



Collibra Data Intelligence Cloud

Data Lineage

Collibra Data Intelligence Cloud - Data Lineage

Release date: July 9, 2023

Revision date: July 06, 2023

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

https://productresources.collibra.com/docs/collibra/latest/Content/to_data-lineage.htm

Contents

Contents	ii
Collibra Data Lineage	1
Technical lineage	1
The lineage harvester	27
Technical lineage via Edge	51
Collibra Data Lineage service instances	53
Supported data sources for technical lineage	54
Technical lineage viewer	83
Business Summary Lineage	109
Differences between Technical lineage and diagrams with Business Summary Lineage	111
Synchronization: Continue on error	114
Technical lineage troubleshooting	116
Working with the lineage harvester	158
Creating a Technical lineage via the lineage harvester	159
Delete the technical lineage of a data source	306
Working with technical lineage via Edge	308
Create technical lineage via Edge	308
Delete the technical lineage of a data source on Edge	311
Migrate the technical lineage of a data source	313
Troubleshooting for technical lineage via Edge	316
Working with custom technical lineage	320
Example	320
Creating custom technical lineage via lineage harvester	321

Working with Tableau	343
Tableau terminology	343
Tableau asset types and domain types	345
Tableau operating model	346
Technical lineage for Tableau	363
Supported data sources in Tableau	382
Set up Tableau	384
Prepare a domain for Tableau ingestion	392
Prepare the Data Catalog physical data layer for Tableau stitching	394
Soft delete of Tableau assets	398
Tableau hostname, schema, and system name mapping (Beta)	399
Technical lineage for Tableau via Edge	401
Technical lineage for Tableau via the lineage harvester	419
Working with Power BI service	475
Features	475
Power BI terminology	476
Power BI asset and domain types	477
Power BI operating model	479
Technical lineage for Power BI	496
Supported data sources in Power BI	500
Power BI ingestion considerations and limitations	514
Show DAX calculations on Power BI Column asset pages	517
Power BI prerequisites	519
Prepare a domain for Power BI ingestion	535
Soft delete of Power BI assets	537
Technical lineage for Power BI via Edge	538

Technical lineage for Power BI via the lineage harvester	551
Working with Power BI service (deprecated)	585
Features	585
Power BI terminology (deprecated)	586
Power BI operating model (deprecated)	587
Power BI asset and domain types (deprecated)	594
Overview Power BI integration steps (deprecated)	596
Ingestion results based on Power BI subscriptions (deprecated)	605
Power BI ingestion limitations (deprecated)	607
Supported data sources in Power BI (deprecated)	611
Power BI prerequisites (deprecated)	615
Prepare a domain for Power BI ingestion (deprecated)	627
Power BI and lineage harvester set-up (deprecated)	628
Power BI business logic (deprecated)	657
Technical lineage for Power BI service (deprecated)	661
Schedule jobs (deprecated)	664
Harvesters upgrade (deprecated)	666
Power BI troubleshooting (deprecated)	668
Working with SSRS and PBRS	677
SSRS and PBRS asset and domain types	677
SSRS and PBRS terminology	679
SSRS and PBRS operating model	680
Soft delete of SSRS assets	697
Technical lineage for SSRS and PBRS	698
SSRS and PBRS integration steps	700
Working with Looker	723

Looker terminology	723
Looker operating model	724
Looker asset and domain types	742
Looker business logic	743
Authentication	747
Technical lineage for Looker	748
Soft delete of Looker assets	749
Prepare a domain for Looker ingestion	750
Overview Looker integration steps	752
Schedule Looker ingestion jobs	772
Troubleshooting	773
Working with MicroStrategy	774
MicroStrategy terminology	775
MicroStrategy asset and domain types	775
MicroStrategy operating model	777
Soft delete of MicroStrategy assets	785
MicroStrategy integration steps	787
Working with MicroStrategy (NEW)(Beta)	804
MicroStrategy terminology (NEW)(Beta)	804
MicroStrategy asset and domain types (NEW)(Beta)	806
MicroStrategy operating model (NEW)(Beta)	808
Technical lineage for MicroStrategy (NEW)(Beta)	820
MicroStrategy integration steps (NEW)(Beta)	823

Collibra Data Lineage

Collibra Data Lineage is a product that allows you to trace how data flows from source to destination. It consists of two components to accommodate two different personas:

- A technical lineage for Data Engineers, Data Architects and similar personas.
- A diagram with [Business Summary Lineage](#) for Business Analysts and other business users.

Technical lineage is a detailed lineage graph that shows where data objects are used and how they are transformed. A diagram with the Business Summary 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 diagram with Business Summary Lineage only provides a summary of it.

Note Collibra Data Lineage is only a cloud-only feature.

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. 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

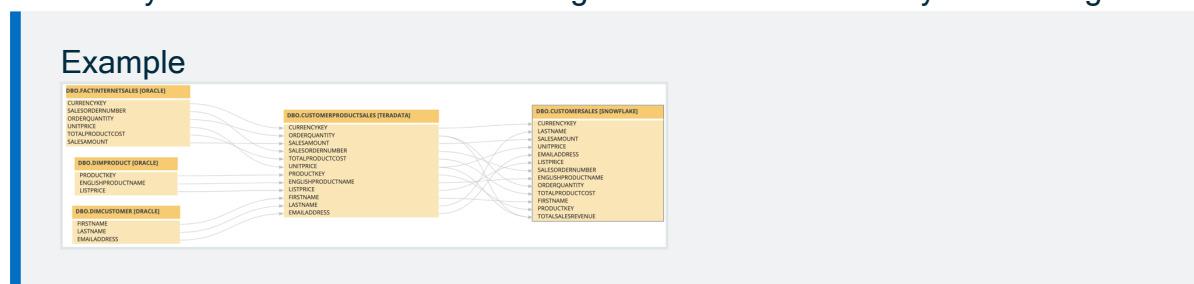
During the technical lineage process, relations of the type "Data Element targets / sources Data Element" are automatically created:

- Between data objects in your data source and assets from [registered data sources](#).
- Between ingested assets from BI sources and Data Catalog assets from registered data sources.

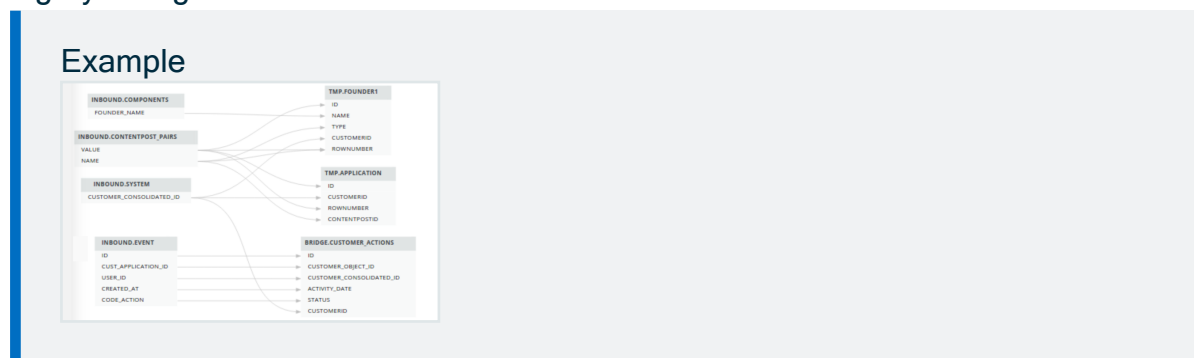
Data objects

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 \(Beta\)](#), which supports the latest MicroStrategy APIs.

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

Note If you want to create a technical lineage for a Google BigQuery database, the project name in the configuration file must be the same as the full name of the Database asset.

Warning Editing the full name of the Schema, Database and System assets may lead to errors during the technical lineage creation process.

The lineage harvester and technical lineage via Edge

The lineage harvester is a command-line tool that you can install, configure and run to create a technical lineage. You also can create a technical lineage on Edge. With technical

lineage via Edge, you can use the Edge user interface, which provides seamless integration with Data Catalog.

The typical workflow for creating a technical lineage is the same whether you use the lineage harvester or Edge. For details, go to [The lineage harvester](#) and [Technical lineage via Edge](#).

If you want to use technical lineage via Edge and the lineage harvester together, you must use the 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](#).

Note Use the lineage harvester pre-release versions only in non-production environments.

For the steps to create technical lineage for the JDBC data sources and ETL tools, go to [Creating technical lineage by using the lineage harvester](#) and [Creating technical lineage via Edge](#). For the list of supported data sources, go to [Supported data sources for technical lineage](#).

Tip

For information about ingesting metadata from the following BI tools and creating a technical lineage via the lineage harvester, go to the dedicated sections:

- Tableau:
 - [Via the Data Catalog user interface](#).
 - [Via the lineage harvester](#).
- [Power BI \(deprecated\)](#)
- [Power BI](#)
- [Looker](#)
- [MicroStrategy](#)
- [MicroStrategy \(NEW\)\(Beta\)](#)
- [SQL Server Reporting Services and Power BI Report Server](#)

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

Transformation logic

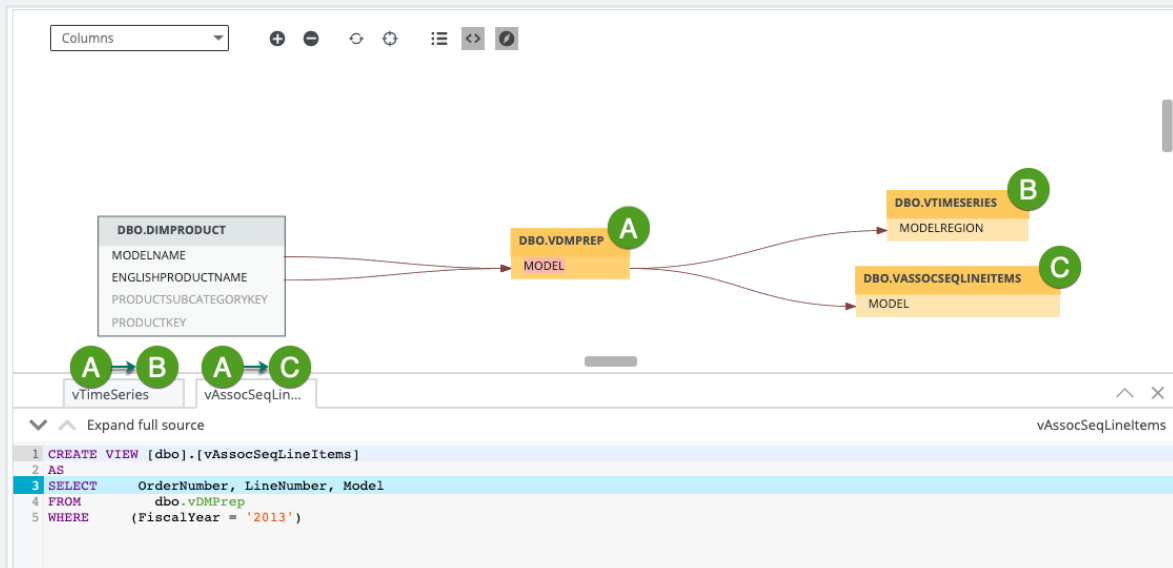
Transformation logic is used to transform source code in a technical lineage diagram that can be visualized in Data Catalog. Collibra Data Lineage [supports](#) the most commonly used transformations.

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.



Automatic stitching for technical lineage

Stitching is a process that creates relations between assets and [data objects](#) representing the same data source. More specifically, stitching creates relations between the following assets:

- The assets that were created when you [prepared](#) Data Catalog's physical data layer for a data source; and
- The data objects in the same data source for which you created a technical lineage and that represent the assets in Data Catalog.

For Colibra Data Lineage to stitch the assets to the data objects, you must prepare the Data Catalog physical data layer to create the database > schema > table > column or system > database > schema > table > column hierarchy. Note that when a table in your data source has a schema and a file as its parents, Colibra Data Lineage uses the schema as the parent for stitching.

When the data sources are scanned, [Collibra Data Lineage service](#) automatically creates and pushes new relations of the type "Data Element targets / sources Data Element":

- Between data objects in your data source and assets from [registered data sources](#).
- Between ingested assets from BI sources and Data Catalog assets from registered data sources.

Example To clarify, in the case of Tableau integration, the Tableau Data Attribute is the target of the Column and the Column is the source of the Tableau Data Attribute.

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.

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 \(Beta\)](#), which supports the latest MicroStrategy APIs.

Tip 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.

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.

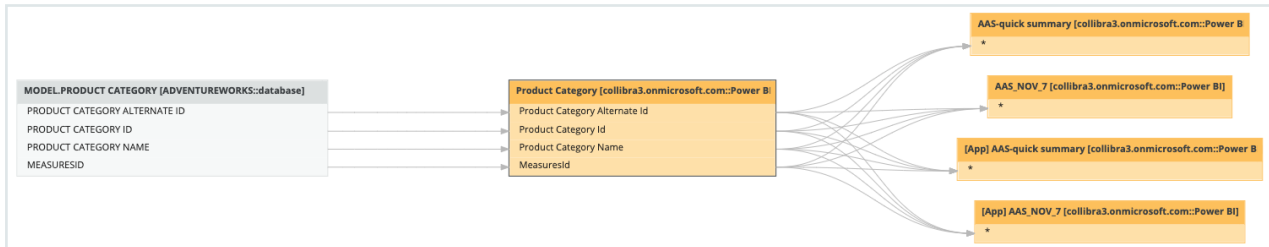
Note This topic applies to MicroStrategy, Power BI, SSRS-PBRS and Tableau. Collibra Data Lineage currently does not offer stitching for Looker assets.

