra Data Lineage service instances are unable to parse DAX. Therefore, stitching between calculated columns in the technical lineage and the corresponding Power BI Column assets in Data Catalog is not possible.

You can, however, show DAX calculations for calculated columns and measures on Power BI Column asset pages. To do so, you only have to ensure that the Calculation Rule attribute type is part of the global assignment of the Power BI Column asset type.

Note All elements in the DAX, even comments for example, are included and shown in the Calculation Rule attribute.

The following additional information on Power BI Column asset pages can also help you interpret the lineage:

- If a calculated column is a measure, the Role in Report attribute has the value "Measure".
- The Technical Data Type attribute indicates the type of column, for example "String" or "Number".

Tip You can enable DAX parsing via Collibra AI. This feature:

- Allows you to create column-level lineage that includes your calculated columns and measures in Power BI.
- Enables stitching between calculated columns in the technical lineage and the corresponding Power BI Column assets in Data Catalog.

For complete information, go to [DAX parsing via Collibra AI](#).

Add the Calculation Rule attribute type to the global assignment

To show DAX calculations, the Calculation Rule attribute type must be part of the global assignment of the Power BI Column asset type. By default, it is not included.

Prerequisites

- You have a [global role](#) that has the **System administration** [global permission](#).

Steps

1. Open the Power BI Column asset type.
 - a. On the main toolbar, click , and then click  **Settings**.
 - » The [Collibra settings page](#) opens.
 - b. Click **Operating Model**.
 - » The [operating model settings](#) appear on the **Asset types** tab page.
 - c. In the overview of asset types, click Power BI Column.
2. In the tab pane, click **Global assignment**.

Tip If the Calculation Rule attribute type already exists in the table, you don't have to do anything more. However, as described in step 6, you might want to ensure that the **Min.** option is set to *1*, so that the attribute type automatically appear on the asset page.

3. Above the table, to the right, click **Edit**.
4. Above the table, to the right, click **Add characteristic**.
 - » The **Add a Characteristic** dialog box appears.

5. Search for and click **Calculation Rule**.
 - » The Calculation Rule attribute type appears at the bottom of the table.
6. If required, edit the minimum or maximum number of occurrences of the characteristic.

Option	Description
Min.	The minimum number of occurrences of the characteristic. Tip Set this option to <i>1</i>, to make the attribute type automatically appear on the asset page.
Max.	The maximum number of occurrences that you can assign to an asset type. Leave this option empty if you don't want a limit to the maximum number of occurrences.

7. Above the table, to the right, click **Save**.

Working with Tableau

This section addresses tasks and concepts that can be of interest to Collibra Admins who are working with Tableau.

Tableau supported data sources

Tableau is business intelligence software that can integrate with various data sources. When you ingest Tableau metadata, Collibra Data Lineage tries to automatically stitch the metadata to data sources registered in Data Catalog. It also creates a technical lineage that shows where metadata is used and how it transforms.

The following table shows the supported data sources in Tableau that have been tested, and whether or not technical lineage and stitching is supported for the data source. We cannot guarantee that stitching works as expected for other data sources or versions.

Note

- If you use custom SQL that is not supported by the Tableau metadata API, the technical lineage might not be complete. For complete information, see the Tableau documentation on [Tableau Catalog support for custom SQL](#) and [Tableau Lineage and custom SQL connections](#).
- If you use stored procedures, lineage is shown between the Tableau Data Source and the Tableau Worksheet, but the database information is missing, so stitching cannot be achieved.

Tip For stitching, you must correctly prepare the [Data Catalog physical data layer](#).

Data source	Version	Support for technical lineage	Support for stitching
Amazon Redshift	1.2.34.1058 and newer	Yes	Yes
Azure SQL server	Newest version	Yes	Yes
Azure SQL Data Warehouse	Newest version	Yes	Yes
Azure Synapse Analytics	Newest version	Yes	Yes
Dremio	20.0.0	Yes	Yes
Google BigQuery	Newest version	Yes	Yes
Greenplum	6.10 and newer	Yes	Yes
HiveQL (SQL-like statements)	2.3.5 and newer	Yes	Yes
IBM DB2	11.5 and newer	Yes	Yes
Oracle	11g, 12c and newer	Yes	Yes
PostgreSQL	9.4, 9.5 and newer	Yes	Yes
Microsoft SQL Server	2014, 2016 and newer	Yes	Yes
MySQL	5.7, 8 and newer	Yes	Yes

Data source	Version	Support for technical lineage	Support for stitching
Netezza	7.2.1.0 and newer	Yes	Yes
SAP HANA	2.00.40 and newer	Yes	Yes
Snowflake	Newest version	Yes	Yes
Spark SQL	2.4.3 and newer	Yes	Yes
Sybase Adaptive Server Enterprise	16.0 SP02 and newer	Yes	Yes
Teradata	15.0, 16.20.07.01 and newer	Yes	Yes

Test your connectivity with the Tableau server

Before you run the lineage harvester, you need to test your connectivity with the Tableau server.

Connectivity requires authentication. The user/token that you intend to use to ingest Tableau assets must be able to authenticate to your Tableau APIs via the command line, from the server on which you intend to install and run the lineage harvester.

Warning As of October 2022, Tableau is enforcing multi-factor authentication for Tableau Cloud Admin users. However, the lineage harvester doesn't support multi-factor authentication. Therefore, Tableau Cloud users with an Admin role must use token-based authentication. This does not affect Tableau Server users or Tableau Cloud users with an Explorer role.

Important By default, your Tableau token is valid for one year; however, you must use the token at least once every two weeks for it to remain valid. If a token is unused for two weeks, it expires. The life of a token can never be greater than one year, but you can configure it to be less than a year. The two-week usage frequency condition remains valid, regardless.

To ensure that you can authenticate and connect to the Tableau server, try the following procedures.

Make the signin API call using a cURL command

1. Create a JSON file called "signin.json".

The file should contain the following:

- For username/password authentication:

```
{
  "credentials": {
    "name": "YOUR_USER",
    "password": "YOUR_PASSWORD",
    "site": {
      "contentUrl": "YOUR_SITE_ID"
    }
  }
}
```

- For personal token-based authentication:

```
{
  "credentials": {
    "personalAccessTokenName": "YOUR_TOKEN_NAME",
    "personalAccessTokenSecret": "YOUR_TOKEN_SECRET",
    "site": {
      "contentUrl": "YOUR_SITE_ID"
    }
  }
}
```

2. Test this on your machine by running the following command:

curl "https://YOUR_TABLEAU_URL/api/3.7/auth/signin" -H "Content-Type: application/json" -X POST -d @signin.json

Tip To test on a Windows machine, you need to:

- a. Download and [install](#) the cURL Command-Line Tool.
- b. In Windows, click **Start > Run**, and then enter `cmd` in the **Run** dialog box.
- c. Run the following command:
`curl "https://YOUR_TABLEAU_URL/api/3.7/auth/signin" -H "Content-Type: application/json" -X POST -d @signin.json`

Check the login request that the lineage harvester sends to the Tableau server

1. Run the lineage harvester with the following parameters:

```
bin/lineage-harvester load-sources -Dakka.http.client.log-unencrypted-network-bytes=1024 -Dakka.loglevel=DEBUG
```

This generates many logs. In the log file, search for “signin”. The entry for “signin” will resemble the following log snippet, in which the login request is shown between curly brackets “{}”:

```
[DEBUG] [11/08/2021 14:03:18.411] [default-akka.actor.default-dispatcher-4] [akka.stream.Log(akka://default/system/StreamSupervisor-1)] [client-plain-text ToNet] Element: SendBytes ByteString(375 bytes)
50 4F 53 54 20 2F 61 70 69 2F 33 2E 37 2F 61 75 | POST /api/3.7/auth/signin HTTP/1
```

2. Verify that the request is the same as the one you used in the `signin.json` file.

Tableau hostname, schema, and system name mapping

To achieve end-to-end lineage and stitching, Collibra Data Lineage must match the full names of data objects in a technical lineage and the full names of their corresponding assets in Data Catalog. However, there are several situations that can impede full-name matching. In such cases, you can include a `hostnameMapping` section in your Tableau

<source ID> configuration file, to map the database, schema or system names that were returned by the Tableau APIs to the actual names of the assets in Data Catalog.

Tip "Mapping" means changing the full name of data objects as they appear in a technical lineage, so that they match the full names of their corresponding assets in Data Catalog.

The following example scenarios can impede full-name matching:

- Tableau can't derive the schema name. In this case, the schema name in the technical lineage is DEFAULT.
- You have schema-less external data sources, such as HiveQL, MySQL or Teradata. In this case, the database name in the technical lineage is also the schema name.
- You have a data access layer between Tableau and your external data source. In this case, Tableau might incorrectly interpret the data access layer as the database name, and the data source as the schema.
- You have data sources that are created based on tables from other data sources in Tableau. These data sources do not have schemas.
- The Tableau APIs returned a technical database or server name that is different than the real name of the database or server.

Warning

- `hostnameMapping` replaces the following deprecated properties:
 - The `databaseMapping` property.
 - The `databases` sub-section of the `collibraSystemNames` section.

`hostnameMapping` must not be used in combination with either of these properties.

For descriptions of these properties, go to the Tableau section in the [Prepare a <source ID> configuration file](#) topic.

If you use the `hostnameMapping` section, you can still use the `collibraSystemName` property in conjunction with the `files`, `connectors` or `cloudfiles` sub-sections.

Example configurations

- The following configuration:
 - Changes the found database name "Test" to "CData".
 - Changes the found schema name "DEFAULT" to "Jan_1_2022".
 - Adds the Colibra system name "TV_testing".

Important The system name must match the name you specified for the `id` property in the [lineage harvester configuration file](#), including for case-sensitivity.

```
"hostnameMapping": {  
  "found_dbname=Test;found_hostname=*;found_schema=DEFAULT": {  
    "dbname": "CData",  
    "schema": "Jan_1_2022",  
    "dialect": "spark",  
    "colibraSystemName": "TV_testing"  
  }  
}
```

- The following configuration:
 - For all found databases on the host "abc.net", changes their names to "CData".
 - Changes the found schema name "DEFAULT" to "Jan_1_2022".

```
"hostnameMapping": {  
  "found_dbname=*;found_hostname=abc.net;found_schema=DEFAULT": {  
    "dbname": "CData",  
    "schema": "Jan_1_2022",  
    "dialect": "spark",  
  }  
}
```

- The following configuration:
 - Changes the found database name "Test" to "CData".
 - Changes the found schema name "DEFAULT" to "Jan_1_2022".

```

"hostnameMapping": {
  "found_dbname=Test;found_hostname=*;found_schema=DEFAULT": {
    "dbname": "CData",
    "schema": "Jan_1_2022",
    "dialect": "spark",
  }
}

```

- The following configuration:
 - Changes the found database name "Test" to "CData".

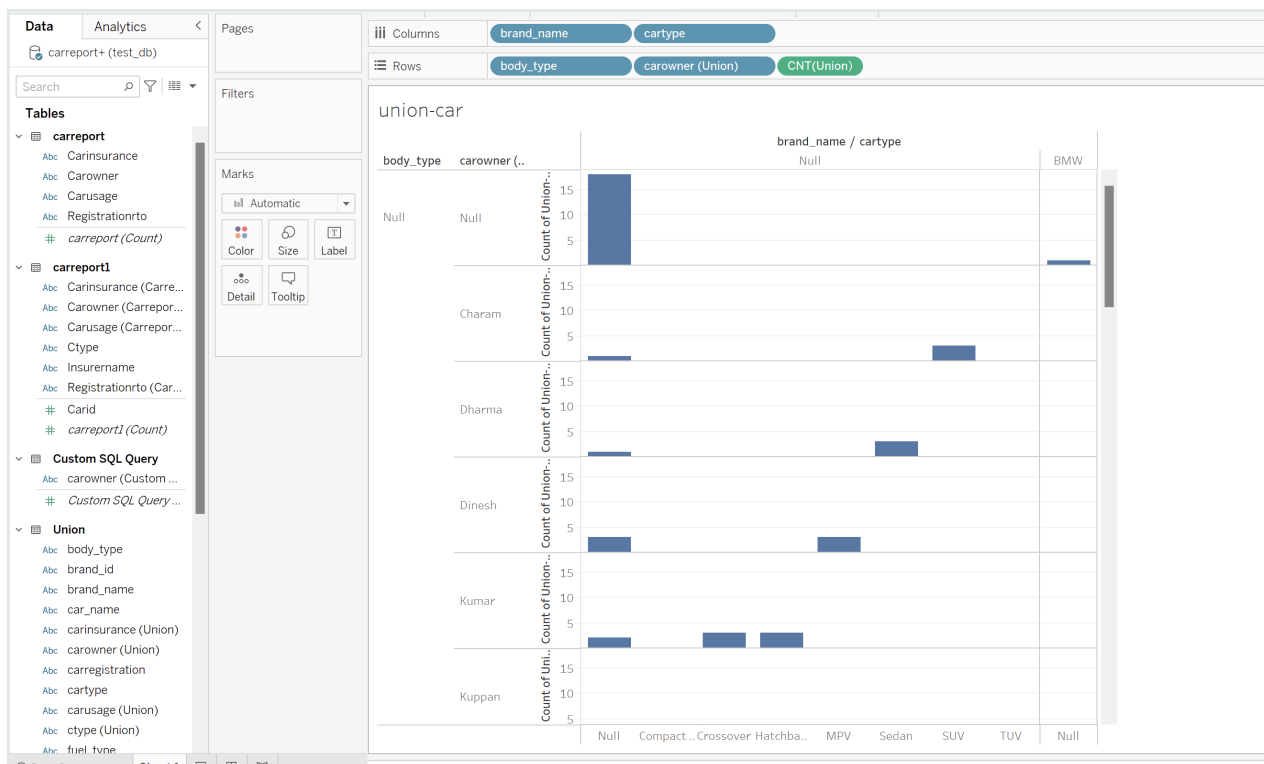
```

"hostnameMapping": {
  "found_dbname=Test;found_hostname=*;found_schema=DEFAULT": {
    "dbname": "CData",
  }
}

```

Working with unions and custom SQL

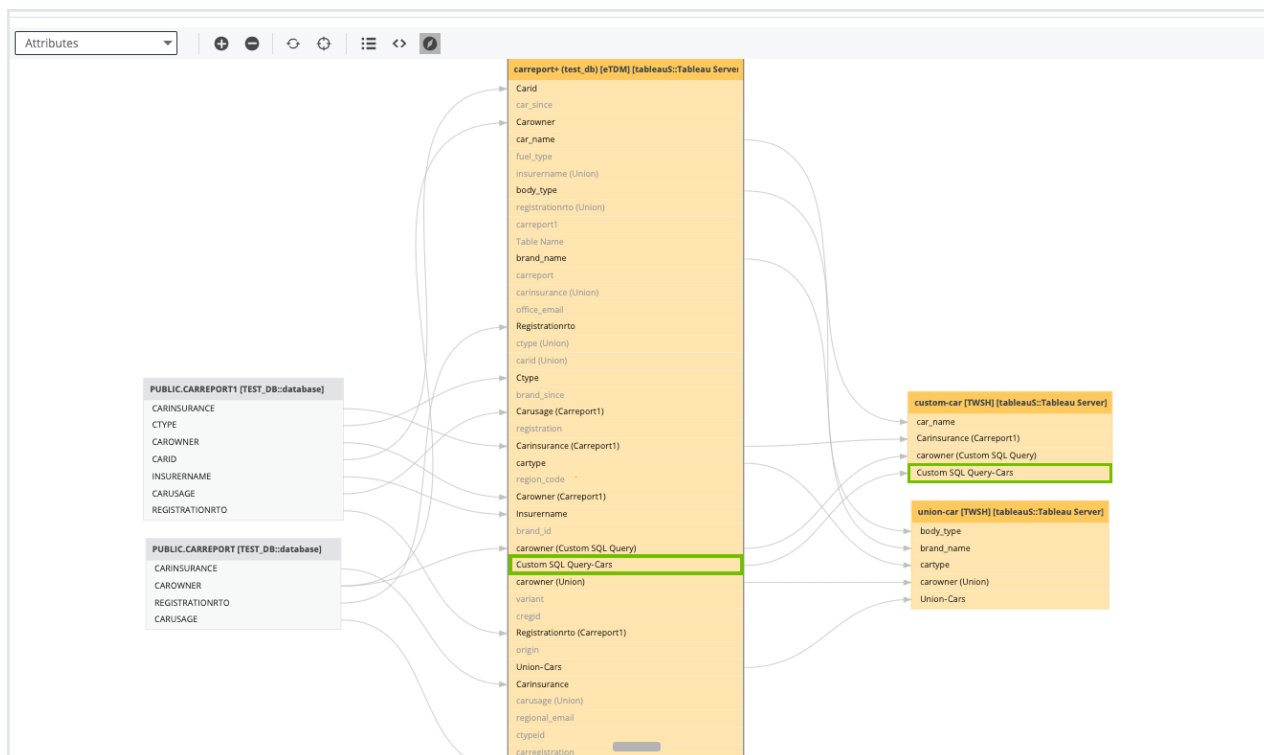
This topic describes how Collibra Data Lineage handles unions and custom SQL in Tableau.



Using custom SQL

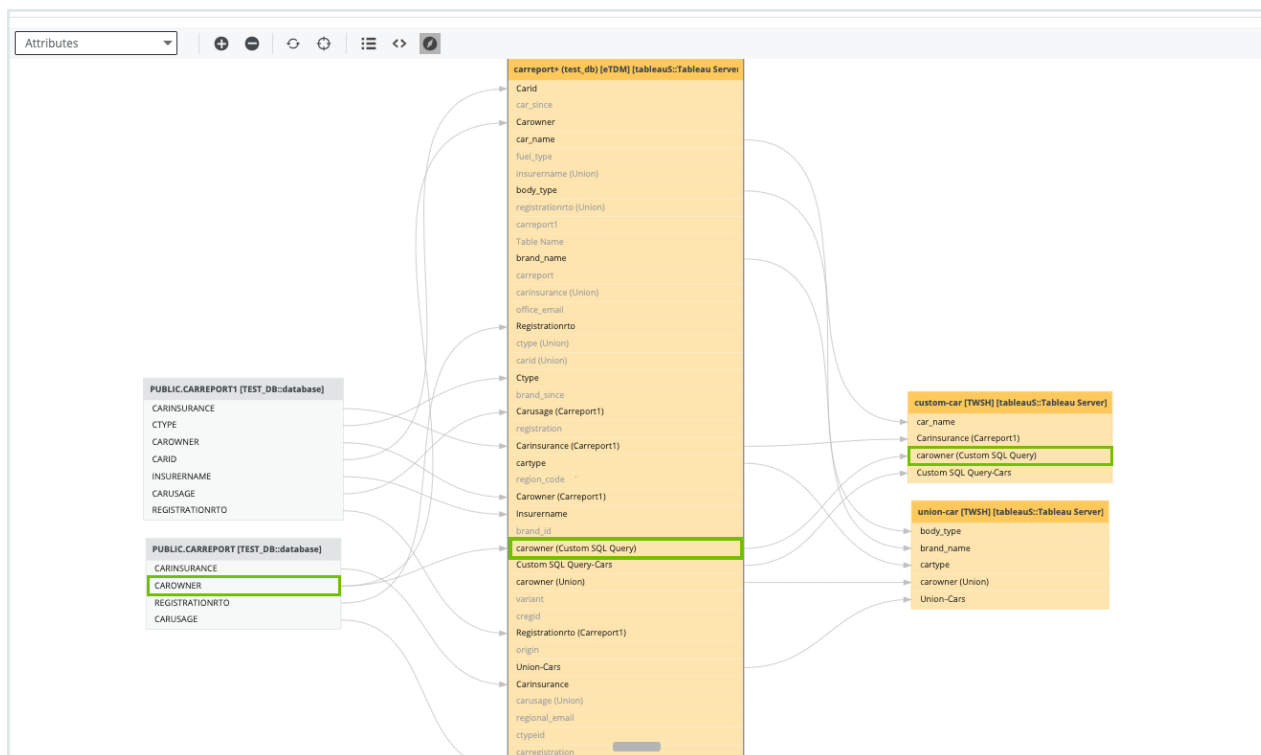
When you use custom SQL queries in Tableau, Tableau creates a new “table” with the name “Custom SQL query” and the type “Table”. The table includes the columns you selected and a column with the name “Custom SQL query”. This column is ingested in Collibra as a Tableau Data Attribute asset with the type “Table”, and it is always a Measure. The asset itself is shown in the dataset node, but it doesn’t have lineage from the upstream database node.

Example: "Custom SQL Query-Cars"



Columns that are mentioned in the custom SQL query are ingested with lineage from the upstream database node. The names of such columns are appended with the text "(Custom SQL Query)".

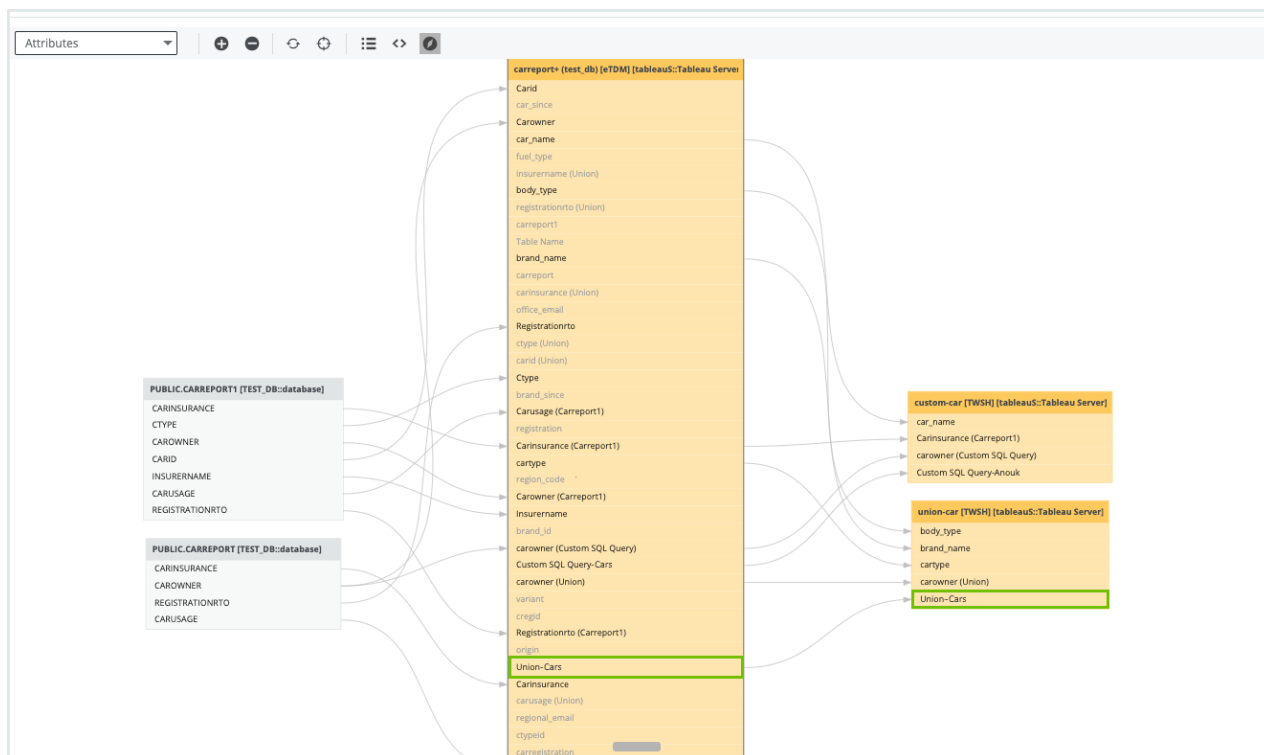
Example: "carowner (Custom SQL Query)"



Creating unions

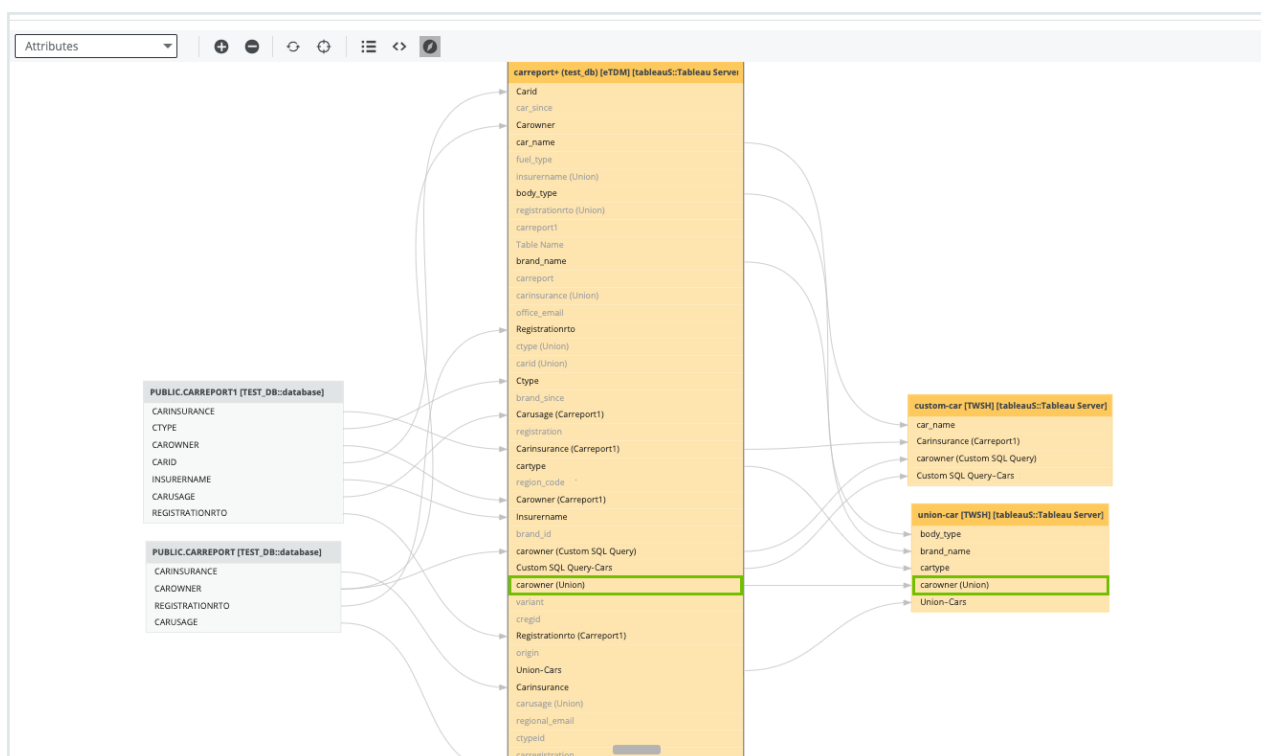
When you create a union, Tableau creates a new “table”. You can name it as you like, for example “Union-Cars”. The table includes all of the columns that are part of the union, and a column with the same name as the table. In this example, “Union-Cars”. This column is ingested in Collibra as a Tableau Data Attribute asset with the type “Table”, and it is always a Measure. The asset itself is shown in the dataset node, but it doesn’t have lineage from the upstream database node.

Example: “Union-Cars”



Columns in the table are ingested with lineage from the dataset node to the report node. The Column asset names are appended with the text "(Union)". There is no lineage from the upstream database node.

Example: "carowner (Union)"



Migrating Tableau assets to the new Tableau operating model

The end of life of Jobserver and all related Jobserver integrations has been announced for September 30, 2024. If you previously integrated Tableau via Jobserver and you created custom asset types, attribute types, or relation types, you'll very likely want to migrate all of that to the new [operating model](#), which is specific to integration via the [lineage harvester](#) or [Edge](#).

If you don't migrate your assets - if you merely start integrating Tableau via the lineage harvester or Edge - you'll lose all of the custom characteristics and metadata associated with your Tableau assets.

In this section, we provide instruction on how to migrate your Tableau assets and all associated metadata to the new operating model, via the lineage harvester or Edge.

About the Tableau migration

This section describes the terminology and methodology for migrating your existing Tableau assets to the new [Tableau operating model](#).

Terminology

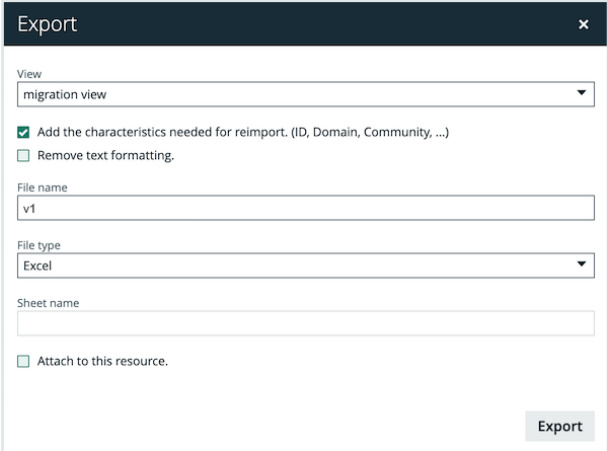
Term	Description
Tableau integration v1	The process of integrating and synchronizing Tableau metadata via the Data Catalog UI, including: • The Tableau assets that were created in the process. • Any custom asset types, attribute types and relation types. • Any customizations to the Tableau asset types. • Any customizations to your Tableau assets, for example added attributes and relations. • Any tags that you added to your Tableau assets. • The specific Tableau ingestion results, which differ from the v2 ingestion results.
Tableau integration v2	The process of integrating and synchronizing Tableau metadata via the lineage harvester, including: • The Tableau assets that were created in the process. • The specific Tableau ingestion results, which differ from the v1 ingestion results.

Methodology

The following is our methodology for migrating Tableau integration v1 metadata to the new operating model. For procedural steps, go to [Migration procedure](#).

Note The purpose of this document is to guide you through the migration of assets that were created via step 1 in the table below. That step is included here merely to present the complete context, from ingesting assets via Tableau integration v1, through migration.

No.	Step	Details
1	Integrate and synchronize Tableau metadata via Tableau integration v1.	Over time, you have likely customized the Tableau asset types, created custom attribute types and relation types, and added attributes and relations to your Tableau v1 assets. When you switch to Tableau integration v2, you want to ensure that you won't lose any of those customizations. All manually created asset types, attribute types and relation types will be migrated. You can migrate whatever you want, for example all manually created asset types, attribute types and relation types. Any characteristics from the v1 operating model that are not part of the v2 operating model can also be migrated. Note We recommend using the v2 operating model for asset types, but you can add any characteristics that suit your needs.
2	Integrate the same Tableau metadata into a different domain, this time via Tableau integration v2.	After successful integration, you will have: • Tableau integration v1 assets and their custom attributes and relations. • Tableau integration v2 assets. Important The new Tableau operating model is only available in Collibra versions 2021.10 and newer.
3	Create custom attribute types.	For every out-of-the-box v2 attribute type for which the value was manually edited in the v1 assets, create a custom asset type in Collibra, to ensure that the values of those attribute types in your v1 assets are not overwritten. For example, in your v2 integration, you have a Tableau Workbook asset with a description that was brought in from Tableau during the integration. However, for the same Tableau Workbook asset that was created in the v1 integration, you manually added lots of valuable details to the description and you don't want to lose those details. In this case, create a custom attribute type named Description Custom, so that the v2 description won't overwrite your detailed v1 description.

No.	Step	Details
4	Create a view in the domain to which you will import assets in step 9.	In the domain to which you will import assets, create a view with the characteristics you want to migrate. If, for example, you want to migrate the Data Source Type attribute from the Tableau v1 Data Source asset type, add the Data Source Type attribute to view in the relevant domain. Important Any characteristics that are not included in the view will be ignored during export.
5	Export Tableau v1 assets.	Export the Tableau assets from the v1 integration, to an Excel file, along with the custom attributes and relations that need to be transferred. Let's call this file "v1". When you export the assets, ensure that the view that you created in the previous step is selected in the Export dialog box. For example, as identified in the following image, you select the "migration view". You also need to ensure that the Add the characteristics needed for reimport is selected. 
6	Export Tableau v2 assets.	Export the Tableau Workbook assets from the v2 integration, to an Excel file, with any attributes and relations that are needed for re-import back into Collibra. Let's call this file "v2".

No.	Step	Details
7	Map the assets in files v1 and v2.	Open both files and map the asset Names in file v1 with the Asset IDs of the same assets in file v2. Tip To accomplish this mapping, you can perform a VLOOKUP based on the asset Name in file v1. For more information, see Microsoft's documentation on how to use the VLOOKUP function in Excel.
8	Copy the required characteristics from file v1 to file v2.	With your Excel files open side-by-side, copy characteristics from file v1 to file v2. For example, let's say that in file v1, you have a Tableau Data Source asset with the Data Source Type attribute type. Find for the corresponding Tableau Data Model asset in file v2 (because that v2 asset type maps to the Tableau Data Source v1 asset type) and copy the Data Source Type attribute value from file v1 to file v2. Ensure the column header is "Data Source Type".
9	Import your Excel file v2 back into Collibra.	Once you've reviewed your v2 file, to see that column headers and values look correct, you can import the file back into Collibra. This will add to your v2 assets all of the customized attributes that you curated for your v1 assets.
10	Verify the migration results	Compare your Tableau integration v1 assets to the respective Tableau integration v2 assets. Verify that the metadata that you manually added to your integration v1 assets has been retained in your v2 assets.
11	Delete your Tableau integration v1 assets and custom assets.	If you've reviewed the migration results and everything looks fine, you can delete your Tableau integration v1 assets and any assets of custom asset types.

Migration procedure

This topic shows the procedure to ingest metadata in Collibra via lineage harvester (Tableau integration v2) and migrating your Tableau assets.

Steps

1. Integrate Tableau via Jobserver (Tableau integration v1).
For complete information, see [Working with Tableau via Jobserver](#).
2. Integrate Tableau via the [lineage harvester](#) or [Edge](#) (Tableau integration v2).
When integration is complete, you will have:
 - A domain in Collibra with custom Tableau integration v1 assets and their custom attributes and relations.
 - A domain in Collibra with Tableau integration v2 assets.
3. For every out-of-the-box v2 attribute type for which the value was manually edited in the v1 assets, create a custom asset type in Collibra, to ensure that the values of those attribute types in your v1 assets are not overwritten. For more information, see [About the Tableau migration](#).
4. In the domain to which you will import assets in step 9, [create a view](#) with the characteristics you want to migrate.

Important Any characteristics that are not included in the view will be ignored during export.

5. [Export](#) Tableau v1 assets to an Excel file. Name this file "v1".
6. Export Tableau v2 assets to an Excel file. Name this file "v2".
7. Map the assets in files v1 and v2.

Tip To accomplish this mapping, you can perform a VLOOKUP based on the asset Name in file v1. For more information, see Microsoft's documentation on how to use the [VLOOKUP function in Excel](#).

8. Copy the required characteristics from file v1 to file v2.
9. Once you've reviewed your v2 file, to see that column headers and values look correct, you can import the file back into Collibra.
10. Verify that the metadata that you manually added to your integration v1 assets has been retained in your v2 assets.
11. If you've reviewed the migration results and everything looks fine, you can delete your Tableau integration v1 assets and any assets of custom asset types.