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

Qtesting1

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 database 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:

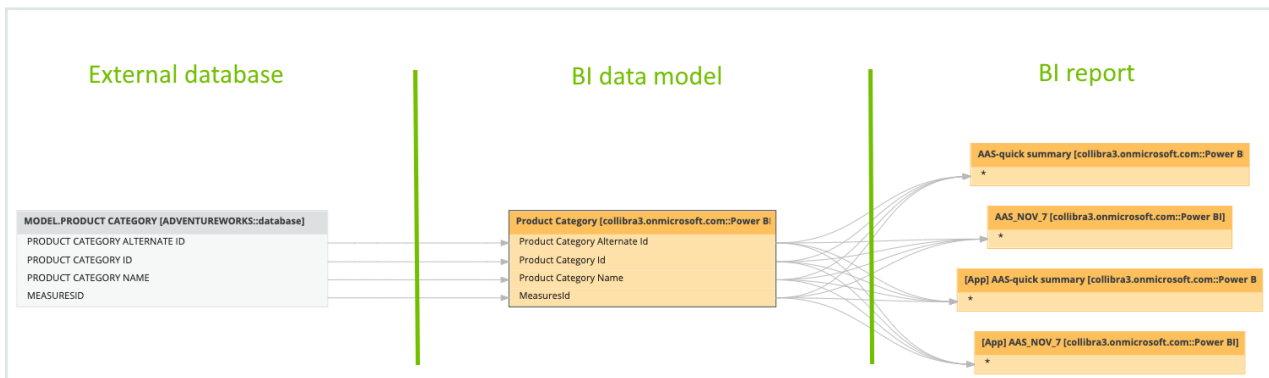
- A technical lineage that shows the flow of data from the data source to your BI tool.
- Assets in Data Catalog that represent the data objects in your data source. Among these assets are:
 - Assets that represent the data set you used to create the report in your BI tool. These are assets of child 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.

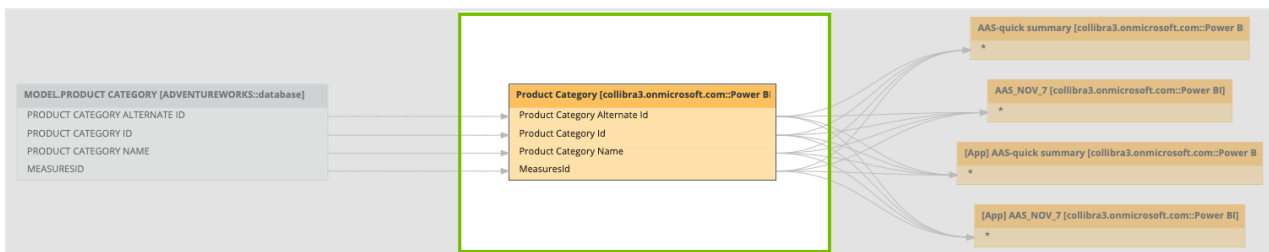
By the fact of its presence in the technical lineage, we know that the lineage harvester collected the source code from the BI tool and identified the flow of data from this data source to the BI data model. This node is a prerequisite for stitching. If it is not shown in the technical lineage, stitching 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

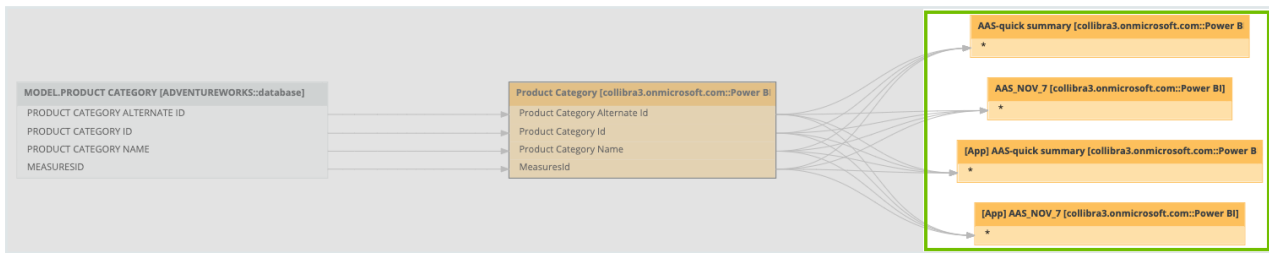


This node represents the data set that you used to create the report in your BI tool. It will always be shown in the technical lineage, because it is the target of the database table and the source of the BI report.

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 Colibra 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.

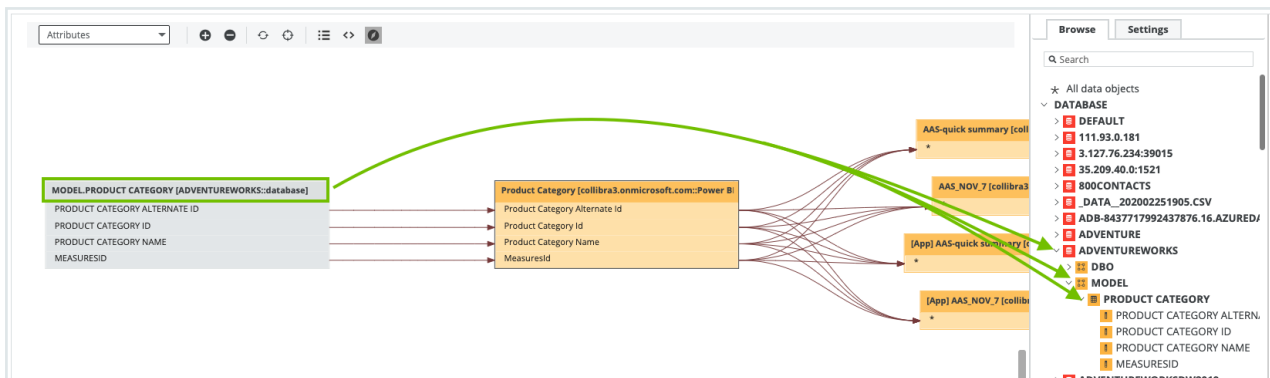
How to achieve stitching

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 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
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 broken stitching scenarios and suggestions for resolving the issue, go to [Broken stitching and possible solutions](#).

Broken stitching and possible solutions

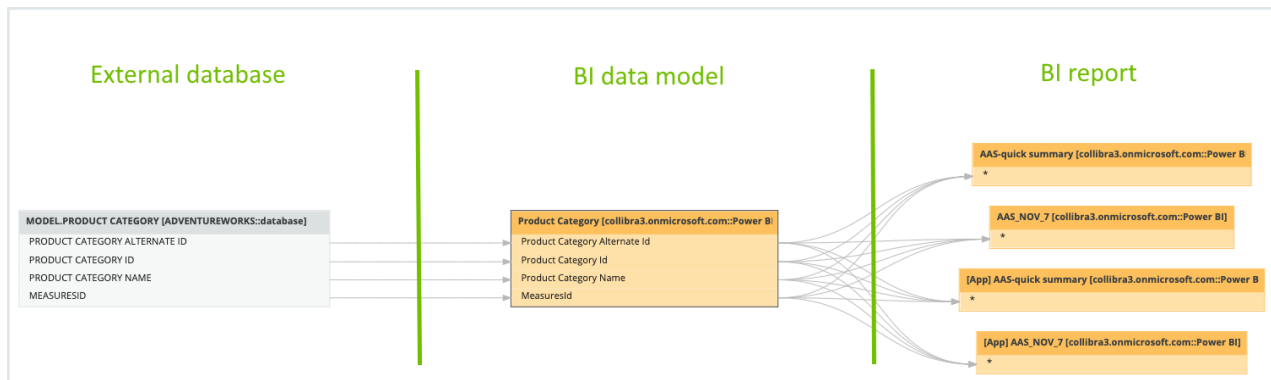
This topic provides some examples where stitching is broken, and some advice on how to achieve stitching.

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. Colibra 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.



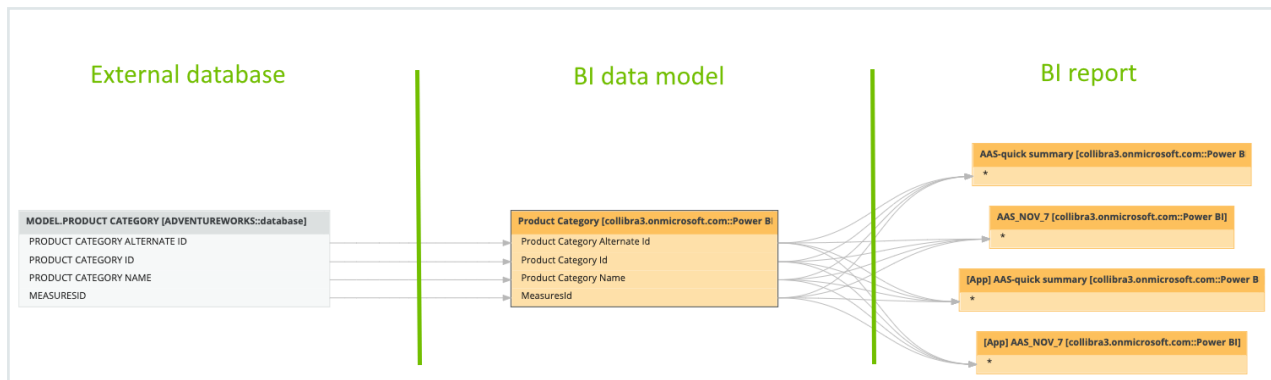
Note If you have integrated MicroStrategy, there will be four nodes. Instead of the BI data model node, there will be two nodes, one representing MicroStrategy Attributes, the other representing MicroStrategy Facts. There could also be four groups if you are integrating SQL Server Reporting Services (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 broken stitching and possible solutions

Here are a few common examples of broken stitching and possible solutions for achieving stitching.

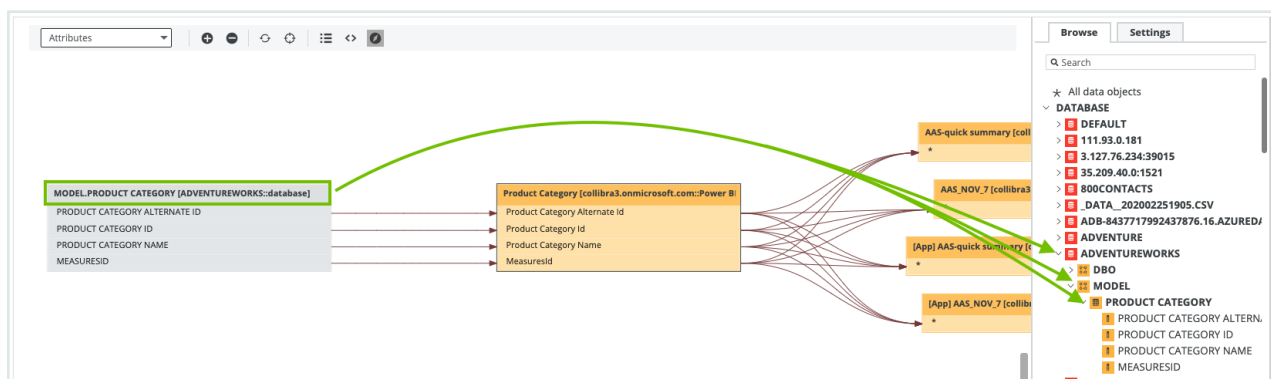
Power BI: database names don't match



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 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:

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

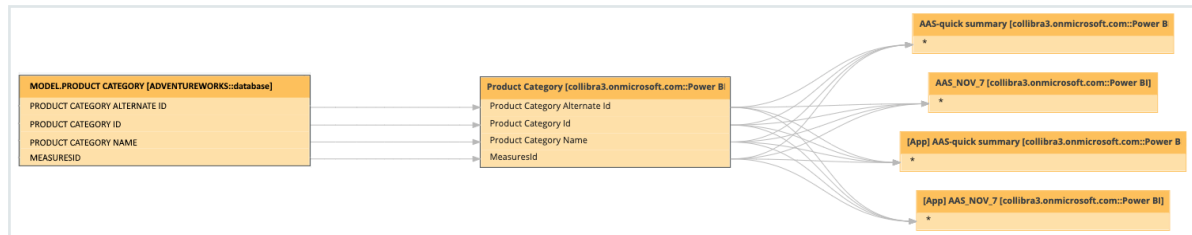
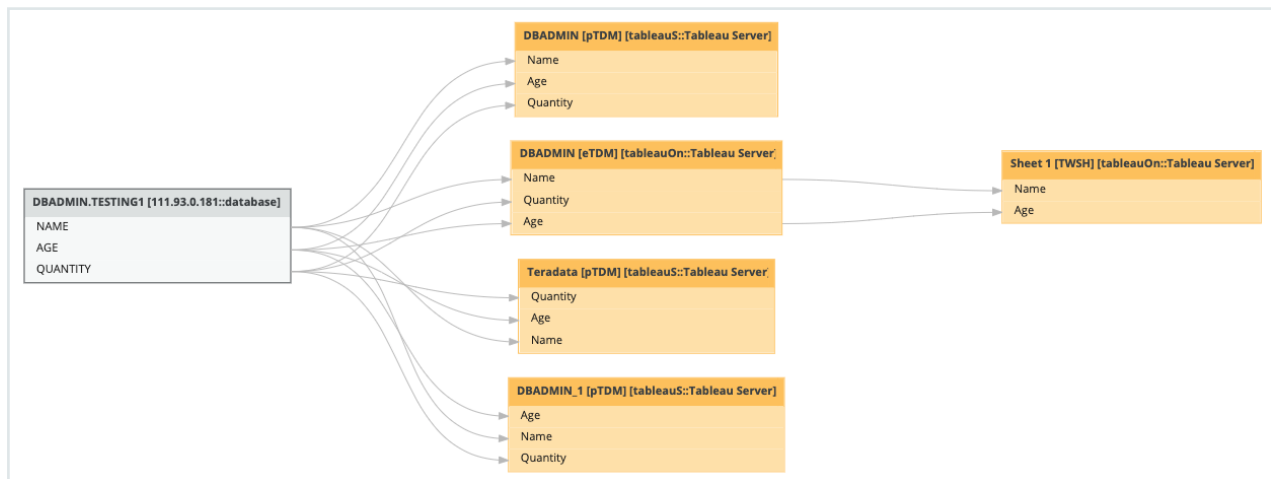