Naming convention

When you synchronize Tableau, Colibra follows a strict naming convention for the names of the new assets. Each asset has a display name and full name. The full name represents the asset path from asset to the database it belongs to. You can freely edit the display name. However, you should never edit the full name, because Data Catalog needs it for a successful migration. Changing the full name may also break the synchronization process.

Warning We highly recommend that you not edit the full names of any Tableau assets. Doing so will likely lead to errors during the migration and synchronization process.

Soft delete of "Missing from source" assets

When you integrate Tableau, data objects in the data source are ingested as assets in Data Catalog. But what if, during synchronization, some of the data objects can no longer be found in the data source because they were moved or deleted? In that case, the status of the corresponding assets of the missing data objects becomes "Missing from source". We refer to this asset status evolution as a "soft delete". If you want, you can then run the [Delete Missing Assets workflow](#) to permanently delete the assets, or manually delete them.

Note If, for example, you remove the permissions to access a certain data object and then run the lineage harvester, the status of the corresponding asset in Data Catalog changes to "Missing from source". If you then add back the permissions to the data object and run the lineage harvester, the status of the asset will revert to the status it had before "Missing from source".

Delete the "Missing from source" assets

The [Delete Missing Assets workflow](#) enables you to delete all assets with the status "Missing from source". You can download the workflow file from the [Colibra Developer Portal](#) and deploy it in your Colibra environment.

Important Be sure to review assets before you delete them, as they might contain important information that will also be deleted.

Tip If you manually delete assets that are represented in a technical lineage, they are still shown in the technical lineage. To delete the corresponding assets of missing data objects and also delete the assets from the technical lineage, you have to:

1. Run the lineage harvester, or wait for your scheduled synchronization job to run.
 - » The technical lineage is refreshed and the status of the assets in Data Catalog becomes “Missing from source”.
2. Run a workflow to delete all assets with the status “Missing from source”, or manually delete them.

Missing stitching and possible solutions

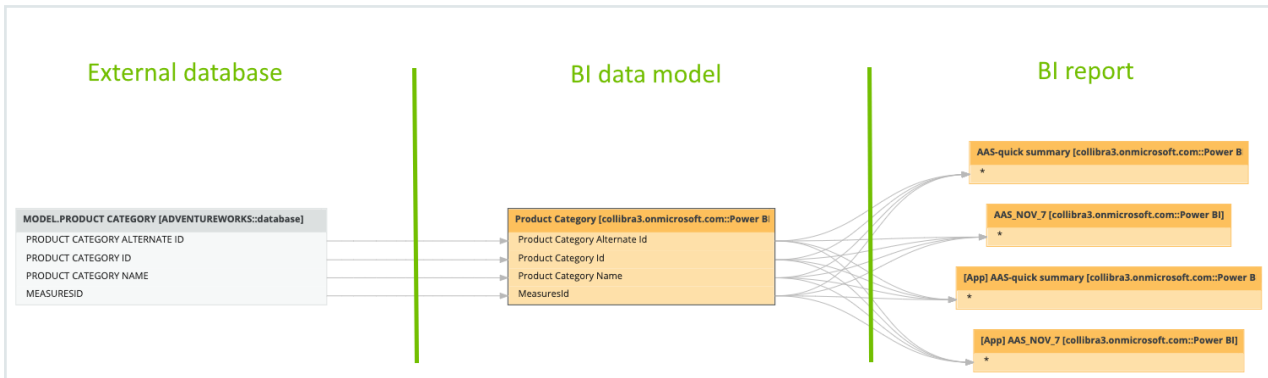
This topic provides some examples where stitching is missing, and some advice on how to achieve stitching.

For a more in-depth examination of stitching, how it works, and what causes stitching to break, go to [Stitching for BI tool integrations](#).

Note This is relevant for MicroStrategy, Power BI, SSRS-PBRS and Tableau. Collibra Data Lineage currently does not offer stitching for Looker assets.

The technical lineage graph without stitching

Go to the relevant asset page and click the **Technical Lineage** tab. There will most likely be three nodes in the technical lineage graph.



Note If you have integrated MicroStrategy, there will be four nodes. Instead of the BI data model node, there will be two nodes, one representing MicroStrategy Attributes, the other representing MicroStrategy Facts. There could also be four groups if you are integrating SQL Server Reporting Services (SSRS) or Power BI Report Server (PBRs) and have a shared data set.

No.	Node	Description
1	External database	This node represents the table from the database you used to create the report in your BI tool. This node is a prerequisite for stitching. If it is not shown in the technical lineage, stitching is not possible.
2	BI data model	This node represents the data set that you used to create the report in your BI tool. This node is always stitched because Collibra Data Lineage knows the full name of the data set in your BI tool, and it creates the corresponding BI Data Set asset with the exact same name. This is referred to as BI stitching.
3	BI report	This node represents the report you created in your BI tool. It is always part of the technical lineage. Like the BI data model node, this node is always stitched because Collibra Data Lineage knows the full name of the report in your BI tool, and it creates the corresponding BI Report asset with the exact same name.

Example reasons for missing stitching and possible solutions

Here are a few common examples of missing stitching and possible solutions for achieving stitching.

Any data source: useCollibraSystemName property is set to "true", but system names are not specified

When preparing your [lineage harvester configuration file](#), you can set the useCollibraSystemName property to true, to indicate that you want to use the system or server name of a data source to match to the System asset you created when you [prepared the physical data layer](#). This is useful when you have multiple databases with the same name, in different systems or servers.

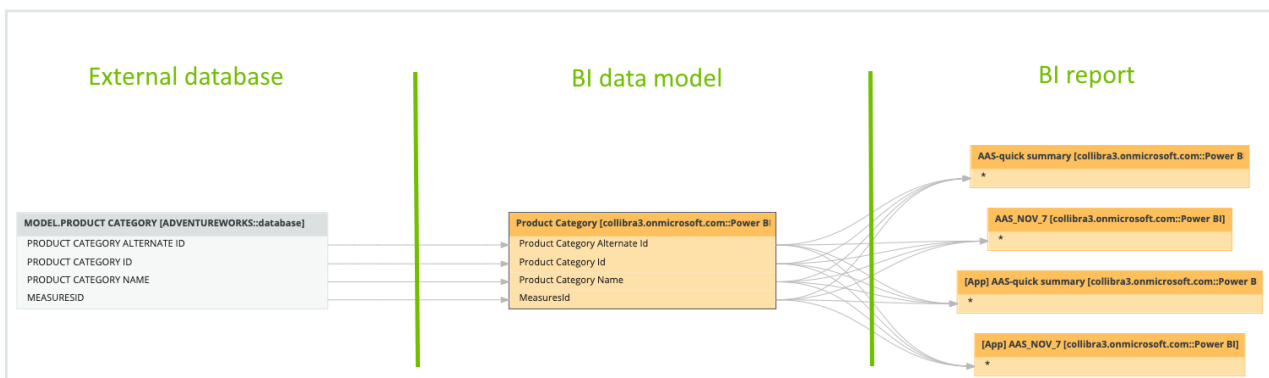
By default, the useCollibraSystemName property is set to false. If you want to use it, set it to true.

Important

- If you set useCollibraSystemName to true, you have to use the collibraSystemName property (either in your lineage harvester configuration file or in your [<source-ID> configuration file](#), depending on your data source) to specify the name of the System asset you created when preparing the physical data layer.
- If you set the useCollibraSystemName property to true, but you don't define the collibraSystemName in the [<source ID> configuration file](#), the system name in the technical lineage is DEFAULT. In this case, stitching is missing.

If you set the useCollibraSystemName property to false, the lineage harvester ignores the collibraSystemName property.

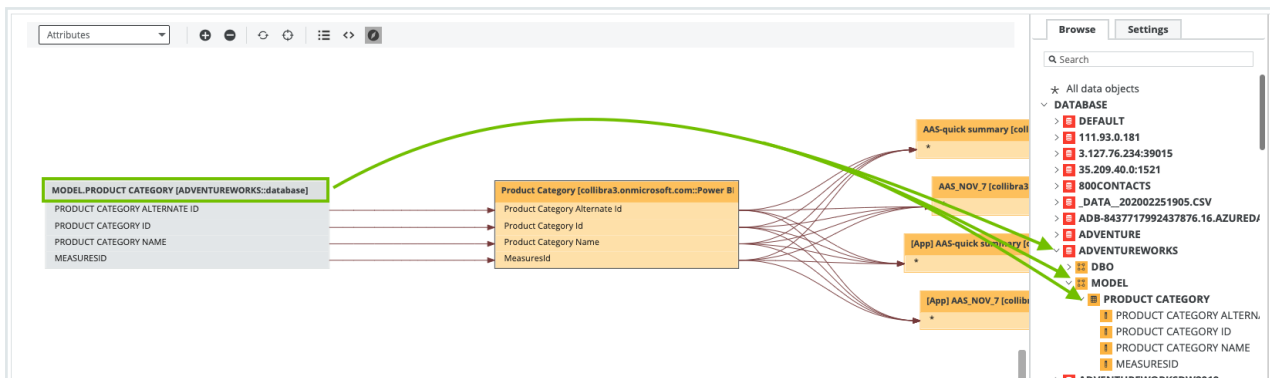
Power BI: database names don't match



First, let's look at the name of an unstitched database table in the technical lineage graph:

```
MODEL.PRODUCT CATEGORY [ADVENTUREWORKS::database]
```

We can identify the following:



- The database name: ADVENTUREWORKS
- The schema name: MODEL
- The table name: PRODUCT CATEGORY

Now let's find the table in the **Stitching** tab:

1. Click the **Settings** tab.
2. Click **Show status**.
3. Click the **Stitching** tab.
 - » The Stitching tab shows a list of all tables that exist in Data Catalog and on the Colibra Data Lineage service instance.

Use the Search field to find the unstitched database table **PRODUCT CATEGORY**.

Sources	
Stitching	
Search product category	
Full asset path	Found in
AAS-MODEL > MODEL > PRODUCT CATEGORY	Catalog
ADVENTUREWORKS > MODEL > PRODUCT CATEGORY	Technical Lineage
colibra3.onmicrosoft.com > powerbicolibraczech > BrnoWorkspace > Datasets > AAS_NOV_7 > Product Category	Catalog & Technical Lineage

In the **Found in** column, the value "Technical Lineage" confirms what we already know: the table was found only in the technical lineage. An exactly matching asset was not identified in Colibra.

Now try to find a likely match. Look for a table that has the same name and the value "Catalog" in the **Found in** column.

The table shown in the following image looks like a match. The schema and table names match exactly; only the database names differ.

Sources	
Stitching	
Search product category	
Full asset path	Found in
AAS-MODEL > MODEL > PRODUCT CATEGORY	Catalog
ADVENTUREWORKS > MODEL > PRODUCT CATEGORY	Technical Lineage
collibra3.onmicrosoft.com > powerbicollibra3.cz > BrnoWorkspace > Datasets > AAS_NOV_7 > Product Category	Catalog & Technical Lineage

To achieve stitching:

1. Prepare the database mapping section of your **Power BI <source ID> configuration file** as follows:

```
{
  "found_dbname=adventureworks;found_hostname=*": {
    "dbname": "aas-model",
  }
}
```

2. Run the lineage harvester again.
 - » Stitching is achieved

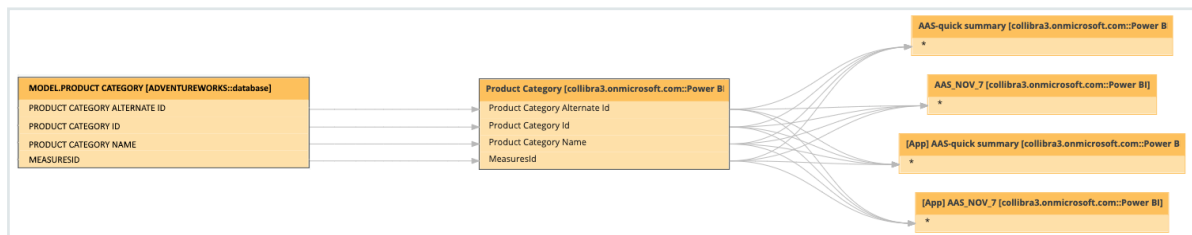
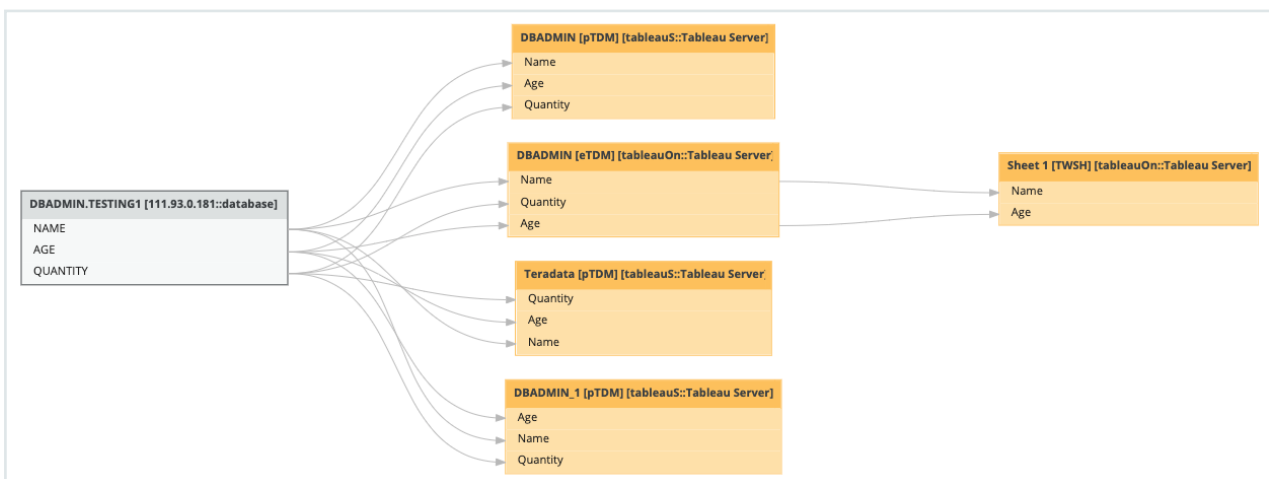


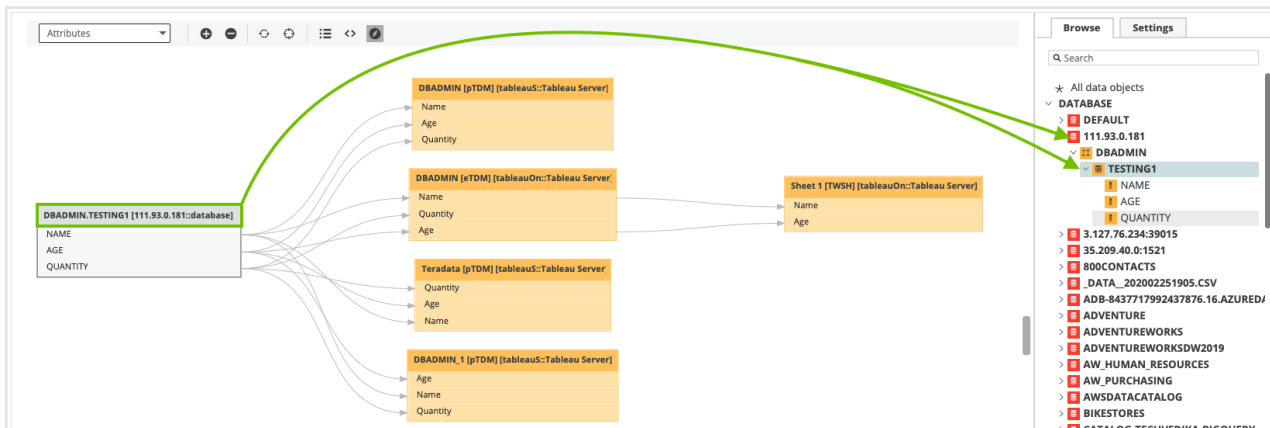
Tableau: database names don't match



First, let's look at the name of the unstitched database table in the technical lineage graph:

DBADMIN.TESTING1 [111.93.0.181::database]

We can identify the following:



- The database name: 111.93.0.181
- The schema name: DBADMIN
- The table name: TESTING1

Now let's find the table in the **Stitching** tab:

1. Click the **Settings** tab.
2. Click **Show status**.
3. Click the **Stitching** tab.
 - » The Stitching tab shows a list of all tables that exist in Data Catalog and on the Collibra Data Lineage service instance.

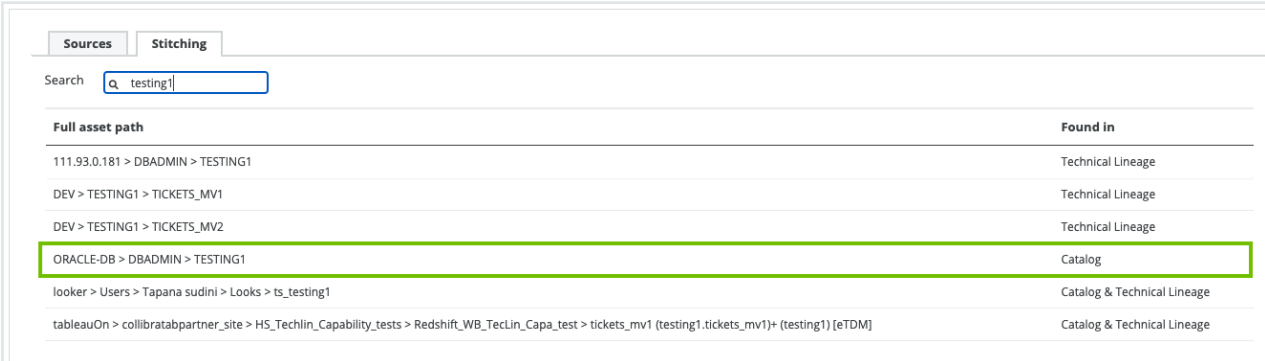
Use the Search field to find the unstitched table TESTING1.

Sources		Stitching
Search		testing1
Full asset path		Found in
111.93.0.181 > DBADMIN > TESTING1		Technical Lineage
DEV > TESTING1 > TICKETS_MV1		Technical Lineage
DEV > TESTING1 > TICKETS_MV2		Technical Lineage
ORACLE-DB > DBADMIN > TESTING1		Catalog
looker > Users > Tapana sudini > Looks > ts_testing1		Catalog & Technical Lineage
tableauOn > collibratabpartner_site > HS_TechLin_Capability_tests > Redshift_WB_TecLin_Capa_test > tickets_mv1 (testing1.tickets_mv1)+ (testing1) [eTDM]		Catalog & Technical Lineage

In the **Found in** column, the value "Technical Lineage" confirms what we already know: the table was found only in the technical lineage. An exactly matching asset was not identified in Colibra.

Now try to find a likely match. Because `DBADMIN.TESTING1` `[111.93.0.181::database]` was found only in the technical lineage, we know the match we're looking for must have the value "Catalog" in the **Found in** column.

The table shown in the following image looks like a match. The schema and table names match exactly; only the database names differ.



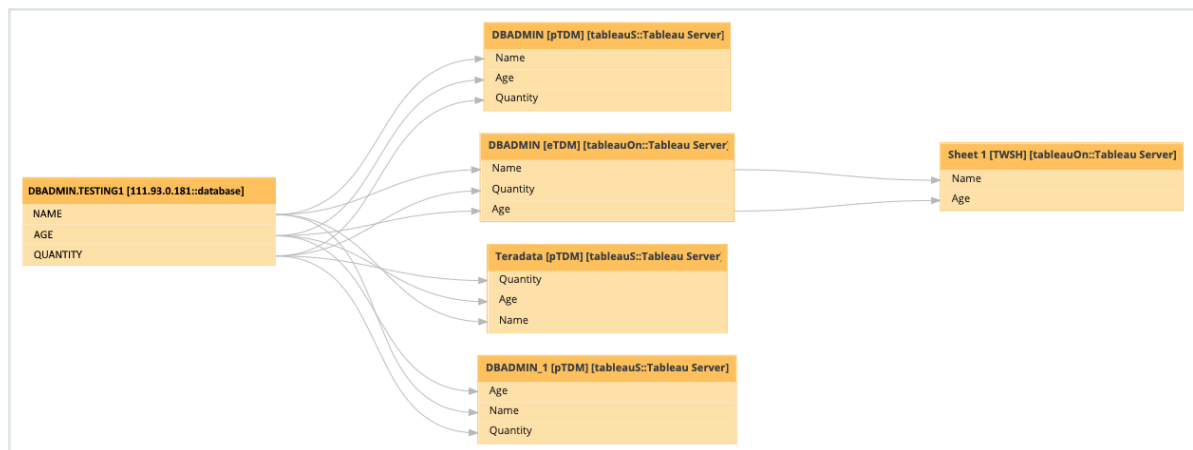
Full asset path	Found in
111.93.0.181 > DBADMIN > TESTING1	Technical Lineage
DEV > TESTING1 > TICKETS_MV1	Technical Lineage
DEV > TESTING1 > TICKETS_MV2	Technical Lineage
ORACLE-DB > DBADMIN > TESTING1	Catalog
looker > Users > Tapana sudini > Looks > ts_testing1	Catalog & Technical Lineage
tableauOn > collibratabpartner_site > HS_Techlin_Capability_tests > Redshift_WB_TecLin_Capa_test > tickets_mv1 (testing1.tickets_mv1)+ (testing1) [eTDM]	Catalog & Technical Lineage

To achieve stitching:

1. Configure the `hostnameMapping` section in your Tableau `<source ID>` configuration file as follows:

```
"hostnameMapping": {
  "found_dbname=111.93.0.181;found_hostname=*: {
    "dbname": "oracle-db",
    "dialect": "oracle"
  }
}
```

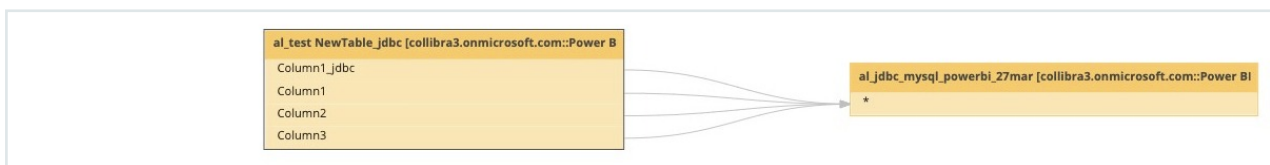
2. Run the lineage harvester again.
 - » Stitching is achieved



Power BI: Unsupported Power Query M function or calculated columns

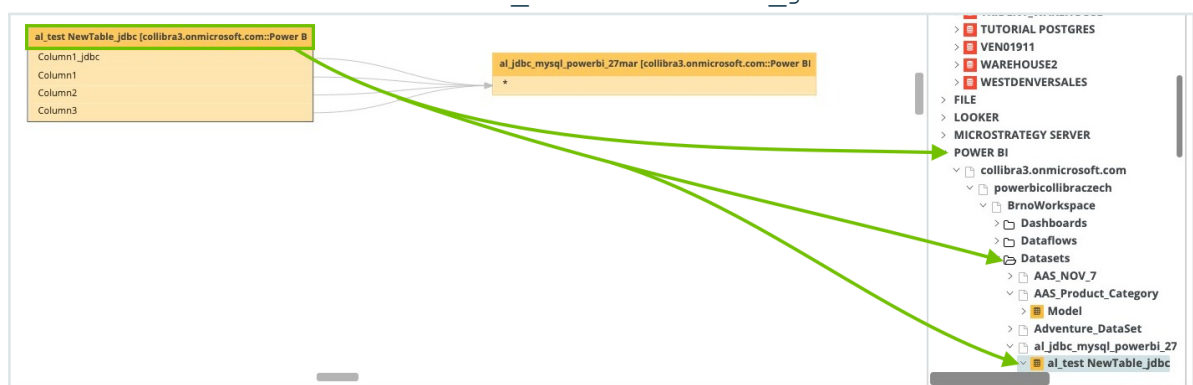
Collibra Data Lineage does not support DAX. Therefore, calculated columns result in missing lineage, as do unsupported Power Query M function.

In this example, notice that the database node, which should be situated to the left of the BI data model node, is missing from the technical lineage graph:



We can identify that these nodes represent, respectively:

- A Power BI data set table named `al_test NewTable_jdbc`



- A Power BI report named `al_jdbc_mysql_powerbi_27Mar`

Note The same scenario can surface for Tableau if, for example, you do not have sufficient permission, or if you have stored procedures or custom SQL that is not supported by Tableau Catalog.

Examine the BI data set table, to see if you can identify a problem

1. Click the **Settings** tab.
2. Click **Show status**.
 - » The Sources tab shows a list of all data sources on the Collibra Data Lineage service instance.
3. Click one or more checkboxes, to show the transformations and source code fragments for specific data sources, or clear all checkboxes, to show for all data sources.

Transformations and source code fragments are shown in the transformations table below. We can quickly identify that there was an analyze error, because the MySQL.Database function is not supported.

All transformations				
Full-text search		Q NewTable_jdbc	Filter by	ERROR
		Export All Transformations		
ID	Name	Status code	Status description	Group name
8935	PowerBI Table: al_test NewTable_jdbc PowerBI Dataset: al_jdbc_mysql_powerbi_27mar in workspace: BrnoWorkspace [8e86429d-f985-4a81-818d-8e05ac256a74]	ANALYZE_ERROR	Function not implemented: MySQL.Database	None
<pre>1 let 2 Source = MySQL.Database("3.132.214.161", "al_test", [ReturnSingleDatabase=true]), 3 al_test_NewTable_jdbc = Source([Schema="al_test",Item="NewTable_jdbc"])[Data] 4 in 5 al_test_NewTable_jdbc 6 Function not implemented: MySQL.Database</pre>				

To achieve stitching, ensure that your queries only include [supported Power Query M functions](#). We also encourage you to create an Ideation ticket via the [Collibra Integrations Ideation Portal](#), if you'd like to request support for a particular function.

Tableau: Insufficient Tableau permissions

If you use a Viewer role or an Explorer role in Tableau, without access to external assets, stitching is always missing. To achieve stitching, ensure that your Tableau user has access to the external assets for which you want lineage.

For complete details on the effect of roles and permissions on technical lineage and stitching, including how to provide necessary permissions to get lineage and automatic stitching with an Explorer role, go to [Tableau roles and permissions](#).

Tableau: Unsupported custom SQL in Tableau

Tableau Catalog doesn't support all custom SQL and stored procedures. For complete information, see the Tableau documentation [Tableau Catalog support for custom SQL](#).

If a Tableau data object uses an unsupported custom SQL or stored procedure, the external database node, which should be situated to the left of the BI data model node, is missing in the technical lineage graph, as shown in the following example image.



Synchronization: Continue on error

This feature allows for continuous processing of an import or synchronization job, even if one or more commands fail. Before the release of this feature, calls to the Import and Sync APIs either fully succeed or fully fail. You might wait for a lengthy import or synchronization job to complete, only to have it fail completely because of a single error.

Now, commands that have validation errors and those that failed to execute are skipped, allowing the processing of valid commands to continue until the job is complete or until an error threshold is met. The error threshold is determined by the "Number of failed commands before stopping import job" setting in [Colibra Console](#). The default value is 100.

This feature is relevant for the `full-sync` and `sync` commands.

For more information, see the Import API Documentation in the [Colibra Developer Portal](#).

Benefits of this feature

- Errors are skipped and valid commands are processed, instead of immediate and complete failure of the job.

- All errors are identified at once, reducing the chances of running a job multiple times, only to discover additional errors.
- Complete error information, including the resource identifier, to quickly identify the source and reason for errors.

Job results

The following table shows the four possible job results for an import or synchronization job:

Job result	Description
Success	The job was completed without errors.
Completed With Error	Errors were detected, but the error threshold was not reached and the job was completed.
Aborted	The error threshold was exceeded, at which point, the job was stopped. All commands that were executed before the stoppage stay committed.
Failure	The job was stopped and any executed commands were rolled back.

List of errors

You can view the results of a synchronization job in the [Activities list](#).

Admin Istrator Edit Reset password								
Overview	Delete ⌵							
Groups	☐	Created	Name	Status	Job Result ↑	Started	Finished	Results ⌵
Responsibilities	☐	8/11/2022 10:15 AM	Synchronization of batch for I...	Completed	Success	8/11/2022 10:15 AM	8/11/2022 10:15 AM	Results ⌵
History	☐	8/10/2022 2:42 PM	Synchronization of batch for I...	Completed	Success	8/10/2022 2:42 PM	8/10/2022 2:42 PM	Results ⌵
Activities	☐	8/11/2022 10:24 AM	Synchronization of batch for I...	Completed	Completed With Error	8/11/2022 10:24 AM	8/11/2022 10:24 AM	Results ⌵
	☐	8/10/2022 2:10 PM	Synchronization of batch for I...	Completed	Success	8/10/2022 2:10 PM	8/10/2022 2:10 PM	Results ⌵
	☐	8/12/2022 4:06 PM	Import	Completed	Success	8/12/2022 4:06 PM	8/12/2022 4:06 PM	Results ⌵
	☐	8/12/2022 4:28 PM	Synchronization of batch for I...	Completed	Success	8/12/2022 4:28 PM	8/12/2022 4:28 PM	Results ⌵
	☐	8/12/2022 4:27 PM	Import	Error	Failure	8/12/2022 4:27 PM	8/12/2022 4:27 PM	Results ⌵
	☐	8/10/2022 2:39 PM	Synchronization of batch for I...	Completed	Success	8/10/2022 2:39 PM	8/10/2022 2:39 PM	Results ⌵
	☐	8/10/2022 2:10 PM	Import	Completed	Success	8/10/2022 2:10 PM	8/10/2022 2:10 PM	Results ⌵
	☐	8/10/2022 2:51 PM	Synchronization of batch for I...	Completed	Success	8/10/2022 2:51 PM	8/10/2022 2:52 PM	Results ⌵
	☐	8/11/2022 10:21 AM	Synchronization of batch for I...	Completed	Completed With Error	8/11/2022 10:21 AM	8/11/2022 10:21 AM	Results ⌵
	☐	8/11/2022 10:16 AM	Synchronization of batch for I...	Completed	Aborted	8/11/2022 10:16 AM	8/11/2022 10:16 AM	Results ⌵
	☐	8/11/2022 1:23 PM	Synchronization of batch for I...	Completed	Completed With Error	8/11/2022 1:23 PM	8/11/2022 1:24 PM	Results ⌵
	☐	8/10/2022 2:34 PM	Synchronization of batch for I...	Completed	Success	8/10/2022 2:34 PM	8/10/2022 2:35 PM	Results ⌵
	☐	8/10/2022 2:15 PM	Import	Completed	Success	8/10/2022 2:15 PM	8/10/2022 2:15 PM	Results ⌵
	☐	8/11/2022 10:20 AM	Synchronization of batch for I...	Completed	Completed With Error	8/11/2022 10:20 AM	8/11/2022 10:21 AM	Results ⌵

When you click **Results** in the relevant row, a dialog box opens, showing a general summary of the job. For jobs with the job result Completed With Error, Aborted, or Failure,

the dialog box includes a link to a list of errors. The list of errors includes the following information:

- The resource type.
- The index number.
- The resource identifier.
- An error message.

Collibra Data Lineage service instances

The Collibra Data Lineage service processes and analyzes the harvested metadata from [supported \(meta\)data sources](#) and uploads it to Data Catalog. The Collibra Data Lineage service processes or stores only metadata, but not actual data.

When you run the lineage harvester or synchronize the technical lineage on Edge, the lineage harvester or technical lineage via Edge firstly connects to any available Collibra Data Lineage service instance to determine your cloud provider and geographic location of your Collibra Data Intelligence Cloud environment. Then, the lineage harvester or technical lineage via Edge sends the harvested metadata to the Collibra Data Lineage service instance with the same cloud provider and geographic location.

Currently, your metadata can be processed on one of the following Collibra Data Lineage service instances:

Server	IP address	DNS name
techlin-aws-ca	15.222.200.199	techlin-aws-ca.collibra.com
techlin-aws-eu	18.198.89.106	techlin-aws-eu.collibra.com
techlin-aws-sg	13.228.38.245	techlin-aws-sg.collibra.com
techlin-aws-us	54.242.194.190	techlin-aws-us.collibra.com
techlin-azure-eu	51.105.241.132	techlin-azure-eu.collibra.com
techlin-azure-us	20.102.44.39	techlin-azure-us.collibra.com
techlin-gcp-au	35.197.182.41	techlin-gcp-au.collibra.com
techlin-gcp-ca	34.152.20.240	techlin-gcp-ca.collibra.com

Server	IP address	DNS name
techlin-gcp-eu	35.205.146.124	techlin-gcp-eu.collibra.com
techlin-gcp-sg	34.87.122.60	techlin-gcp-sg.collibra.com
techlin-gcp-uk	35.234.130.150	techlin-gcp-uk.collibra.com
techlin-gcp-us	34.73.33.120	techlin-gcp-us.collibra.com

Important You have to allow all Collibra Data Lineage service instances in your geographic location. For example, if your data is located in Europe, you have to allow the following Collibra Data Lineage service instances: techlin-aws-eu and techlin-gcp-eu. In addition, we highly recommend that you always allow the techlin-aws-us instances as a backup, in case the lineage harvester cannot connect to other Collibra Data Lineage service instances.

Business stewards

This section caters primarily to the following business stewards:

Types of business stewards	What you want from Collibra Data Lineage
Governance roles: • Data Governance Consultant • Data Governance Manager • Data Intelligence Director • Data Quality Officer • Data Steward • Enterprise Data Steward	• Easily identify and view certified reports. • View business lineage. • Assign business terms to BI assets. • Tell a story about the data.
Analyst roles: • Business Analyst • Data Analyst • Data Scientist • Quantitative User Researcher • Operations Manager • Program Manager • Product Manager • Project Manager	• Use dashboards for an overall view of the most important information. • Certify and view reports. • Check technical lineage for data set life cycles. • Check for missing data and request new integrations, if necessary. • Identify data owners.

Technical lineage

Automatic stitching for technical lineage

BI tool business logic

Technical lineage and stitching for BI tool integrations

Business lineage

Differences between Technical lineage and business lineage

BI integration concepts

dclxiii

dclxvi

dclxviii

dclxxi

dclxxxii

dclxxxiv

dclxxxvii

Technical lineage

Technical lineage is a detailed [lineage graph](#) that shows how data transforms and flows from source to destination across its entire lifecycle. It enables you to easily discover where tables and columns are used and how they relate to each other.