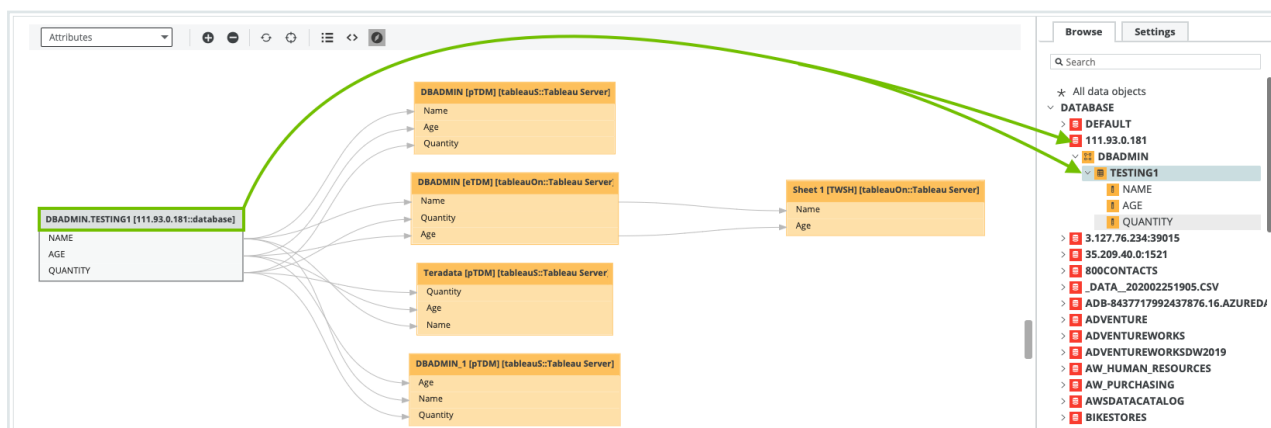
Tableau: database names don't match



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:

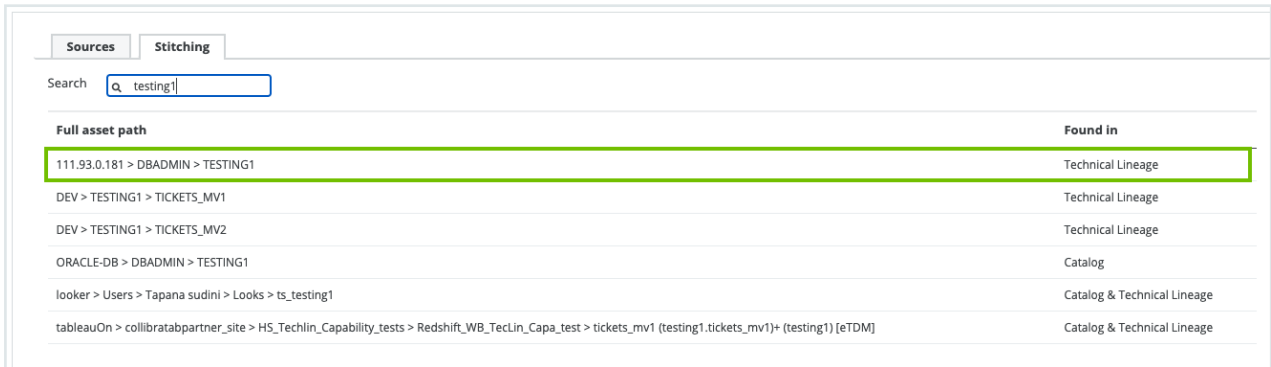


- The database name: 111.93.0.181
- The schema name: DBADMIN
- The table name: TESTING1

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 Colibra Data Lineage service instance.

Use the Search field to find the unstitched database table TESTING1.



The screenshot shows the 'Stitching' tab in the Colibra Data Lineage interface. A search bar at the top contains the text 'testing1'. Below the search bar is a table with two columns: 'Full asset path' and 'Found in'. The table lists several assets, with the first row highlighted in green. The highlighted row shows the full asset path '111.93.0.181 > DBADMIN > TESTING1' and is found in the 'Technical Lineage'.

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.

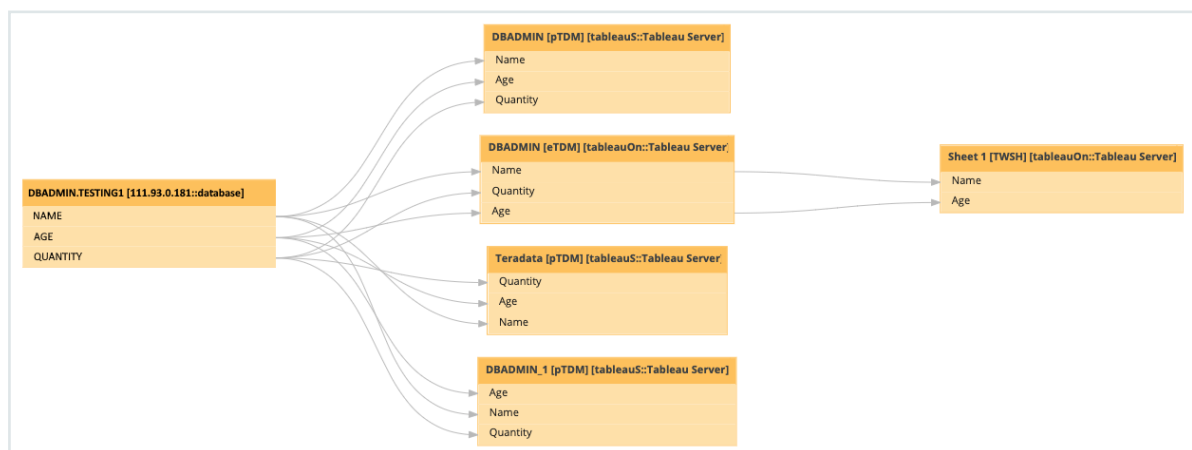
Sources Stitching	
Search testing	
Full asset path	Found in
111.93.0.181 > DBADMIN > TESTING1	Technical Lineage
DEV > TESTING1 > TICKETS_MV1	Technical Lineage
DEV > TESTING1 > TICKETS_MV2	Technical Lineage
ORACLE-DB > DBADMIN > TESTING1	Catalog
looker > Users > Tapani sudini > Looks > ts_testing1	Catalog & 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 `databaseMapping` property in your Tableau <source ID> configuration file as follows:

```
{
  "databaseMapping": {
    "111.93.0.181": "oracle-db",
  }
}
```

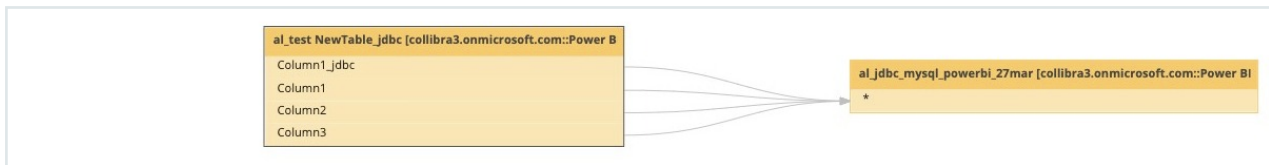
2. Run the lineage harvester again.
 - » Stitching is achieved



Power BI: Unsupported Power Query M function or calculated columns

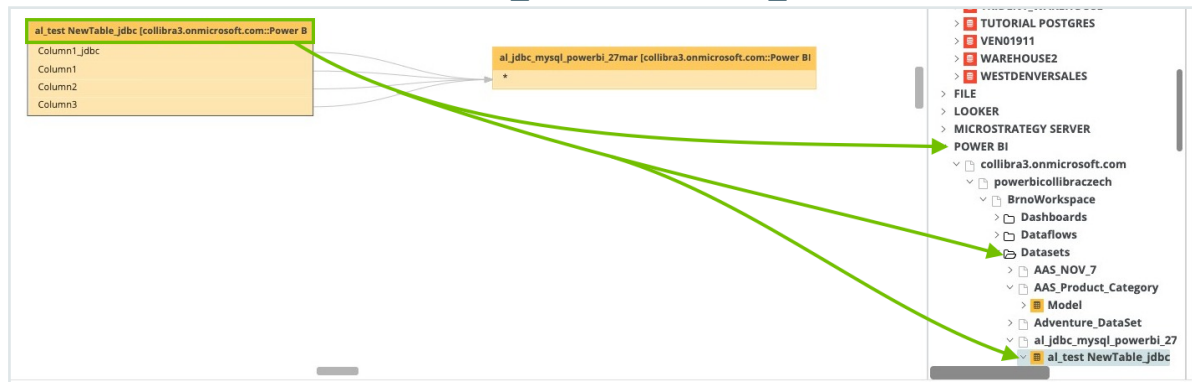
Collibra Data Lineage does not support DAX. Therefore, calculated columns result in missing lineage, as do unsupported Power Query M function.

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

```

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

```

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.

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.

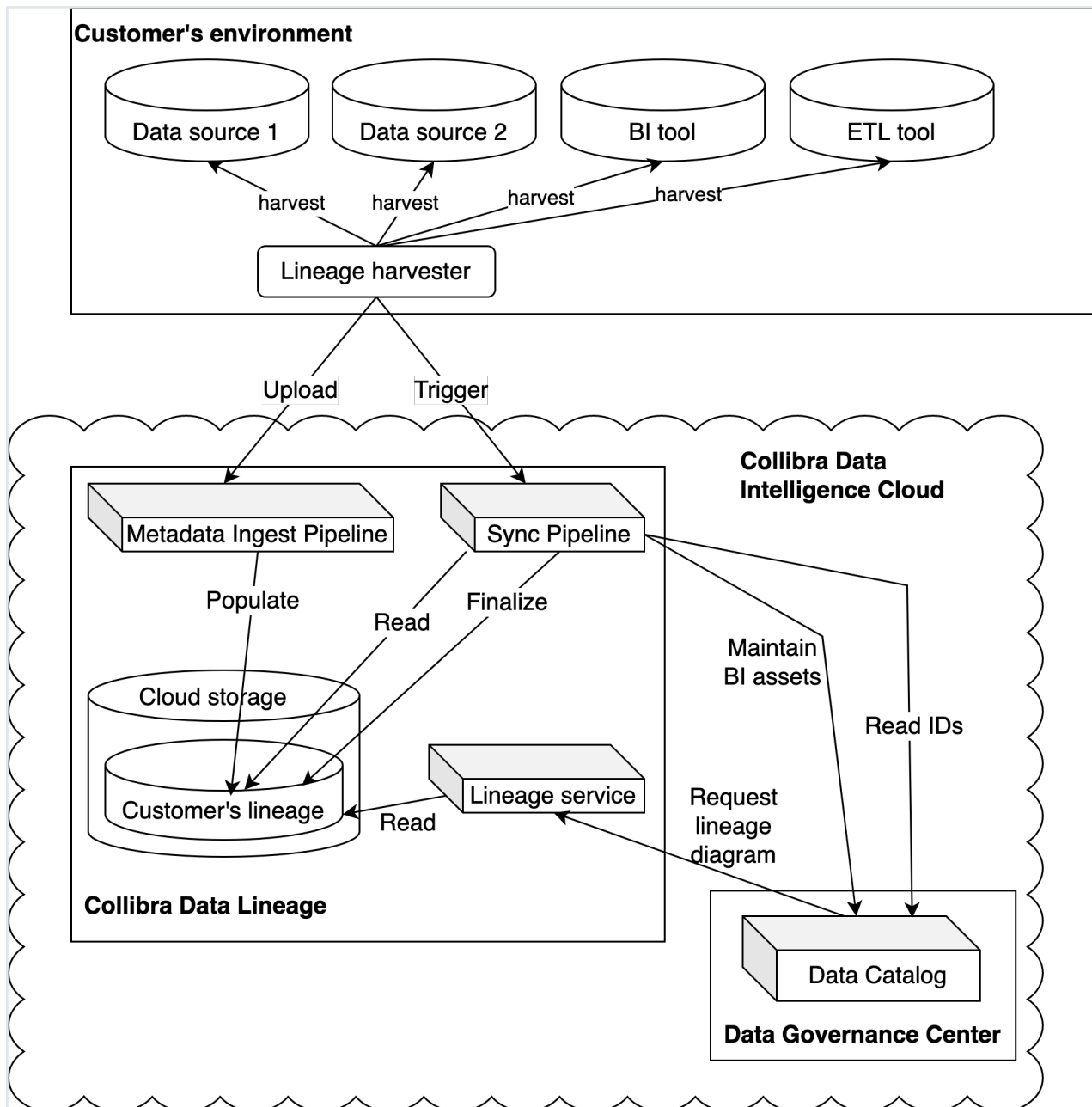
Permissions

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 to 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.