In this topic

- [Asset types](#)
- [Data objects](#)
- [Exporting technical lineage information](#)
- [Naming convention](#)

Asset types

You can view a technical lineage for the following asset types:

- Table
- Column
- Looker Look
- MicroStrategy Report
- MicroStrategy Table
- MicroStrategy Column
- Power BI Report
- Power BI Table
- Power BI Column
- SSRS Report
- SSRS Table
- SSRS Column
- Tableau Worksheet
- Tableau Data Attribute

Note The Technical lineage tab is only shown if you have a [global role](#) with the following [global permissions](#):

- Catalog
- Technical Lineage

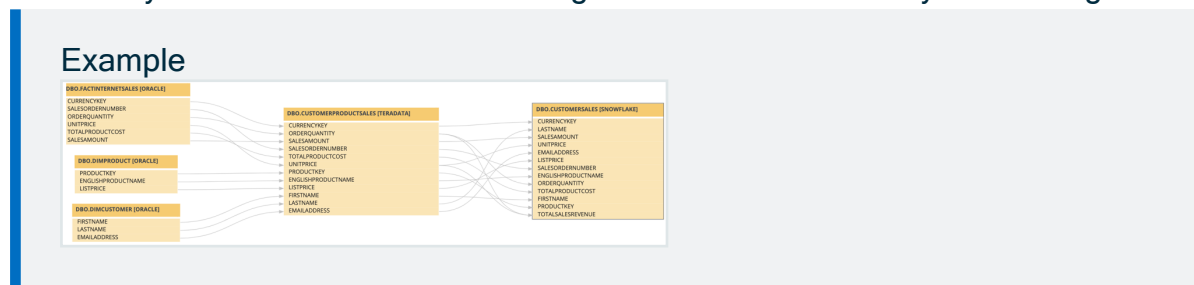
During the technical lineage process, relations of the type "Data Element targets / sources Data Element" are automatically created:

- Between data objects in your data source and their corresponding assets in Data Catalog, including the asset that you create when [preparing the Data Catalog physical data layer](#).
- If you are integrating a BI tool, between ingested assets from BI sources and Data Catalog assets from registered data sources.

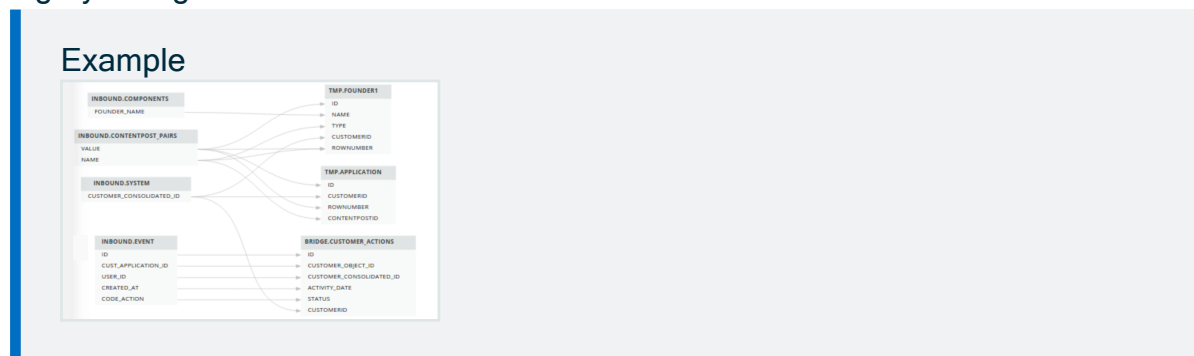
Data objects

You can see two types of data objects in your technical lineage:

- Data objects from your data source that are [stitched](#) to assets in Data Catalog and for which you created the technical lineage. These assets have a yellow background.



- Other objects, for example temporary tables and columns, that are collected from your data sources, but are not stitched to assets in Data Catalog. These objects have a gray background.



Note Collibra Data Lineage:

- Does not support stitching for Looker assets.
- Supports stitching for MicroStrategy assets only if you use the new integration method, which supports the latest MicroStrategy APIs.

Exporting technical lineage information

You can export technical lineage information and transformation details to formats such as PDF and PNG. For complete information, go to [Export the technical lineage information](#) and [Export technical lineage transformation details](#).

Naming convention

When you create a technical lineage, Data Catalog follows a strict naming convention for the [full names](#) of assets. Each asset has a display name and full name. You can freely edit the display name. However, do not edit the full name, because Data Catalog needs it to refresh data sources for which you created the technical lineage and to refresh the technical lineage itself.

When you prepare the Data Catalog physical data layer and the configuration file, you should always use the full name as the name of the corresponding data object in your data source for the following assets:

- System
- Database
- Schema

Warning Editing the full name of the System, Database and Schema assets may lead to errors during the technical lineage creation process.

Tip To stitching assets outside of the traditional *system > database > schema > table > column* hierarchy, you can use [Custom technical lineage](#) with the batch-definition option.

Automatic stitching for technical lineage

Stitching is a process that creates relations between [data objects](#) from a data source and their corresponding assets in Collibra. When the data sources are scanned, Collibra Data Lineage automatically creates new relations of the type "Data Element targets / sources Data Element":

- Between data objects in your data source and their corresponding assets in Data Catalog, including the asset that you create when [preparing the Data Catalog physical data layer](#).
- If you are integrating a BI tool, between ingested assets from BI sources and Data Catalog assets from registered data sources.

The importance of preparing the Data Catalog physical data layer

To stitch data objects in your data sources to their corresponding assets in Collibra Data Intelligence Cloud, the full names of the data objects and assets must match exactly. The full names are constructed according to the full path of the data objects in your data source:

(system name) > database name > schema name > table name > column name

However, when you register a data source via Jobserver or by running the lineage harvester, only assets of the following asset types are created in Data Catalog:

- Schema
- Table
- Column

Therefore, you have to create a Database asset and create a relation between it and the relevant Schema asset, to construct the full path hierarchy required for full name matching. If you set the `useCollibraSystemName` property to `true` in your [lineage harvester configuration file](#), you also need to create a System asset and create a relation between it and the Database asset. We refer to this as preparing the Data Catalog physical data layer.

Important This does not apply if you register a data source via Edge because in that case, Collibra automatically creates the system > database > schema > table > column hierarchy.

Note If you don't [prepare](#) the Data Catalog physical data layer, Data Catalog creates a technical lineage without stitching. As a result, when you click the Technical lineage tab on any Column, Table, Tableau Data Attribute, Power BI Column or SSRS Column asset page, you get the message **The current asset doesn't have a technical lineage yet**. However, you can use the [Browse tab pane](#) to view the technical lineage of data objects in data sources for which you created the technical lineage.

Tip For a more in-depth look at BI tool stitching, specifically the relationship between technical lineage and stitching, what causes stitching to break, and how to achieve stitching: go to the following topics:

- [Stitching for BI tool integrations](#)
- [Missing stitching and possible solutions](#)

Stitching issues

To stitch assets in Data Catalog to data objects collected by the lineage harvester, the Collibra Data Lineage service looks at the full path of the assets in Data Catalog and the full path of data objects in your data source. Stitching is based on the full path of objects with the following structure: (system) > database > schema > table > column. If the full paths match, the Collibra Data Lineage automatically stitches the data objects to the existing assets in Data Catalog. To indicate this, the assets have a yellow background in the technical lineage graph. Note that in Collibra, full paths are case-sensitive.

If the full path of an asset in Data Catalog does not match (including for case-sensitivity) the full path of a data object in your data source, Collibra Data Lineage cannot stitch them. To indicate this, the data objects have a gray background in your technical lineage graph. To fix stitching issues, you must check the full path of the assets in Data Catalog and make sure they match the full path of the data objects that are shown in the technical lineage graph. If you change the full path, make sure to run the lineage harvester again. Note that in Collibra, full paths are case-sensitive.

Note Collibra Data Lineage:

- Does not support stitching for Looker assets.
- Supports stitching for MicroStrategy assets only if you use the new integration method, which supports the latest MicroStrategy APIs.

You can use the [Stitching tab page](#) to easily find the full path of assets in Data Catalog and data objects that were collected by the lineage harvester. The Stitching tab page also shows an overview of all assets and data objects that are stitched successfully.

Table assets and View assets without columns

If you ingest a table or a view that does not have any columns, the Table or View asset that is created in Data Catalog will not have any relations to Column assets. In this case:

- Stitching cannot be achieved, because stitching is based on columns, not tables or views.
- On the [Stitching tab page](#), the **Found in** column will reflect that the table or view:
 - Is found in the Technical lineage.
 - Is not found in Catalog.

BI tool business logic

BI tool business stewards work with BI reports to make business decisions. Collibra Data Lineage offers BI tool business stewards several advantages:

- Easily find certified BI tool content.
- Shop for reports.
- Find where content is stored in your BI tool.
- Trace BI tool data to its sources.
- Get information about a BI report in a single location.

Note Due to limitations of the Looker REST API, Data Catalog cannot stitch Looker assets and corresponding assets in Data Catalog. The Looker REST API does not provide transformations in Looker that are needed for stitching.

BI asset pages

Depending on the asset type, the asset page shows different information ingested from your BI tool. For complete information, go to [BI tool operating models](#).

You can find a specific asset pages by searching in [Data Marketplace](#) or by looking in the Data Catalog BI domain in which you ingested the metadata.

Details

An asset page contains attributes and relations to other assets. This information is synchronized from your BI tool. You can, however, add additional characteristics, tags or comments directly via the asset page.

If you want to use a report, you can add it to the [Data Basket](#) and check it out.

Business Analysts Community

Looker Catalog

LKLK

Average_age

Looker Look Implemented

Add to Data Basket

Details

Tags

Comments

Diagram

Pictures

Technical Lineage

Responsibilities

References

History

Files

URL

Visits count

Favorites count

Document creation date

Document modification date

Document last accessed date

used in Report

uses Report

is grouped into Business Dimension

groups Report

https://collibra.looker.com/looks/12

2

0

7/17/2019

7/17/2019

7/20/2020

Name	Domain	Definition	Description
Web Analytics and Product Us...	Looker Catalog		

Name	Domain	Definition	Description
Average_age	Looker Catalog		

Name	Domain	Description
Shared	Looker Catalog	

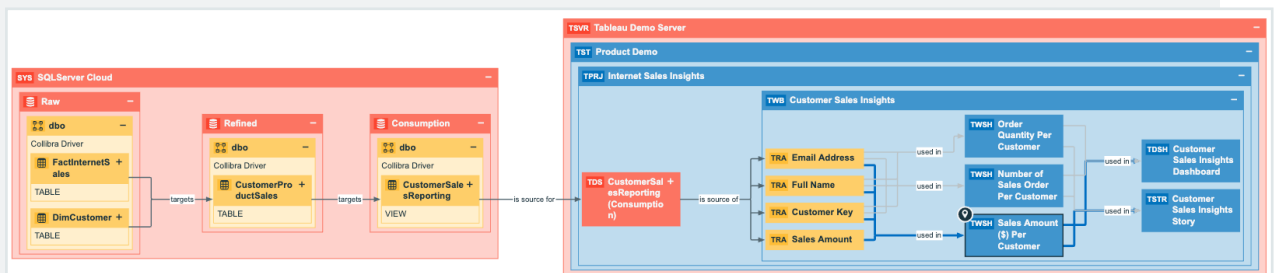
Name	Domain	Definition	Description
Query 456 part 1	Looker Catalog		
Query 456 part 2	Looker Catalog		

Business diagrams

Diagrams is a feature that allows you to interact with assets based on their relations in an easy-to-read diagram. Diagrams help you to quickly understand how assets are related. As such, the diagram can show a high-level presentation of a data set or report. If the BI assets are **stitched** to registered assets in Data Catalog, you can also **see the stitching results** in the diagram.

Tip For each **supported BI tool**, we include the JSON code and instruction on how to create a diagram view of the BI tool operating model in your Collibra environment. For complete information, go to **BI tool operating models**, select your BI tool, and then scroll down to the section "Create an operating model diagram view".

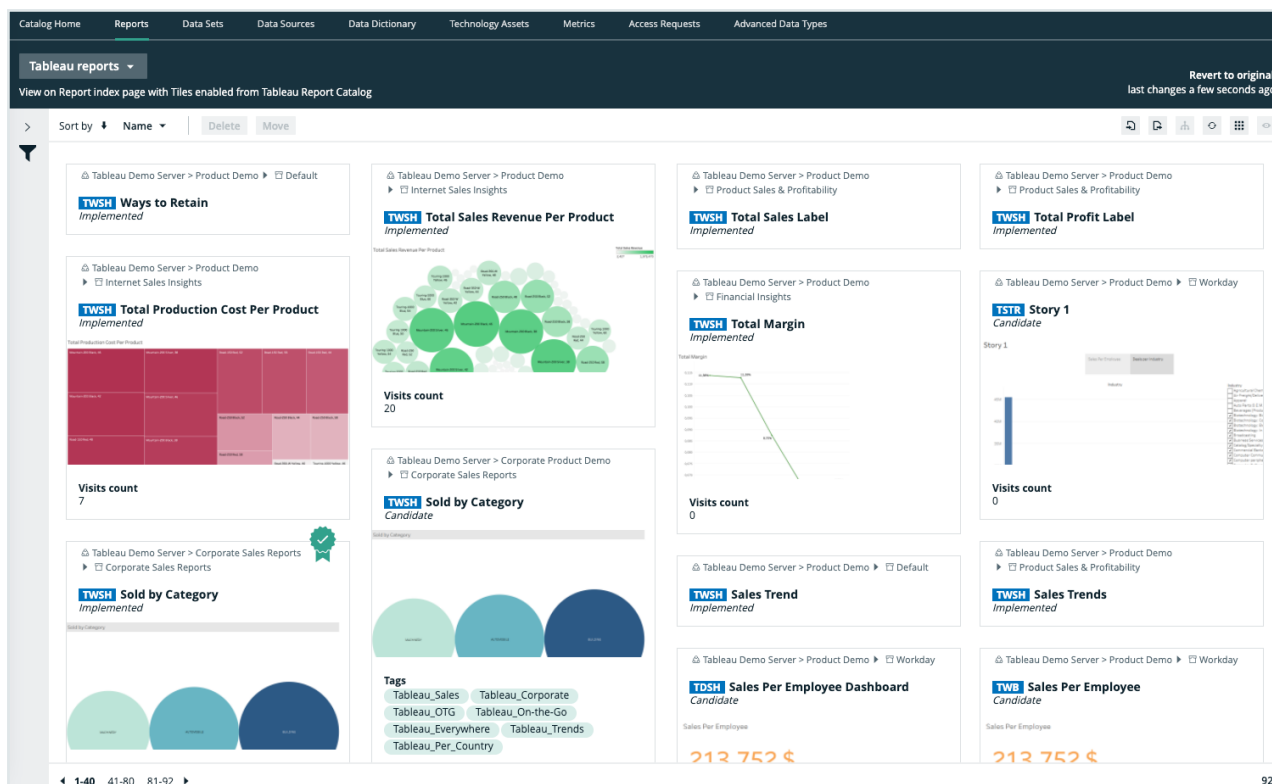
Example The following diagram shows the *Customer Sales Insights* Tableau Workbook, which is stored in the *Internet Sales Insights* Tableau Project. The Tableau Workbook contains Tableau Report Attributes that have the *CustomerSalesReporting* Tableau Data Source as source. This Tableau Data Source is stitched to the *CustomerSalesReporting* Table asset in the *SQL Server Cloud* data source.



Report views

Collibra Data Lineage enables you to find all ingested BI asset types in a single location.

In the **Reports** tab page in Data Catalog you can see an overview of all BI Report assets and their children. Optionally, you can **create a view** with a **filter** to only show, for example, Tableau assets. This is useful if you quickly want to see all reports or if you want find specific reports, for example certified reports or the most frequented reports.



Technical lineage and stitching for BI tool integrations

BI tools, such as Power BI and Tableau, allow you to build reports that help you visualize and understand your data. To trust the data in your report, it's essential to know where the data came from. Collibra Data Lineage allows you to create a technical lineage, to trace the data from your data sources to your reports. Stitching then creates relations between the data objects in your technical lineage and the corresponding assets in Data Catalog, to give you a complete picture of your data landscape and all critical metadata.

In this topic, we examine the relationship between technical lineage and stitching.

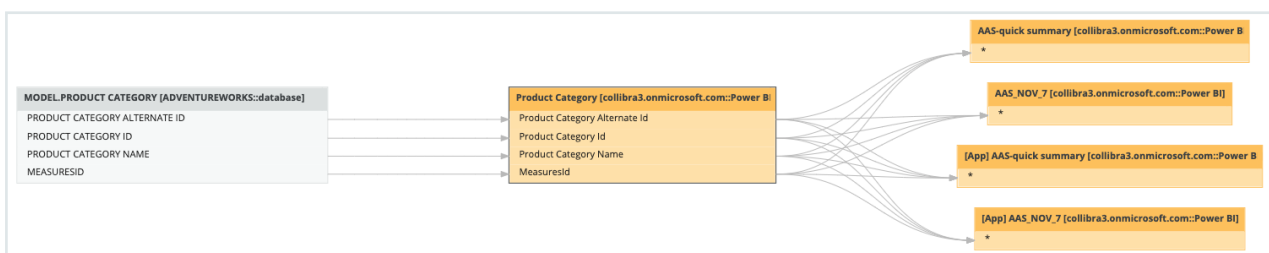
Note This topic applies to MicroStrategy, Power BI, SSRS-PBRS and Tableau. Collibra Data Lineage currently does not offer stitching for Looker assets.

For a more technical perspective, see [Technical overview of BI tool lineage](#) and [Missing stitching and possible solutions](#).

Stitching: The bridge between ingestion and technical lineage

Keep in mind that metadata ingestion (which results in the creation of assets in Collibra) and technical lineage are separate and independent concepts. The single, seamless process of integrating a BI tool for the purpose of technical lineage could lead one to think otherwise.

A technical lineage illustrates the flow of data in your external data sources. It does not inherently tell you anything about your assets in Collibra. The bridge between the metadata you ingest as assets in Data Catalog and the technical lineage, is stitching. As it concerns a technical lineage graph, stitching or the lack of stitching is reflected only in the color of the nodes in the technical lineage.



- A yellow node indicates stitching. Specifically:
 - There is an asset in Collibra with a full name that exactly matches the data object in the technical lineage.
 - A relation of the type "Data Element targets / sources Data Element" is created between the asset and the data object, and shown on the asset page.
 - In the **Stitching** tab, the **Found In** column indicates that the database table was found in both Data Catalog and the technical lineage.

Sources		Stitching
Search		testing1
Full asset path		Found in
111.93.0.181 > DBADMIN > TESTING1		Technical Lineage
DEV > TESTING1 > TICKETS_MV1		Technical Lineage
DEV > TESTING1 > TICKETS_MV2		Technical Lineage
ORACLE-DB > DBADMIN > TESTING1		Catalog
looker > Cars > Customers > EMEA > HND_ACRA_7		Catalog & Technical Lineage
tableauOn > collibratabpartner_site > HS_Techlin_Capability_tests > Redshift_WB_TecLin_Capa_test > tickets_mv1 (testing1.tickets_mv1)* (testing1) [eTDM]		Catalog & Technical Lineage

- A gray node indicates a lack of stitching.
 - There is no asset in Data Catalog with a full name that exactly matches the name of the data object.
 - In the **Stitching** tab, the **Found In** column indicates that the database table was found only in the technical lineage.

Sources Stitching	
Search testing1	
Full asset path	Found in
111.93.0.181 > DBADMIN > TESTING1	Technical Lineage
DEV > TESTING1 > TICKETS_MV1	Technical Lineage
DEV > TESTING1 > TICKETS_MV2	Technical Lineage
ORACLE-DB > DBADMIN > TESTING1	Catalog
looker > Users > Tapana sudini > Looks > ts_testing1	Catalog & Technical Lineage
tableauOn > collibratabpartner_site > HS_Techlin_Capability_tests > Redshift_WB_TecLin_Capa_test > tickets_mv1 (testing1.tickets_mv1)+ (testing1) [eTDM]	Catalog & Technical Lineage

If the database node is missing from the technical lineage graph, we refer to this as "missing stitching". This can happen if, for example, your BI tool has limited support for custom SQL, or if your integration includes a data source that is not yet supported by Collibra Data Lineage. In these situations, the relations required to recognize the database are not exposed.

Tip If you can't view a technical lineage because you lack the permissions, you can still identify stitching by viewing a [diagram](#). A relation of the type "Data Element targets / sources Data Element" between, for example, a Tableau Data Attribute asset and Column asset in a diagram, indicates stitching.

Full path, full name matching

When you integrate your BI tool, the full names of the assets that are created in Data Catalog reflect the full paths (also considered the full names) of the corresponding data objects in the external data source. The full paths to data objects follow this hierarchy:

(system name) > database name > schema name > table name > column name

The system name is only relevant if you specify one as part of your pre-integration preparation. For complete information, go to [Prepare the Data Catalog physical data layer](#).

To stitch assets in Data Catalog to data objects in the technical lineage, Collibra Data Lineage looks at the full names of assets in Data Catalog and the full names of data

objects in your data source, which figure in the technical lineage. If there is an exact match in the full names, stitching is achieved.

Note The full path represents the full name of an asset, not the display name. As such, you can change the display name of an asset without breaking stitching, but if you change the full name of an asset, and it no longer exactly matches the full name of the corresponding data object, stitching will break.

If an ingestion job was successful, and it's true that the full names of the assets in Data Catalog are taken directly from the full names of the corresponding data objects, then how is it possible that the full names don't match? The possibilities are addressed in the following section.

What causes stitching to break?

The following scenarios result in a lack of stitching:

Scenario	Why stitching breaks
During integration of your BI tool, the API returns a technical name, IP address, or hostname of the database, instead of the true name of the database.	The database name returned by the API doesn't match the name of the Database asset you created when you prepared the Data Catalog physical data layer.

Scenario	Why stitching breaks
You have registered a schema-less data source, for example HiveQL, MySQL or Teradata.	The full names of assets don't match because the full path hierarchy is altered because of the lack of a schema name. See an example Let's say you ingest a HiveQL data source via Edge. Note that Edge gives the name "CDATA" for the database. The full path to a column is something like: <pre>Hive_123 (system) > CDATA (database) > Hive_ABC (schema) > Table > Column</pre> Because HiveQL is database-less, the value that you give for the <code>database</code> property in your configuration file is used as the schema name in the technical lineage, and the value you give for <code>collibraSystemName</code> is used as the database name. But if <code>useCollibraSystemName</code> is set to <code>true</code>, then the value of <code>collibraSystemName</code> is also used as the system name. In that case, in the full path to the column, the system name and the database name are the same: <pre>Hive_123 (system) > Hive_123 (database) > Hive_ABC (schema) > Table > Column</pre> Notice the mismatch between the database names. The <code>externalDbName</code> property tells the lineage harvester to use the value that you specify here for the database name in the technical lineage, specifically "CDATA". This ensures that the full paths match and stitching is preserved.
You haven't prepared the Data Catalog physical data layer, or did so incompletely or erroneously.	• The database name returned by the API doesn't match the name of the Database asset you created when you prepared the Data Catalog physical data layer. • The name of the System asset you created doesn't match the name of the system of the data source that you register, as specified in the configuration file. • You forgot to create the required relation between the Database asset and the Schema asset that was created when you registered your data source.

Scenario	Why stitching breaks
A database query includes a function or query that Collibra Data Lineage does not support.	The relations required to recognize the database are not exposed, resulting in "missing stitching".
You experience a rare exception, for example, SAP label names v. technical names	When connecting to an SAP HANA data source, some BI tools use the label name instead of the technical name. This can result in a mismatch between the name of the data source in the technical lineage and the Database asset in Collibra.

Creating the technical lineage

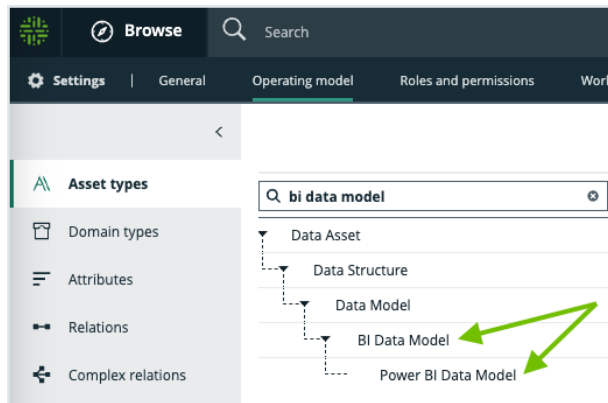
Let's start with a lifehack: create the technical lineage without giving any thought to stitching. Specifically, prepare your source ID configuration file as you want, for filtering or to specify a system name, but don't worry about database mapping. Run the lineage harvester and analyze the technical lineage, to see what the APIs return for the database names. You can then set up database mapping in your source ID file and run the harvester again.

What you've done so far

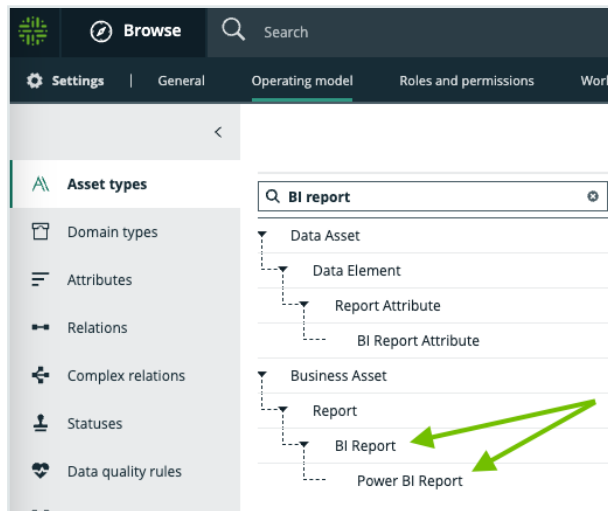
- You've pulled in data from a data source to your BI tool, and with that data set, you've created a report.
- You've either:
 - Prepared a lineage harvester configuration file and run the lineage harvester.
 - Added the relevant Edge capability and run the Edge job.

You now have:

- A technical lineage that shows the flow of data from the data source to your BI tool.
- Assets in Data Catalog that represent the data objects in your data source. Among these assets are:
 - Assets that represent the data set you used to create the report in your BI tool. These are assets of child asset types of the BI Data Model asset type, for example Power BI Data Model and Tableau Data Model assets.



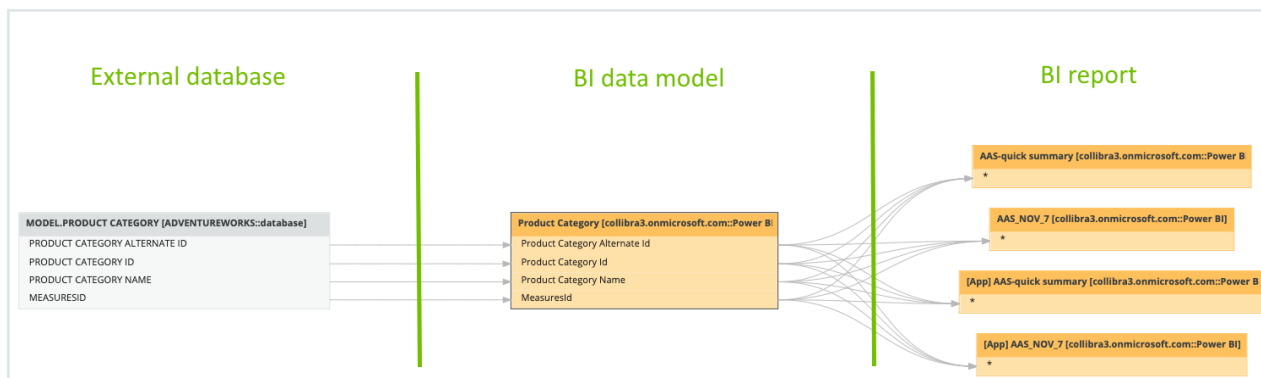
- Assets that represent the report in your BI tool. These are assets of child asset types of the BI Report asset type, for example Power BI Report and Tableau Report assets.



Analyze the technical lineage

Go to the asset page of your BI Data Model asset and click the **Technical Lineage** tab. As shown in the following image, there will most likely be three nodes or groupings of nodes:

- The external database.
- The BI data model.
- The BI report.



The first thing we notice is that the database node has a gray background and the other two have a yellow background. The yellow nodes represent BI assets and data objects. As such, we say that this part of the technical lineage graph depicts BI lineage.

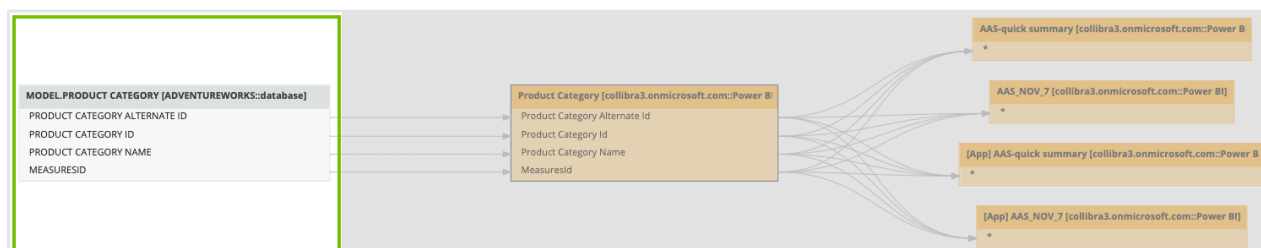
Ultimately, what we want is for all three nodes to have the yellow background. Technically speaking, that means:

- Lineage is confirmed upstream of the BI lineage.
- The data sources that feed into the database node are shown.

Let's examine more closely these three nodes.

Note If you are integrating MicroStrategy, there will be four groups of nodes. In reference to the previous image, the BI data model node will consist of two groups of nodes, one representing MicroStrategy Attributes and one representing MicroStrategy Facts. There could also be four groups if you are integrating SQL Server Reporting Services (SSRS) or Power BI Report Server (PBRs) and have a shared data set.

The external database



This node represents the table from the database you used to create the report in your BI tool. It is returned by the API and is shown in the technical lineage, as long as:

- You have the required roles and permissions in your BI tool, to access the data in your data sources. For example, in Tableau, you need certain roles and permissions to access external data objects.
- There are no unsupported custom SQL transformations or functions.
- No errors have caused the integration to fail.

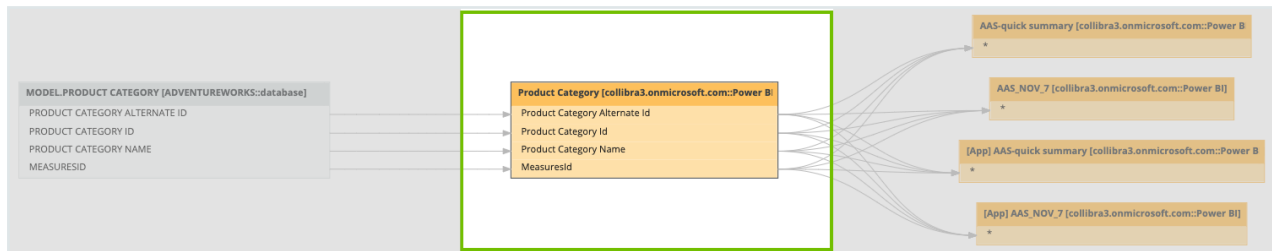
By the fact of its presence in the technical lineage, we know that the lineage harvester collected the source code from the BI tool and identified the flow of data from this data source to the BI data model. This node is a prerequisite for stitching. If it is not shown in the technical lineage, stitching is not possible.

Note The node might be yellow if you previously ingested metadata in Data Catalog that matches the database tables used in your dataset.

The gray background indicates that there might be a Table asset in Collibra that corresponds with this database table, but their full names do not exactly match.

Tip Look closely at the names of these nodes, to correctly identify if the nodes represent data objects from the data source or from your BI tool. In this example, you can tell by the names that the two yellow nodes are the BI data set and BI report nodes. When you view a technical lineage, it could be that the database and BI data set are stitched, and the BI report node does not appear in the technical lineage. This could be the case if you're viewing the lineage at the column level, and the attribute that the column represents is not used in the report. At first glance, one might incorrectly think that the database node, which is essential for stitching, is not shown.

The BI data model

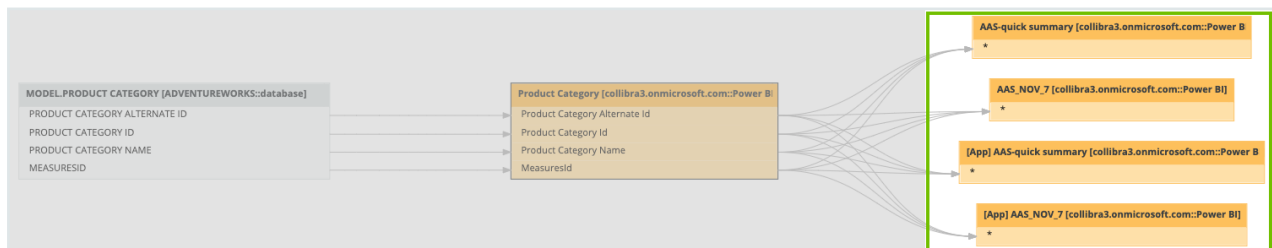


This node represents the data set that you used to create the report in your BI tool. It will always be shown in the technical lineage, because it is the target of the database table and the source of the BI report.

The yellow background indicates that the name of the BI Data Model asset in Data Catalog matches exactly the name of this data set in the technical lineage.

Note This node is always stitched because Collibra Data Lineage knows the full name of the data set in your BI tool, and it creates the corresponding BI Data Model asset with the exact same name. This is referred to as BI stitching.

The BI report



This node (or grouping of nodes) represents the report you created in your BI tool. It is always part of the technical lineage.

Like the BI data model node, this node is always stitched because Collibra Data Lineage knows the full name of the report in your BI tool, and it creates the corresponding BI Report asset with the exact same name.

Tip While the BI report node is always part of the technical lineage, it might not initially be visible when you view the technical lineage. If, for example, you're viewing the lineage at the column level, and the attribute that the column represents is not used in the report, there will be no arrow leading to the report node in the technical lineage. In this case, right-click on the data model node and click **Table lineage** to pull back and view the table-level lineage. The BI report node will appear and you will see which columns/data attributes are used in the report.

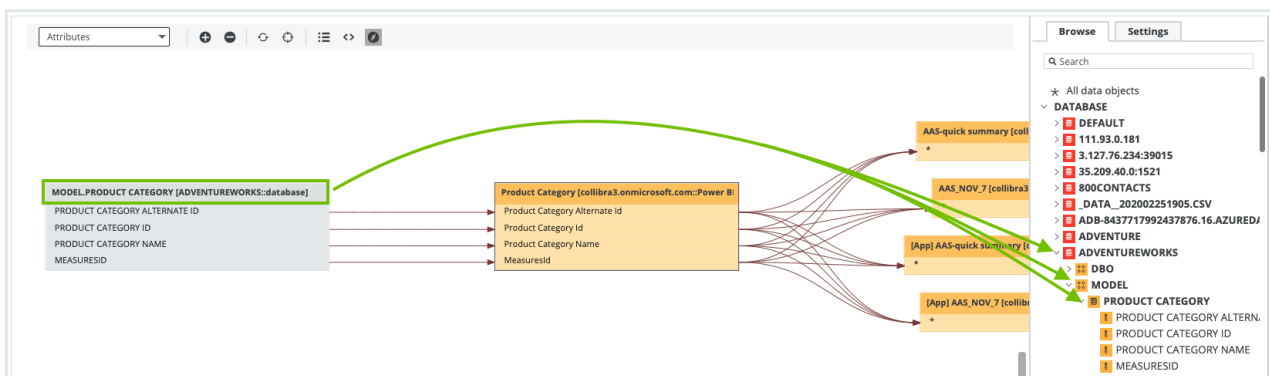
How to achieve stitching

Let's have a look at a typical database mismatching scenario.

First, let's look at the name of an unstitched database table in the technical lineage graph:

```
MODEL.PRODUCT CATEGORY [ADVENTUREWORKS::database]
```

We can identify the following:



- The database name: ADVENTUREWORKS
- The schema name: MODEL
- The table name: PRODUCT CATEGORY

Now let's find the table in the **Stitching** tab:

1. Click the **Settings** tab.
2. Click **Show status**.
3. Click the **Stitching** tab.
 - » The Stitching tab shows a list of all tables that exist in Data Catalog and on the Colibra Data Lineage service instance.

Use the Search field to find the unstitched database table `PRODUCT CATEGORY`.

Sources Stitching	
Search product category	
Full asset path	Found in
AAS-MODEL > MODEL > PRODUCT CATEGORY	Catalog
ADVENTUREWORKS > MODEL > PRODUCT CATEGORY	Technical Lineage
colibra3.onmicrosoft.com > powerbicolibraczech > BrnoWorkspace > Datasets > AAS_NOV_7 > Product Category	Catalog & Technical Lineage

In the **Found in** column, the value "Technical Lineage" confirms what we already know: the table was found only in the technical lineage. An exactly matching asset was not identified in Collibra.

Now try to find a likely match. Look for a table that has the same name and the value "Catalog" in the **Found in** column.

The table shown in the following image looks like a match. The schema and table names match exactly; only the database names differ.

Sources Stitching	
Search product category	
Full asset path	Found in
AAS-MODEL > MODEL > PRODUCT CATEGORY	Catalog
ADVENTUREWORKS > MODEL > PRODUCT CATEGORY	Technical Lineage
colibra3.onmicrosoft.com > powerbicolibraczech > BrnoWorkspace > Datasets > AAS_NOV_7 > Product Category	Catalog & Technical Lineage

To [achieve stitching](#), you need to create a source ID configuration file and configure database mapping.

For more missing stitching scenarios and suggestions for resolving the issue, go to [Missing stitching and possible solutions](#).

Business lineage

Business lineage is a [diagram](#) that includes, among others, relations of the type "Data Element sources / targets Data Element". It is not a separate diagram view, but refers to any diagram that contains that relation type. It allows you to trace data flows between registered databases and, as such, provides a summary of a technical lineage.

Tip See also [Differences between Technical lineage and business lineage](#).

You can [create](#) a new [diagram view](#) including business lineage or you can select one of the existing diagram views that shows the relation "Data Element sources / targets Data Element" between Column assets of registered data sources and between BI assets and assets of registered data sources.

Creating a business lineage

Business lineage is automatically created as part of the technical lineage process.

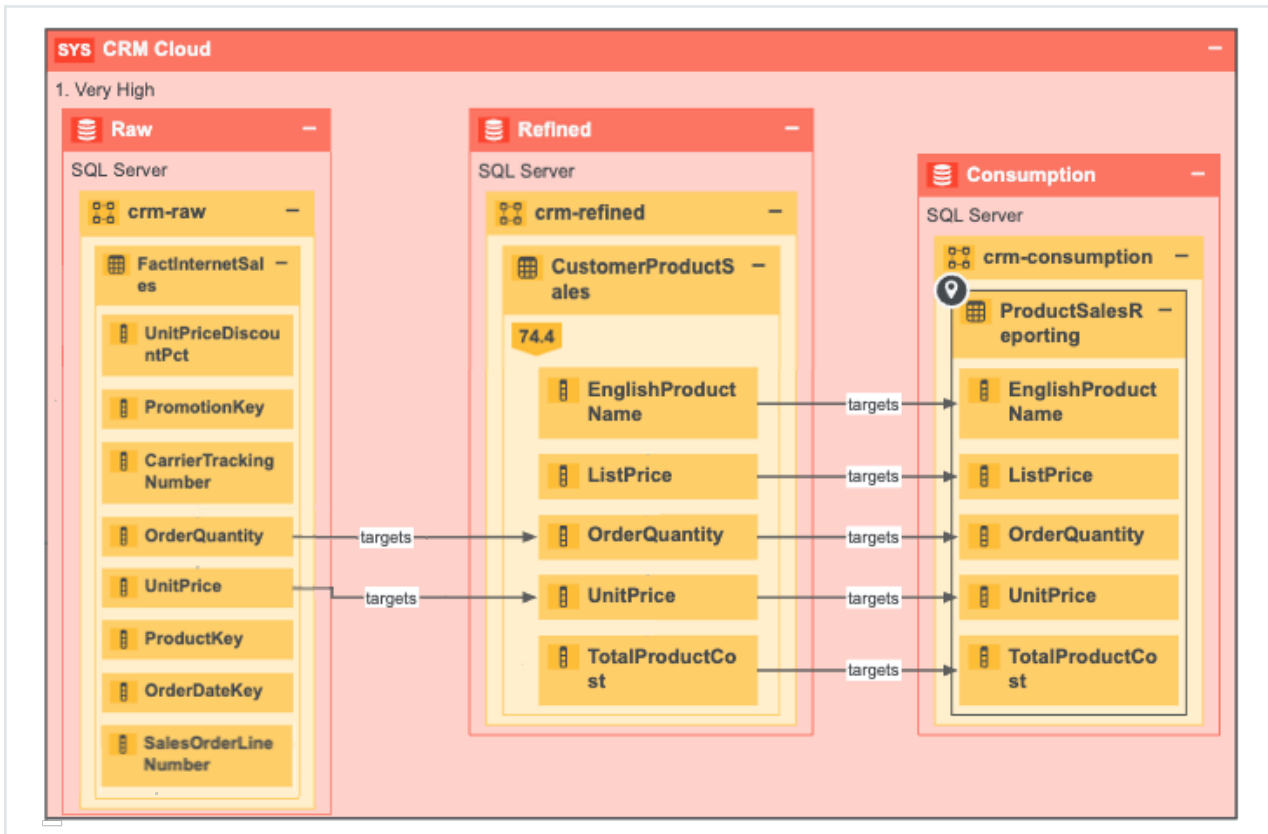
To create a business lineage, you:

1. [Register](#) the data sources for which you want to view a business lineage.
2. Do one of the following:
 - [Prepare](#) a configuration file that mentions the relevant data sources, and then [create](#) a technical lineage via the lineage harvester.
 - Create a [Technical lineage via Edge](#) for the relevant data sources.

Once the data sources are scanned, the Collibra Data Lineage service automatically pushes relations of the type "Data Element sources / targets Data Element" to Collibra Data Intelligence Cloud.

Example of a business lineage

In this business lineage, you see that the Column assets of the Table asset CustomerProductSales have a relation of the type "Data Element sources / targets Data Element" to Column assets of other Table assets.



Differences between Technical lineage and business lineage

Technical lineage is a detailed lineage graph that shows where data objects are used and how they are transformed. Business lineage shows the relations between Data Assets in Data Catalog after [stitching](#). Both map the flow of data, but a technical lineage provides a detailed overview of the data flow, while a business lineage only provides a summary of it.

[Business lineage](#) and technical lineage are both visual representations of nodes. However, there are some key differences between them.

Business lineage	Technical lineage
A business lineage helps Business Analysts and other business stewards to understand their data by providing a summary of the technical lineage.	A technical lineage helps Data Engineers, Data Architects and similar personas to easily navigate to data objects in the data flows and find relevant source code fragments by providing a detailed lineage graph.

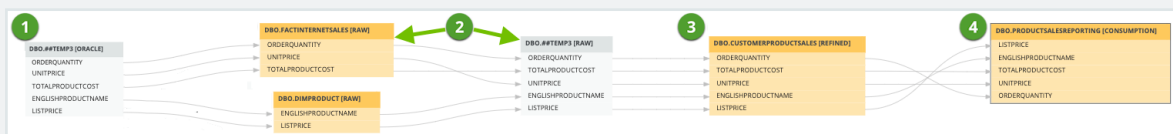
Business lineage	Technical lineage
A business lineage is accessible via the Diagram tab on all asset pages.	A technical lineage is accessible via the Technical Lineage tab pane of all Table assets, all Column assets, and some BI assets.
A diagram shows assets and relations as defined in its diagram view. In the case of a business lineage, the diagram shows, amongst others, relations of the type "Data Element targets / sources Data Element" between assets that exist in Data Catalog. Relations of this type are automatically created as part of the technical lineage process.	A technical lineage shows relations of the type "Data Element targets / sources Data Element" between all data objects in the data source. Relations of this type are automatically created as part of the technical lineage process. Note The data objects that you see in the technical lineage are: • Data Element assets for which you created the technical lineage, • Other objects, for example temporary tables and columns, that the lineage scanner collected from your data sources, but are not assets in Data Catalog.
A business lineage shows how registered data sources relate to each other.	Technical lineage shows how all data sources for which you create a technical lineage relate to each other. If the data source, or a part of the data source, is not registered in Data Catalog, the dependencies between the data elements in the data sources are still shown.

Example

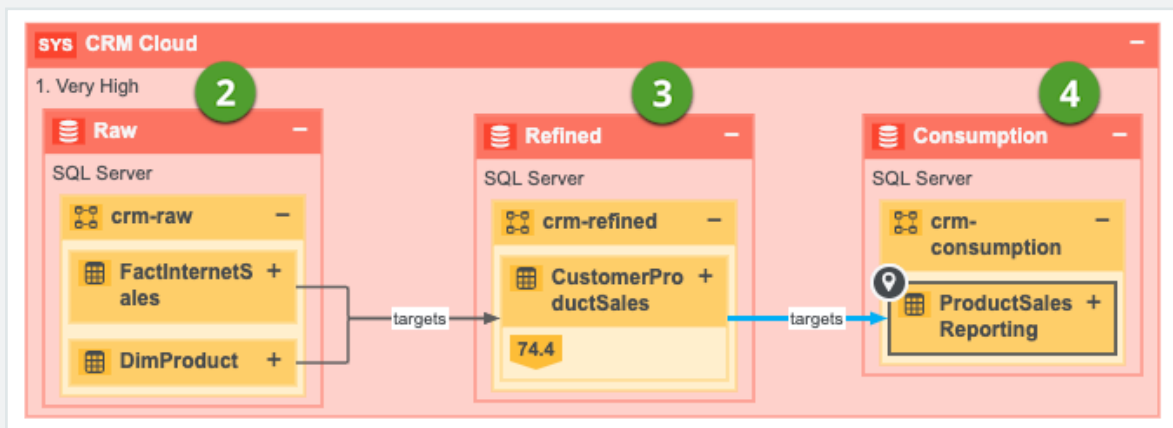
You have created a technical lineage for four different databases:

- The first database, *Oracle*, is not ingested in Data Catalog and therefore has no assets in Data Catalog.
- The second database, *Raw*, contains tables that are ingested in Data Catalog, but also tables that are not ingested and therefore are not assets.
- The third and fourth database, *Refined* and *Consumption*, only contains data objects that are also assets in Data Catalog.

The Technical lineage shows the data flow from all data objects in the first database, to the second, the third, and the fourth. Databases or data objects that are not ingested in Data Catalog and therefore are not assets, have a gray background.



The business lineage only shows the relations between data objects that are also assets in Data Catalog, which means the data flow from assets in the second database to assets in the third, to assets in the fourth. The first database, which wasn't ingested, will not be shown on the diagram.



Dependencies

A dependency is a data object that is targeted by another data object. This is represented by a relation of the type "Data Element targets / sources Data Element", where the dependency is the tail.

There are two type of dependencies:

- a direct dependency: a data object that is the tail of a relation of the type "Data Element targets / sources Data Element".

Example If column A targets column B, then column B is the direct dependency of column A.

- an indirect dependency: a data object that is the target of a direct or another indirect dependency.

Example Column A targets column B, which on its turn targets column C. This means that column A indirectly targets column C, so column C is the indirect dependency of column A.

BI integration concepts

This section addresses BI tool-specific integration concepts for business stewards.

BI tool terminology

The following tables show the supported BI tool terminology and corresponding asset types and terminology in Collibra Data Intelligence Cloud.

Note Keep in mind, it is possible that your organization has renamed the out-of-the-box asset types.

Steps

Tableau term	Description	Collibra equivalent
Site	A site is a stand-alone collection of content, such as projects, workbooks and users. Each site has its own URL and its own set of users.	Subcommunity and Tableau Site asset

Tableau term	Description	Collibra equivalent
Project	A project organizes related content resources. Content resources are workbooks, views and data sources.	Tableau Project asset
Workbook	A workbook is a collection of views.	Tableau Workbook asset
Dashboard	A dashboard is a collection of views from multiple worksheets.	Tableau Dashboard asset
Worksheet	A worksheet contains a single view, along with shelves, legends, and the Data pane.	Tableau Worksheet asset
Tableau data source	Tableau Data Sources consist of metadata that describe the connection information, information about how to access or refresh the data and customizations.	Tableau Data Model asset
Dimension	Dimensions contain qualitative values (such as names, dates, or geographical data).	Attribute type Role in Report on a Tableau Data Attribute asset page
Measure	Measures contain numeric, quantitative values that you can measure.	Attribute type Role in Report on a Tableau Data Attribute asset page
Tableau data attribute	Tableau Data Attributes define a property of a Tableau data entity.	Tableau Data Attribute asset
Tableau data entity	Tableau Data Entities are an abstraction of the physical implementation of database tables, used for Tableau report creation.	Tableau Data Model asset
Tableau data model	Tableau Data Models are an abstraction for the physical implementation of databases, schemas, files, etc., used for Tableau report creation.	Tableau Data Model asset
Tableau server	A Tableau server is a server on which Tableau users can publish data sources, as a means to share the data connections they've defined.	Tableau Server asset

Published versus embedded data sources

You can create data sources in Tableau when you connect to data. After you set up the data sources in Tableau, you can publish data sources as standalone resources, or you can publish workbooks with the data sources embedded in.

Unless you take actions to publish the data source separately, the data source is published as embedded in a workbook by default. For more information, see the Tableau documentation on [Publishing data separately or embedded in workbooks](#).

Collibra Data Lineage ingests metadata of data sources as assets of the Tableau Data Model asset type, regardless of the way the data sources are published.

eTDM and pTDM

When you ingest a Tableau data source in Collibra, each asset is identified as eTDM or pTDM with [eTDM] or [pTDM] added to the asset name.

eTDM stands for embedded Tableau Data Model, which indicates that the asset represents the data source that is embedded in a workbook in Tableau. pTDM stands for published Tableau Data Model, which indicates that the asset represents the data source that is published separately in Tableau.

For a data source that is both published separately and embedded in a workbook, Collibra Data Lineage ingests the metadata in one of the following ways:

- If the metadata of the embedded data source matches that of the published data source, Collibra Data Lineage ingests the metadata only from the published data source to avoid duplication.
- If the metadata of the embedded data source contains more fields than that of the published data source, Collibra Data Lineage ingests metadata from both the published and embedded data sources.

As a result, a Tableau workbook can have one of the following relations:

- To the published and embedded data source.
- To the published data source only.

Power BI term	Description	Asset type in Collibra
Capacity	A resource that hosts Power BI Workspaces.	Power BI Capacity
Dashboard	A collection of Power BI tiles with metrics from one or more Reports and Data Models.	Power BI Dashboard
Dataflow	A collection of tables that are created and managed in workspaces in the Power BI service.	Power BI Data Flow
Datamart	A self-service analytics solutions, enabling users to store and explore data that is loaded in a fully managed database.	Power BI Data Mart
Data Set	A collection of data that is used to create a Power BI report.	Power BI Data Model
Data Set Column	A column in a Power BI Data Model.	Power BI Column
Data Set Table	A table in a Power BI Data Model.	Power BI Table
Report	A detailed view of a Power BI Data Model, with visualizations of findings and insights.	Power BI Report
Server or Tenant	A visual analytics platform for creating and storing Power BI Reports and Data Models.	Power BI Server
Tile	An element representing data on the Power BI Dashboard.	Power BI Tile
Workspace	A collection of Power BI Dashboards, Reports and Data Models.	Power BI Workspace
MicroStrategy term	Description	Asset type in Collibra
Attribute / Fact	A detailed view of a MicroStrategy visualization, with findings and insights.	MicroStrategy Data Entity
Attribute Form / Fact expression	Additional descriptive information about an attribute.	MicroStrategy Data Attribute

MicroStrategy term	Description	Asset type in Collibra
Column	A column in a MicroStrategy data model.	MicroStrategy Data Attribute
Dataset	A collection of data that is used to create MicroStrategy reports.	MicroStrategy Data Model
Document	A collection of grid and graph reports that can be viewed at the same time, along with images and text.	MicroStrategy Document
Dossier	A collection of MicroStrategy chapters and pages.	MicroStrategy Dossier
Folder	A collection of MicroStrategy reports and data models.	MicroStrategy Folder
Intelligent Cube	A set of data that can be shared as a single in-memory copy among many different reports created by multiple users.	MicroStrategy Data Model
Project	A collection of MicroStrategy visualizations, report attributes and tables.	MicroStrategy Project
Report	A detailed view of a MicroStrategy data model, with visualizations of findings and insights.	MicroStrategy Report
Server	A visual analytics platform for creating and storing MicroStrategy reports and data models.	MicroStrategy Server
Visualization	A visual representation of the data in a dossier, such as a grid, line chart, or heat map.	MicroStrategy Visualisation

Looker term	Description	Asset type in Collibra
Dashboard	A collection of Looker tiles with metrics from one or more Looker Looks.	Looker Dashboard
Explore	A collection of data that is used to define Looker Dimensions and Measures.	Looker Data Set
Dimensions, Measures	An atomic unit of data that is used in a Looker Look or Looker Tile. It represents a column in a Looker Data Set.	Looker Data Set Column

Looker term	Description	Asset type in Collibra
Folder or Space	A container that stores Looker Looks, Dashboards and other folders.	Looker Folder
Look	A detailed view of a Looker Data Set, with visualizations of findings and insights.	Looker Look
Dimensions, Measures	An atomic unit of data that is used in a Looker Look or Looker Tile. It represents the actual use a Looker Data Set Column.	Looker Report Attribute
Query	A query that creates a simple report in a Looker Tile or Looker Look.	Looker Query
Looker instance	A platform to create Looker Dashboards and rich visualizations.	Looker Tenant
Tile or Dashboard element	An element that represents data on the Looker Dashboard.	Looker Tile

SSRS-PBRS term	Description	Asset type in Collibra
Column	A column in an SQL Server Reporting Services Report Data Set.	SSRS Column
Data Set	A collection of data that is used to create an SQL Server Reporting Services Report.	SSRS Data Model
Folder	A collection of SQL Server Reporting Services and Power BI Report Server Reports and Data Sets.	SSRS Folder
KPI	A key performance indicator of SQL Server Reporting Services.	SSRS KPI

SSRS-PBRS term	Description	Asset type in Collibra
Mobile report	A detailed view of an SQL Server Reporting Services Data Set, with visualizations of findings and insights.	SSRS Report
Paginated report	A detailed view of an SQL Server Reporting Services Data Set, with visualizations of findings and insights.	SSRS Report
Parameter	A column that is part of an SQL Server Reporting Services Data Set and that is used in a KPI.	SSRS Parameter
Power BI Report Server report	A detailed view of a Power BI Data Model, with visualizations of findings and insights.	Power BI Report
SQL Server Reporting Services or Power BI Report Server server or tenant	A visual analytics platform for creating and storing SQL Server Reporting Services and Power BI Report Server Reports and Data Sets.	SSRS Server
Table	A table in an SQL Server Reporting Services Report Data Set.	SSRS Table

BI asset types and domain types

BI tool integration uses a specific subset of out-of-the-box [asset types](#) and [domain types](#).

The following table shows the asset and domain types that are used for the BI tool integrations. Above each asset type you can see the parent asset types in the breadcrumbs.

Note Keep in mind, it is possible that your organization has renamed the out-of-the-box asset types.

Asset type	Description	Domain type
Business Asset › Business Dimension › BI Folder › Looker Folder	A container that stores Looker Looks, Dashboards and other folders.	BI Catalog
Business Asset › Report › BI Report › Looker Dashboard	A collection of Looker tiles with metrics from one or more Looker Looks.	BI Catalog
Business Asset › Report › BI Report › Looker Look	A detailed view of a Looker Data Set, with visualizations of findings and insights.	BI Catalog
Business Asset › Report › BI Report › Looker Query	A query that creates a simple report in a Looker Tile or Looker Look.	BI Catalog
Business Asset › Report › BI Report › Looker Tile	An element that represents data on the Looker Dashboard.	BI Catalog
Data Asset › Data Element › Data Attribute › BI Data Attribute › Looker Data Set Column	An atomic unit of data that is used in a Looker Look or Looker Tile. It represents a column in a Looker Data Set.	BI Catalog
Data Asset › Data Element › Report Attribute › BI Report Attribute › Looker Report Attribute	An atomic unit of data that is used in a Looker Look or Looker Tile. It represents the actual use a Looker Data Set Column.	BI Catalog

Asset type	Description	Domain type
Data Asset ▶ Data Set ▶ BI Data Set ▶ Looker Data Set	A collection of data that is used to define Looker Dimensions and Measures.	BI Catalog
Technology Asset ▶ Server ▶ BI Server ▶ Looker Tenant	A platform to create Looker Dashboards and rich visualizations.	BI Catalog

Asset type	Description	Domain type
Business Asset ▶ Business Dimension ▶ BI Folder ▶ MicroStrategy Folder	A collection of MicroStrategy reports and data models.	BI Catalog
Business Asset ▶ Business Dimension ▶ BI Folder ▶ MicroStrategy Project	A collection of MicroStrategy visualizations, report attributes and tables.	BI Catalog
Business Asset ▶ Report ▶ BI Report ▶ MicroStrategy Dossier	A collection of MicroStrategy chapters and pages.	BI Catalog
Business Asset ▶ Report ▶ BI Report ▶ MicroStrategy Document	A collection of grid and graph reports that can be viewed at the same time, along with images and text.	BI Catalog
Business Asset ▶ Report ▶ BI Report ▶ MicroStrategy Report	A detailed view of a MicroStrategy data model, with visualizations of findings and insights.	BI Catalog

Asset type	Description	Domain type
Data Asset › Data Element › Data Attribute › BI Data Attribute › MicroStrategy Data Attribute	A column in a MicroStrategy data model.	BI Catalog
Data Asset › Data Element › Report Attribute › BI Report Attribute › MicroStrategy Visualization	A detailed view of a MicroStrategy visualization, with findings and insights.	BI Catalog
Data Asset › Data Structure › Data Entity › BI Data Entity › MicroStrategy Data Entity	A detailed view of a MicroStrategy visualization, with findings and insights.	BI Catalog
Data Asset › Data Structure › Data Model › BI Data Model › MicroStrategy Data Model	A collection of data that is used to create MicroStrategy reports.	BI Catalog
Technology Asset › Server › BI Server › MicroStrategy Server	A visual analytics platform for creating and storing MicroStrategy reports and data models.	BI Catalog

Asset type	Description	Domain type
Business Asset ▶ Business Dimension ▶ BI Folder ▶ Power BI Capacity	A resource that hosts Power BI Workspaces.	BI Catalog
Business Asset ▶ Business Dimension ▶ BI Folder ▶ Power BI Workspace	A collection of Power BI Dashboards, Reports and Data Models.	BI Catalog
Business Asset ▶ Report ▶ BI Report ▶ Power BI Dashboard	A collection of Power BI tiles with metrics from one or more Reports and Data Models.	BI Catalog
Business Asset ▶ Report ▶ BI Report ▶ Power BI Report	A detailed view of a Power BI Data Model, with visualizations of findings and insights.	BI Catalog
Business Asset ▶ Report ▶ BI Report ▶ Power BI Tile	An element representing data on the Power BI Dashboard.	BI Catalog
Data Asset ▶ Data Element ▶ Data Attribute ▶ BI Data Attribute ▶ Power BI Column	A column in a Power BI Data Model.	BI Catalog
Data Asset ▶ Data Structure ▶ Data Entity ▶ BI Data Entity ▶ Power BI Table	A table in a Power BI Data Model.	BI Catalog

Asset type	Description	Domain type
Data Asset ▶ Data Structure ▶ Data Model ▶ BI Data Model ▶ Power BI Data Flow	A collection of tables that are created and managed in workspaces in the Power BI service.	BI Catalog
Data Asset ▶ Data Structure ▶ Data Model ▶ BI Data Model ▶ Power BI Data Mart		BI Catalog
Data Asset ▶ Data Structure ▶ Data Model ▶ BI Data Model ▶ Power BI Data Model	A collection of data that is used to create a Power BI report.	BI Catalog
Technology Asset ▶ Server ▶ BI Server ▶ Power BI Server	A visual analytics platform for creating and storing Power BI Reports and Data Models.	BI Catalog

Asset type	Description	Domain type
Business Asset ▶ Business Dimension ▶ BI Folder ▶ SSRS Folder	A collection of SQL Server Reporting Services and Power BI Report Server Reports and Data Sets.	BI Catalog
Business Asset ▶ Report ▶ BI Report ▶ SSRS KPI	A key performance indicator of SQL Server Reporting Services.	BI Catalog

Asset type	Description	Domain type
Business Asset › Report › BI Report › SSRS Report	A detailed view of an SQL Server Reporting Services Data Set, with visualizations of findings and insights.	BI Catalog
Data Asset › Data Element › Data Attribute › BI Data Attribute › SSRS Column	A column in an SQL Server Reporting Services Report Data Set.	BI Catalog
Data Asset › Data Element › Report Attribute › BI Report Attribute › SSRS Parameter	A column that is part of an SQL Server Reporting Services Data Set and that is used in a KPI.	BI Catalog
Data Asset › Data Set › BI Data Set › SSRS Data Model	A collection of data that is used to create an SQL Server Reporting Services Report.	BI Catalog
Data Asset › Data Element › Data Attribute › BI Data Attribute › Power BI Table › SSRS Table	A table in an SQL Server Reporting Services Report Data Set.	BI Catalog
Technology Asset › Server › BI Server › SSRS Server	A visual analytics platform for creating and storing SQL Server Reporting Services and Power BI Report Server Reports and Data Sets.	BI Catalog

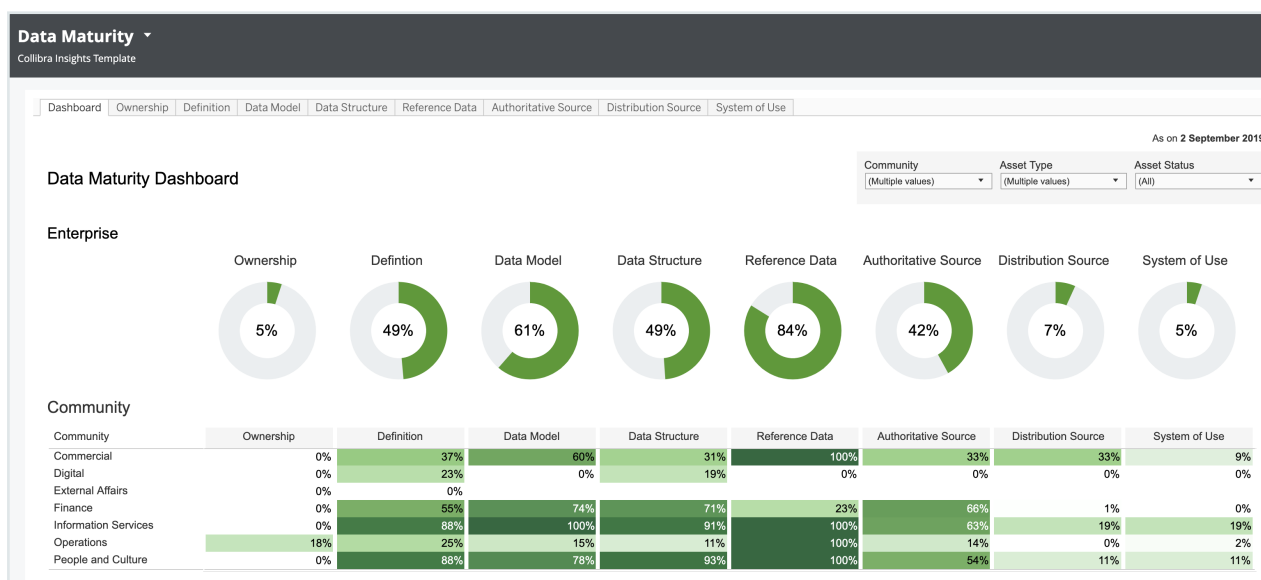
Asset type	Description	Domain type
Business Asset › Business Dimension › BI Folder › Tableau Project	Collection of Tableau workbooks and data sources.	BI Catalog
Business Asset › Business Dimension › BI Folder › Tableau Site	Collection of content (workbooks, data sources, users, ...) that's walled off from any other content on that instance of Tableau Server.	BI Catalog
Business Asset › Report › BI Report › Tableau View › Tableau Dashboard	A collection of several worksheets and supporting information, shown on a single screen, so that you can simultaneously compare and monitor a variety of data.	BI Catalog
Business Asset › Report › BI Report › Tableau View › Tableau Worksheet	A worksheet is a single sheet on which you can build views of your data.	BI Catalog
Business Asset › Report › BI Report › Tableau Workbook	Collection of sheets. A sheet can be a worksheet, a dashboard or a story.	BI Catalog
Data Asset › Data Element › Data Attribute › BI Data Attribute › Tableau Data Attribute	A specification that defines a property of a Tableau data entity. Examples: CustomerBirthDate, EmployeeFirstName.	BI Catalog
Data Asset › Data Structure › Data Model › BI Data Model › Tableau Data Model	An abstraction from the physical implementation of database, schema, file, etc., used for Tableau report creation.	BI Catalog

Asset type	Description	Domain type
Technology Asset ▶ Server ▶ BI Server ▶ Tableau Server	A visual analytics platform for creating interactive dashboards and rich visualisations	BI Catalog

Add a BI report to a Collibra dashboard

You can create a [dashboard](#) and use the [Collibra Insights widget](#) to curate it with your BI reports, or add reports to an existing dashboard.

The Collibra Insights widget is a reporting widget that you can use to show any report from a webpage.



Note Before configuring dashboard reports, you need to set up Insights Data Access. You can then configure dashboard reports via the Collibra Data Intelligence Cloud settings, or generate the dashboard reports via a URL from tools such as Tableau Online, Power BI, Qlik, or ThoughtSpot. Note that Insights Data Access is not a free, out-of-the-box product. For more information, go to [Insights Data Access](#).

Important Whether or not a third-party webpage can be embedded is determined by the security settings of the third party.

Configure the Collibra Insights widget

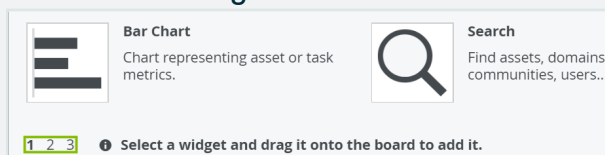
Requirements and permissions

- If you want to manage your own dashboard: Ensure that you have a [global role](#) that has the **Manage Your Own Views, Dashboards, Search Filters** [global permission](#).
- If you want to manage another user's dashboard: Ensure that you have a global role that has the **Manage Shared Views, Dashboards, Search Filters** global permission.
- [Insights Data Access](#) is set up.
- To add a BI report, you need access to the report in the relevant BI tool.

Steps

1. [Access](#) the dashboard for which you want to configure the Collibra Insights widget, or [create](#) a new dashboard.
2. Click  → **Edit widgets**.
3. Find the **Collibra Insights** widget and drag it onto the dashboard.

Tip You can find the widget by using the page numbers, to browse the collection of widgets.



- On the **Report** tab, in the **Enter a URL to view** field, enter the address of the webpage that contains the report that you want to be shown on the widget.

- Clear the **Run in sandbox** option. You might not be able to connect to your BI tool if this option is selected.
- On the **Widget** tab, enter the required information:

Field	Description
Show title above the widget	True: Shows the default title on the widget. You can also enter your own title. False: Does not show the title on the widget.
Show description above the widget	True: Shows the default description on the widget, below the title. You can also enter your own description. False: Does not show the description on the widget.
Refresh every	True: Automatically refreshes the counter according to the refresh rate you select. False: Does not automatically refresh the counter.

- Click **Save**.
- In the upper-right corner of the page, click **Close & save edits**.

Note If you are not yet signed in to the BI tool for which you are adding a report, you might be prompted to do so before the report appears on the dashboard.

Technical stewards

This section caters primarily to the following technical stewards:

Types of technical roles	What you want from Collibra Data Lineage
• Data Engineer • Enterprise Data Architect • Integrations Engineer • Artificial Intelligence / Machine Learning Engineer • Systems Engineer • Data Infrastructure Engineer • Data Quality Engineer	• Collect and evaluate data. • Consult data models in Data Catalog. • Perform impact analysis. • Evaluate data security. • Ensure the success of data migration from one data source to another.

Supported SQL syntax	dcciv
Supported transformation details	dccvi
Technical lineage for Snowflake ingestion methods	dccxiv
Transformation logic	dccxviii
Technical lineage export types	dccxix
BI integration concepts	dccxxiv
Technical lineage viewer	dcclxix

Supported SQL syntax

The SQL syntax used in your data sources has an impact on the technical lineage.

Technical lineage supports SQL syntax that is relevant to process data for all [supported data sources](#). This includes:

- DML (Data Manipulation Language) statements that are used to move and transform data. For example, *INSERT*, *UPDATE* and *MERGE*.

Note Technical lineage supports the extraction of DML statements from supported procedures, but it does **not support** all SQL syntax.

- DDL (Data Definition Language) statements:
 - that impact the technical lineage. For example, *ALTER TABLE*, which you use to add or rename columns.
 - that are used to transform data. For example, *CREATE A TABLE AS SELECT*.
- Relevant syntax constructs. For example, nested subselects, aliases, different join methods, synonyms and cross-database references.

Example You want to create a technical lineage for a Teradata source that has the following SQL syntax:

- ALTER TYPE
- ALTER PROCEDURE
- CREATE/REPLACE AUTHORIZATION
- MLOAD (MultiLoad)
- RECORD (FastLoad)
- BEGIN/END QUERY LOGGING
- Functions with schema, for example `schema_name.function.name(args...)`
- Functions with conversation, for example `function_name(args...) RETURNS VARCHAR(<number>) CHARACTER SET LATIN`
- Macro argument attributes

Collibra Data Lineage will successfully parse this SQL syntax.