Note Hidden Tableau worksheets were briefly excluded from the Tableau ingestion, as part of code changes in the 2022.11 release. This change was reverted. Instead, we urge you to review the worksheets that have the attribute "Visible on server" in Tableau set to "false". If a worksheet is not hidden, the "Visible on server" attribute in Collibra is now set to "yes". You can track the hiding or unhiding of the worksheet in the **History** tab of the asset page. If you have Collibra Data Marketplace, you can filter out hidden worksheets by excluding those that have "Visible on server" set to "false".

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

Changed in version	New lineage harvester improvements
2023.07	• Collibra Data Lineage now supports the following Power Query M functions: ◦ <code>Cube.Transform</code> ◦ <code>Cube.AddAndExpandDimensionColumn</code> ◦ <code>Table.FromList</code> ◦ <code>AnyalYSISServices.Databases</code> These functions enable technical lineage between SAP HANA (using <code>SapHana.Database</code>) and Analysis Services (using <code>AnalysisServices.Databases</code>), 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 <code>displaySampleQueries</code> 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 <code>PARTITIONED BY</code> parameters in <code>CREATE TABLE</code> 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 <code>useCollibraSystemName</code> property as <code>true</code>, 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:

Changed in version	New lineage harvester improvements
	◦ The Snowflake driver to address the CVE-2023-30535 vulnerability. ◦ The BigQuery driver to mitigate the CVE-2022-45688 vulnerability. • 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

Changed in version	New lineage harvester improvements
	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 to the repository. • When you create technical lineage for Snowflake by using the SQL-API ingestion method, ◦ <code>QUERY_TAG</code> values are now shown in the transformation window for lineage queries. ◦ The lineage harvester optimized the results of the <code>columns_joined</code> 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 <code>ignore-source</code> 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.

Changed in version	New lineage harvester improvements
	• When ingesting MySQL data sources, the Collibra Data Lineage service instances now support the “as” keyword as optional in “create table” statements. Previously, parsing failed if the “as” keyword was missing.

Changed in version	New lineage harvester improvements
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.

Changed in version	New lineage harvester improvements
	Note Query parameters are supported, but core parameters are not. • 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 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 maplet had an associated shortcut, technical lineage in Collibra would be broken up. Now, there is end-to-end lineage within PowerCenter even when a maplet 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 sys-

Changed in version	New lineage harvester improvements
	tem 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 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 workarounds. • 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 Colibra 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 Colibra. • 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 Colibra. 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 Colibra. 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.

Changed in version	New lineage harvester improvements
	• 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. • 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>.

Changed in version	New lineage harvester improvements
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>colibraSystemName</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.
2022.02	
1.4.4	The lineage harvester now supports: • Technical lineage for Matillion. Redshift and Snowflake projects in Matillion are supported. • Snowflake syntax for the <code>CONNECT BY</code> clause.

Technical lineage via Edge

You can use the technical lineage features on 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. Technical lineage via Edge can collect transformation

logic like SQL scripts and ETL scripts from a specific location, for example a database table or a folder on a file system.

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

Using technical lineage via Edge

To create a technical lineage for different data sources via Edge, you must [enable the features](#) in the Collibra settings or in Collibra Console.

You can define how technical lineage via 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 Tableau, Power BI, 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.

After you create the connections and configure technical lineage via Edge for different data sources, you can manually synchronize the capabilities or add a synchronization schedules.

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](#).

Tip

For information about ingesting metadata from the following BI tools and creating a technical lineage via Edge, go to the dedicated sections:

- [Tableau](#)
- [Power BI](#)

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 and be able to create new relations between these assets.

Specific permissions might be required to access different data sources. Select a data source in the Overview of Collibra-provided JDBC drivers topic to see the required permissions to create a technical lineage.

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

Server	IP address	DNS name
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
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.

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.

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
Amazon Redshift	1.2.34.1058 and newer	JDBC, Folder	SQL based input without stored procedures.
Azure SQL server	Newest version	JDBC, Folder	SQL based input and stored procedures.
Azure SQL Data Warehouse	Newest version	JDBC, Folder	SQL based input and stored procedures.
Azure Synapse Analytics	Newest version	JDBC, Folder	SQL based input and stored procedures.
Google BigQuery	Newest version	JDBC, Folder	SQL based input without stored procedures.
Greenplum	6.10 and newer	JDBC, Folder	SQL based input.
HiveQL (SQL-like statements)	2.3.5 and newer	JDBC, Folder	SQL based input and connection via an AWS host.

JDBC data source type	Supported versions	Connection type	Scope
IBM Db2	11.5 and newer	JDBC, Folder	SQL based input without stored procedures.
Oracle	11g, 12c and newer	JDBC, Folder	SQL based input and stored procedures.
PostgreSQL	9.4, 9.5 and newer	JDBC, Folder	SQL based input without stored procedures.
Microsoft SQL Server	2014, 2016 and newer	JDBC, Folder	SQL based input and stored procedures.
MySQL	5.7, 8 and newer	JDBC, Folder	SQL based input without stored procedures.
Netezza	7.2.1.0 and newer	JDBC, Folder	SQL based input without stored procedures.
SAP Hana	2.00.40 and newer	JDBC, Folder	SQL based input and 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. Important Collibra Data Lineage supports SQL based input and SAP HANA Information views are supported for SAP HANA on-premises. However, calculated views are not supported for SAP HANA Cloud.

JDBC data source type	Supported versions	Connection type	Scope
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.
Spark SQL	2.4.3 and newer	JDBC, Folder	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.
Sybase Adaptive Server Enterprise	16.0 SP02 and newer	JDBC, Folder	SQL based input without stored procedures.
Teradata	15.0, 16.20.07.01 and newer	JDBC, Folder	SQL based input, including BTEQ scripts.

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
Amazon Redshift	1.2.34.1058 and newer	JDBC connection, Shared Storage connection	SQL based input without stored procedures.
Azure SQL server	Newest version	JDBC connection, Shared Storage connection	SQL based input and stored procedures.
Azure SQL Data Warehouse	Newest version	JDBC connection, Shared Storage connection	SQL based input and stored procedures.
Azure Synapse Analytics	Newest version	JDBC connection, Shared Storage connection	SQL based input and stored procedures.
Google BigQuery	Newest version	JDBC connection, Shared Storage connection	SQL based input without stored procedures.
Greenplum	6.10 and newer	JDBC connection, Shared Storage connection	SQL based input.
HiveQL (SQL-like statements)	2.3.5 and newer	JDBC connection, Shared Storage connection	SQL based input and connection via an AWS host.

JDBC data source type	Supported versions	Connection type	Scope
IBM Db2	11.5 and newer	JDBC connection, Shared Storage connection	SQL based input without stored procedures.
Oracle	11g, 12c and newer	JDBC connection, Shared Storage connection	SQL based input and stored procedures.
PostgreSQL	9.4, 9.5 and newer	JDBC connection, Shared Storage connection	SQL based input without stored procedures.
Microsoft SQL Server	2014, 2016 and newer	JDBC connection, Shared Storage connection	SQL based input and stored procedures.
MySQL	5.7, 8 and newer	JDBC connection, Shared Storage connection	SQL based input without stored procedures.
Netezza	7.2.1.0 and newer	JDBC connection, Shared Storage connection	SQL based input without stored procedures.

JDBC data source type	Supported versions	Connection type	Scope
SAP Hana	2.00.40 and newer	JDBC connection, Shared Storage connection	SQL based input and 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.
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 .
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.
Sybase Adaptive Server Enterprise	16.0 SP02 and newer	JDBC connection, Shared Storage connection	SQL based input without stored procedures.
Teradata	15.0, 16.20.07.01 and newer	JDBC connection, Shared Storage connection	SQL based input, including BTEQ scripts.

ETL tools

The following table shows the supported ETL tools.

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
Azure Data Factory	2	API	Commonly supported transformations and activities in Azure Data Factory. For details, go to Supported transformation details .
IBM InfoSphere DataStage	11.5 and newer	Folder	Commonly used DataStage ETL components including SQL overrides and transformation details. Collibra Data Lineages supports IBM InfoSphere DataStage transformation logic. You have to prepare a folder with all data objects that you want to process.
Informatica Intelligent Cloud Services, specifically Cloud Data Integration	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.
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.

Tip [Data Integration](#) is one of the Informatica Intelligent Cloud services.

ETL tool	Supported versions	Connection type	Scope
Matillion	Newest version	API	SQL based input without stored procedures. The lineage harvester can only access Redshift and Snowflake projects.
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.

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. The API connection type is not supported for Informatica Intelligent Cloud Services (IICS) and Matillion yet on Edge. You can use Shared Storage connections when you create the technical lineage for IICS and Matillion on Edge.

ETL tool	Supported versions	Connection type	Scope
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.
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.
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.

ETL tool	Supported versions	Connection type	Scope
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.
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.

BI tools

The following table shows the supported BI tools.

Lineage harvester

The following table shows the supported BI tools.

BI tool	Tested versions	Connection type
Tableau	Newest	API. You have to prepare: • lineage harvester configuration file for Tableau ingestion. • Optionally, a Tableau <source ID> configuration file.
Power BI (deprecated)	Newest	API and XMLA endpoints. You have to run the Power BI harvester and the lineage harvester to ingest Power BI metadata. Note Integration via the Power BI harvester is deprecated. We will continue to fix issues, but the development of new features and improvements is discontinued.
Power BI	Newest	API. The new Power BI integration includes many enhancements, including consolidated harvesters, meaning you no longer need the Power BI harvester. You only need to prepare: • lineage harvester configuration file for Power BI ingestion. • Optionally, a Power BI <source ID> configuration file.
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.

BI tool	Tested versions	Connection type
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.

BI tool	Tested versions	Connection type
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 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 repository, see the MicroStrategy repository documentation.

BI tool	Tested versions	Connection type
MicroStrategy (NEW)(Beta)	Newest	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.

Tip

For information about ingesting metadata from the following BI tools and creating a technical lineage via the lineage harvester, go to the dedicated sections:

- Tableau:
 - [Via the Data Catalog user interface](#).
 - [Via the lineage harvester](#).
- [Power BI \(deprecated\)](#)
- [Power BI](#)
- [Looker](#)
- [MicroStrategy](#)
- [MicroStrategy \(NEW\)\(Beta\)](#)
- [SQL Server Reporting Services and Power BI Report Server](#)

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

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
Tableau	Newest	API	Technical Lineage for Tableau
Power BI	Newest	API	Technical Lineage for Power BI

For information about ingesting metadata from the following BI tools and creating a technical lineage via Edge, go to the dedicated sections:

- [Tableau](#)
- [Power BI](#)

Custom technical lineage

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

To custom technical lineage on Edge, add the Technical Lineage for Custom Lineage and synchronize the technical lineage.

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

Authentication

Technical lineage supports the following means of authentication:

- For all data sources, except for [external directories](#): username and password.
- Google BigQuery data sources: username and password or a service account key file. For more information, see the [Google BigQuery documentation](#).
- Power BI: username and password or service principal.
- Snowflake: username and password or key pair authentication.
- Tableau: username and password or token-based authentication.
- No other authentication methods are supported.

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:

- [IBM DataStage](#)
- [Informatica PowerCenter](#)
- [Informatica Intelligent Cloud Services](#)
- [Snowflake](#)
- [SQL Server Integration Services](#)
- For the supported transformations in Azure Data Factory, go to [Supported Azure Data Factory transformation details and data sources](#).

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](#).

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 • 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. Collibra Data Lineage does not access the Informatica PowerCenter data sources, so Collibra 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 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 • 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 - Direct login - Stored procedures - The <code>COPY INTO</code> DML command - Streams 2	- Data Definition Language (DDL) queries - 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 `CREATE STREAM` statement.

SQL Server Integration Services (SSIS)

Collibra Data Lineage supports the following non-exhaustive list of transformations in SQL Server Integration Services:

- 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

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 keyword SELECT, or else they will be skipped. Furthermore, if a comment precedes the keyword SELECT, the query will be skipped.

Supported Azure Data Factory transformation details and data sources

Collibra Data Lineage supports the most commonly used [transformations](#) and [data sources](#) in Azure Data Factory ([Beta](#)) .

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 {@context} • 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.

Collibra Data 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

Data sources	Dataset type	Linked service type
Open Database Connectivity (ODBC)	OdbcTable	Odbc
On-premises Oracle database	OracleTable	Oracle
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

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 [Creating a Technical lineage via the lineage harvester](#) and [Create technical lineage via Edge](#).

For more information about Snowflake, go to [Snowflake Documentation](#).

Data Lineage integrations available in beta

Collibra Data Intelligence Cloud [supports](#) many data sources and metadata sources, such as ETL tools or BI sources, for which you can create a technical lineage or which you can ingest.

Before Collibra releases a new lineage harvester, we test the new lineage harvester and technical lineage via Edge integrations extensively. However, we cannot foresee all possible use cases and scenarios. To further improve the lineage harvester and technical lineage via Edge, you can now test new integrations via the lineage harvester and via in beta. After a testing period, the new integrations will be made available for all Collibra Data Lineage users.