Not supported SQL syntax

Technical lineage does not support the following SQL syntax:

- DML statements that you use to access data in complex structures such as JSON objects or structs.
- Triggers, foreign keys and indexes.
- Cursors, functions or dynamic queries.
- Streams queries.

Tip This is not an exhaustive list. If the SQL syntax that you use is not supported, you can add an idea in the [Collibra Integrations Ideation Portal](#). We will evaluate the SQL syntax for inclusion.

Tip Dynamic SQL statements yield limited results. For example, SSRS uses the columns defined by the first SELECT statement in a stored procedure to determine the columns in the result set. Therefore, if you want a full ingestion, you need a static SQL statement. Fortunately, you can transform dynamic SQL statements into static statements. If the dynamic SQL can be logged at the runtime of a table, the dynamic query is transformed into a static query that can be extracted by Collibra Data Lineage and processed without limitations.

Supported transformation details

Collibra Data Lineage supports the most commonly used [transformations](#) in the following sources:

- [Azure Data Factory](#)
- [Databricks Unity Catalog \(Beta\)](#)
- [dbt Cloud](#)
- [IBM DataStage](#)
- [Informatica PowerCenter](#)
- [Informatica Intelligent Cloud Services](#)
- [Snowflake](#)
- [SQL Server Integration Services](#)

Azure Data Factory

Collibra Data Lineage supports the most commonly used [transformations](#) and [data sources](#) in Azure Data Factory.

Supported transformations

The following tables shows a non-exhaustive list of supported and unsupported transformations.

Supported transformations	Unsupported transformations
• Aggregate¹ • Alter Row • Assert • Derived Column¹ • Exists • External Call² • Filter • Flatten¹ • Join • Lookup • Parse¹ • Pivot³ • Rank • Select¹ • Sink⁴ • Sort • Source • Split • Stringify • Surrogate Key • Union • Unpivot • Window¹	• Some reserved variables names, for example {<code>@context</code>} • Flowlets

Limitations

1. Transformations that contain column patterns or rule-based mappings can only be partially analyzed because they generate column names on the fly during the actual data flow run. If technical lineage is detected from a dynamically generated column, it is given the placeholder Dynamic Column in the technical lineage viewer.
2. In the Mapping section of the editor, column patterns are not supported and not displayed in the technical lineage graph. Note that Auto mapping uses column patterns behind the scenes and is therefore not supported either.
3. Pivoted columns can only be inferred when explicit values are provided in the Pivot Key tab. When columns cannot be inferred, a placeholder Pivoted Columns is added.
4. The SQL scripts and rule-based mappings in the transformation are not supported.

Supported data sources

The following table shows a non-exhaustive list of supported sources with the corresponding dataset and linked service types.

CollibraData Lineage supports all data format types that are supported in Azure Data Factory, including binary, Excel file, Delimited text, JSON, Parquet, and so on.

Data sources	Dataset type	Linked service type
Amazon Redshift	AmazonRedshiftTable	AmazonRedshift
Azure Blob storage	AzureBlob	AzureBlobStorage
Azure Data Lake Storage Gen2	AzureBlobFSFile	AzureBlobFS
Azure Data Lake Store	AzureDataLakeStoreFile	AzureDataLakeStore
Azure Databricks Delta Lake	AzureDatabricksDeltaLake	AzureDatabricksDeltaLake
Azure SQL Managed Instance	AzureSqlMITable	AzureSqlMI
Azure SQL Server database	AzureSqlTable	AzureSqlDatabase
Azure Synapse Analytics	AzureSqlDWTable	AzureSqlDW
DB2 data source	Db2Table	Db2
Google Cloud Storage	GoogleCloudStorageLocation	GoogleCloudStorage
Microsoft Access	MicrosoftAccessTable	MicrosoftAccess
Microsoft Azure Cosmos Database	CosmosDbSqlApiCollection	CosmosDb
Open Database Connectivity (ODBC)	OdbcTable	Odbc
On-premises Oracle database	OracleTable	Oracle

Data sources	Dataset type	Linked service type
REST	RestResource	RestService
Salesforce	SalesforceObject	Salesforce
Salesforce Marketing Cloud	SalesforceMarketingCloudObject	SalesforceMarketingCloud
Salesforce Service Cloud	SalesforceServiceCloudObject	SalesforceServiceCloud
SAP Business Warehouse (open hub)	SapOpenHubTable	SapBW
SFTP server	SftpLocation	Sftp
Snowflake	SnowflakeTable	Snowflake
SQL Server	SqlServerTable	SqlServer

Databricks Unity Catalog (Beta)

Collibra Data Lineage support for Databricks Unity Catalog is in [Beta testing](#), because the system tables feature in Databricks Unity Catalog is in Public Preview. Collibra Data Lineage support for Databricks Unity Catalog will go GA when the system tables feature in Databricks Unity Catalog goes GA.

Collibra Data Lineage retrieves lineage information from the lineage system tables that build on the Unity Catalog's data lineage feature, and visualizes lineage down to the column level, including notebooks, which are also captured in the lineage system tables. For examples of how Unity Catalog captures and presents data lineage, go to [Capture and view data lineage with Unity Catalog](#) in the Databricks documentation.

Note that while CollibraData Lineage retrieves lineage information from notebooks, it does not ingest or include the notebook assets in the technical lineage.

Important Unity Catalog does not provide source code for each transformation. Therefore, the source code pane in [the technical lineage graph](#) generated by CollibraData Lineage will not display any source code.

dbt Cloud

Collibra Data Lineage supports materialization, and tables and views are treated like tables by default. You can customize the setting in one of the following ways so that the tables and views are treated like views:

- If you use technical lineage via Edge, specify the `materializedMapping` property in the [<source ID> configuration file](#).
- If you use the lineage harvester, specify the `materializedMapping` property in the **Source Configuration** field in [the Technical Lineage for dbt Cloud capability](#).

IBM DataStage

IBM DataStage uses jobs with stages instead of transformations. IBM Datastage has three job types: parallel jobs, sequence jobs and server jobs. For a list of all job stages per job type in IBM DataStage, read the [IBM documentation](#).

Technical lineage for DataStage supports runtime parameters in parameter set files. To include the runtime parameters, ensure to export DataStage files with executables. For more information, go to [Prepare an external directory folder for the lineage harvester](#) if you use the lineage harvester, or [Create a technical lineage via Edge for DataStage](#).

Informatica PowerCenter transformations

The following table shows a non-exhaustive list of supported and unsupported transformations in Informatica PowerCenter.

Supported transformations	Unsupported transformations
• Aggregator • Expression¹ • Filter • Input • Joiner • Lookup • Mapplet • Normalizer • Output • Pre- and post-session SQL commands • Rank • Router • Sorter • Source • SQL in the <code>translate_db_type</code> function • Target • Transaction Control • Union • Update Strategy	• Data Masking • Java • Sequence Generator • Stored Procedure² • Web Services • XML
Note 1. The transformation is shown if the column (expression) is using at least one column from another connected transformation. 2. The stored procedures are stored and run in the databases that Informatica PowerCenter connects to. Colibra Data Lineage does not access the Informatica PowerCenter data sources, so Colibra Data Lineage collects the stored procedure names but does not support the Stored Procedure transformation.	

Informatica Intelligent Cloud Services

The following table shows a non-exhaustive list of supported taskflows and unsupported tasks in Informatica Intelligent Cloud Services.

Supported taskflows	Unsupported tasks
• Taskflow • Linear Taskflow	• Parallel tasks • Parallel tasks with decision • Sequential tasks • Sequential tasks with decision • Single task

The following table shows a non-exhaustive list of supported and unsupported transformations and constructions in Informatica Intelligent Cloud Services. Specifically, transformations and constructions in the [Cloud Data Integration service](#).

Supported transformations	Unsupported transformations, functions and constructions
• Data-driven conditions • Expression, including custom expressions in the supported transformations • Filter • Joiner, including join conditions • Lookup • Mapplet • Pre SQL and post SQL commands • Router • Sequence Generator • Source • Stored Procedure • Target • Union	• Aggregator • Cleanse • Data Masking • Deduplicate • Hierarchy Builder • Hierarchy Parser • Hierarchy Processor • Input • Java • Labeler • Machine Learning • Normalizer • NEXTVAL • Parse • Python • Rank • Rule Specification • Structure Parser • Transaction Control • Velocity • Verifier • Web Services

Snowflake

You can create technical lineage for Snowflake by using SQL Snowflake ingestion mode or SQL-API Snowflake ingestion mode. Collibra Data Lineage supports different queries

and transformations for each ingestion method. For more information about the ingestion methods, go to [Technical lineage for Snowflake ingestion methods](#).

SQL Snowflake ingestion mode

With the SQL Snowflake ingestion mode, Collibra Data Lineage does not support the following non-exhaustive list of transformations:

- Snowflake Scripting
- Snowpark

SQL-API Snowflake ingestion mode

With the SQL-API Snowflake ingestion mode, Collibra Data Lineage supports the Data Manipulation Language (DML) statements from the following sources. The table also shows a non-exhaustive list of unsupported queries and transformations.

Supported transformations	Unsupported queries and transformations
• Using a driver • Data Definition Language (DDL) queries • Direct login • Stored procedures • The <code>COPY INTO</code> DML command • Streams 2	• Queries or query paths that are not executed 1 • Sequences, including generating new values • Snowflake Scripting • Snowpark • Snowpipes
Note 1. If you create technical lineage for Snowflake by using the JDBC connection type, only queries or query paths that are executed are supported. For example, if a SQL query contains a CASE statement, the technical lineage will only show lineage from the WHEN path that was executed. However, if you use the folder connection type to ingest Snowflake, SQL queries that include all paths of a CASE statement will be parsed and reflected in the technical lineage. 2. Collibra Data Lineage supports lineage that uses streams as a source and lineage on tables that has streams. Collibra Data Lineage does not support lineage on a <code>CREATE STREAM</code> statement. 3. If the data sharing consumer moves data from the shared view to a table, Collibra Data Lineage does not support lineage from the table.	

SQL Server Integration Services (SSIS)

Collibra Data Lineage supports the following non-exhaustive list of transformations and component types in SQL Server Integration Services:

Supported transformations	Supported component types
• Aggregate • Cache Transform • Conditional Split • Data Conversion • Derived Column • Fuzzy Grouping • Lookup • Merge Join • Multicast • OLE DB Command • Row Count • Script Component • Slowly Changing Dimension • Sort • Union All	• Microsoft.ADONETDestination • Microsoft.DataReaderSourceAdapter • Microsoft.XmlSourceAdapter • Microsoft.ScriptComponentHost

Important

- Collibra Data Lineage supports SQL, but cannot parse other languages or scripts, for example SHELL and BAT scripts.
- SQL statements from Excel are not supported.
- All SQL queries must be preceded by the SELECT or WITH keyword, or else they will be skipped. If a comment precedes the SELECT or WITH keyword, the query will be parsed as expected.
- Collibra Data Lineage does not create lineage for disabled executables.

Technical lineage for Snowflake ingestion methods

To create technical lineage for Snowflake, you can use the following connection types:

- The JDBC connection. With this connection type, you can choose to use the SQL or SQL-API Snowflake ingestion modes.

- The folder connection type if you use the lineage harvester or the Shared Storage connection if you use technical lineage via Edge.

You can use different ingestion modes and connection types to collect and process the metadata of your Snowflake data sources with one technical lineage license. For example, you can use both the SQL-API ingestion mode and the folder or Shared Storage connection type. In this way, technical lineage is created based on the query execution and also provides a full coverage of stored procedures.

The JDBC connection type

You can use the JDBC connection type to establish connection to your Snowflake data sources. Collibra Data Lineage collects and processes the metadata from the data sources to create technical lineage.

With the JDBC connection type, you can choose to use the SQL or SQL-API Snowflake ingestion modes. These modes are complementary and are designed to address different needs and use cases.

SQL Snowflake ingestion modes

With this ingestion mode, Collibra Data Lineage retrieves lineage from the database schema and views, providing a design lineage. You can understand the data flow at the schema level from the generated technical lineage.

Note that stored procedures are not supported.

SQL-API Snowflake ingestion modes

Introduced in the 2023.02 release, the SQL-API mode retrieves lineage from views and executed database queries, providing an operational style of lineage. This mode accesses much more information and may take longer for lineage processing.

Stored procedures are supported. However, if a stored procedure is defined but not executed, the generated technical lineage does not include lineage for that stored procedure.

The technical lineage is based on Snowflake's interpretation of modified objects. Therefore, Collibra Data Lineage cannot show lineage for queries that Snowflake does not interpret or interprets differently than expected. For example, technical lineage does not include indirect lineage, as Snowflake does not interpret indirect lineage. Indirect lineage is the lineage that includes a column that does not appear in the target table but is used as a filter for data moving to the target table.

Additionally, if database queries contain conditional statements, the technical lineage includes lineage only for the conditions that were executed. Only the executed path of a `CASE WHEN/THEN` or `IF` statement is shown in lineage for each executed query instance.

If you use the lineage harvester, set the `mode` property in [the lineage harvester configuration file](#) to indicate which ingestion mode you want to use.

If you use technical lineage via Edge, use the **Ingestion Method** field in [the technical lineage for Snowflake capability](#) to select the ingestion mode you want to use.

The folder or Shared Storage connection type

With this connection type, you must prepare the SQL queries. The SQL queries can come from a log, stored procedure definitions, and so on. Collibra Data Lineage processes each conditional statement to create the technical lineage for all possible conditions.

If you use the lineage harvester, you must [prepare a SQL directory](#) and add your SQL queries to the folder.

If you use technical lineage via Edge, you must [add your SQL queries to the Shared Storage connection folder](#) and use the Technical Lineage for SqlDirectory capability to create the technical lineage.

See the following table for a summary of the connection types and ingestion modes.

Connection type	Ingestion mode	Details	Release date
JDBC	SQL	Collibra Data Lineage extracts metadata and information about the Snowflake database schemas and views to calculate lineage. This is the default mode. You can use the technical lineage to understand the flow of data at the schema level.	2020
	SQL-API	Collibra Data Lineage parses SQL from views and schemas, and additionally gets lineage information from the ACCESS_HISTORY system view, which is a log of all queries that are run on the system. The SQL-API mode supports stored procedures and other orchestration methods, for example, application queries and ad-hoc queries. You can use the technical lineage to see the operational lineage from executed queries.	2023.02
Folder or Shared Storage connection	Not applicable	Collibra Data Lineage retrieves lineage from the SQL queries that you upload to a SQL directory. The technical lineage captures all lineage paths from the SQL queries.	Folder - 2020 Shared Storage connection - 2023.05

For more information about the supported queries and transformation, go to [Supported transformation details](#).

For an overview of the steps to create technical lineage, go to [Create a technical lineage via the lineage harvester](#) and [Create a technical lineage via Edge](#).

For more information about Snowflake, go to [Snowflake Documentation](#).

Transformation logic

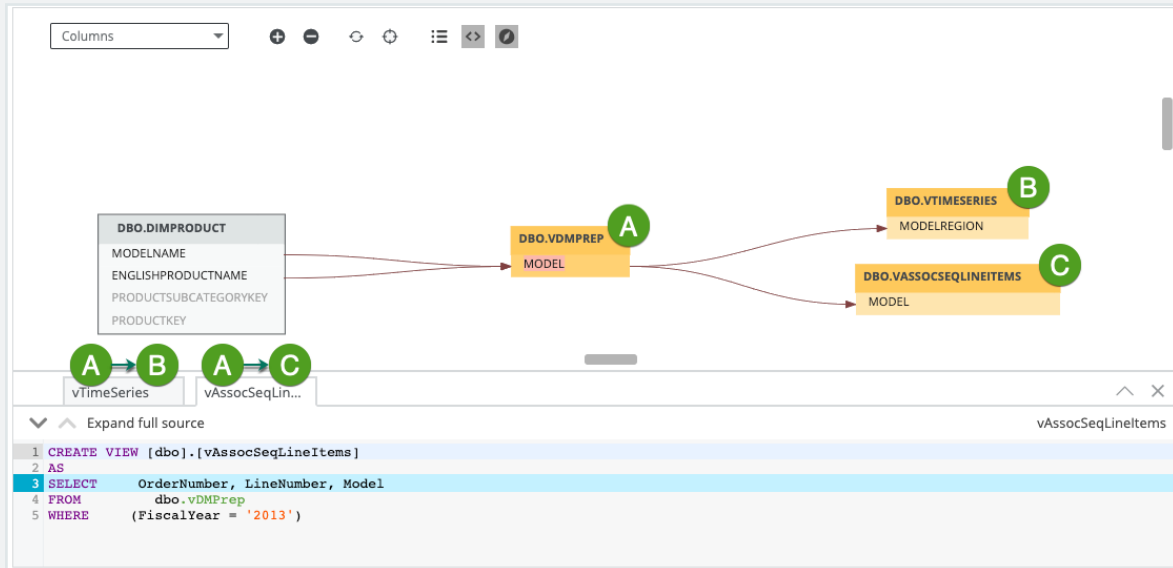
Transformation logic is used to transform source code in a technical lineage diagram that can be visualized in Data Catalog. Collibra Data Lineage [supports](#) the most commonly used transformations.

Collibra Data Lineage enables you to trace how your data flows between multiple data sources and, at the same time, see the source code of each part of your technical lineage. By following the transformations in your technical lineage, you can easily find a specific source code fragment.

Tables and columns in a technical lineage can have incoming and outgoing transformations. When you right-click on a table or column and click either Transformations (IN) or Transformations (OUT), the source code pane shows the following:

- The name of the source code fragment. On the [Sources tab page](#), you can see the analysis log files of this source code fragment.
- If a table or column has more than one transformation, there are tabs for each source code fragment.
- The source code of the fragment. The source code that is relevant for the selected column or table is highlighted.

Example You want to see the outgoing transformations of column A to columns B and C. When you right-click column A and then click **Transformations (OUT)**, you see that there are two tabs containing source code. The first tab shows the outgoing source code from column A to column B. The second tab shows the source code from column A to column C.



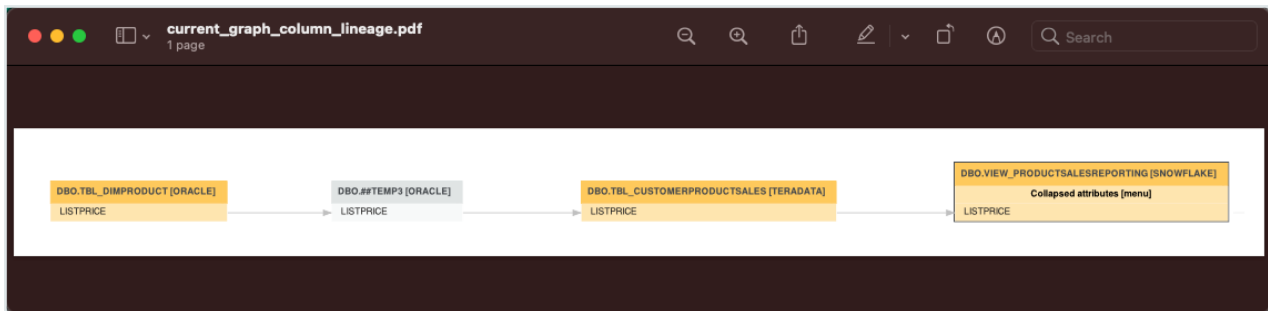
Technical lineage export types

If you want to share a [technical lineage graph](#) of your technical lineage, you can export the information to one of the following export types, via the [Settings tab pane](#):

- [PDF](#)
- [PNG](#)
- [Graph CSV](#)
- [Full Batch CSV](#)
- [JSON Lineage](#)

PDF and PNG

The PDF and PNG exports show only the technical lineage graph of the selected table or column.



Graph CSV

The CSV export option generates a ZIP file with the following CSV file:

File name	File content
current_graph_column_lineage.csv	The technical lineage graph of the selected column or table.

Full Batch CSV

The Full CSV option generates a ZIP file with the following CSV files:

File name	File content
current_graph_column_lineage.csv	The technical lineage graph of the selected column or table.
full_batch_column_lineage.csv	The technical lineage graph of the full technical lineage.

Example

The current_graph_column_lineage CSV file and the full_batch_column_lineage CSV files show the same information, but with a different scope. These files show how data flows from source to target.

	1	2	3	4	5	6	7	8	9	10	11	12
	source_system	source_database	source_schema	source_table	source_column	target_system	target_database	target_schema	target_table	target_column	procedure_names	query_names
14	CRMSys	REFINED	DBO	CUSTOMERPI	SALESAMOUNT	SYSTEM1	CONSUMPTION	DBO	CUSTOMERS	SALESAMOUNT		CustomerSalesReporting
15	CRMSys	REFINED	DBO	CUSTOMERPI	SALESORDERNL	SYSTEM1	CONSUMPTION	DBO	CUSTOMERS	SALESORDERNUMBER		CustomerSalesReporting
16	CRMSys	REFINED	DBO	CUSTOMERPI	SALESTERRITOR	SYSTEM1	CONSUMPTION	DBO	CUSTOMERC	SALESTERRITORYCOUNTRY		CustomerChurnReporting
17	CRMSys	REFINED	DBO	CUSTOMERPI	SALESTERRITOR	SYSTEM1	CONSUMPTION	DBO	CUSTOMERC	SALESTERRITORYKEY		CustomerChurnReporting
18	CRMSys	REFINED	DBO	CUSTOMERPI	SALESTERRITOR	SYSTEM1	CONSUMPTION	DBO	CUSTOMERC	SALESTERRITORYREGION		CustomerChurnReporting
19	CRMSys	REFINED	DBO	CUSTOMERPI	TOTALPRODUCT	SYSTEM1	CONSUMPTION	DBO	PRODUCTSAI	TOTALPRODUCTCOST		ProductSalesReporting
20	CRMSys	REFINED	DBO	CUSTOMERPI	UNITPRICE	SYSTEM1	CONSUMPTION	DBO	PRODUCTSAI	UNITPRICE		ProductSalesReporting
21	DEFAULT	RAW	DBO	##TEMP1	ADDRESSLINE1	DEFAULT	RAW	DBO	##TEMP2	FULLADDRESS	join_tables_anonymize_push_to_refined_zone	join_tables_anonymize_push_to_refined_zone
22	DEFAULT	RAW	DBO	##TEMP1	ADDRESSLINE2	DEFAULT	RAW	DBO	##TEMP2	ADDRESSLINE2	join_tables_anonymize_push_to_refined_zone	join_tables_anonymize_push_to_refined_zone
23	DEFAULT	RAW	DBO	##TEMP1	BIRTHDATE	DEFAULT	RAW	DBO	##TEMP2	BIRTHDATE	join_tables_anonymize_push_to_refined_zone	join_tables_anonymize_push_to_refined_zone
24	DEFAULT	RAW	DBO	##TEMP1	CITY	DEFAULT	RAW	DBO	##TEMP2	FULLADDRESS	join_tables_anonymize_push_to_refined_zone	join_tables_anonymize_push_to_refined_zone
25	DEFAULT	RAW	DBO	##TEMP1	COMMUTEDIST	DEFAULT	RAW	DBO	##TEMP2	COMMUTEDIST	join_tables_anonymize_push_to_refined_zone	join_tables_anonymize_push_to_refined_zone
26	DEFAULT	RAW	DBO	##TEMP1	COUNTRYREGI	DEFAULT	RAW	DBO	##TEMP2	COUNTRYREGI	join_tables_anonymize_push_to_refined_zone	join_tables_anonymize_push_to_refined_zone
27	DEFAULT	RAW	DBO	##TEMP1	CUSTOMERALT	DEFAULT	RAW	DBO	##TEMP2	CUSTOMERALT	join_tables_anonymize_push_to_refined_zone	join_tables_anonymize_push_to_refined_zone
28	DEFAULT	RAW	DBO	##TEMP1	CUSTOMERKEY	DEFAULT	RAW	DBO	##TEMP2	CUSTOMERKEY	join_tables_anonymize_push_to_refined_zone	join_tables_anonymize_push_to_refined_zone
29	DEFAULT	RAW	DBO	##TEMP1	DATEFIRSTPUR	DEFAULT	RAW	DBO	##TEMP2	DATEFIRSTPUR	join_tables_anonymize_push_to_refined_zone	join_tables_anonymize_push_to_refined_zone

No	Column	Description
1	source_system	The name of the source system. Note This column is only shown when <code>useCollibraSystemName</code> is set to <code>true</code> in the lineage harvester configuration file.
2	source_database	The name of the source database.
3	source_schema	The name of the source schema.
4	source_table	The name of the source table.
5	source_column	The name of the source column.
6	target_system	The name of the target system. Note This column is only shown when <code>useCollibraSystemName</code> is set to <code>true</code> in the lineage harvester configuration file.
7	target_database	The name of the target database.
8	target_schema	The name of the target schema.

No	Column	Description
9	target_table	The name of the target table.
10	target_column	The name of the target column.
11	procedure_name	The name of the stored procedure. This column remains empty when an object in your technical lineage doesn't have stored procedure. Warning This column is deprecated and will be removed in the future.
12	query_name	The name of the specific source code fragment or transformation detail. You can use this name to search for more information in the Sources tab page.

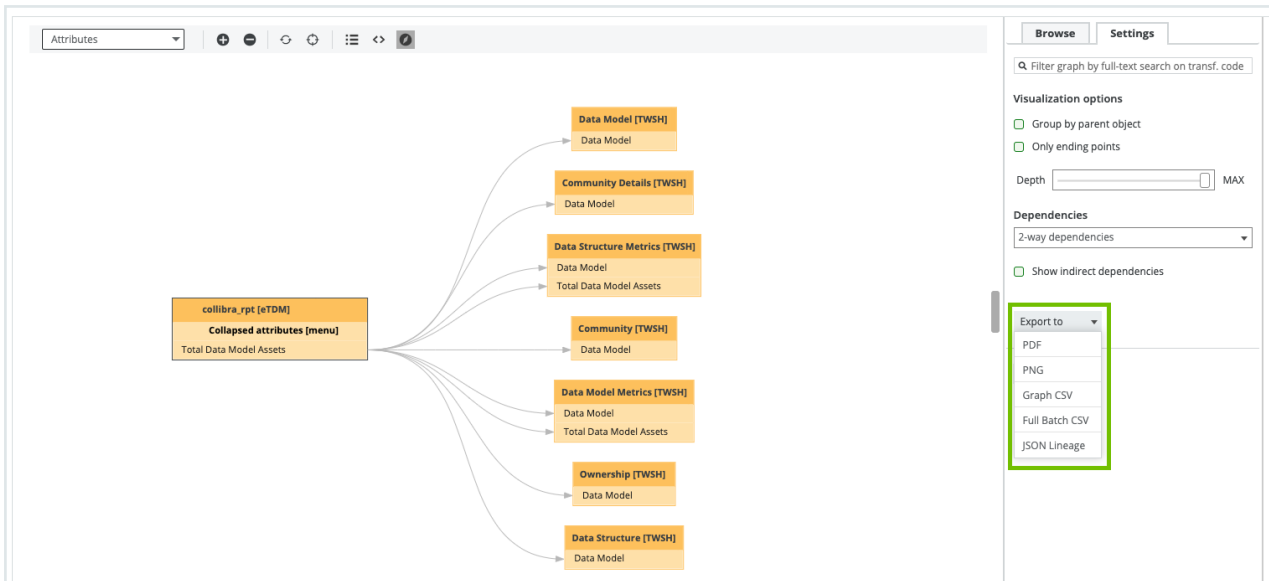
Tip The names of the source and target objects indicate the full path of the object. For example, the full name of a column is (system) > database > schema > table > column. This path is used to [stitch](#) your technical lineage objects to assets in Data Catalog.

JSON Lineage

This export option generates a JSON file that is formatted in the same manner that is required for creating a custom technical lineage.

Export the technical lineage information

If you want to share a [technical lineage graph](#) or the [transformation logic](#) of your technical lineage, for example with colleagues who don't have access to Collibra, you can export the information. For complete details on the various export options, see [Technical lineage export types](#).



Steps

1. In the [Technical lineage viewer](#), click the [Settings](#) tab.
2. On the **Settings** tab, click **Export**.
3. Click the [export type](#).



» The technical lineage information is downloaded.

Export technical lineage transformation details

If you want to download analyzing and parsing errors for a data source, you can export the transformation details of one or more data sources on [the Sources tab page](#) of the technical lineage viewer.

ID	Name	Status code	Status description	Group name
26728	ACCOUNT_MANAGERS	ANALYZE_ERROR	Problem with the source table for column "C.CUST_ADDRESS_COUNTRY_ID" at line 3, column 3. Please verify that the column exists in the DDL/schema provided.	OE.ACCOUNT_MANAGERS
26729	CUSTOMERS_VIEW	PARSING_ERROR	Unable to proceed with syntax parsing at line 24 column 32. c.cust_geo_location.sdo_point_x location_x ^	OE.CUSTOMERS_VIEW
26733	OC_CUSTOMERS	ANALYZE_ERROR	Problem with the source table for column "OC_CUSTOMERS" at line 5, column 41. Please verify that the column exists in the DDL/schema provided.	OE.OC_CUSTOMERS
26734	OC_CORPORATE_CUSTOMERS	ANALYZE_ERROR	Problem with the source table for column "OC_CUSTOMERS" at line 5, column 41. Please verify that the column exists in the DDL/schema provided.	OE.OC_CORPORATE_CUST OMERS
26735	OC_ORDERS	ANALYZE_ERROR	Problem with the source table for column "OC_CUSTOMERS" at line 1, column 196. Please verify that the column exists in the DDL/schema provided.	OE.OC_ORDERS
26740	BSLN_INTERNAL	PARSING_ERROR	Unable to proceed with syntax parsing at line 1 column 1.	PACKAGE

OracleCloud Full-text search Filter by ERROR + Export Selected Transformations

1. In the [Technical lineage viewer](#), click the [Settings tab](#).
2. Click **Show lineage**.
3. Select the data sources for which you want to download the transformation details. If you want to download the transformation details for all data sources, do not select any data source.
4. Click **Export Selected Transformations**.
 - » A ZIP file that contains an **errors.csv** file is downloaded.

This section addresses BI tool-specific integration concepts for technical stewards.

This topic provides information about the technical lineage that is created when you ingest BI tool metadata in Data Catalog.

For a business perspective, see [Technical lineage and stitching for BI tool integrations](#).

When you ingest Tableau metadata in Data Catalog, a technical lineage for Tableau Data Attribute assets is automatically created.

Permissions

If you have a Data Catalog global role with the Catalog and Technical lineage [global permissions](#), you can see the technical lineage of Tableau assets by clicking on the Technical lineage tab on the asset page of any of the following asset types:

- Table
- Column
- Tableau Data Attribute
- Tableau Worksheet

Technical lineage graph

The [technical lineage graph](#) shows relations of the type "Data Element sources / targets Data Element" between Tableau assets and other data objects in the data flow, for example between a Column asset and a Tableau Data Attribute asset. These relations are created during the Tableau ingestion process as a result of automatic stitching.

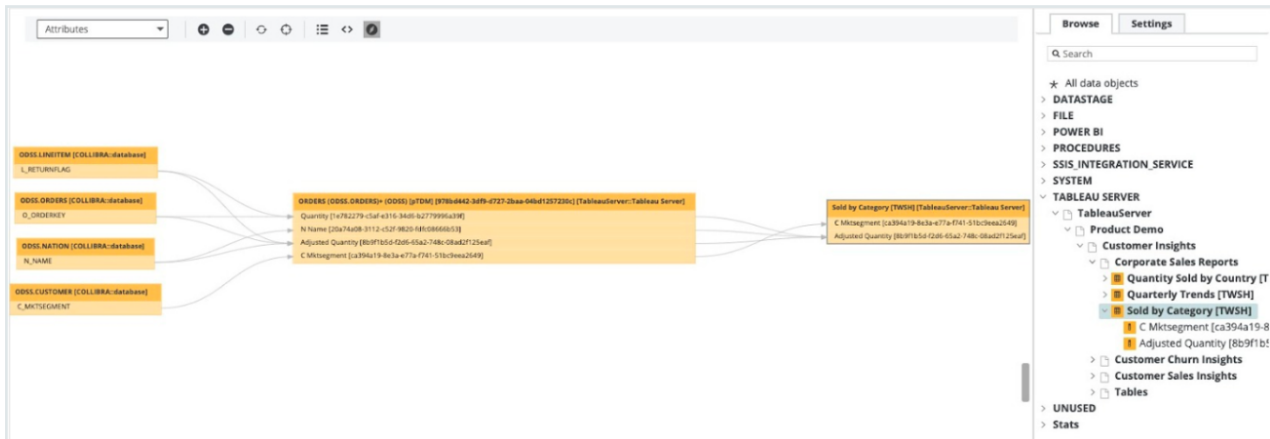
Note If you use a Tableau [<source ID> configuration file](#) and don't specify a value for the relevant `collibraSystemName` property, the designation "UNDEFINED" will be shown in the technical lineage.

Note If you use custom SQL that is not supported by the Tableau metadata API, the technical lineage might not be complete. For complete information, see the Tableau documentation on [Tableau Catalog support for custom SQL](#) and [Tableau Lineage and custom SQL connections](#).

Example

The following technical lineage shows how data flows from a PostgreSQL data source to Tableau. It shows relations of the type "Data Element sources / targets Data Element"

between the Column assets of the database and Tableau Data Attribute assets in Tableau. For example, Column asset *L_RETURNFLAG* has a relation of the type "Data Element sources / targets Data Element" to the Tableau Data Attribute assets *Quantity* and *Adjusted Quantity*.



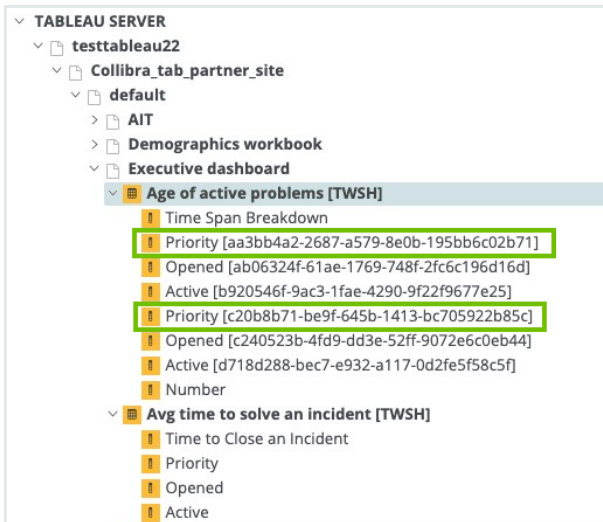
UUIDs in the Tableau technical lineage

Collibra Data Lineage uses unique full names to create a technical lineage and stitch objects within the technical lineage. Full names in Collibra are constructed in accordance with the hierarchy of data objects in Tableau, for example:

- Server > Site > Project > Workbook > Worksheet > Field
- Server > Site > Project > Workbook > Data Model > Column

In Collibra, every node in this hierarchy must have a unique name. However, in Tableau, the names of data objects do not have to be unique. As such, if Tableau data objects in a technical lineage hierarchy have the same full name, Collibra Data Lineage adds the UUIDs of the corresponding assets to the names in the technical lineage, to maintain uniqueness.

In the following example image, the names of the assets Priority, Opened and Active in the technical lineage have been appended with their UUIDs.



Note

- UUIDs are not added to the names of the assets themselves; they are only added to the names of the data objects in the technical lineage.
- The UUID is always part of the full name of an asset, regardless of whether or not it is a duplicate.

How to resolve UUIDs in names in a technical lineage

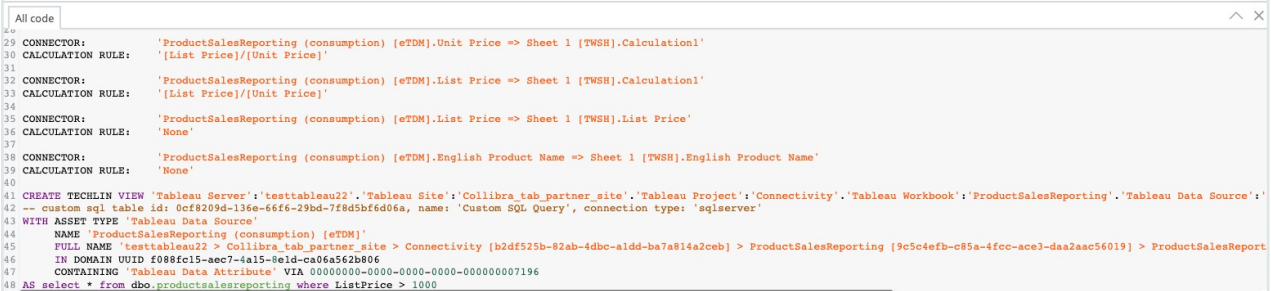
To keep Collibra Data Lineage from adding UUIDs to the names of the data objects in a technical lineage, ensure that the names of all fields and columns in Tableau are unique.

Generally, Tableau doesn't allow you to create two fields or columns with the same name. However, hierarchy fields and non-hierarchy fields can have the same name. Duplication of names can also happen if:

- A Tableau worksheet is using two different data sources that have columns with the same name.
- You create a virtual connection that contains multiple data sources that have columns with the same name.
- There are multiple data sources in Tableau with the same name.

Sources tab page

The [Sources tab page](#) shows, for each Tableau data source and Tableau Worksheet, the transformation and calculation rules that the Collibra Data Lineage service analyzed and processed, and the results of the analysis. It also shows the TECHLIN VIEW query definitions, based on custom SQL queries.



```
40 All code
29 CONNECTOR: 'ProductSalesReporting (consumption) [eTDM].Unit Price => Sheet 1 [TWSH].Calculation1'
30 CALCULATION RULE: '[List Price]/[Unit Price]'
31
32 CONNECTOR: 'ProductSalesReporting (consumption) [eTDM].List Price => Sheet 1 [TWSH].Calculation1'
33 CALCULATION RULE: '[List Price]/[Unit Price]'
34
35 CONNECTOR: 'ProductSalesReporting (consumption) [eTDM].List Price => Sheet 1 [TWSH].List Price'
36 CALCULATION RULE: 'None'
37
38 CONNECTOR: 'ProductSalesReporting (consumption) [eTDM].English Product Name => Sheet 1 [TWSH].English Product Name'
39 CALCULATION RULE: 'None'
40
41 CREATE TECHLIN VIEW 'Tableau Server': 'testtableau22'. 'Tableau Site': 'Collibra_tab_partner_site'. 'Tableau Project': 'Connectivity'. 'Tableau Workbook': 'ProductSalesReporting'. 'Tableau Data Source':
42 -- custom sql table id: 0cf8209d-136e-66f6-29bd-7f8d5bf6d06a, name: 'Custom SQL Query', connection type: 'sqlserver'
43 WITH ASSET TYPE 'Tableau Data Source'
44 NAME 'ProductSalesReporting (consumption) [eTDM]'
45 FULL NAME 'testtableau22 > Collibra_tab_partner_site > Connectivity [b2df525b-82ab-4dbc-ald-d-ba7a814a2ceb] > ProductSalesReporting [9c5c4efb-c85a-4fcc-ace3-daa2aac56019] > ProductSalesReport
46 IN DOMAIN UUID f088fc15-aec7-4a15-8e1d-ca06a562b806
47 CONTAINING 'Tableau Data Attribute' VIA 00000000-0000-0000-0000-000000007196
48 AS select * from dbo.productsalesreporting where ListPrice > 1000
```

If a parameter is used in a Tableau worksheet, it is shown in the worksheet source code, for example:

PARAMETERS: 'parameter1'.

If a parameter is used in a calculation rule, it is also shown under the Tableau data source for data sources in the calculation rule, for example:

CALCULATION RULE: '[List price]/[parameter1]'

The success rate of the analysis indicates how complete the technical lineage is. There are a few limitations that prevent the Collibra Data Lineage service from processing all Tableau metadata.

Important The Collibra Data Lineage service might not be able to process all complex Tableau metadata. This means that the success rate of a Tableau ingestion might not be 100%.

Note
With the implementation of SQL extension v2 in Collibra 2024.01.1, queries and their relevant assets are now combined into a single line in the transformations table. Because of this, the source code processed count in the Done column is reduced. The lineage itself is unchanged.

Sources

Stitching

Selection	Source ID	Scanner type	Success rate	Done	Parsing Error	Analyze Error	Last sync time
☐	ssrs	SSRS	100 %	32	0	0	2023-12-04 09:00:41 UTC
☒	ssrs	SQL	94 %	16	1	0	2023-12-04 09:00:41 UTC
☐	looker	SQL	98 %	247	3	0	2023-12-04 08:47:37 UTC
☐	tableauOn	SQL	100 %	49	0	0	2023-12-04 08:47:22 UTC
☐	tableauOn	TABLEAU	100 %	922	0	0	2023-12-04 08:47:17 UTC
☐	powerbi2	SQL	100 %	1	0	0	2023-12-04 08:47:16 UTC

ssrs

Full-text search

Filter by

All

Export Selected Transformations

ID	Name	Status code	Status description	Group name
88138	None	DONE	Analysis succeeded, queries 3	None
1 {"id":"00000000-0000-0000-0000-100000000030","name":"SSRS Table"} 2 {"id":"00000000-0000-0000-0000-100000000029","name":"SSRS Column"} 3 {"id":"00000000-0000-0000-0000-000000007047","role":"contains","coRole":"is part of","sourceAssetTypeName":"SSRS Table","targetAssetTypeName":"SSRS Column"}				
88141	Source Code 1	DONE	Analysis succeeded, queries 1	None
88142	Source Code 3	DONE	Analysis succeeded, queries 1	None
88143	Source Code 5	DONE	Analysis succeeded, queries 1	None
88144	Source Code 7	DONE	Analysis succeeded, queries 1	None
88145	Source Code 9	PARSING_ERROR	Unable to proceed with syntax parsing at line 2 column 1. Person.testing_procedure_TS ^	None
88146	Source Code 11	DONE	Analysis succeeded, queries 1	None

Error codes

The `Errors summary` represents a summary of all errors per Tableau site. The `continue on error` feature allows for continuous processing of an import or synchronization job, even if one or more commands fail.

Sources

Stitching

Selection	Source ID	Scanner type	Success rate	Done	Parsing Error	Analyze Error	Last sync
☐	tableau	TABLEAU	100 %	1	0	0	2023-01-01 00:00:00 UTC
☒	tableau	TABLEAU	0 %	0	0	0	None UTC

Full-text search

Filter by

All

ID	Name	Status code	Status description	Group name
-10	Errors summary	ERROR	None	None
1 1/1 sites have failed. 2 Processing site collibratabpartnersite has failed 3 No closing quotation 4				
1	Transformation	DONE	None	None

Warning codes

Warning codes indicate:

- Issues that might affect the technical lineage, but do not stop the processing.
- Issues that you can resolve.

ID	Name	Status code	Status description	Group name
4639	Parent project not found	WARNING	Part 1 out of 1	Missing content
4697	REST datasource not found	WARNING	Part 1 out of 6	Mismatched ID

Element	Description																				
ID	The warning ID number.																				
Name	The name of the warning. Possible values are: • Empty name • Field relation not found • Parent project not found • Parent workbook not found • Parent database not found • Datasource not found • Worksheet not found • REST datasource not found • Not found in remote fields • Multiple datasources • Query parsing error • Invalid Collibra system names • Invalid hostname mapping																				
Status code	The status label. The value is always WARNING.																				
Status description	Identifies a grouping of warnings. Warnings of the same type (meaning they have the same group name and name) are grouped together in "parts" of up to 100 warnings. Example In this example, there are 250 Configuration > Invalid Collibra system names warnings, grouped into parts 1, 2 and 3: <table><tr><th>ID</th><th>Name</th><th>Status code</th><th>Status description</th><th>Group name</th></tr><tr><td>4714</td><td>Invalid Collibra system names</td><td>WARNING</td><td>Part 1 out of 3</td><td>Configuration</td></tr><tr><td>4715</td><td>Invalid Collibra system names</td><td>WARNING</td><td>Part 2 out of 3</td><td>Configuration</td></tr><tr><td>4716</td><td>Invalid Collibra system names</td><td>WARNING</td><td>Part 3 out of 3</td><td>Configuration</td></tr></table>	ID	Name	Status code	Status description	Group name	4714	Invalid Collibra system names	WARNING	Part 1 out of 3	Configuration	4715	Invalid Collibra system names	WARNING	Part 2 out of 3	Configuration	4716	Invalid Collibra system names	WARNING	Part 3 out of 3	Configuration
ID	Name	Status code	Status description	Group name																	
4714	Invalid Collibra system names	WARNING	Part 1 out of 3	Configuration																	
4715	Invalid Collibra system names	WARNING	Part 2 out of 3	Configuration																	
4716	Invalid Collibra system names	WARNING	Part 3 out of 3	Configuration																	

Element	Description
Group name	The type, or category, of warning. Possible values are: • Configuration • Mismatched ID • Missing content

The following table shows the complete set of warning codes, by group and name.

Group name	Name	Description
Missing content	Empty name	Raised during the processing of databases, tables, columns, worksheets and dashboards. Contains the following lines: • Database with id DATABASE_ID is skipped • Table with id TABLE_ID is skipped • Column with id COLUMN_ID is skipped • Worksheet with id WORKSHEET_ID is skipped • Dashboard with id DASHBOARD_ID is skipped Indicates that the <code>name</code> property of a database, table, column, worksheet or dashboard, which has a specified value for the <code>id</code> property, has a <code>null</code> value or it is empty: Example for a database: <pre> { "data": { "databasesConnection": { "nodes": [{ "id": "DATABASE_ID", "name": null, ... }] } } } </pre> Note The <code>name</code> property is considered empty if the value is <code>null</code> or if it is empty.

Group name	Name	Description
Missing content	Parent database not found	Raised during the processing of tables. Contains the line: Table with id TABLE_ID is skipped Indicates that the parent database for a table with TABLE_ID was not found in the previously processed databases. Possible causes: • The database property is not present in the JSON file. • The database property is empty: "database": {}. • The DATABASE_ID is not present for the id property. <pre> { "data": { "tablesConnection": { "nodes": [{ "id": "TABLE_ID", "database": { "id": "DATABASE_ID" } } ...] } } } </pre>

Group name	Name	Description
Missing content	Parent project not found	Raised during the processing of projects, workbooks and published data sources. Contains the following lines: • Workbook with id WORKBOOK_ID is skipped • Published datasource with id DATASOURCE_ID is skipped • Project with id PROJECT_ID has unreachable parent project Indicates that the parent project of a project, workbook, or published data source was not found in the previously processed projects. Possible causes: • The project property is not present in the JSON file. • The project property is empty: "project": {}. • The PROJECT_ID is not present for the id property. Example for a workbook: <pre> { "workbooks": { "workbook": [{ "project": { "id": "PROJECT_ID" }, "id": "WORKBOOK_ID", ... }] } } </pre> Example for a published datasource: To identify the PROJECT_ID, first find the DATASOURCE_LUID of the published data source, as returned by the metadata API: <pre> { "data": { "datasourcesConnection": { "nodes": [</pre>

Group name	Name	Description
		<pre>{ "__typename": "PublishedDatasource", "id": "DATASOURCE_ID", "luid": "DATASOURCE_LUID" ... }</pre> Then, in the data returned by the REST API, reference the DATASOURCE_LUID to identify the PROJECT_ID of the data source.: <pre>{ "datasources": { "datasource": [{ "id": "DATASOURCE_LUID", "project": { "id": "PROJECT_ID", ... } }] } }</pre> Example for a project: PARENT_PROJECT_ID is not found: <pre>{ "projects": { "project": [{ "id": "PROJECT_ID", "parentProjectId": "PARENT_ PROJECT_ID", ... }] } }</pre> Project is not skipped in this case. The new parent project is created with name Unknown project PARENT_PROJECT_ID.

Group name	Name	Description
Missing content Mismatched ID	Parent workbook not found	Raised during processing of worksheets, dashboards, REST-only views and embedded data sources. Contains the following lines: • Worksheet with id WORKSHEET_ID is skipped • Dashboard with id DASHBOARD_ID is skipped • View with id VIEW_ID is skipped (rest only) • Embedded data source with id DATASOURCE_ID is skipped Indicates that the parent workbook of a worksheet, dashboard or view with a specified ID was not found in the previously processed workbooks. Possible causes: • The <code>workbook</code> property is not present in the JSON file. • The <code>workbook</code> property is empty: <code>"workbook": {}</code>. • <code>WORKBOOK_ID</code> is not present for the <code>luid</code> property. • mismatched ID issue. Example for a worksheet: <pre> { "data": { "sheetsConnection": { "nodes": [{ "id": "WORKSHEET_ID", "workbook": { "luid": "WORKBOOK_ID" ... } }] } } } </pre> Example for a dashboard: <pre> { "data": { "dashboardsConnection": { </pre>

Group name	Name	Description
		<pre>"nodes": [{ "id": "DASHBOARD_ID", "workbook": { "luid": "WORKBOOK_ID" } } ...]</pre> Example for an embedded data source: <pre>{ "data": { "dashboardsConnection": { "nodes": [{ "id": "DASHBOARD_ID", "workbook": { "luid": "WORKBOOK_ID" } } ...] } } }</pre> Note Use the <code>luid</code> property, not the <code>id</code> property, to find a workbook by ID.

Group name	Name	Description
Missing content	Worksheet not found	Raised during the processing of dashboards. Contains the line: Worksheet with id WORKSHEET_ID is skipped for dashboard with id DASHBOARD_ID Indicates that a worksheet with a given ID was not found in the previously processed worksheets. <pre> { "data": { "dashboardsConnection": { "nodes": [{ "id": "DASHBOARD_ID", "sheets": [{ "id": "WORKSHEET_ID" }, ...] }] } } } </pre> Possible cause: WORKSHEET_ID is not present for the id property.

Group name	Name	Description
Mismatched ID	REST datasource not found	Raised during the processing of published data sources. Contains the line: Published datasource with id DATASOURCE_ID is skipped Indicates that a data source with DATASOURCE_ID could not be matched with the DATASOURCE_LUID returned by the REST API, resulting in a mismatched ID. <pre>{ "data": { "datasourcesConnection": { "nodes": [{ "__typename": "PublishedDatasource", "id": "DATASOURCE_ID", "luid": "DATASOURCE_LUID" ... }] } } }</pre> During processing, information returned by the metadata API and the REST API is combined. Collibra Data Lineage then looks to the DATASOURCE_LUID property in the REST metadata to identify the correct project ID, which is lacking from the information returned by the metadata API. This only applies to published data sources, as embedded data sources are assigned to workbooks, not projects.

Group name	Name	Description
Missing content	Datasource not found	Raised during the processing of embedded data sources. Contains the line: Embedded datasource with id EMBEDDED_DATASOURCE_ID references non existing published datasource with id PUBLISHED_DATASOURCE_ID Indicates that an embedded data source with EMBEDDED_DATASOURCE_ID references a published data source with PUBLISHED_DATASOURCE_ID, which was not found in the previously processed data sources. <pre> { "data": { "datasourcesConnection": { "nodes": [{ "__typename": "EmbeddedDatasource", "id": "EMBEDDED_DATASOURCE_ ID", "upstreamDatasources": [{ "id": "PUBLISHED_ DATASOURCE_ID", ... }] }] } } } </pre> Possible cause: PUBLISHED_DATASOURCE_ID is not present for the id property.

Group name	Name	Description
Missing content	Field relation not found	Raised during the processing of data source fields. Contains the lines: - Referenced field with id FIELD_ID is skipped - Report field with id FIELD_ID is skipped Indicates that a field with a given FIELD_ID was not found in remote fields, which is needed to create relations. <pre> { "data": { "datasourcesConnection": { "nodes": [{ "id": "DATASOURCE_ID", "fieldsConnection": { "nodes": [{ "__typename": "DatasourceField", "remoteField": { "id": "FIELD_ID" ... </pre> Possible cause: An embedded datasource has a calculated field that is not mapped to any published data source field. This can occur: - During the processing of referenced fields. In this case, the relation between the two Tableau Data Attributes cannot be created. - During the processing of report fields. In this case, the relation between the Tableau Data Attribute and the Tableau Data Worksheet cannot be created.

Group name	Name	Description
Missing content	Multiple datasources	Raised during the processing of custom SQL queries. Contains the line: Custom sql query with id QUERY_ID contains columns of NUMBER_OF_DATASOURCES datasources. Found best datasource: DATASOURCE_ID Indicates that a query with QUERY_ID has matched multiple data sources. Only one data source can be used: datasource with DATASOURCE_ID. The warning is caused by the fact that there is no direct relation between the query and the data source. The algorithm tries to find the best data source, based on a comparison of the list of query columns and the data source columns. To verify this, do the following: - Find the query with QUERY_ID and the columns (see COLUMN_ID) in the table JSON data: <pre> { "data": { "tablesConnection": { "nodes": [{ "__typename": "CustomSQLTable", "id": "QUERY_ID", "columnsConnection": { "nodes": [{ "id": "COLUMN_ID", ... </pre> - Find the data source with DATASOURCE_ID in the data source JSON data. It should contain all of the columns (see COLUMN_ID) that are used by the query: <pre> { </pre>

Group name	Name	Description
		<pre>"data": { "datasourcesConnection": { "nodes": [{ "id": "DATASOURCE_ID", "fieldsConnection": { "nodes": [{ "id": "COLUMN_ID" }] } }] } }</pre> The data source found for this query (meaning DATASOURCE_ID) might not be the right one for the TECHLIN VIEW definition. In this case, the data source DATASOURCE_ID might have the wrong relations between the Tableau Data Attribute asset and the Column asset.

Group name	Name	Description
Missing content	Datasource not found	Raised during the processing of custom SQL queries. Contains the line: Custom sql query with id QUERY_ID is skipped Indicates that query with QUERY_ID contains columns that are not referenced by any data source fields, so the data source can't be assigned to the query. <pre>{ "data": { "tablesConnection": { "nodes": [{ "__typename": "CustomSQLTable", "id": "QUERY_ID", "columnsConnection": { "nodes": [{ "id": "COLUMN_ID", ... }] } }] } } }</pre>

Group name	Name	Description
Missing content	Query parsing error	Raised during the processing of custom SQL queries. Contains the line: Error parsing query with id QUERY_ID, error: ERROR Indicates that there is an issue when deriving column names from a query for a custom SQL with QUERY_ID. <pre> { "data": { "tablesConnection": { "nodes": [{ "__typename": "CustomSQLTable", "id": "QUERY_ID", "query": "QUERY" }] } } } </pre> Custom SQL is still processed as TECHLIN VIEW with no columns.
Configuration	Invalid Collibra system names	Raised during the processing of the collibraSystemNames section in the <source ID> configuration file. Contains the lines: • Collibra system name not found for database with hostname "DB_HOSTNAME" • Collibra system name not found for file with path "FILE_PATH" • Collibra system name not found for connector with url "CONNECTION_URL" • Collibra system name not found for cloud file with name "CLOUD FILE PATH"
Configuration	Invalid hostname mapping	Raised during the processing of the hostnameMapping section the <source ID> configuration file. Contains the line: Collibra system name not found for database "DB_NAME" host "HOST_NAME" and schema "SCHEMA_NAME"

When you ingest Power BI metadata in Data Catalog, Collibra Data Lineage automatically creates a technical lineage for assets of the following types:

- Power BI Report
- Power BI Table
- Power BI Column

To view the technical lineage, go to the asset page of any asset of these types, and then click the **Technical Lineage** tab.

The screenshot shows the 'FullName' asset page in the 'Power BI Demo Catalog'. The asset is a 'Power BI Column' with a status of 'Candidate'. The left sidebar contains navigation tabs: Details, Tags, Comments, Diagram, Pictures, Technical Lineage (highlighted with a green box), Responsibilities, References, and History. The main content area is titled 'Technical Data Type' and shows 'String' as the data type. Below this, it states 'is target of Column' and displays a table with one row: 'FullName' in the 'Name' column and 'Consumption' in the 'Domain' column. Further down, it states 'is part of Data Entity' and displays a table with one row: 'CustomerSalesReporting' in the 'Name' column and 'Power BI Demo Catalog' in the 'Domain' column.

Name	Domain
FullName	Consumption

Name	Domain
CustomerSalesReporting	Power BI Demo Catalog

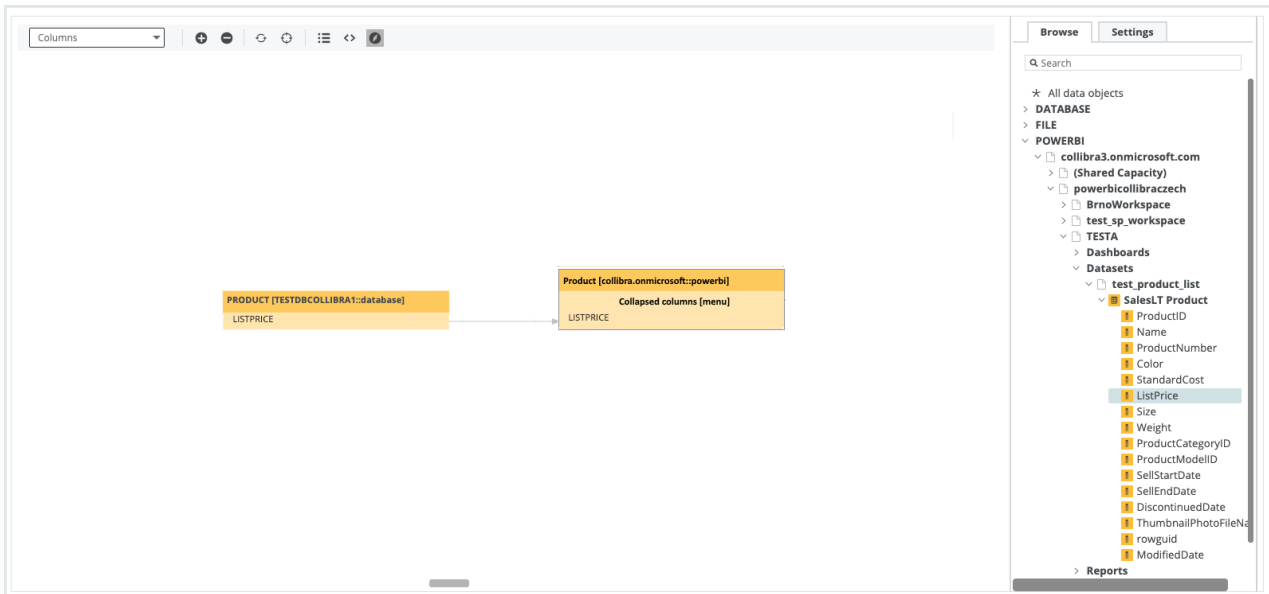
Note If you ingest Power BI for the first time or if you change your geolocation or cloud provider, you have to restart the DGC service before you can see your technical lineage.

Technical lineage graph

The technical lineage graph shows relations of the type "Data Element targets / sources Data Element" between BI assets and other data objects in the data flow, for example Column assets or Power BI Column assets. These relations are created during the Power BI ingestion process as a result of automatic stitching.

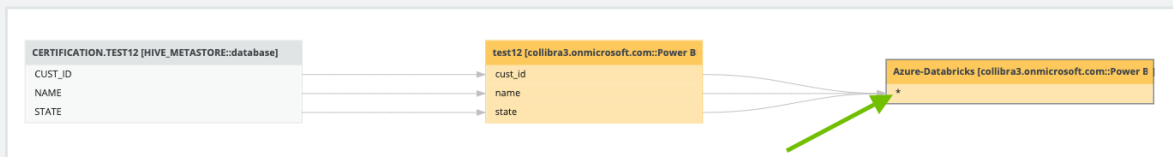
Example

The following technical lineage shows the relation of the type "Data Element targets / sources Data Element" between the Column asset *LISTPRICE* and the Power BI Column asset *ListPrice*.



Note

When harvesting Power BI, report attributes are not returned by the API. Therefore, for a given report, Colibra Data Lineage creates a dummy report attribute. This dummy report attribute is identified in the technical lineage by an asterisk (*), as shown in the following example image. Links are drawn from all data attributes in the data set that were used to create the report, to the dummy report attribute.



Tip Does your database or schema have the name "Default" in the technical lineage graph? This is the case if you use a [Power Query M function](#) that doesn't have the schema or database name specified, or if Power BI hasn't returned the database or schema name. In this case, you can configure database and schema mapping in your <source ID> configuration file, to provide the name of the database or schema. This allows you to achieve stitching and view the lineage you need. For more information, go to [Broken stitching and possible solutions](#).

Sources tab page

The Sources tab page shows the transformation details that were analyzed and processed on the [Collibra Data Lineage service instances](#) and the results of this analysis. The success rate of the analysis indicates how complete the technical lineage is.

Important The Collibra Data Lineage server can process most, but not all, complex Power BI metadata. This means that the success rate of a Power BI ingestion can be very high, but almost never 100%.

Example

The following image shows that you have created a technical lineage for four data sources. Power BI has a success rate of 83%. When you use the transformation logs to investigate the errors, you see that the Collibra Data Lineage service instance couldn't process some elements of the Power BI metadata, for example because they are not supported or there is an issue in the [configuration file](#) or the [Power BI setup](#).

Sources					
Stitching					
Selection	Source ID	Scanner type	Success rate	Done	Pa
☐	Powerbi	POWERBI	83 %	15	3
☐	PowercenterSQLServer	INFA	100 %	7	0
☐	PostgreSQLCloud	SQL	100 %	7	0
☐	SAPHana	SQL	99 %	101	0
All transformations Full-text search Q Filter by All ▾					
ID	Name		Status code	Status description	
0	Source Code 134		DONE	None	
1	Source Code 135		DONE	None	
2	Source Code 136		DONE	None	
3	Source Code 137		DONE	None	
4	Source Code 138		DONE	None	
5	Source Code 139		DONE	None	
6	Source Code 140		DONE	None	
7	sap.hana.db/GET_TARGET_DATA		PARSING_ERROR	sap.hana.db/GET_TARGET_DATA: Unsupported calculation view type SCRIPT_BASED	
8	Power BI		DONE	None	
9	Power BI Demo		ANALYZE_ERROR	Workspace "Power BI Demo": dataset information not retrieved missing for 2 dataset(s) - The request returned an error: (404) Not Found	

Note

With the implementation of SQL extension v2 in Collibra 2024.01.1, queries and their relevant assets are now combined into a single line in the transformations table. Because of this, the source code processed count in the Done column is reduced. The lineage itself is unchanged.

Sources		Stitching					
Selection	Source ID	Scanner type	Success rate	Done	Parsing Error	Analyze Error	Last sync time
☐	ssrs	SSRS	100 %	32	0	0	2023-12-04 09:00:41 UTC
☒	ssrs	SQL	94 %	16	1	0	2023-12-04 09:00:41 UTC
☐	looker	SQL	98 %	247	3	0	2023-12-04 08:47:37 UTC
☐	tableauOn	SQL	100 %	49	0	0	2023-12-04 08:47:22 UTC
☐	tableauOn	TABLEAU	100 %	922	0	0	2023-12-04 08:47:17 UTC
☐	powerbi2	SQL	100 %	1	0	0	2023-12-04 08:47:16 UTC

ssrs

Full-text search

Filter by

All

Export Selected Transformations

ID	Name	Status code	Status description	Group name
88138	None	DONE	Analysis succeeded, queries 3	None
1 {"id":"00000000-0000-0000-0000-100000000030","name":"SSRS Table"} 2 {"id":"00000000-0000-0000-0000-100000000029","name":"SSRS Column"} 3 {"id":"00000000-0000-0000-0000-000000007047","role":"contains","coRole":"is part of","sourceAssetTypeName":"SSRS Table","targetAssetTypeName":"SSRS Column"}				
88141	Source Code 1	DONE	Analysis succeeded, queries 1	None
88142	Source Code 3	DONE	Analysis succeeded, queries 1	None
88143	Source Code 5	DONE	Analysis succeeded, queries 1	None
88144	Source Code 7	DONE	Analysis succeeded, queries 1	None
88145	Source Code 9	PARSING_ERROR	Unable to proceed with syntax parsing at line 2 column 1. Person.testing_procedure_TS ^	None
88146	Source Code 11	DONE	Analysis succeeded, queries 1	None

When you ingest MicroStrategy metadata in Data Catalog, Collibra Data Lineage automatically creates a technical lineage.

To view the technical lineage, click the **Technical lineage** tab on the asset page of any of the following asset types:

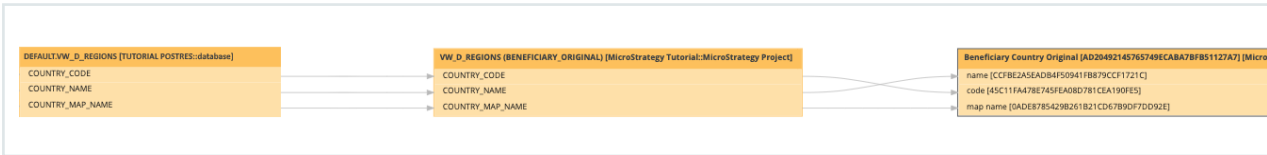
- Table
- Column
- MicroStrategy Data Attribute
- MicroStrategy Report

The **Technical lineage** tab is only shown if you have the Data Catalog global role with the Catalog and Technical lineage [global permissions](#).

Note If you ingest MicroStrategy for the first time or if you change your geolocation or cloud provider, you have to restart the DGC service before you can see the technical lineage.

Technical lineage graph

The technical lineage graph shows relations of the type "Data Element targets / sources Data Element" between BI assets and other data objects in the data flow, for example Column assets or MicroStrategy Data Attribute assets. These relations are created during the MicroStrategy ingestion process as a result of automatic stitching.



MicroStrategy API limitations

Limitation	Details
Report attributes	When harvesting MicroStrategy, report attributes are not returned by the API. Therefore, for a given report, Collibra Data Lineage creates a dummy report attribute. This dummy report attribute is identified in the technical lineage by an asterisk (*), as shown in the following example image. Links are drawn from all data attributes in the data set that were used to create the report, to the dummy report attribute. The diagram shows a technical lineage graph where data from a source table 'DEFAULTVW_D_REGIONS [TUTORIAL POSTRES:database]' flows through a view 'VW_D_REGIONS (BENEFICIARY_ORIGINAL) [MicroStrategy Tutorial:MicroStrategy Project]' to a data attribute 'Beneficiary Country Original [AD20492145765749ECABA78F851127A7] [MicroStrategy Tutorial:MicroStrategy Project]'. This data attribute then flows into a final data attribute 'Beneficiary Country Original [AD20492145765749ECABA78F851127A7] [MicroStrategy Tutorial:MicroStrategy Project]' which has attributes 'name [CCFBE25EADB4F50941FB879CCF1721C]', 'code [45C11FA478E745FEA08D781CEA190FE5]', and 'map name [0ADE8785429B261821CD67B9DF7DD92E]'. The final data attribute is identified by an asterisk (*) in the technical lineage, indicating it is a dummy report attribute.

Limitation	Details
Reports	The integration supports all common MicroStrategy reports. However, due to limitations with the MicroStrategy APIs: • The following report subtypes are not supported: data mart, bulk export, incremental refresh, and transaction. • The following report extended types are not supported: MDX, Query Builder, Freeform XQuery, and Data Import. Any log messages referring to skipped end points are due to the MicroStrategy API limitations. They are not due to an error or lack of functionality on the part of Collibra Data Lineage.
Freeform SQL	The integration supports MicroStrategy Freeform SQL. However, due to limitations with the MicroStrategy APIs, it is only supported for reports, not cubes. Note Freeform SQL is supported for reports (not cubes or dossiers) if you have MicroStrategy update10 or newer, or MicroStrategy ONE.

UUIDs in the MicroStrategy technical lineage

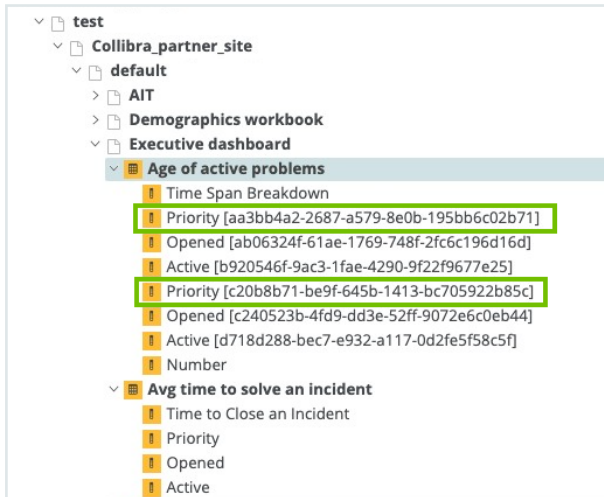
Collibra Data Lineage uses unique full names to create a technical lineage and stitch objects within the technical lineage. Full names in Collibra are constructed in accordance with the hierarchy of data objects in MicroStrategy, for example:

- Server > Project > Folder > Report > Data Entity > Data Attribute
- Server > Project > Folder > Dossier > Data Entity > Data Attribute
- Server > Project > Folder > Document > Data Entity > Data Attribute

In Collibra, every node in this hierarchy must have a unique name. However, in MicroStrategy, the names of data objects do not have to be unique. As such, if MicroStrategy data objects in a technical lineage hierarchy have the same full name,

Collibra Data Lineage adds the UUIDs of the corresponding assets to the names in the technical lineage, to maintain uniqueness.

In the following example image, the names of the assets Priority, Opened and Active in the technical lineage have been appended with their UUIDs.