The following table shows the technical lineage via lineage harvester integrations that are currently in beta.

Lineage harvester integrations	Available in lineage harvester version	Description	How to use
MicroStrategy integration, using the latest MicroStrategy APIs	2023.05 and newer	Benefits: • Supports the latest MicroStrategy APIs. • Allows for technical lineage and automatic stitching. • No longer dependent on a direct connection to the repository.	See MicroStrategy integration steps (NEW) (Beta)

Warning The lineage harvester and technical lineage via Edge beta integrations offer early access to new integrations. However, we can only allow a limited number of customers to test the integrations and give feedback. We will make the integrations available for all customers after processing the feedback and improving Collibra Data Lineage.

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.

Tip For more information about the technical lineage for [Looker](#) or [Power BI](#), we highly advise you to read the dedicated sections in the user guide.

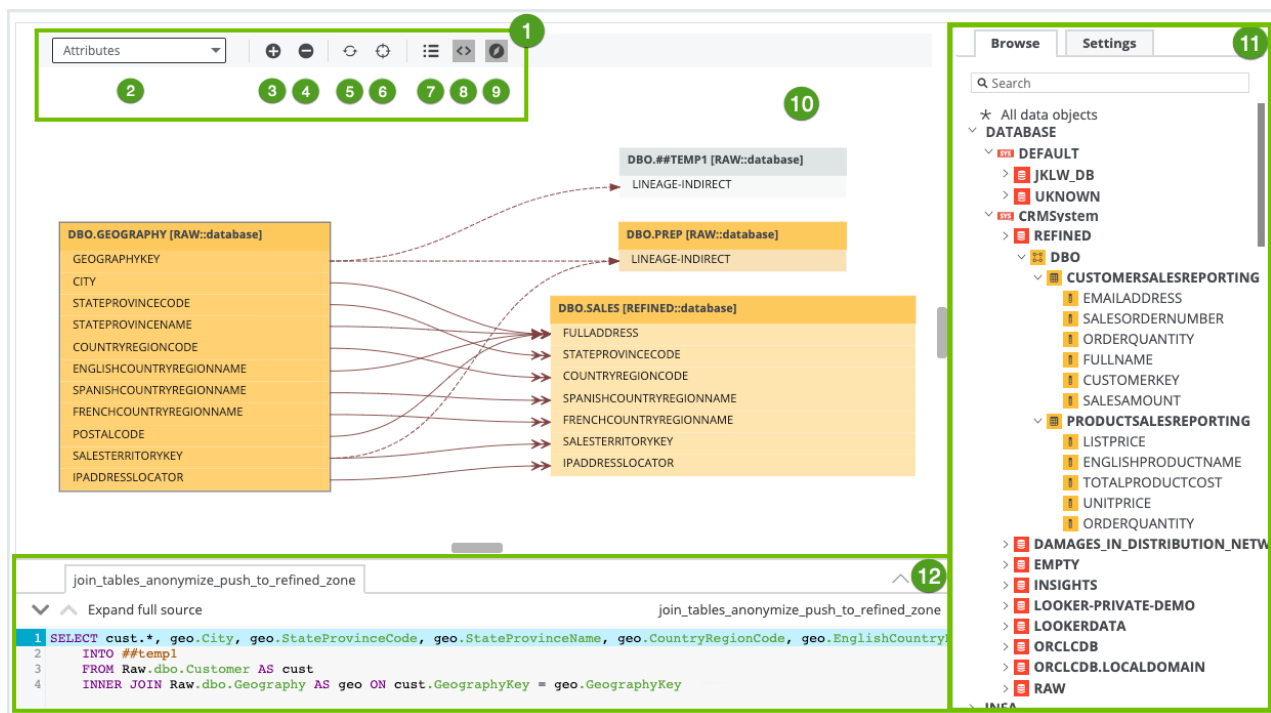
Technical lineage tab

You can only see the Technical lineage tab on a Column or Table asset page when you have the following prerequisites:

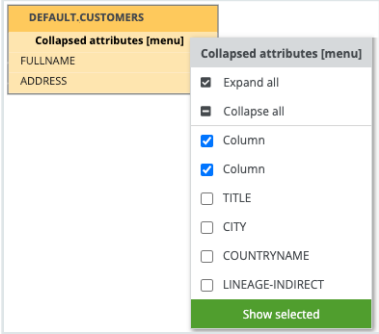
- You have a [global role](#) with the Catalog [global permission](#), for example, Catalog Author.
- You have a global role with the Technical lineage [global permission](#).

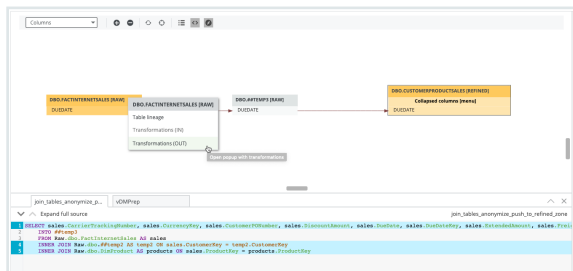
Note View permissions are not enforced in Collibra Data Lineage. This means that anyone with the Technical Lineage global permission can see all of the assets in a technical lineage graph, regardless of their view permissions as determined at the community or domain level.

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.

No	Name	Description
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.
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.
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.
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

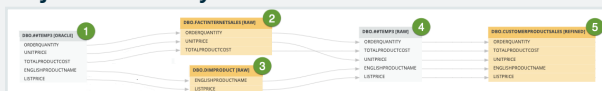
The background color of a node indicates whether or not the data object was stitched to an asset in Data Catalog, and whether something went wrong.

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 (Beta), which supports the latest MicroStrategy APIs.
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.



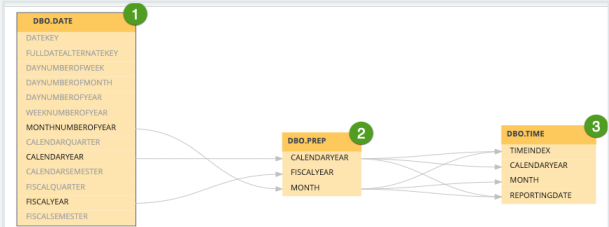
Font colors

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.

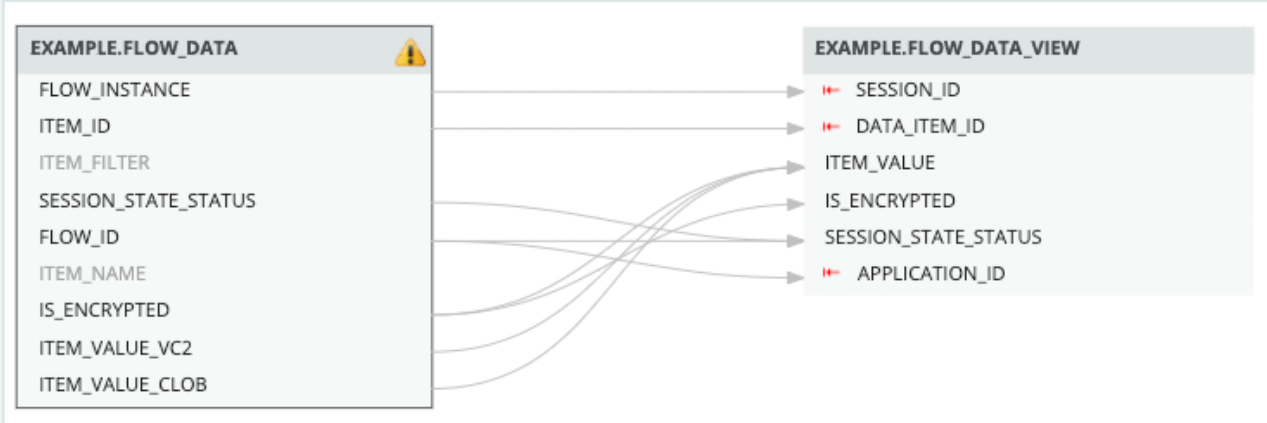


Icons

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:

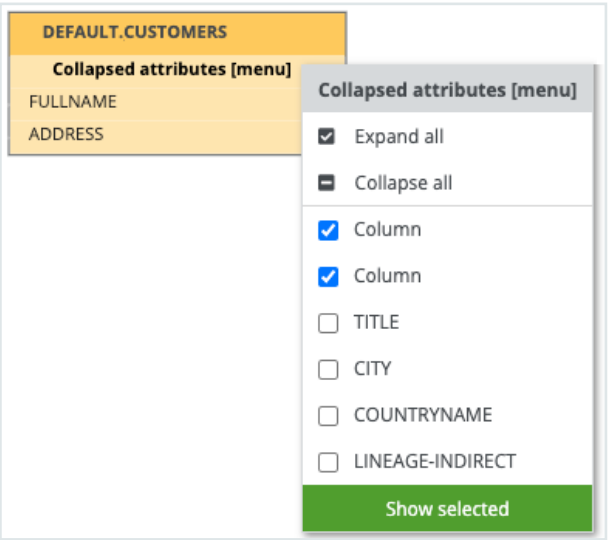
Arrow type	Description
Single	Shows the full lineage without skipping certain data objects.
Double	Shows that there are hidden data objects in the technical lineage graph. This happens when only the endpoints of the technical lineage flow are shown.

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.



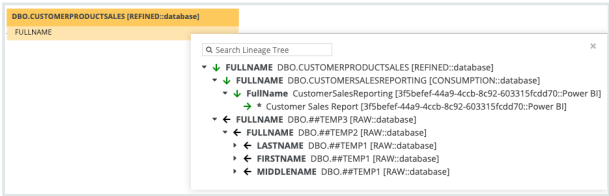
Collapsed attributes menu

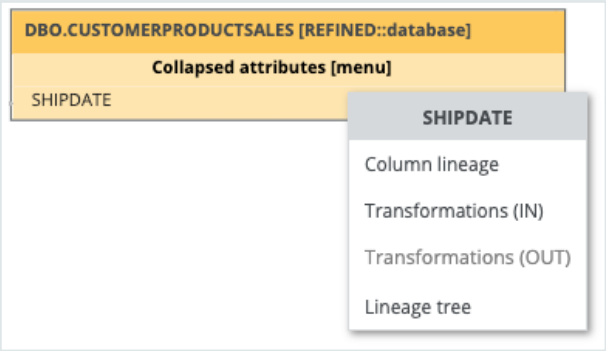
If you select a specific column in a table with multiple columns, you can click **Collapsed attributes [menu]** to show all columns, collapse all columns or only show selected columns in the same table.



Right-click menu

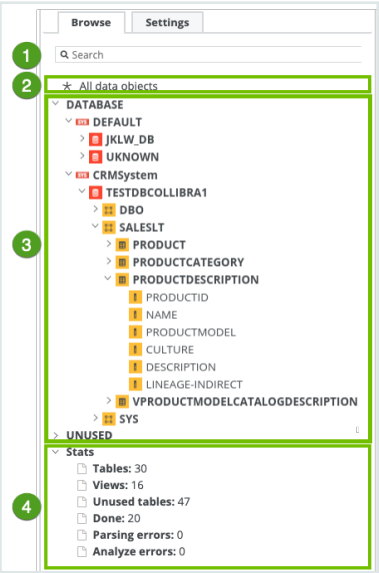
If you right-click a node, you can perform several specific actions on that node.

Functionality	Description
Column/Table lineage	Switch to the technical lineage graph of the selected column or table.
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. You can enter the name of a database, schema, table or column. Searching for a system name is not supported.
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.

No	Name	Description
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.

Browse Settings

1

2

Visualization options

☐ Group by parent object
☐ Only ending points

Depth MAX

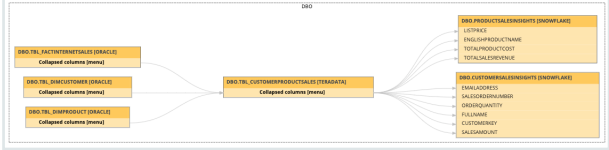
3

4 ☐ Show indirect dependencies

5 Export ▾

6 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, Colibra 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 by using the lineage harvester or technical lineage via Edge, 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
```

Sources

Stitching

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 transformations

Full-text search

2

Filter by

All

Export All Transformations

4

1

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 to

Show lineage

5

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

3

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.
	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.
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.

No	Name	Description
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

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](#).

The screenshot displays the Collibra Data Lineage service interface. It features a 'Sources' tab and a 'Stitching' tab. A search bar is located at the top left, with a magnifying glass icon and a search button. Below the search bar is a table with two columns: 'Full asset path' and 'Found In'. The table lists various data assets and their locations. On the right side, there is a 'Browse' tab and a 'Settings' tab. The 'Settings' tab contains options for visualization and dependencies. A 'Show lineage' button is located at the bottom of the settings panel.

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 > SQL201	Catalog
JD DEFAULT > INVENTORYMANAGEMENT > SQL281	Catalog
JD DEFAULT > INVENTORYMANAGEMENT > SQL721	Catalog
JD DEFAULT > INVENTORYMANAGEMENT > SQL763	Catalog
JD DEFAULT > INVENTORYMANAGEMENT > SQL763	Catalog

Settings

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.
2	Full asset path	The full path to all data objects on the Collibra Data Lineage service and all assets in Data Catalog.

No	Name	Description
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 .

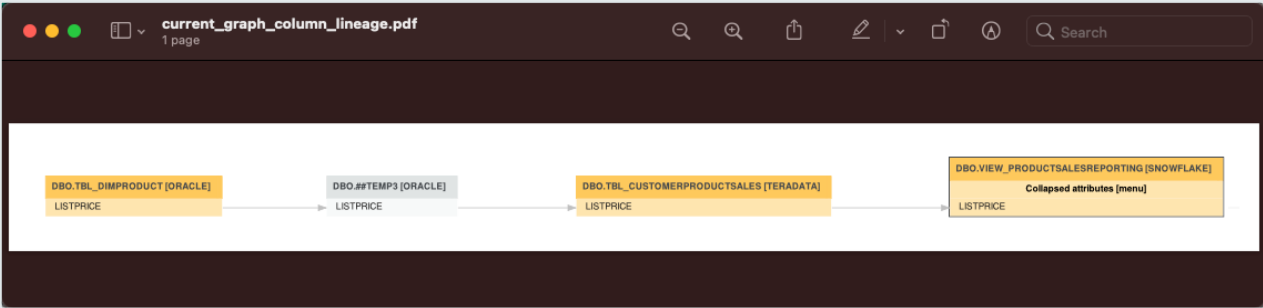
Technical lineage export types

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.

Steps

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.

Business Summary Lineage

The Business Summary Lineage is a representation of relations of the type "Data Element sources / targets Data Element" in a [business diagram](#). 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.

Note Click [here](#) for an overview of the differences between Technical lineage and a diagram with Business Summary Lineage.

You can [create](#) a new [diagram view](#) including the Business Summary 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.

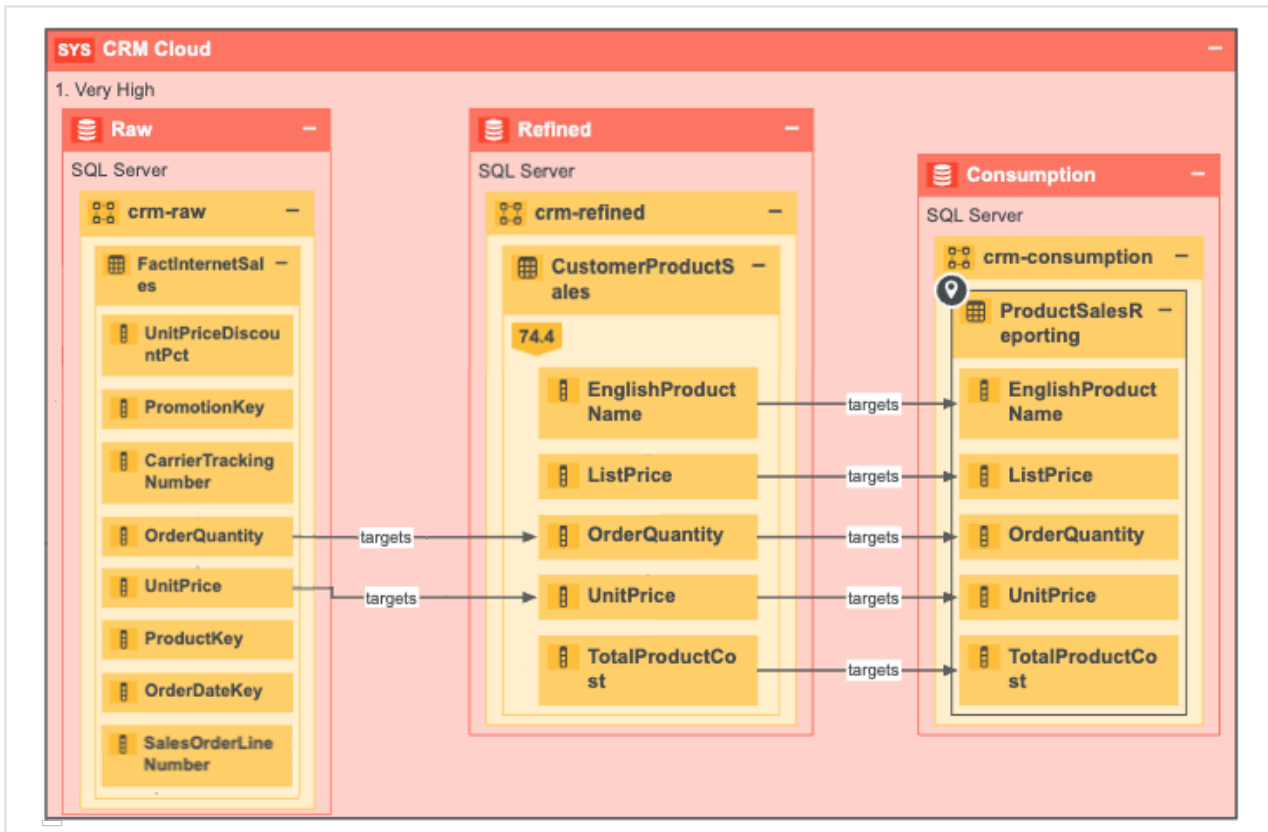
Before you can view a diagram with Business Summary Lineage, you have to:

- [Register](#) the data sources that you want to see in a diagram with Business Summary Lineage.
- [Prepare](#) a configuration file to create a technical lineage.
- Use the [lineage harvester](#) or [technical lineage via Edge](#) to upload the data sources in your configuration file to the Collibra Data Lineage service where they are scanned and processed.

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 diagram with Business Summary Lineage

In this business diagram, 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 diagrams with Business Summary Lineage

Technical lineage is a detailed lineage graph that shows where data objects are used and how they are transformed. A diagram with the Business Summary 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 diagram with Business Summary Lineage only provides a summary of it.

The [Business Summary Lineage](#) and a technical lineage are both visual representations of nodes. However, there are some key differences between them.

Tip For information on the steps required to create a technical lineage, including how to [prepare](#) the Data Catalog physical data layer, see [About technical lineage](#).

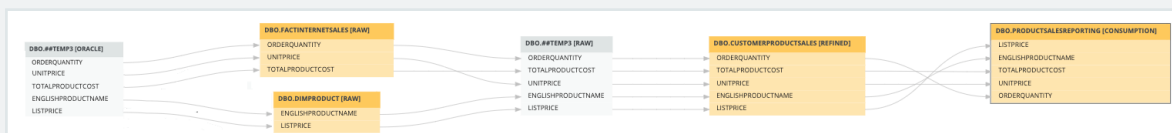
Business Summary Lineage	Technical lineage
A diagram with a Business Summary Lineage helps Business Analysts and other business users to understand their data by providing a summary of the technical lineage.	A technical lineage helps Data Engineers, Data Architects and similar personas to easily navigate to data objects in the data flows and find relevant source code fragments by providing a detailed lineage graph.
A diagram containing Business Summary Lineage is accessible via the Diagram tab pane of all assets.	A technical lineage is accessible via the tab pane of all Table assets and Column assets. You can view a technical lineage via the tab pane of Table assets and Column assets if you added their database as data sources in the configuration file .
A diagram shows assets and relations as defined in its diagram view. In the case of a Business Summary 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 diagram with a Business Summary 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

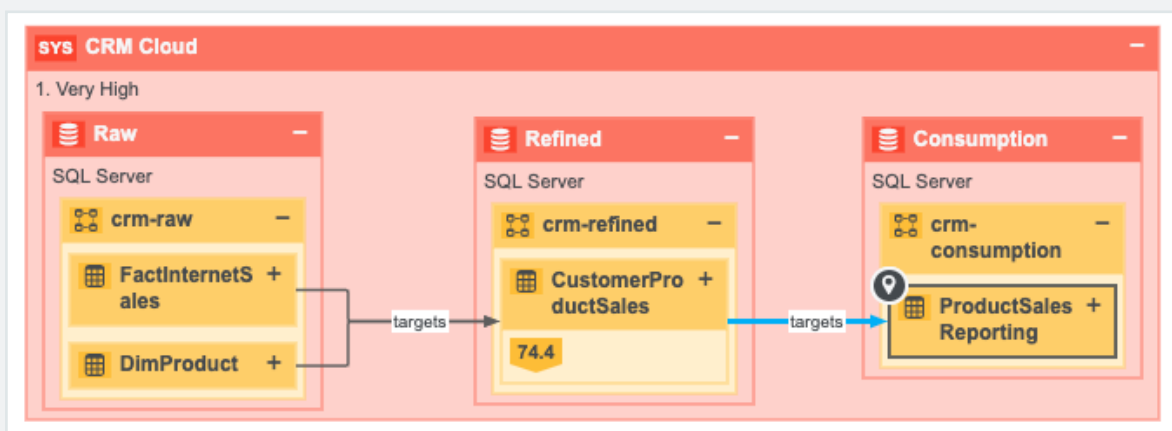
You have created a technical lineage for four different databases:

- 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.

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.



A diagram with Business Summary 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.

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 [Collibra 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 [Collibra 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.

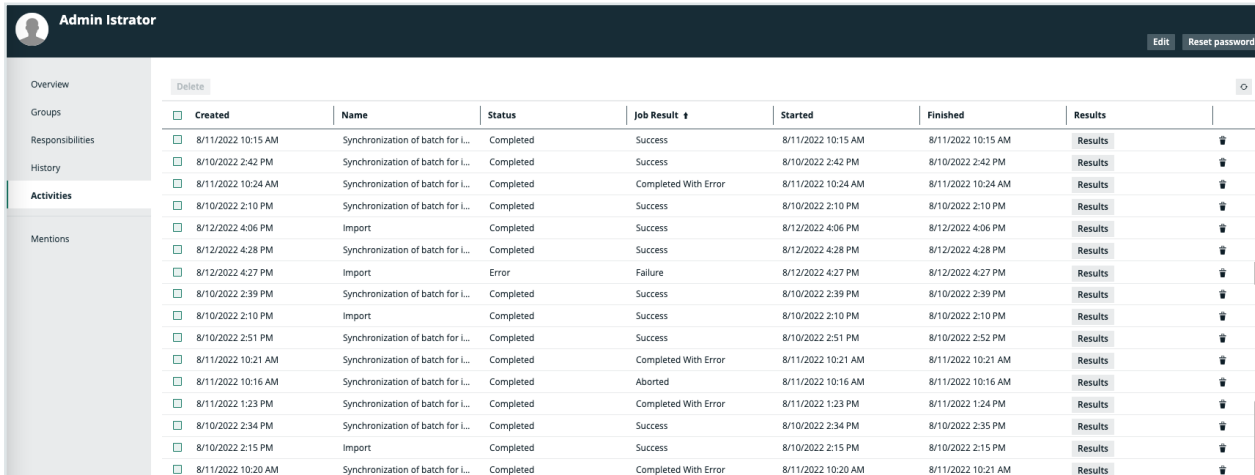
Job results

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](#).



Created	Name	Status	Job Result	Started	Finished	Results	
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	
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	
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.

Import			
Error List			
Number of Errors: (23)			
Resource Type	Index	Resource Identifier	Error Message
Asset	2277	{"externalSystemId":"ad046ece110634..."}	The maximum limit of relations (1) for this source (testtableau22 > Colibra_tab_p...
Asset	2276	{"externalSystemId":"ad046ece110634..."}	The maximum limit of relations (1) for this source (testtableau22 > Colibra_tab_p...
Asset	2279	{"externalSystemId":"ad046ece110634..."}	The maximum limit of relations (1) for this source (testtableau22 > Colibra_tab_p...
Asset	2283	{"externalSystemId":"ad046ece110634..."}	The maximum limit of relations (1) for this source (testtableau22 > Colibra_tab_p...
Asset	2268	{"externalSystemId":"ad046ece110634..."}	The maximum limit of relations (1) for this source (testtableau22 > Colibra_tab_p...
Asset	2287	{"externalSystemId":"ad046ece110634..."}	The maximum limit of relations (1) for this source (testtableau22 > Colibra_tab_p...
Asset	2281	{"externalSystemId":"ad046ece110634..."}	The maximum limit of relations (1) for this source (testtableau22 > Colibra_tab_p...
Asset	2275	{"externalSystemId":"ad046ece110634..."}	The maximum limit of relations (1) for this source (testtableau22 > Colibra_tab_p...
Asset	2289	{"externalSystemId":"ad046ece110634..."}	The maximum limit of relations (1) for this source (testtableau22 > Colibra_tab_p...
Asset	2273	{"externalSystemId":"ad046ece110634..."}	The maximum limit of relations (1) for this source (testtableau22 > Colibra_tab_p...

Technical lineage troubleshooting

This section describes what you can do when you encounter issues running the [lineage harvester](#), browsing through a [technical lineage](#) or [stitching](#) data source objects in your data source to existing assets in Data Catalog.

Technical lineage general troubleshooting

This topic contains the following information:

- [Most common issues](#)
- [Testing connectivity](#)
- [Password errors](#)

Most common issues

The following messages or other issues can appear when you run the lineage harvester, view a technical lineage or upload the new relations to Data Catalog via Collibra Data Lineage.

Tip For a list of all error codes and messages that the lineage harvester displays, please see the [lineage harvester error codes](#) section.

Problem	Solution
No suitable driver is found. Source '<datasource>' failed with exception: java.sql.SQLException: No suitable driver found for jdbc:snowflake://<hostname>	This error occurs when the configuration for the JDBC driver in the lineage harvester configuration file is not correct. For example a trailing slash is added to the host name. Examine the <code>hostname</code> in the lineage-harvester.conf file and ensure the hostname does not contain any slash (/).
You get the following error message: Could not find or load main class lineage.lineage-harvester-<version nr.>	This error message appears when the folder path to the lineage harvester is invalid. Check the folder path and make sure that it does not contain whitespaces.
You get the following error message: Failed to load file '<file-name>'. If the file is not in UTF-8, please convert it accordingly.	This error message appears if the lineage harvester tries to read a non-UTF-8 SQL file of a data source with connection type <code>SqlDirectory</code>. To solve this issue, convert all SQL files to UTF-8 and rerun the lineage harvester.
The lineage harvester does not connect to hosts using a proxy server.	Technical lineage supports proxy server authentication. For complete details, including the necessary commands, see Connecting to a proxy server.

Problem	Solution
You get the following error message or a similar certificate error: <pre>Source '<data source name>' failed with exception: javax.net.ssl.SSLHandshakeException: General SSLEngine problem</pre>	This message appears when the proxy server sends an unexpected certificate to the lineage harvester or when the default Java TrustStore is empty or outdated. First update Java and rerun the lineage harvester to see if that resolves the issue. If the same error message is shown, try the following: On Windows Note • In the following example commands, we refer to the techlin-gcp-us instance. You should refer to the correct Collibra Data Lineage service instance in the geographic location of your Collibra Data Intelligence Cloud environment. • Tableau is used as an example in the following steps. You can replace <code>tableau</code> in the example commands with your data source. 1. Run the following command to extract the certificate from the Tableau server: <pre>keytool -printcert -rfc - sslserver techlin-gcp-us.- collibra.com:443 > tableau- cert.crt</pre> Tip Replace the URL <code>techlin-gcp-us.collibra.com</code> with the URL for your Tableau server, which you specify in the lineage harvester configuration file. This will create a file named <code>tableau-cert.crt</code> in the folder where you run this command. 2. Run the following command to find the

Problem	Solution
	location of your JAVA_HOME: <pre>echo %JAVA_HOME%</pre> » The location path will be something like the following: C:\Program Files\Java\jdk-17.0.2 3. Use the location path of your JAVA_HOME in the following command, to import the tableau-cert.crt file into the cacerts file found above. <pre>keytool -importcert -file tableau-cert.crt -alias "TableauProdServerCert" -keystore "C:\Program Files\Java\jdk-17.0.2\cacerts"</pre> Note You can specify a different alias, if you want. 4. Run the following command: <pre>keytool -list -keystore "C:\Program Files\Java\jdk-17.0.2\lib\security\cacerts" findstr "Tableau"</pre> 5. Enter the keystore password. Tip The password is typically changeit. » A list of all certificates that match the Tableau string in the "C:\Program Files\Java\jdk-17.0.2\cacerts" file is shown.

Problem	Solution
	Tip In the list of certificates, look for the one that you imported in step 3. If it's listed, it means the "C:\Program Files\Java\jdk-17.0.2\cacerts" file has the certificate needed to validate the Tableau server. 6. Run the following command to have the lineage harvester use the cacerts file that you just updated. <pre>set JAVA_OPTS=- Djavax.net.ssl.trustStore="C :\Program Files\Java\jdk- 17.0.2\lib\security\cacerts" - Djavax.net.ssl.trustStorePas sword="changeit"</pre> 7. Run the following command to test the synchronization: <pre>./lineage-harvester.bat full-sync -s tableau</pre> On Linux

Problem	Solution
	Note • In the following example commands, we refer to the <code>techlin-gcp-us</code> instance. You should refer to the correct Collibra Data Lineage service instance in the geographic location of your Collibra Data Intelligence Cloud environment. • If you want to add an existing certificate to the Java TrustStore, instead of creating a new Keystore, replace "<code><your keystore name></code>" in steps 2 and 3, with the path to the <code>cacerts</code> file in your Java installation, for example <code>%JAVA_HOME%\jre\lib\cacerts</code>. 1. Use the following command to get a certificate from the corresponding <code>techlin-gcp-us.com</code> site, which is part of the CollibraData Lineage infrastructure: <pre>openssl x509 -in <(openssl s_client -connect techlin-gcp-us.collibra.com:443 -prexit 2>/dev/null) -out techlin-gcp-us.crt</pre> Tip If you already have a correctly formatted certificate on the server, you can skip this step. 2. Add the certificate to the Java TrustStore: <pre>keytool -importcert -file techlin-gcp-us.crt -alias techlin-gcp-us -keystore <your keystore name> -storepass changeit</pre>

Problem	Solution
	3. Run the lineage harvester and use the new TrustStore using the following parameter: <pre>-Djavax.net.ssl.trustStore= e=<your keystore name></pre> Example To synchronize your data sources again, run the following command: <pre>./bin/lineage- harvester full-sync - Djavax.net.ssl.trustSt ore=mykeystore</pre>