Note

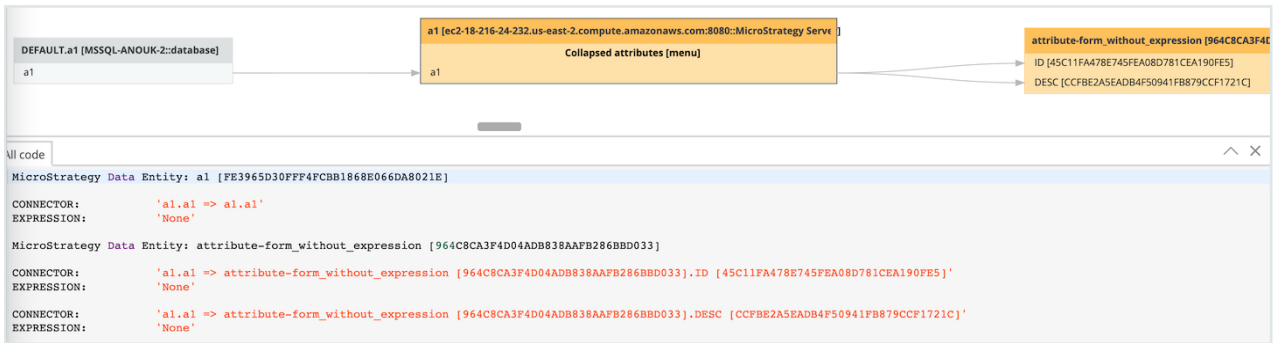
- UUIDs are not added to the names of the assets themselves; they are only added to the names of the data objects in the technical lineage.
- The UUID is always part of the full name of an asset, regardless of whether or not it is a duplicate.

To keep Collibra Data Lineage from adding UUIDs to the names of the data objects in a technical lineage, ensure that the names of all data objects in MicroStrategy are unique.

Sources tab page

The [Sources tab page](#) shows the expressions that the Collibra Data Lineage service analyzed and processed, and the results of the analysis. It also shows the TECHLIN VIEW query definitions, based on custom SQL queries.

Note MicroStrategy uses the term "expressions" instead of "transformations".



Source code is provided for the following MicroStrategy asset types:

- MicroStrategy Document
- MicroStrategy Dossier
- MicroStrategy Report
- MicroStrategy Data Entity

The success rate of the analysis indicates how complete the technical lineage is.

For example, the following image shows that you have created a technical lineage for two data sources. SAP HANA has a success rate of 83%. When you use the transformation logs to investigate the errors, you see that the Collibra Data Lineage service instance couldn't process some elements of the SAP HANA metadata, for example because they are not supported or because there is an issue in the [configuration file](#).

SourcesStitching						
Selection	Source ID	Scanner type	Success rate	Done	Parsing Error	Analyze Error
☐	PostgreSQLCloud	SQL	100 %	7	0	0
☐	SAPHana	SQL	83 %	101	0	1
All transformationsFull-text search Filter byAll ▾						
ID	Name	Status code	Status description		Group name	
0	Source Code 134	DONE	None		None	
1	Source Code 135	DONE	None		None	
2	Source Code 136	DONE	None		None	
3	Source Code 137	DONE	None		None	
4	Source Code 138	DONE	None		None	
5	Source Code 139	DONE	None		None	
6	Source Code 140	DONE	None		None	
7	sap.hana.db/GET_TARGET_DATA	PARSING_ERROR	sap.hana.db/GET_TARGET_DATA Unsupported calculation view type: SCRIPT_BASED		None	

Note

With the implementation of SQL extension v2 in Collibra 2024.01.1, queries and their relevant assets are now combined into a single line in the transformations table. Because of this, the source code processed count in the Done column is reduced. The lineage itself is unchanged.

Sources		Stitching					
Selection	Source ID	Scanner type	Success rate	Done	Parsing Error	Analyze Error	Last sync time
☐	ssrs	SSRS	100 %	32	0	0	2023-12-04 09:00:41 UTC
☒	ssrs	SQL	94 %	16	1	0	2023-12-04 09:00:41 UTC
☐	looker	SQL	98 %	247	3	0	2023-12-04 08:47:37 UTC
☐	tableauOn	SQL	100 %	49	0	0	2023-12-04 08:47:22 UTC
☐	tableauOn	TABLEAU	100 %	922	0	0	2023-12-04 08:47:17 UTC
☐	powerbi2	SQL	100 %	1	0	0	2023-12-04 08:47:16 UTC

ssrs

Full-text search

Filter by

All

Export Selected Transformations

ID	Name	Status code	Status description	Group name
88138	None	DONE	Analysis succeeded, queries 3	None
1 {"id":"00000000-0000-0000-0000-100000000030","name":"SSRS Table"} 2 {"id":"00000000-0000-0000-0000-100000000029","name":"SSRS Column"} 3 {"id":"00000000-0000-0000-0000-000000007047","role":"contains","coRole":"is part of","sourceAssetTypeName":"SSRS Table","targetAssetTypeName":"SSRS Column"}				
88141	Source Code 1	DONE	Analysis succeeded, queries 1	None
88142	Source Code 3	DONE	Analysis succeeded, queries 1	None
88143	Source Code 5	DONE	Analysis succeeded, queries 1	None
88144	Source Code 7	DONE	Analysis succeeded, queries 1	None
88145	Source Code 9	PARSING_ERROR	Unable to proceed with syntax parsing at line 2 column 1. Person.testing_procedure_TS ^	None
88146	Source Code 11	DONE	Analysis succeeded, queries 1	None

When you ingest Looker metadata, you automatically create a technical lineage for Looker Look assets. If you have the right permissions to view the technical lineage, you can go to a Looker Look asset page and click the **Technical lineage** tab, which allows you to access the technical lineage.

Business Analysts Community

Looker Catalog

LKLK

Average Lifetime Revenue

Looker Look

Implemented

0

0

<

URL

<https://colibra.looker.com/looks/>

Visits count

0

Favorites count

0

Document creation date

11/17/2020

Document modification date

11/17/2020

used in Report

Name ↑	Domain	Definition	Description
Web Analytics and Product Us...	Looker Catalog		

Details

Tags

Comments

Diagram

Pictures

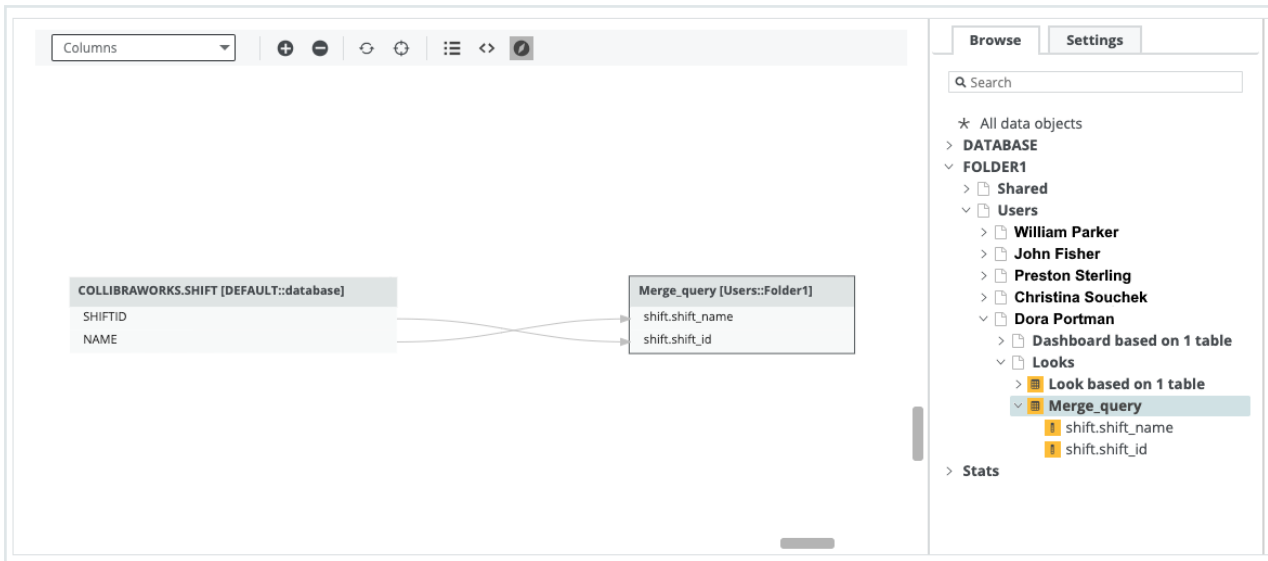
Technical Lineage

Responsibilities

References

History

Files



When you ingest SQL Server Reporting Services (SSRS) and Power BI Report Server (PBRs) metadata in Data Catalog, you automatically create a technical lineage for SSRS Column assets. Each SSRS Column asset page has a Technical lineage tab page that shows the technical lineage of that asset Column asset.

We cannot access PBRs lineage information. As a result, you can only create a technical lineage for SSRS Column assets.

Note If you ingest SSRS and PBRs for the first time, or if you change your geolocation or cloud provider, you might have to restart the DGC service before you can see your technical lineage.

Business Analysts Community
MyPBR5

RSCL

food_code
SSRS Column ⓘ

Candidate
0
0

Add characteristic <

Details
Tags
Comments

Diagram
Pictures

Technical Lineage

Responsibilities
References
History
Files

sources Data Element ⓘ
No data available

targets Data Element ⓘ

Name ↑	Domain	Description
food_code	MyPBR5	

is part of BI Data Model

Name ↑	Domain	Description
ds_redshift_test	MyPBR5	

Technical lineage graph

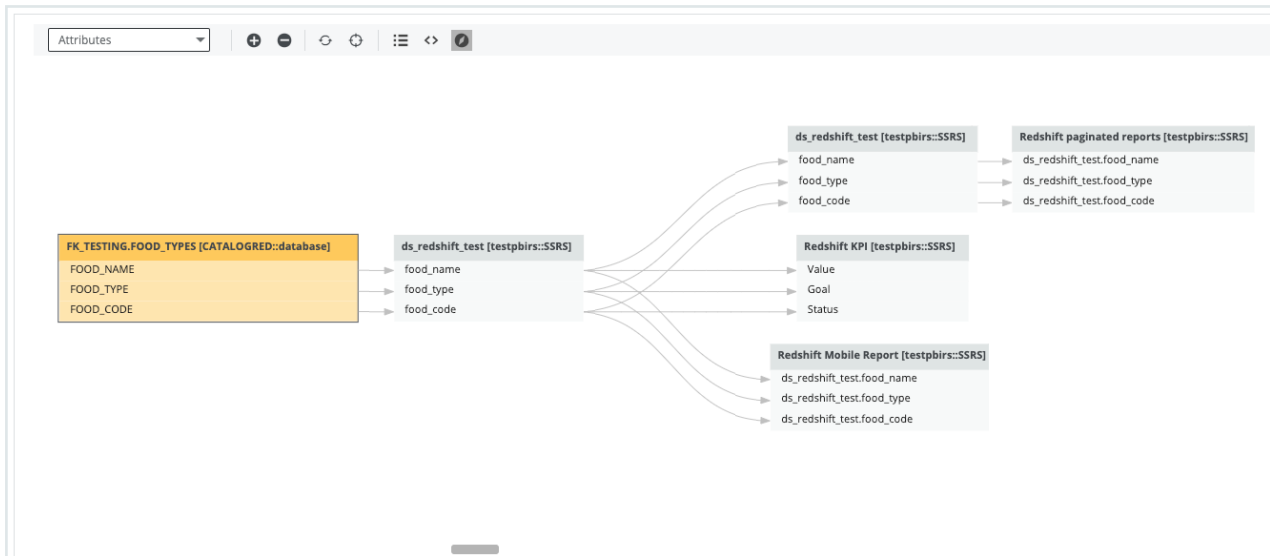
The technical lineage graph shows relations of the type "Column is source for / is target of Data Attribute" between BI assets and other data objects in the data flow, for example Column assets or Power BI Column assets. These relations are created during the ingestion process as a result of automatic stitching.

For more information about the technical lineage, see the [Collibra Data Lineage](#) section in the documentation.

Example

The following technical lineage shows the relation of the type "Data Element sources / targets Data Element" between the Column assets *FOOD_NAME*, *FOOD_TYPE* and

FOOD_CODE and the SSRS Column assets *food_name*, *food_type* and *food_code*.



Sources tab page

The Sources tab page shows the transformation details that the Collibra Data Lineage service analyzed and processed and the results of this analysis. The success rate of the analysis indicates how complete the technical lineage is.

Important The Collibra Data Lineage service can process most, but not all complex metadata. This means that the success rate of an ingestion job can be very high, but might not be 100%.

Note

With the implementation of SQL extension v2 in Collibra 2024.01.1, queries and their relevant assets are now combined into a single line in the transformations table. Because of this, the source code processed count in the Done column is reduced. The lineage itself is unchanged.

Sources		Stitching					
Selection	Source ID	Scanner type	Success rate	Done	Parsing Error	Analyze Error	Last sync time
☐	ssrs	SSRS	100 %	32	0	0	2023-12-04 09:00:41 UTC
☒	ssrs	SQL	94 %	16	1	0	2023-12-04 09:00:41 UTC
☐	looker	SQL	98 %	247	3	0	2023-12-04 08:47:37 UTC
☐	tableauOn	SQL	100 %	49	0	0	2023-12-04 08:47:22 UTC
☐	tableauOn	TABLEAU	100 %	922	0	0	2023-12-04 08:47:17 UTC
☐	powerbi2	SQL	100 %	1	0	0	2023-12-04 08:47:16 UTC

ssrs

Full-text search

Filter by

All

Export Selected Transformations

ID	Name	Status code	Status description	Group name
88138	None	DONE	Analysis succeeded, queries 3	None
1 {"id":"00000000-0000-0000-0000-100000000038","name":"SSRS Table"} 2 {"id":"00000000-0000-0000-0000-100000000029","name":"SSRS Column"} 3 {"id":"00000000-0000-0000-0000-000000007047","role":"contains","coRole":"is part of","sourceAssetTypeName":"SSRS Table","targetAssetTypeName":"SSRS Column"}				
88141	Source Code 1	DONE	Analysis succeeded, queries 1	None
88142	Source Code 3	DONE	Analysis succeeded, queries 1	None
88143	Source Code 5	DONE	Analysis succeeded, queries 1	None
88144	Source Code 7	DONE	Analysis succeeded, queries 1	None
88145	Source Code 9	PARSING_ERROR	Unable to proceed with syntax parsing at line 2 column 1. Person.testing_procedure_TS ^	None
88146	Source Code 11	DONE	Analysis succeeded, queries 1	None

Providing ODBC database names in Power BI

You can create a technical lineage for ODBC data sources in Power BI. However, ODBC database names often can't be determined. When a database name can't be determined, it's given a substitute name, which is the ODBC connection string.

This substitute name can be seen in the technical lineage, but it is merely a placeholder that doesn't carry any meaning if you're trying to identify the database it represents in the technical lineage. A bigger problem is that if you want to stitch the ODBC database to assets in Data Catalog, the substitute name won't match with any ingested databases, so stitching won't work.

To ensure that the true database names appear in the technical lineage, and to ensure successful stitching, you can use a [Power BI <source ID> configuration file](#) to provide the true system names of the ODBC databases in Power BI.

Tip The name "<source ID>" refers to the value of the `sourceId` property in the [configuration file](#). If, for example, the value of the `sourceId` property in the lineage harvester configuration file is `power-bi-source-1`, then the name of your <source ID> configuration file should be *power-bi-source-1.conf*.

Example of the <source ID> configuration file

For each ODBC database in Power BI, add the following content to the JSON file:

```
{
  "found_dbname=DSN_MYDATABASE;found_hostname=ODBC": {
    "dbname": "DB001",
    "schema": "MYSCHEMA",
    "dialect": "oracle",
    "collibraSystemName": "oracle-system-name"
  }
}
```

Property	Description
found_ dbname=<substitute database name>;found_ hostname=<server name>	<code>found_dbname</code> is the substitute database name. You need to convert it to uppercase and replace every non-alphanumeric character by an underscore (_). In this example, the substitute name is “dsn=MYDATABASE”, so you should use “DSN_MYDATABASE”. Note The substitute name is the ODBC connection string, which can be lengthy when it includes the driver and parameters in full. <code>found_hostname</code> should be “ODBC”, but you can also use an asterisk (*).
dbname	The true system name of the ODBC database in Power BI.
schema	The name of the default schema of the ODBC database in Power BI. If no schema is specified and the lineage harvester fails to find a specific schema, it uses the default schema.

Property	Description
dialect	The dialect of the ODBC connection. The dialect must be one of the supported SQL dialects. If no dialect is specified, "mssql" is used, by default. Tip You can enter one of the following values: • <i>azure</i>, for an Azure SQL Server data source. • <i>bigquery</i>, for a Google BigQuery data source. • <i>mssql</i>, for a Microsoft SQL Server data source. • <i>oracle</i>, for an Oracle data source. • <i>redshift</i>, for an Amazon Redshift data source. • <i>snowflake</i>, for a Snowflake data source. • <i>sybase</i>, for a Sybase data source.
collibraSystemName	The system or server name of a database. Important Because you are using a <source ID> configuration file only for the purpose of providing the true system name of an ODBC database in Power BI, you are not required to: • Set the <code>useCollibraSystemName</code> property in the lineage harvester configuration file to <code>true</code>. • Specify a Collibra system name in the <source ID> configuration file. However, if the <code>useCollibraSystemName</code> property is set to <code>true</code> in the lineage harvester configuration file, then you must specify a Collibra system name in the <source ID> configuration file.

Supported Power Query M functions

Power Query is a data transformation and preparation engine. It uses a scripting language called Power Query M formula language—also known as M—for all transformations.

M is considered a "mashup" language. The Power Query engine filters and combines data from supported data sources. The "mashed up" data is then expressed using M. M is used by Power BI. It is not relevant to other integrations in Collibra.

The Collibra Data Lineage service performs lexical and syntax analysis of M. With regard to syntax analysis, the Collibra Data Lineage service instances currently support the following functions.

Note Not all functions have an impact on the technical lineage, so even though an error is raised for any unsupported functions, it might not mean that your lineage is incomplete. We are working to support the most common Power Query functions. If you have a Power Query function that is not yet supported and it's very important to you, please create an Ideation ticket.

For complete information on these functions, see the [Microsoft documentation on accessing data functions](#).

- **Backend-accessing data functions that impact the lineage diagram**

- AmazonRedshift.Database
- AnalysisServices.Database

Note

- This function is fully supported if no MDX queries are used.
- If MDX queries are used and they resemble SQL, they will be parsed by the SQL parser.
- We don't currently support this function if used with MDX queries that resemble DAX, as the Collibra Data Lineage service instances can't parse such queries.

- AnalysisServices.Databases
- Csv.Document
- Databricks.Contents
- Databricks.Catalogs
- Excel.Workbook
- File.Contents
- GoogleBigQuery.Database
- Odbc.DataSource
- Odbc.Query
- Oracle.Database
- PostgreSQL.Database
- SapHana.Database
- Snowflake.Database

- Sql.Database
- Sql.Databases
- Sybase.Database
- Teradata.Database
- Web.Contents
- **Transformations that impact the lineage diagram**
 - Cube.AddAndExpandDimensionColumn
 - Cube.Transform
 - PowerBI.Dataflows

Note We only support dataflows without parameters that contain the following information:

- workspace ID
- dataflow ID
- entity (Power BI Table) ID

- PowerPlatform.Dataflows

Note We only support dataflows without parameters that contain the following information:

- workspace ID
- dataflow ID
- entity (Power BI Table) ID

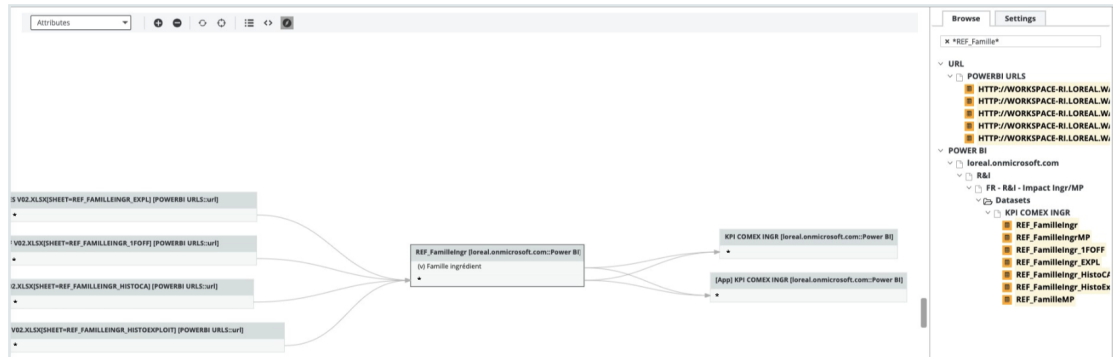
- Replacer.ReplaceText
- Replacer.ReplaceValue
- Table.AddColumn
- Table.AddIndexColumn
- Table.Combine

Additional information

If Collibra Data Lineage can't determine the column names in a source file or database, but the PowerBI column names are known and there is only one source file or database, then corresponding database/file columns are created and technical lineage is preserved. However, if column names can't be determined and there are multiple source files or databases, as is the case when the Table.Combine function is used, then it's not possible to know which source column corresponds to the Power BI column. This results in an error

and the technical lineage is broken.

To resolve this issue, a dummy column with the value “*” is created in the source table and the Power BI table:



This preserves the technical lineage at the table level:



- Table.CombineColumns
- Table.DuplicateColumn
- Table.ExpandTableColumn
- Table.FromList
- Table.FromRecords
- Table.FromRows
- Table.NestedJoin
- Table.PromoteHeaders
- Table.RemoveColumns
- Table.RenameColumns
- Table.ReorderColumns
- Table.ReplaceValue
- Table.SelectColumns
- Table.SplitColumn
- Table.Unpivot
- Table.UnpivotOtherColumns

- Table.TransformColumnNames

Note Only the following parameters are supported: Text.Upper, Text.Lower, and Text.Proper.

- Value.NativeQuery

Note Query parameters are supported. Core parameters are not supported.

- **Transformations that don't impact the lineage diagram**

- Table.AddKey
- Table.AlternateRows
- Table.Buffer
- Table.Distinct
- Table.ExpandListColumn
- Table.FillDown
- Table.FillUp
- Table.FindText
- Table.FirstN
- Table.InsertRows
- Table.IsEmpty
- Table.LastN
- Table.MaxN
- Table.MinN
- Table.Range
- Table.RemoveFirstN
- Table.RemoveLastN
- Table.RemoveMatchingRows
- Table.RemoveRows
- Table.RemoveRowsWithErrors
- Table.Repeat
- Table.ReplaceErrorValues
- Table.ReplaceKeys
- Table.ReplaceMatchingRows
- Table.ReplaceRows
- Table.ReverseRows

- Table.SelectRows
- Table.SelectRowsWithErrors
- Table.Skip
- Table.Sort
- Table.TransformColumns
- Table.TransformColumnTypes
- Table.First
- Table.Last
- Table.Max
- Table.Min
- Table.SingleRow
- **Unsupported transformations**

Note Using unsupported transformations can cause parsing errors.

- Table.FromRecords
- SharePoint.Tables
- Folder.Files
- PowerBI REST API.Navigation
- DB2.Database
- Table.ExpandRecordColumn
- Table.Group

Working with Power Query parameters

Power BI Power Query is a data transformation and data preparation engine. It gets data from your data sources and the Power Query Editor, and performs the extract, transform, and load (ETL) processing of data.

You can use Power Query parameters to store and manage values that can be reused. Parameters give you the flexibility to dynamically change the output of your queries, depending on their values. For complete information on creating and managing parameters, see the [Microsoft documentation](#).

Power BI parameters in technical lineage

Power BI parameters are configured at the dataset level and can be used in reports. When you integrate Power BI, the Power BI APIs return all parameters that are loaded in a report.

Important When you select the **Enable load** option for a parameter, Power BI loads the columns from the parameterized table into its memory. The lineage harvester can then harvest these columns and create the full lineage.

If the **Enable load** option is not selected for a parameter:

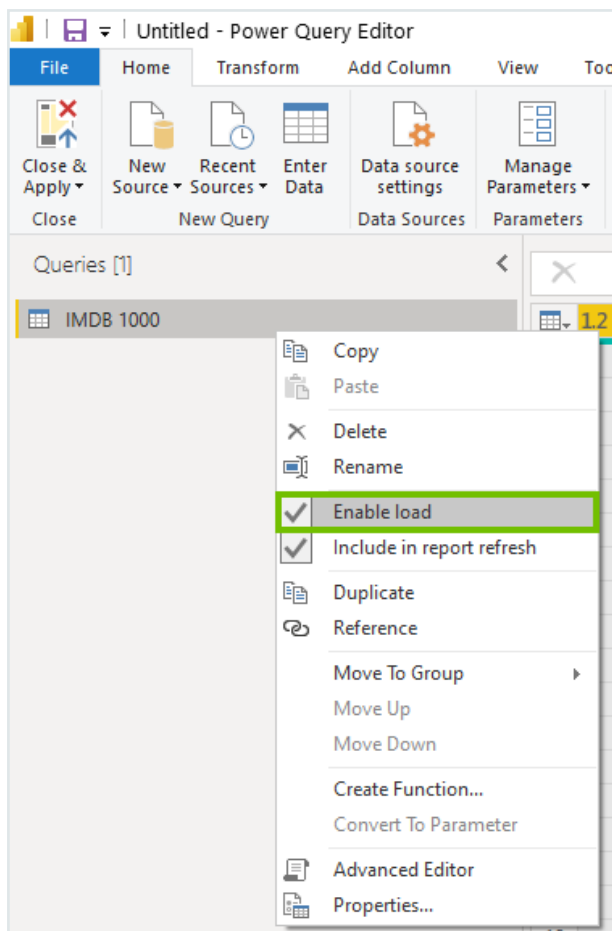
- The Power BI APIs can recognize the parameterized table, but not the columns in the table. In this case, Collibra Data Lineage can only create a table-level lineage; columns cannot be shown.
- If the parameter is used with, for example, the Table.AddColumn function or a similar function, a parsing error will be produced, because the Collibra Data Lineage service instance won't know which column to add.

Parameters of unsupported Power Query M functions are not supported. For the lists of supported and unsupported Power Query M functions, see [Supported Power Query M functions](#). Likewise, global parameters are not supported. Global parameters are parameters that are not specific to a Power Query M function.

Before Collibra Data Lineage introduced support for parameters, if you had a dataset or a report that had parameters, the following error message was shown: "Could not process lineage. Please check if the Power Query expression contains schema, table(s) and column(s)."

Select the Enable load option for a parameter

In the **Home** tab of the Power Query Editor, right-click the parameter, and then select **Enable load**.



Ensure that the **Enable load** option is selected for all parameters.

Warning If you change the **Enable load** setting for a parameter, you must refresh the relevant data set. If the data set is not refreshed, the metadata processing fails due to an "Unknown identifier" analyze error.

Technical lineage viewer

The technical lineage viewer shows the technical lineage and allows you to edit the view. You can access the technical lineage viewer via the Technical lineage tab on Column and Table [asset pages](#) and BI assets of the same level.

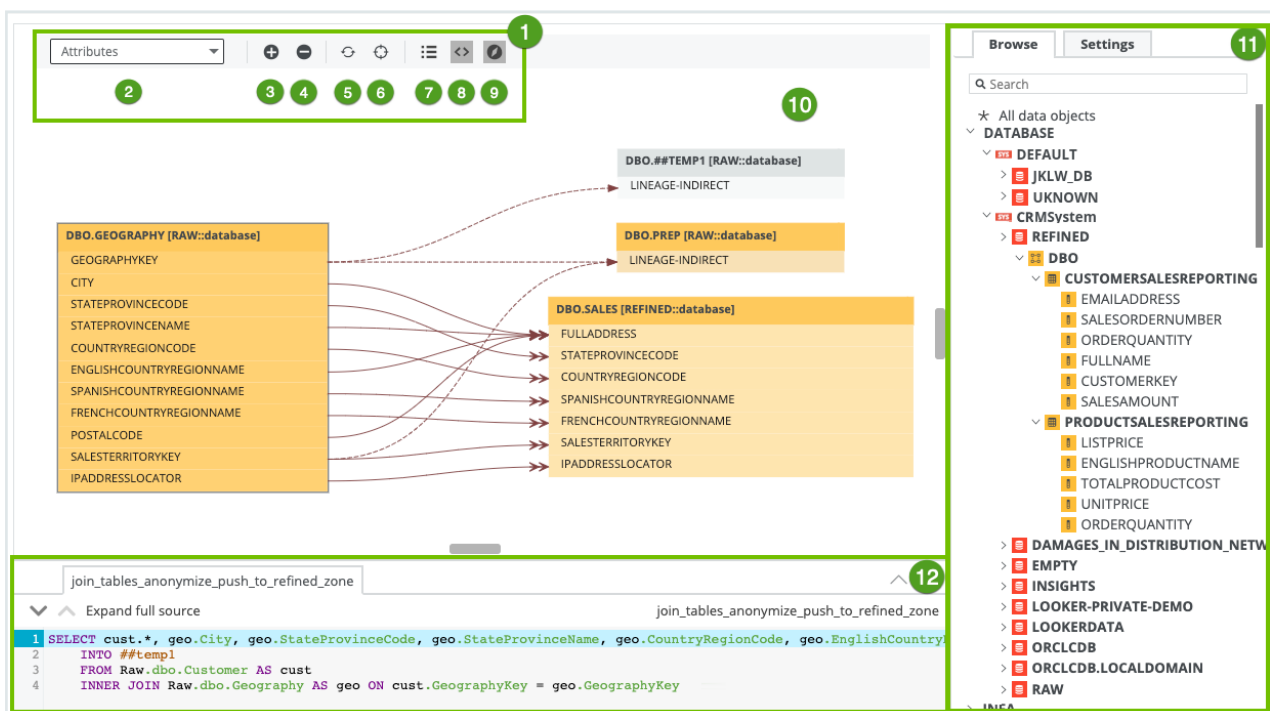
Technical lineage tab

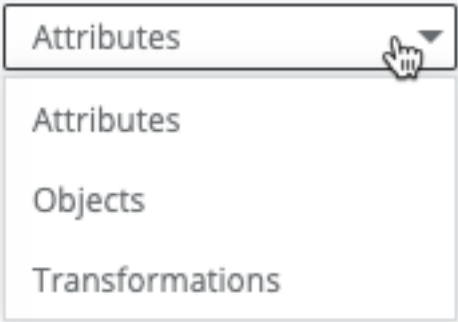
You can only see the Technical lineage tab on a Column or Table asset page when you have the following prerequisites:

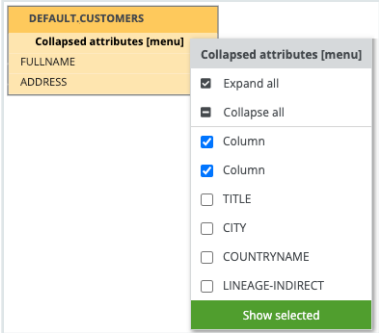
- You have a [global role](#) with the Catalog [global permission](#), for example, Catalog Author.
- You have a global role with the Technical lineage [global permission](#).

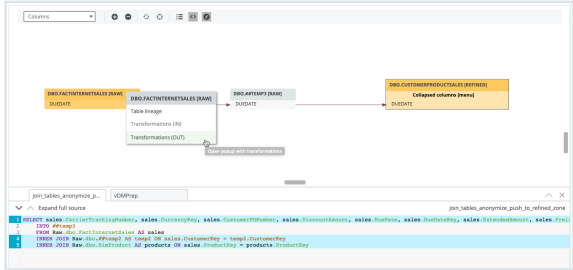
Note View permissions are not enforced in Collibra Data Lineage. This means that anyone with the Technical Lineage global permission can see all of the assets in a technical lineage graph, regardless of their view permissions as determined at the community or domain level.

Technical lineage viewer



No	Name	Description
1	Toolbar	The toolbar to work with technical lineage. The toolbar helps you to edit basic settings that apply to the entire lineage.
2		Drop-down list to determine which details (attributes, objects or transformations) you want to show in the technical lineage graph.
3	+	Button to zoom in on the technical lineage.
4	-	Button to zoom out on the technical lineage.
5	↺	Button to refresh the technical lineage. This discards all the changes that you made to the technical lineage and restores it to the initial state.
6	📍	Button to reposition the technical lineage to the starting position.
7	☰	Button to show or hide the legend panel.
8	<>	Button to show or hide the source code pane.
9	⦿	Button to show or hide the Browse and Settings tab panes.

No	Name	Description
10	Technical lineage graph	The actual visualization of the traceability of the current data object, according to your selection in the Browse tab pane. If you select a specific column in a table with multiple columns, you can click Collapsed columns [menu] to show all other columns, collapse all columns or only show selected columns in the same table.  Tip Data objects that are stitched to assets in Data Catalog have a yellow background. Other data objects that the Collibra Data Lineage collected from your data source, but are not stitched and therefore are not assets in Data Catalog, have a gray background.

No	Name	Description
11	Tab panes	Tab panes that contain useful tools to browse through your technical lineage or determine which content is visualized in the technical lineage.
	Browse tab pane	This pane can be used to search for specific data objects or show statistics on the amount of tables and views in use. More information .
	Settings tab pane	This pane can be used to search for transformation code, edit the visualization of the technical lineage, see the status of the source code , check the stitching results or export your technical lineage to PDF, PNG or CSV. More information .
12	Source code pane	The source code pane shows the source code of specific data objects. It can be used to easily find issues in the data flow. The source code pane is shown when you click  in the toolbar or when you right-click a column or table and click Transformations (IN) or Transformations (OUT) which shows the transformation logic in the source code pane. 

The technical lineage graph

The technical lineage graph consists of nodes and edges. Each node represents a corresponding object in a data source. Each edge shows a relation between nodes.

Nodes and edges in the technical lineage graph show how data flows from source to destination. Understanding the nodes and edges better, enriches your technical lineage experience.

Consider the following visual elements in the technical lineage graph:

- [Relation types](#)
- [Messages](#)
- [Colors](#)
- [Icons](#)
- [Arrows](#)
- [Collapsed attributes menu](#)
- [Right-click menu](#)

Relation types

The technical lineage graph shows relations between columns in the graph. The Collibra Data Lineage creates and shows the following relation type between [stitched](#) assets and other data objects:

Head	Role	Co-role	Tail	ID
Data Element	targets	sources	Data Element	00000000-0000-0000-0000-000000007069

Messages

The technical lineage graph might show different messages to alert you. The following messages are the most common:

Message	Description
No object found, try using a wildcard %	When a data object name was entered in the search field on the Browse tab pane, this message is shown if the data object does not exist or a system name was entered. The following rules apply when you search for a data object: • Use the percent sign (%) wildcard character if needed. • Enter a database, schema, table or column name. • Do not enter a system name.
Nodes count exceeds the limit 350. Edges count exceeds the limit 1,000.	The technical lineage graph exceeds the limit of 350 nodes or 1,000 edges and is too large to display. This happens, for example, if you have a table with many columns and you try to show the technical lineage of all columns in a table in one graph. Note You cannot manually change this limit.
Depth was auto-adjusted to <number>. Graph was too large to display at once.	The technical lineage graph exceeds the edge limit, which results in the automatic adjustment of the flow depth for the graph. The adjusted depth value is determined by the number of the edges that exceed the maximum edge limit. When the flow depth is automatically adjusted to a lower value than the actual graph size, you can find the  icon in the technical lineage graph. To view the truncated lineage, right click the innermost node, and select Table lineage from the menu. The lineage information of the selected table is displayed.
The current asset doesn't have a technical lineage yet.	This message is shown if you didn't create a technical lineage for the data source of the asset. Use the Browse tab pane to navigate through the data object for which a technical lineage graph is available.