You get the following error messages: In the lineage harvester log file: <pre>java.lang.Exception: No native library found for os.name=Linux, os.arch=x86_64, paths= [/org/sqlite/native/Linux/x86_ 64:/usr/java/packages/ lib/amd64:/usr/lib64:/lib64:/lib:/u sr/lib]</pre> In the console: <pre>Failed to load native library:<sqlite-file-name>. osinfo: Linux/x86_64 java.lang.UnsatisfiedLinkError: /tmp/<sqlite-file-name>: failed to map segment from shared object: Operation not permitted</pre>	The lineage harvester uses a temporary file containing an SQLite database as a cache file. That means that you need write permission to the /tmp folder. If this action failed, you can specify another directory with write permissions using – Dorg.sqlite.tmpdir=<path to a temp directory>. Example You have a temporary directory with write permissions. The path to this directory is custom/temp. Run the lineage harvester with the following command: <pre>./bin/lineage-harvester - Dorg.sqlite.tmpdir=custo m/temp full-sync</pre>

Problem	Solution
The lineage harvester configuration file is not specified correctly. <pre>harvester.error.ConfigurationProblem: Bad configuration: .sources[0] (expected ',' or '}' got '"')</pre>	The lineage harvester configuration file includes JSON syntax errors. Examine the lineage harvester configuration file and correct any errors. In this error message, there is a missing comma (,) or curly bracket ({}).
You get the following error message: <pre>Technical lineage is not enabled for this Catalog instance.</pre>	First make sure that there are no spelling errors in the <code>Data Catalog</code> section of the configuration file. If your configuration file is configured correctly, but the issue is not solved, create a support ticket to enable Technical lineage for your Collibra Data Intelligence Cloud instance in Salesforce.
You get the following error message: <pre>The size of the import file is too large (max size: 10 MB).</pre>	The file you are trying to upload exceeds the size limit for uploaded files. Contact Collibra Support to increase the maximum file size.
You get the following error message: <pre>Source 'X' was never successfully processed..</pre>	This message appears when a source that is specified in the lineage harvester configuration file has never been successfully processed by the Collibra Data Lineage service. You can either: • Remove source 'X' from the configuration file, and then run the command again. • Run a full-sync of source X, and then re-run the command that previously failed.
Technical lineage is unavailable because the selected table does not contain columns.	Technical lineage only includes tables that have columns. Add a relation of the type "Table contains/is part of Column" between your Table asset and Column assets.

Problem	Solution
You get the following message in your technical lineage: <pre>The current asset doesn't have a technical lineage yet.</pre>	This message appears if one or more of the following situations apply: • The data source of the current asset is not included in the configuration file. If you want a technical lineage for this asset, add its data source to the configuration file. • You have upgraded to the lineage harvester 1.3.0 or newer or you created a technical lineage for the first time. In this case, you may need to restart your DGC service before you can see the technical lineage. • You see parsing errors. For more information, see the Sources tab page. • The full name of one or more relevant assets does not match (including for case-sensitivity) any of the names of the assets in the configuration file, which causes automatic stitching to fail. Make sure that the information in the configuration file and the Data Catalog physical data layer matches: ◦ The relevant assets have relations between each other, for example <i>Technology asset groups/is grouped by Technology asset</i> → <i><Database asset> contains/is part of <Schema asset> contains/is part of <Table asset> contains/is part of <Column asset></i>. ◦ The full name of your System asset matches the name of your system or the name you used in the configuration file. ◦ The full name of your Database asset matches the name of your database or, for Google BigQuery your project,

Problem	Solution
	or the name you used in the configuration file. ◦ The full name of your Schema asset matches the name of the Schema of the data source or the name you used in the configuration file. Tip Make sure that the full path of each asset in Data Catalog matches the full path of the corresponding data object from your data source on the Stitching tab page. Note that in Collibra, full paths are case-sensitive.
You get one of the following messages: • Nodes count exceeds the limit 350. • Edges count exceeds the limit 1000.	This message appears when the technical lineage graph exceeds the limit of 350 nodes or 1,000 edges, and is too large to build. 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. If you see this message, we recommend that you browse through the technical lineage graph on the object level or select a single column in the Browse tab pane. Note You cannot manually change these limits.

Problem	Solution
You get the following error message in your technical lineage for a Microsoft SQL Server data source: "Oops, no data flow found in your SQL scripts. Make sure you upload DML queries like insert, update, merge that moves data between the tables."	This error message appears when you run the lineage harvester to create a technical lineage for a Microsoft SQL Server data source without having the correct permissions to the SQL Server. As a result, the lineage harvester processes empty files and there is no technical lineage available for this data source. Make sure you have at least the VIEW DEFINITION permission or sysadmin role in Microsoft SQL Server. Note If you use multiple users, make sure that each one of them has the proper permissions.

Problem	Solution
You get the following error message: <pre>net.snowflake.client.jdbc.Snowflake SQLLoggedException: JDBC driver internal error: Fail to retrieve row count for first arrow chunk: sun.misc.Unsafe or java.nio.DirectByteBuffer.<init> (long, int) not available.</pre>	The issue is related to the Arrow library, a dependency of the Snowflake JDBC driver. The issue has not been resolved in the Snowflake JDBC driver; to get rid of the error, set the <code>JAVA_OPTS</code> 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: <pre>set JAVA_OPTS="-Djdk.- mod- ule.il- legalAccess=permit"</pre> ◦ If you use OpenJDK 17 or higher: <pre>set JAVA_OPTS="--add- opens- =java- a.base/java.nio=ALL- UNNAMED"</pre> 2. In the same command line, enter the following command: <pre>.\bin\lineage-har- vester.bat full-sync</pre> Note The <code>set</code> command is specific to the Windows Command Shell. The command is different if you are using PowerShell.

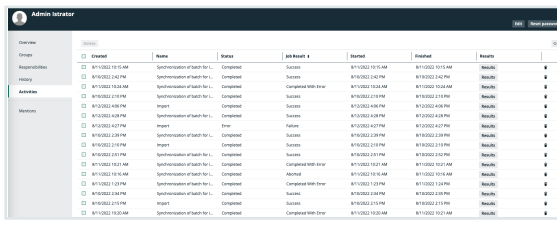
Problem	Solution
	On Linux Enter the following command: • If you use OpenJDK 16: <pre> JAVA_OPTS="-Djdk.- mod- ule.il- legalAccess=permit" ./bin/lineage-harvester full-sync </pre> • If you use OpenJDK 17 or higher: <pre> JAVA_OPTS="--add-opens- =java- a.base/java.nio=ALL- UNNAMED" ./bin/lineage- harvester full-sync </pre>
You get the following error message: <pre> Source 'SnowflakeInfo' failed with exception: net.snowflake.client.jdbc.Snowflake SQLException: SQL compilation error: Database 'SNOWFLAKE' does not exist or not authorized. </pre>	To access the Snowflake shared ready-only database, you need a user with a role that has the IMPORTED PRIVILEGES privilege. If the privilege is not assigned to the default role in Snowflake, 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>

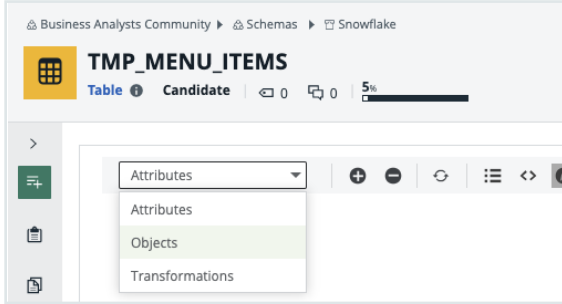
Problem	Solution
The import job fails. Note If the import job fails during import and the failing job is rolled back, you can have both old and new relations. The old relations were created during the first job and the new relations are created after the rollback. If more than one job is triggered, only the failed job is rolled back.	First, check the following: • The asset ID must exist. • The structure of the data must be correct. • The cardinality of relation types between asset types. Then, rerun the import of relations.
Relations are not changed as expected.	Check whether the lineage harvester refreshed the data source via a scheduled job. If the import job failed, then the data source was not refreshed and the previously created relations stay the same. If that happened, rerun the lineage harvester to import again.
Manual relations are overwritten.	We recommend that you do not manually add relations of the type "Data Element targets / sources Data Element" between asset types that are imported via the scheduled jobs. These relations are overwritten every time the scheduled job synchronizes the data source.
Ingesting Looker or Power BI assets fails.	For more information, see the following sections: • Looker troubleshooting. • Power BI troubleshooting

Problem	Solution
You get the following error message: <pre>java.lang.OutOfMemoryError: Java heap space</pre>	This error message indicates that Java does not have enough memory allocated to finish the task. This error can happen anytime during Harvester run. Follow these steps to increase the maximum heap size. Note 4 GB RAM is sufficient in most cases, but more memory could be needed for larger harvesting tasks. On Windows Run the following command: <pre>set JAVA_OPTS=-Xmx4g && .\bin\lineage-harvester.bat full-sync</pre> In this example, 4g means 4 GB. Note The <code>set</code> command is specific to the Windows Command Shell. The command is different if you are using PowerShell. Tip If you want to check the default maximum heap size, run the following command: <pre>java -XX:+PrintFlagsFinal -version findstr MaxHeapSize</pre> On Linux Run the following command: <pre>JAVA_OPTS=-Xmx4g ./bin/lineage-harvester full- sync</pre> In this example, 4g means 4 GB.

Problem	Solution
	Tip If you want to check the default maximum heap size, run the following command: <pre>java -XX:+PrintFlagsFinal -version grep MaxHeapSize</pre>
You get the following error message: <pre>java.lang.NoSuch.MethodError</pre>	This error message indicates that the JAVA_HOME was not specified; therefore, the harvester was using a previous version of Java. With the following commands, you can specify the Java version to 11, which is needed to successfully run the lineage harvester: • <code>export JAVA_HOME=/us</code> • <code>echo \$JAVA_HOME</code>

Problem	Solution
You get the following error message: <pre>Error: A JNI error has occurred, please check your installation and try again Exception in thread "main" java.lang.UnsupportedClassVersionError: harvester/Harvester has been compiled by a more recent version of the Java Runtime (class file version 55.0), this version of the Java Runtime only recognizes class file versions up to 52.0 ...</pre>	In this error message, class file versions up to 52.0 indicates that Java 8 was used; however, lineage harvester requires Java version 11 or newer. If there are multiple versions of Java installed, the lineage harvester might pick Java 8 instead of Java 11. You can run the command <code>java -version</code> to check your Java version. To resolve this issue, set the path to the correct Java installation directory, in the <code>JAVA_HOME</code> environment variable. To do so, run the lineage harvester with the following command: On Windows: <pre>set JAVA_HOME=\path\to\java_11_dir && .\bin\lineage-harvester.bat full-sync</pre> Note The <code>set</code> command is specific to the Windows Command Shell. The command is different if you are using PowerShell. On Linux: <pre>JAVA_HOME=/path/to/java_11_dir ./bin/lineage-harvester full-sync</pre>

Problem	Solution
You get a <code>NegativeArraySizeException</code> error.	A <code>NegativeArraySizeException</code> error is shown if your Java Virtual Machine (JVM) has the string compaction feature disabled. In that case, calling the <code>getBytes</code> results in an attempt to allocate triple the size of the string's value, which can exceed the size limit. To resolve this error, try running the lineage harvester with the string compaction feature enabled, by running the following command: <pre>JAVA_OPTS='-XX:+CompactStrings -Xmx8g' ./lineage-harvester full-sync -s <source ID></pre>
Synchronization of a data source fails completely or with some errors.	A synchronization job that is completed without any errors has the Success status. Other possible statuses, Completed With Error, Aborted and Failure, are determined in part by the value of the "Number of failed commands before stopping import job" setting, in Colibra Console. For complete information, see Synchronization: Continue on error option. You can view the results of a synchronization job in the Activities list.  In the Activities list, click Results in the relevant row to view the details of a synchronization job. The details are intended to help you resolve the errors. To help reduce the chance of an aborted synchronization job, consider increasing the value of the "Number of failed commands before stopping import job" setting.

Problem	Solution
The technical lineage viewer does not show the technical lineage.	If a technical lineage graph does not show up, change the technical lineage graph details on the tool bar to Objects.  For more information, go to Technical lineage viewer.

Testing connectivity

You can check whether the lineage harvester can connect to the [Collibra Data Lineage service instance](#) and Data Catalog.

1. Run the lineage harvester in command line.
2. Run the following command: `test-connection`.
 - » The result shows if the lineage harvester can connect to the Collibra Data Lineage service instance and Data Catalog.

The logs will also show the IP addresses of the Collibra Data Lineage service instances that you have to allow.

Password errors

If you mistyped the password or want to change an existing password, go to the [lineage harvester](#) folder > **config/pwd.conf** and delete the lines below. As a result, the lineage harvester will ask for the password again.

Tip If you have the lineage harvester version 1.3.0 or newer, you can also provide your [passwords](#) via stdin or a password manager.