Message	Description
Technical lineage cannot be shown.	The technical lineage graph cannot be shown, because there are too many data objects. This happens, for example, when you created a technical lineage for multiple data source and you click All data objects in the Browse tab pane. Use the Browse tab pane to view specific parts of the technical lineage graph or click the suggested data objects to see their graph.

Colors

The technical lineage graph shows different colors to indicate which [data objects](#) are stitched to assets in Data Catalog and which are not.

Background colors

The background color of a node indicates whether or not the data object was stitched to an asset in Data Catalog, and whether something went wrong.

A node has one of three background colors:

Color	Description
Yellow	Data objects from your data source that are stitched to assets in Data Catalog
Gray	Data objects, for example temporary tables and columns, that Collibra Data Lineage collects from your data sources, but are not stitched to assets in Data Catalog. Note Collibra Data Lineage: - Does not support stitching for Looker assets. - Supports stitching for MicroStrategy assets only if you use the new integration method, which supports the latest MicroStrategy APIs.

Color	Description
Red	Attributes that are automatically assigned to a data object, because of missing DDL statements. If you want to remove objects with a red background, change the statements and rerun the lineage harvester or synchronize the technical lineage again if you use technical lineage via Edge.

Since a technical lineage shows how data flows from source to destination, it is possible to see a lineage graph with both yellow, red and gray nodes.

Example The following technical lineage graph shows two nodes with a gray background and three nodes with a yellow background. Node 1 and 4 contain data objects that are not stitched to assets in Data Catalog while nodes 2, 3 and 5 contain existing assets in Data Catalog that were stitched to the corresponding data objects when you created the technical lineage.

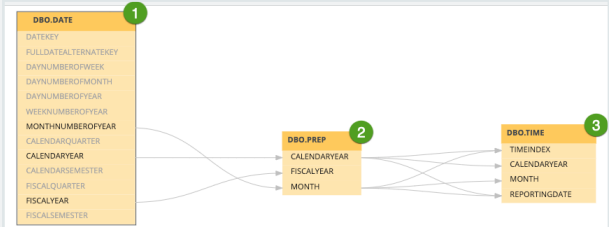
Font colors

The font color of data objects in the technical lineage graph indicates whether or not there is a relation between this data object and one or more other data objects.

A node has one of two font colors:

Color	Description
Black	At least one direct or indirect relation exists between the data object and another. Tip When a column flows from one table to another, the lineage reflects the direct dependency between the column in the source table and the column in the target table. This is considered a direct lineage. An indirect lineage, on the other hand, shows indirect dependencies. For example, if a JOIN clause is used in a query, the columns in the resulting view are generated by the JOIN clause; in other words, by an indirect dependency, not an actual flow of data.
Gray	No relation exists between the data object and another.

Example The following technical lineage graph shows three nodes. The node 1 contains data objects that have no incoming or outgoing edges to other data objects in the technical lineage. Nodes 2 and 3 only contain data objects that have a relation to other data objects in the technical lineage.

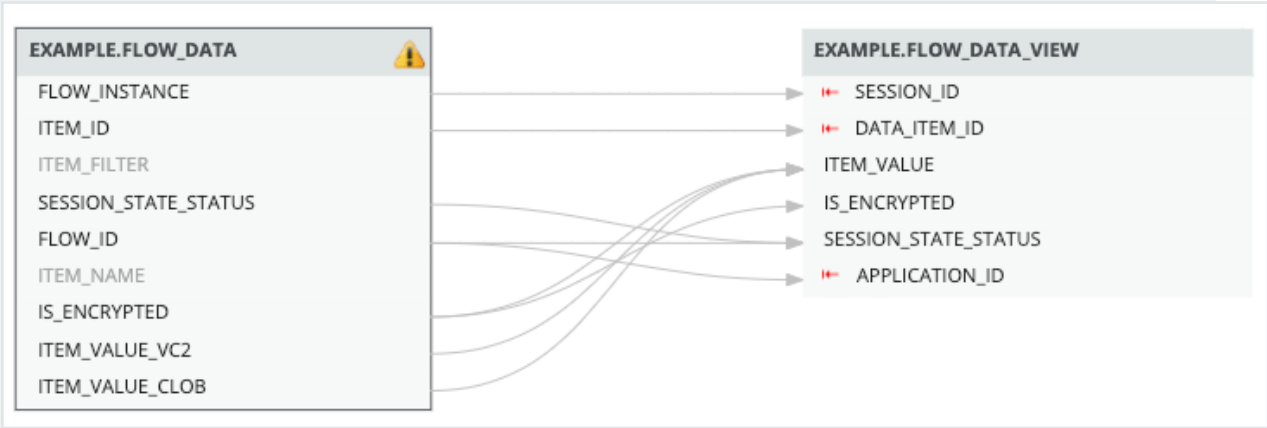


Icons

Collibra uses various icons in the technical lineage graph.

Icon	Description
	The name of a table was found by the full-text search in the source code on which the analysis failed. Consequently, the lineage flow of the table is probably incomplete. If you click Show failed SQLs on the right click menu of the table, the failed SQL queries appear in the source code pane at the bottom of the page.
	The lineage is cyclic, for example $A \rightarrow B \rightarrow C \rightarrow A$. It only appears if you enabled the only ending points option in the Settings tab pane.
	A relation for the data objects exists, but it isn't shown, for example because you set the technical lineage flow depth to a lower value than the actual graph size.

Example The following Technical lineage graph shows two nodes. The first node has an icon to indicate that the lineage flow you currently see is probably incomplete. The second node has three data objects that have a relation to other data objects, but the edges that represent that relation are not shown.



Arrows

Arrows are incoming or outgoing edges that show how the data flows from source to destination. They represent relations of the type "Data Element sources / targets Data Element".

There are two ways in which an arrow can be shown:

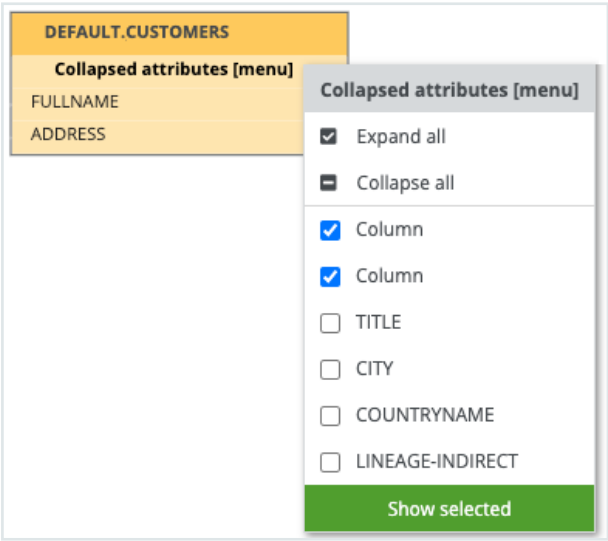
Arrow type	Description
Single	Shows the full lineage without skipping certain data objects.
Double	Shows that there are hidden data objects in the technical lineage graph. This happens when only the endpoints of the technical lineage flow are shown.

Example The following Technical lineage graph shows three nodes. Edges with double arrows are shown between node 1 and 3. These edges indicate that there are other nodes between these nodes in the full technical lineage flow. Node 2 has outgoing edges with single arrows. These edges indicate that there is a direct relation between node 2 and 3.



Collapsed attributes menu

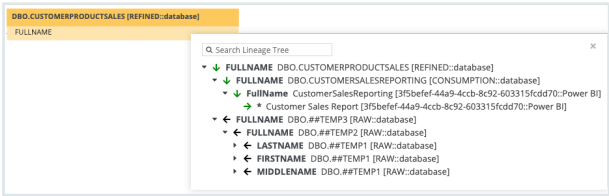
If you select a specific column in a table with multiple columns, you can click **Collapsed attributes [menu]** to show all columns, collapse all columns or only show selected columns in the same table.

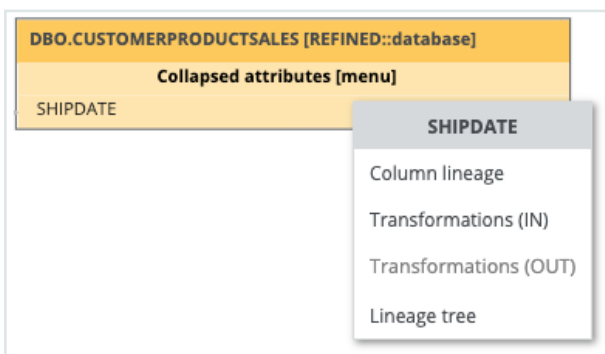


Right-click menu

If you right-click a node, you can perform several specific actions on that node.

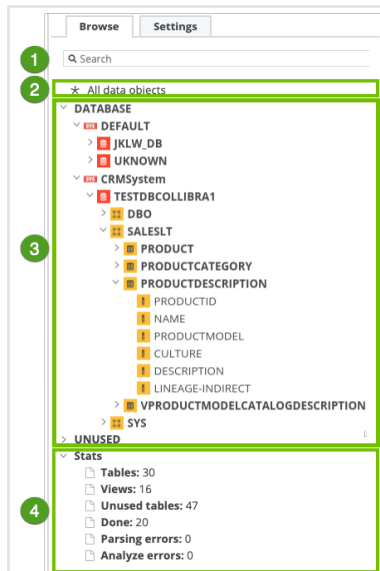
Functionality	Description
Column/Table lineage	Switch to the technical lineage graph of the selected column or table.

Functionality	Description
Transformation (IN)	Show the transformation logic of the incoming source code fragments in the source code pane .
Transformation (OUT)	Show the transformation logic of the outgoing source code fragments in the source code pane .
Lineage tree	Show an alternative way to view the flow of data objects, called the lineage tree. The lineage tree is particularly useful if there are many nodes in a lineage. It enables you to see the entire lineage in one pop-up, which means you no longer have to scroll through the technical lineage graph to see the full lineage. The lineage tree uses arrows to visualize the traceability of data objects: • Green arrows represent outgoing edges. • Black arrows represent incoming edges. 
Custom features	When the lineage flow of the table is incomplete or there is an issue in the source code of a data object, the right-click menu shows the Show failed SQLs option. If you click this option, the source code pane opens and shows the SQL queries that failed.



Technical lineage Browse tab pane

The **Browse** tab pane allows you to navigate to and search for a specific **data object** within the technical lineage tree.



No	Name	Description
1	Search	A search field that you can use to find a specific data object. Enter the name of a database, schema, table, or column. Searching for a system name is not supported. When you hover over the search field, a tool tip is displayed, which shows the available search methods: • To search for a schema, table, or column with a specific name, enter <code><nodetype> : <name></code>, where <code><nodetype></code> is the node type, which can be schema, table, or column, and <code><name></code> is the respective schema name, table name, or column name. Searching for a specific system or database is not supported. • For hierarchical searches, enter two or more combinations of <code><nodetype> : <name></code> separated by a space. For example, you can search by using the following combinations: ◦ <code>database:<dbname> schema:<schemaname></code> ◦ <code>database:<dbname> table:<tablename></code> ◦ <code>database:<dbname> schema:<schemaname> table:<tablename> column:<columnname></code> For a detailed list of node types that you can search in your technical lineage, see the tool tip. • You can use abbreviations for search terms, for example <code>s</code> for schema or <code>proc</code> and <code>p</code> for procedure. For details, see the tool tip. • You can use <code>%</code> as a wildcard.
2	All data objects	A link to the complete technical lineage, showing all data objects in your data sources.

No	Name	Description
3	Navigation tree	A navigation tree in which you can search for specific data objects and visualize them in your technical lineage. The data objects are grouped by node type and have the following structure: system (if applicable) > database > schema > table > column. Note The list of data objects contains all systems, databases, schemas, tables and columns that were collected from the data sources by the lineage harvester. If available, it also shows the technical lineage of BI sources, for example Power BI and Looker. In that case, the structure follows the existing structure in the BI source metadata. Note The UNUSED branch contains data objects that were detected by Collibra Data Lineage, but are not included in any Technical lineage.
4	Stats	Statistics that show which information is or is not visualized in the technical lineage. The statistics contain the following data: • Tables: the amount of tables that are shown in the technical lineage. • Views: the amount of views that are shown in the technical lineage. • Unused tables: the amount of tables in your data source that are not shown in the technical lineage. Tip This metric is hidden when there are no unused tables. • Unused views: the amount of views in your data source that are not shown in the technical lineage. Tip This metric is hidden when there are no unused views. • Done: the amount of queries that were processed successfully. • Parsing errors: the amount of queries with invalid or unidentified syntax. • Analyze errors: the amount of columns that are not linked to a table.

Technical lineage Settings tab pane

The **Settings** tab pane allows you to edit the technical lineage, search for queries and export the technical lineage.

1

2

3

4

5

6

Browse

Settings

Q Filter graph by full-text search on transf. code

Visualization options

☐ Group by parent object

☐ Only ending points

Depth MAX

Dependencies

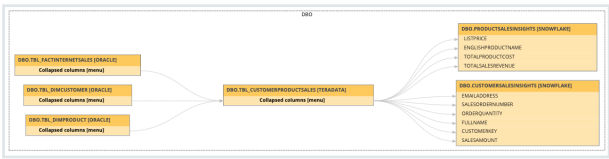
2-way dependencies

☐ Show indirect dependencies

Export

Show status

No	Name	Description
1	Search field	A search field to find specific transformation code in a selected object or attribute. As you type, corresponding object names from the technical lineage appear in a drop-down list. If you press <code>Enter</code> , the technical lineage only shows the parts that contain your search word(s).

No	Name	Description
2	Visualization options	Options to define how you will see the data objects in the technical lineage.
	Group by parent object	Option to group tables and columns together by their hierarchical parent object. Example A schema is the parent object of a table. 
	Only ending points	Option to hide all data objects in the middle of the data flow and only show the ending points of the technical lineage.
	Depth	A slider that determines the maximum flow depth. The relation path length from the first node in the technical lineage graph to any other node is automatically adjusted to the maximum flow depth. If you see  in the technical lineage graph, the flow depth is set to a lower value than the actual graph size.
3	Dependencies	Drop-down to select the dependencies that you want to visualize. You can select one of the following dependencies: • Inbound dependencies only • Outbound dependencies only • 2-way dependencies
4	Show indirect dependencies	Option to include indirect dependencies in a technical lineage. Tip When a column flows from one table to another, the lineage reflects the direct dependency between the column in the source table and the column in the target table. This is considered a direct lineage. By default, Collibra Data Lineage only shows direct lineage. An indirect lineage, on the other hand, shows indirect dependencies. For example, if a JOIN clause is used in a query, the columns in the resulting view are generated by the JOIN clause; in other words, by an indirect dependency, not an actual flow of data.

No	Name	Description
5	Export	Button to export your technical lineage. You can choose among the following export types: • PDF • PNG • CSV • Full CSV • JSON
6	Show status	Button to switch to the Sources tab page, which shows the analysis log files of your data sources and the Stitching tab page, which shows an overview of assets and data objects and shows which are stitched.

Technical lineage Sources tab page

When you create a technical lineage, your data sources are uploaded to the [Collibra Data Lineage service](#) to be analyzed and processed. The Sources tab page shows the transformation details or source code that was analyzed and the results of this analysis.

You can access the Sources tab page by clicking **Show status** on the [Settings tab pane](#).

Note If an analyzed data source has the following result, the data source does not appear in the Sources tab page:

```
Parsing errors: 0
Analysis errors: 0
Done: 0
```


SourcesStitching

6

Selection	Source ID	Scanner type	Success rate	Done	Parsing Error	Analyze Error	Last sync time
☐	bigquery-connectivity-demo-images-edge	SQL	100 %	6	0	0	2023-02-08 15:09:02 UTC
☐	Collibra	SNOWFLAKE_OBJECT_DEPENDENCIES	100 %	16	0	0	2023-05-16 15:11:49 UTC
☐	Collibra	SNOWFLAKE_ACCESS_HISTORY	100 %	2	0	0	2023-05-16 15:11:49 UTC
☐	datastage	SQL	100 %	6	0	0	2023-02-08 15:09:15 UTC
☐	datastage	DATASTAGE	77 %	393	69	48	2023-02-08 15:09:13 UTC
☐	db2	SQL	100 %	20	0	0	2023-02-08 15:09:21 UTC
☐	DB2Edge	SQL	100 %	20	0	0	2023-02-08 14:51:39 UTC
☐	DEMO_WH	SQL	100 %	16	0	0	2023-02-08 14:51:42 UTC
☐	DEMO_WH_EDGE	SQL	100 %	16	0	0	2023-05-16 15:11:50 UTC

All transformationsFull-text search2Filter byAllExport All Transformations4

1

3

ID	Name	Status code	Status description	Group name
1	vAssocSeqLineItems	DONE	Analysis succeeded, queries 1	None
2	vAssocSeqOrders	DONE	Analysis succeeded, queries 1	None
3	vTargetMail	DONE	Analysis succeeded, queries 1	None
4	vTimeSeries	DONE	Analysis succeeded, queries 1	None
5	vDMPPrep	DONE	Analysis succeeded, queries 1	None

BrowseSettings

1

5

Q Filter graph by full-text search on transf. code

Visualization options

☐ Group by parent object

☐ Only ending points

Depth

MAX

Dependencies

2-way dependencies

☐ Show indirect dependencies

Export to

Show lineage

No	Name	Description
1	Summary per data source	A summary per data source. You can also select data sources to filter the results.
	Selected	Checkboxes to filter on a data source in the transformations table. If you select none, the transformations table contains all transformations.
	Source ID	The ID of your data source. You entered this ID in the configuration file.
	Scanner type	The type of scanner that is used to scan the queries in your data source.
	Success rate	The success rate of the data source analysis on the Collibra Data Lineage service. The success rate indicates how complete your technical lineage is. Important The success rate of a technical lineage gives a good indication of the processing success. A success rate less than 100%, however, does not mean processing was unsuccessful. A parsing error, for example, which negatively affects the success rate, does not always negatively affect the completeness of the lineage.
	Done	The amount of queries that were scanned and analyzed.
	Parsing Error	The amount of parsing errors.
	Analyze Error	The amount of analysis errors.
	Last sync time	The last time the data source was uploaded to the Collibra Data Lineage service , for analysis and processing.
2	Search tools	Tools to help you search for specific source code fragments.
	Full-text search	A search field to find specific queries in the log files. Type what you are looking for and press <code>Enter</code> .
	Filter by	A drop-down list to filter the source codes based on their status code.

No	Name	Description
3	Transformations table	The table that contains details of the transformations and source code (fragments). You can filter the rows in the table by selecting data sources in the data source table and by using the search tools. Tip If you click a source code fragment, you can see the log file attached to it. Note Unity Catalog does not provide source code for each transformation. Therefore, when you create technical lineage for Databricks Unity Catalog, the source code pane does not display any source code.
	ID	The ID of the source code fragments or transformation details, which are assigned in chronological order.
	Name	The name of the specific source code fragment or transformation detail. You can also see the source code fragment name in the source code pane in the technical lineage graph.
	Status code	The status of the analysis. A source code fragment or transformation detail can have one of the following status codes: • DONE: All queries are processed successfully. • ERROR: Some queries could not be processed. • PARSING_ERROR: The syntax of some queries is invalid or unidentified. • ANALYZE_ERROR: Some columns are not linked to a table.
	Status description	The description of the status code that provides more information about the analysis and shows how many queries were processed.
	Group name	The name of the package or procedure to which the source code fragment or transformation details belongs.

No	Name	Description
4	Export Selected Transformations	The button to export transformation details for the selected data sources. When you click this button, you download a ZIP file. The ZIP file contains an errors.csv file that includes the transformation details for the selected data sources. If you do not select any data sources, the transformation details for all listed data sources in the transformations table are exported.
5	Show lineage	The button to go back to the technical lineage graph.
6	Sort by each column	The sorting icons that you can use to sort by each column in ascending or descending order. These columns include Scanner type , Success rate , Done , Parsing Error , Analyze Error , and Last sync time .

Analysis results

If you click one of the rows in the Transformations table, a file with the analysis results attached to the source code or transformation details opens. You can use these files to easily find errors in the source code or transformation details of your data source.

2778	None	DONE	Analysis succeeded, queries 0	None
2779	None	ANALYZE_ERROR	Column name " FICT_INCANTATION_ " specified more than once in CREATE query at line 1 , column 1.	None
<pre> 1 CREATE TECHLIN VIEW 'infa': 'REP_FICTION_WF01'. 'folder': 'FICT'. 'workflow': 'WF_FICTION_INCANTATION_ON'. 'session': 'INCANTATION FICTI 2 AS SELECT FICT_INCANTATION_.FICTION_ID, 3 FICT_INCANTATION_.CASE_ID, FICT_INCANTATION_.INCANTATION_FICT_SEND, 4 FICT_INCANTATION_.MY_FICT_ID AS MY_FICT_ID 5 FROM 6 FICTION.FICT_INCANTATION_, FICT_INCANTATION 7 WHERE FICT_INCANTATION_.PROCESS_FICTION_ID=24 </pre>				
2780	None	DONE	Analysis succeeded, queries 0	None
2781	None	DONE	Analysis succeeded, queries 1	None
2782	None	DONE	Analysis succeeded, queries 0	None

If the metadata that Collibra Data Lineage collects from your data source includes SQL queries, the analysis results might display comments from those SQL queries.

Note If a comment ends with a statement separator, for example, `/*select 2 from dual*/;`, the comment is counted as a statement. Consequently, the number of queries that are displayed in the **Done** column under **Summary per data source** might be greater than the actual number of queries parsed.

3	Package.dtsx	DONE	Analysis succeeded, queries 0	None
<pre> 1 /***** 2 multiline comment 2 3 **/ 4 5 ALTER TABLE [SalesLT].[Customer] ADD CONSTRAINT [DF_Customer_NameStyle] DEFAULT ((0)) FOR [NameStyle]; </pre>				
4	Package.dtsx	DONE	Analysis succeeded, queries 0	None
<pre> 1 -- single line comment 2 ALTER TABLE [SalesLT].[Customer] ADD CONSTRAINT [DF_Customer_rowguid] DEFAULT (newid()) FOR [rowguid]; </pre>				
5	Package.dtsx	DONE	Analysis succeeded, queries 0	None

Analyze errors and possible solutions

If the analysis results contain errors, the Status code column shows `ANALYZE_ERROR` and the Name column displays the name of the error. You can find the transformation details by clicking the row.

The following list shows some common errors and possible solutions:

Informatica Intelligent Cloud Services: Missing parameter files

This error occurs when Collibra Data Lineage cannot find the parameter files that are listed in the error message. To resolve this issue:

1. Ensure you have the parameter files.
2. Take one of the following actions:
 - If you use technical lineage via Edge, upload the parameter files to the **Parameter Files** field when you [add the technical lineage for Informatica Intelligent Cloud Services \(IICS\) capability](#).
 - If you use the lineage harvester, specify the `paramFiles` property in [the configuration file](#) to point to the directory in which your parameter files are stored.

iics_dev_source

Full-text search

Q

Filter by

All

ID	Name	Status code
210	Errors summary	ERROR
211	Missing parameter files.	ERROR

1

There are some mapping task parameter files, that were not found:

2

/shared/User_Parameters/EAM_PARAM.txt

3

/shared/User_Parameters/User_Parameters.param

4

/shared/User_Parameters//EAM_PARAM.txt

Informatica Intelligent Cloud Services: No taskflows found in zip root file

This error occurs when the `taskflow` type was specified in the objects to be retrieved, but Collibra Data Lineage could not find any taskflows while creating technical lineage for IICS. To resolve this issue, take any of the following actions:

- If you use technical lineage via Edge, verify the paths and types in the **Objects** field of the [technical lineage for Informatica Intelligent Cloud Services \(IICS\) capability](#).
- If you use the lineage harvester, verify the paths and types in the `objects` property in the [configuration file](#).

Note that the only meaningful types to retrieve are: Taskflow, Workflow, Project and Folder.

Sources

Stitching

Selection	Source ID	Scanner type
☐	Snowflake_edw	SQL
☐	MyDB	SQL
☒	iics_source	IICS

iics_source

Full-text search

Filter by

All

ID	Name	Status code	Status description
1	Undefined system names for connections	ANALYZE_ERROR	None
2	Errors summary	ERROR	None

```
1 0/0 taskflows have failed.
2 1/2 connections have failed.
3 Saving connection fmlMzaEw6oRfHVy5jGKic5 has failed
4 [MSG-LIN-19001]: Connection with name 'CAG_ORCL_PQM' is not defined.. Please check the documentation.
5
6 No taskflows found in zip root file
7 If there are no taskflows inside the zip root file, lineage cannot be generated.
8
```

Informatica PowerCenter: CONFIGURATION

This error occurs in the following scenarios:

- If you use technical lineage via Edge to create technical lineage, certain properties were not specified in the **Source Configuration** field in [the Technical Lineage for Informatica PowerCenter capability](#).
- If you use the lineage harvester to create technical lineage, certain properties were not specified in [the source ID file](#).

The unspecified properties are marked as `UNDEFINED` in the transformation details. To resolve this issue, specify the properties.

All transformations

Full-text search

Q

Filter by

ERROR

Export All Transformations

ID	Name	Status code	Status description	Group name
23894	CONFIGURATION	ANALYZE_ERROR	IMPORTANT: Please, define 'UNDEFINED' values in your configuration file	None

```
1 # Please, define 'UNDEFINED' values in your configuration file
2
3 {
4   "connectionDefinitions": {
5     "PROCMONITOR": {
6       "dbname": "UNDEFINED",
7       "schema": "UNDEFINED"
8     },
9     "INFOMANAGERONOTAP": {
10      "dbname": "UNDEFINED",
11      "schema": "UNDEFINED"
12    },
13     "INFALOAD_NO_DOP_ORHCKGX": {
14       "dbname": "UNDEFINED",
15       "schema": "UNDEFINED"
16     },
17     "INFAPAROKHCKGX": {
18       "dbname": "UNDEFINED",
19       "schema": "UNDEFINED"
20     },
21   }
22 }
```

Informatica PowerCenter: UNRESOLVED PARAMETERS

This error occurs when parameters are missing from the parameter files in the external folder. To resolve this issue:

- If you use technical lineage via Edge, export your parameter files and [add the files to the Shared Storage connection folder](#).
- If you use the lineage harvester, export your parameter files and add the files to [the external folder](#). Also ensure that the `path` property in [the configuration file](#) points to the correct location of the external folder.

12851	UNRESOLVED PARAMETERS	ANALYZE_ERROR	List of unresolved parameters with their headers consisted of folder and session names. Fulfill them and include inside parameter files.	None
<pre> 1 [CERNER_DM.s_CERNER_DM_ENCOUNTER_Load] 2 \$DBCONNECTION_TGT 3 \$DBCONNECTION_Tgt 4 \$DBCONNECTION_Src 5 6 [CERNER_DM.s_CERNER_DM_ENCOUNTER_Step2a_ER_ADMIT_IND_Stage_Load] 7 \$DBCONNECTION_TGT 8 \$DBCONNECTION_Tgt 9 \$DBCONNECTION_Src 10 11 [CERNER_DM.s_CERNER_DM_ENCOUNTER_Step3d_ED_DSCHRG_DISP_STG_Load] 12 \$DBCONNECTION_TGT 13 \$DBCONNECTION_Tgt 14 \$DBCONNECTION_Src 15 16 [CERNER_DM.s_CERNER_DM_ENCOUNTER_Step3c_ED_DSCHRG_DISP_STG_Load] 17 \$DBCONNECTION_TGT 18 \$DBCONNECTION_Tgt 19 \$DBCONNECTION_Src 20 21 [CERNER_DM.s_CERNER_DM_ENCOUNTER_Step3b_ER_ADMIT_IND_Stage_Load] </pre>				

SQL Server Integration Services: CONFIGURATION

This error occurs in any of the following scenarios:

- If you use technical lineage via Edge to create technical lineage, the definitions for certain connections were not specified in the **Source Configuration** field in [the Technical Lineage for SQL Server Integration Services capability](#).
- If you use the lineage harvester to create technical lineage, the definitions for certain connections were not specified in [the source ID file](#).

The connections lacking definitions are listed when you expand the CONFIGURATION analyze error. To resolve this issue, take any of the following actions:

- If you use technical lineage via Edge to create technical lineage,
 1. Copy the connection definitions that are marked as `UNDEFINED` in the error.
 2. Add the copied connection definitions to the **Source Configuration** field in [the Technical Lineage for SQL Server Integration Services capability](#).
 3. Replace `UNDEFINED` with the correct values:
 - Ensure that the values for `collibraSystemName`, `schema`, and `dbname` match the system, schema, and database names in Data Catalog.
 - Select the value for `dialect` from the dialect list that Collibra Data Lineage supports.

For more information about specifying the values, go to the description of the **Source Configuration** field in the [Add a technical lineage capability to your Edge site](#) step.

- If you use the lineage harvester to create technical lineage,
 - a. Copy the connection definitions that are marked as `UNDEFINED` in the error.
 - b. Add the copied connection definitions to the `connection_definitions.conf` file .
 - c. Replace `UNDEFINED` with the correct values:
 - Ensure that the values for `collibraSystemName`, `schema`, and `dbname` match the system, schema, and database names in Data Catalog.
 - Select the value for `dialect` from the dialect list that Collibra Data Lineage supports.

For more information about specifying the values, go to the Configure the connection definitions step in [Prepare an external directory folder for the lineage harvester](#).

671806 CONFIGURATION ANALYZE_ERROR IMPORTANT: Please, define 'UN

```
1 # Please, define 'UNDEFINED' values in your configuration file
2
3 {
4   "Data Source=127.0.0.1,11447;User ID=UserDev;Initial Catalog=ETLAudit;Provider=SQLNCLI11.1;Auto Translate=False"; {
5     "collibraSystemName": "UNDEFINED",
6     "dbname": "UNDEFINED",
7     "schema": "UNDEFINED",
8     "dialect": "UNDEFINED"
9   },
10  "Data Source=127.0.0.1,11463;User ID=DEV;Initial Catalog=DVStage;Provider=SQLNCLI11.1;Auto Translate=False"; {
11    "collibraSystemName": "UNDEFINED",
12    "dbname": "UNDEFINED",
13    "schema": "UNDEFINED",
14    "dialect": "UNDEFINED"
15  },
16  "Data Source=BOFI-UATSQL.DEV.MYBOFI.LOCAL.ALL_RWD_SQL.SDM.AXOS.CO,11445;User ID=a;Initial Catalog=MassMaint;Provider=SQLNCLI11.1;Auto Translate=False"; {
17    "collibraSystemName": "UNDEFINED",
18    "dbname": "UNDEFINED",
19    "schema": "UNDEFINED",
20    "dialect": "UNDEFINED"
21  }
22 }
```

Technical lineage Stitching tab page

The Stitching page shows the full path of assets in Data Catalog and data objects of the data sources for which you created a technical lineage. You can use it to easily check which assets are **stitched** and which are not.

You can access the Stitching tab page by clicking **Show status** on the **Settings** tab pane.

SourcesStitching

Search

Full asset path	Found In
DEFAULT > DEFAULT > EU_COUNTRIES	Catalog & Technical Lineage
DEFAULT > DEFAULT > EU_CUSTOMERS	Catalog & Technical Lineage
DEFAULT > DEFAULT > LINEAGE-CONSTANT	Technical Lineage
DEFAULT > DEFAULT > SIMPLIFIED_US_CUSTOMERS	Catalog & Technical Lineage
DEFAULT > DEFAULT > UNION_EU_US_CUSTOMERS	Catalog & Technical Lineage
DEFAULT > DEFAULT > US_CUSTOMERS	Catalog & Technical Lineage
DEFAULT > DEFAULT > WW_CUSTOMERS	Catalog & Technical Lineage
INF_REP_DEV > COLLIBRASAMPLE > COLLIBRA_SAMPLE > ORACLE > M_DEFAULT_INTERNETSALES > SQ_ORDER_ITEMS.SOURCE_QUALIFIER	Technical Lineage
INF_REP_DEV > COLLIBRASAMPLE > COLLIBRA_SAMPLE > SQL > M_DEFAULT > CONCAT_FULLNAME.EXPRESSION	Technical Lineage
INF_REP_DEV > COLLIBRASAMPLE > COLLIBRA_SAMPLE > SQL > M_DEFAULT > JOIN_CUSTOMER_INTERNETSALES.JOINER	Technical Lineage
INF_REP_DEV > COLLIBRASAMPLE > COLLIBRA_SAMPLE > SQL > M_DEFAULT > SQ_DIMCUSTOMER.SOURCE_QUALIFIER	Technical Lineage
INF_REP_DEV > COLLIBRASAMPLE > COLLIBRA_SAMPLE > SQL > M_DEFAULT > SQ_FACTINTERNETSALES.SOURCE_QUALIFIER	Technical Lineage
JD DEFAULT > INVENTORYMANAGEMENT > SQL17	Catalog
JD DEFAULT > INVENTORYMANAGEMENT > SQL181	Catalog
JD DEFAULT > INVENTORYMANAGEMENT > SQL1201	Catalog
JD DEFAULT > INVENTORYMANAGEMENT > SQL281	Catalog
JD DEFAULT > INVENTORYMANAGEMENT > SQL721	Catalog
JD DEFAULT > INVENTORYMANAGEMENT > SQL763	Catalog
JD DEFAULT > INVENTORYMANAGEMENT > SQL763	Catalog

BrowseSettings

Filter graph by full-text search on transf. code

Visualization options

☐ Group by parent object

☐ Only ending points

Depth MAX

Dependencies

2-way dependencies

☐ Show indirect dependencies

Export

Show lineage

No	Name	Description
1	Search field	A search field to find specific assets or data objects. Type what you are looking for and press Enter.

No	Name	Description
2	Full asset path	The full path to all data objects on the Collibra Data Lineage service and all assets in Data Catalog.
3	Found in	The location where the asset or data object was found. There are three possible locations: • Data Catalog: The asset was found in Data Catalog, but it does not match the full path of a data object on the Collibra Data Lineage service. As a result, there is no technical lineage created for this asset. • Technical lineage: The data object was found in the data source for which you created a technical lineage, but it does not match the full path of an asset in Data Catalog. As a result, the data object is shown in technical lineage with a gray background. • Data Catalog & Technical lineage: An asset and a data object with the same full path were found in Data Catalog and on the Collibra Data Lineage service. As a result, they were stitched and are shown in technical lineage with a yellow background. Note In Collibra, full paths are case-sensitive.
4	Show lineage	The button to go back to the technical lineage graph .

Technical lineage troubleshooting

For complete troubleshooting information, go to the [Collibra Support Portal](#).

If you encounter issues when creating technical lineage for Databricks Unity Catalog, go to [Technical lineage via Edge troubleshooting for Databricks](#) in Collibra Support Portal.