```
{
  "url" : "<URL>",
  "userName" : "<user>",
  "password" : "<password>"
}
```

Technical lineage known issues and limitations

The following table shows known issues and limitations in the current lineage harvester version.

Important The success rate of a technical lineage, as shown in the [Sources tab page](#), 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.

Known issue	Description
The lineage harvester currently does not support Java version 16.	If you have Java version 16 and run the lineage harvester with the <code>full-sync</code> command, the harvester fails during the API key retrieval process. As a workaround, we recommend the following: 1. Set the <code>JAVA_OPTS</code> to the following: <pre>JAVA_OPTS='--illegal-access=deny'</pre> 2. Run the lineage harvester in the same command line window.
Collibra Data Lineage does not reuse the database model or DDL statements from other sources in the lineage harvester configuration file.	Currently, all sources in the lineage harvester configuration file are analyzed separately. As a result, the database model and DDL statements that are used for one source are not taken into account when analyzing another source. As a workaround, we recommend that you make sure that each source has all DDL statements that it needs to be processed properly. Tip Saving the DDL statements in separate files and adding the preview <code>"_"</code> before their names might speed up the analysis of the DDL statements.

Known issue	Description
Harvesting an Amazon Redshift data source fails when using a CDATA JDBC driver.	If you use a CDATA JDBC driver to harvest metadata from an Amazon Redshift data source, you have to set the <code>queryPassthrough</code> property in the connection configuration to <code>true</code>, otherwise the driver fails to execute the query. Note The <code>queryPassthrough</code> property is only required if you are using the CDATA Redshift driver, for ingestion via Edge. The lineage harvester via CLI uses the native Redshift driver, which does not require the <code>QueryPassthrough</code> property.

Lineage harvester messages

A message code is shown in the lineage harvester logs when something goes wrong during the [lineage harvester process](#). The message code indicates which part of the harvesting process was skipped or failed and provides steps to resolve it.

General lineage harvester messages

Message code	Description
MSG-LIN-1001	The current asset does not have a technical lineage yet. Only assets that are processed and stitched by Collibra Data Lineage have a Technical lineage. Look for the asset name in the navigation tree of the Browse tab pane, to see if the asset was processed. • If the asset name is not shown in the navigation tree, ensure that the data source of the asset is included in the configuration file. • If the asset name is shown in the navigation tree, ensure that you correctly prepared the Data Catalog physical data layer for technical lineage before you run the harvester. Specifically, the full path of each asset in Data Catalog must match the full path of the corresponding data object from your data source on the Stitching tab page. Note that in Collibra, full paths are case-sensitive. Less likely factors, such as your lineage harvester version and parsing errors can also lead to this error. For complete troubleshooting information, see Technical lineage general troubleshooting.
MSG-LIN-2501	When the full-sync command was entered to create a technical lineage, synchronization processing failed because a duplicate BI tool data source was added with different source IDs in the lineage harvester configuration file. To resolve this issue, complete the following steps: 1. Enter the <code>list-sources</code> command and review the listed data sources to identify the sources IDs for the same data source. 2. Enter the <code>ignore-source</code> command with one of the source IDs of the duplicated data source specified. 3. Remove the configuration of the data source with the source ID that you ignored in step 2 from the lineage harvester configuration file. 4. Enter the <code>full-sync</code> command to synchronize the technical lineage again.
MSG-LIN-3000	This is an unknown or unclassified lineage harvester error. Create a support ticket to report the issue.

Message code	Description
MSG-LIN-3001	The lineage harvester was able to successfully connect to the Collibra Data Lineage service instances, but received HTTP client error response. If the error message contains Technical lineage is not enabled for this Catalog instance, do the following: • Make sure that the URL to your Collibra Data Intelligence Cloud in the <code>catalog</code> section of the lineage harvester configuration file is correct. • Make sure that the username and password you use to sign in to Collibra are correct. • Make sure that Collibra Data Lineage is enabled for your Collibra environment. If the error message contains Enter a valid URL, do the following: • This error is caused by an invalid URL. Make sure that the URL to your Collibra Data Intelligence Cloud in the <code>catalog</code> section of the lineage harvester configuration file is correct. If the issue persists, please contact Collibra support or your customer success manager.
MSG-LIN-3002	The lineage harvester was able to successfully connect to an instance of the Collibra Data Lineage service, but received an HTTP server error response. Wait a few minutes and then run the lineage harvester again. If the issue persists, please contact Collibra support or your customer success manager.
MSG-LIN-3003	The lineage harvester failed to retrieve the API key of your Collibra Data Intelligence Cloud environment with Data Catalog from the Collibra Data Lineage service instances due to network connectivity issues. To resolve this issue, do the following: • Check your network connectivity. • Make sure you have allowed the IP addresses of all Collibra Data Lineage service instances. • Check your proxy settings. Tip You can test your connectivity using the <code>test-connectivity</code> command.

Message code	Description
MSG-LIN-3004	Unable to determine the geographic location of your Collibra Data Intelligence Cloud environment. When you run the lineage harvester, it firsts connects to any available Collibra Data Lineage service instance to determine your cloud provider and geographic location of your Collibra environment. Then, the lineage harvester sends the harvested metadata to the Collibra Data Lineage service instance with the same cloud provider and geographic location. In this case, the geographic location of your Collibra environment could not be determined. If the issue persists, please contact Collibra support or your customer success manager.

Message code	Description
MSG-LIN-3005	Connection error due to Snowflake DB Client. The lineage harvester encountered an error through the Snowflake database connector SDK. This issue is specific to the Snowflake connector, not the lineage harvester. To resolve this issue, set the <code>JAVA_OPTS</code> 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 - Enter one of the following commands: - If you use OpenJDK 16: <pre>set JAVA_OPTS="-Djdk.moduleillegalAccess=permit"</pre> - If you use OpenJDK 17 or higher: <pre>set JAVA_OPTS="--add-opens=java.base/java.nio=ALL-UNNAMED"</pre> - In the same command line, enter the following command: <pre>.\bin\lineage-harvester.bat full-sync</pre> Note The <code>set</code> 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: <pre>JAVA_OPTS="-Djdk.moduleillegalAccess=permit" ./bin/lineage-harvester full-sync</pre>

Message code	Description
	If you use OpenJDK 17 or higher: <pre> JAVA_OPTS="--add-opens=java.base/java.nio=ALL-UNNAMED" ./bin/lineage-harvester full-sync </pre>
MSG-LIN-4000	The Collibra Data Lineage service instance is unable to connect to Data Catalog. To resolve this issue, try the following: - Check your network connectivity. - Make sure that the URL to your Collibra Data Intelligence Cloud in the <code>catalog</code> section of the lineage harvester configuration file is correct. - Make sure the host names of all databases in the lineage harvester configuration file are correct. If the issue persists, please contact Collibra support or your customer success manager.
MSG-LIN-19001	Connection not defined in config file. This means that a connection to the system or server could not be established. To resolve this issue, try to ensure that you have correctly prepared your Informatica Intelligent Cloud Services <source ID> configuration file.
MSG-LIN-19002	Taskflow failed to process because of missing connection definition. A taskflow could not be processed because one of the mappings in the taskflow refers to a connection that could not be extracted from the <source ID> configuration file. To resolve this issue, try to ensure that you have correctly prepared your Informatica Intelligent Cloud Services <source ID> configuration file.
MSG-LIN-19003	Both the auth and username properties exist in the lineage harvester configuration file. Lineage harvester processing ends. The username property is deprecated. Take the following steps: - Update the lineage harvester configuration file by using the auth property only. - Run the lineage harvester again.

SQL scanner messages

Message code	Steps to resolve the issue
MSG-LIN-5001	This is an unexpected error. Create a support ticket to report your issue.
MSG-LIN-5002	<Object> not found, please provide DDL or object definition. The scanner for SQL statements couldn't successfully complete its analysis, due to a missing object definition. The error message includes the name of the object. This happens, for example, in a scenario whereby a SQL statement such as "CREATE TABLE TMP AS SELECT * FROM ACCOUNTS" is uploaded, but the definition for the table ACCOUNTS was not uploaded. In this case, it's impossible to extract lineage information, as the structure of the table ACCOUNTS is unknown. Therefore "*" cannot be expanded to an actual list of columns, which results in the error. To resolve this issue, if you are uploading SQL statements as files, you need to ensure that you provide DDL for all objects. You can inspect the lineage harvester output to identify the queries that are using the object in the error.

Tableau general troubleshooting

The following messages and issues can appear when you run the lineage harvester, view a technical lineage or upload the new relations to Data Catalog via Colibra Data Lineage.

Problem	Solution
You get connectivity issues with a 401001 error code. Unfortunately, 401001 is a very general error code, returned by a Tableau API, that can refer to many issues, including but not limited to the following: • The lineage harvester configuration file was configured with the wrong password or Tableau site ID. • SSO authentication was used, which is not supported.	Ensure that the user/token that you intend to use to ingest Tableau assets can authenticate to your Tableau APIs via the command line, from the server on which you intend to install and run the lineage harvester. You can test your ability to authenticate by making the signin API call, using a cURL command. You can also try checking the login request that the lineage harvester is sending to the Tableau server. For complete information and guidance on how to test your ability to connect to the Tableau server and authenticate, see Test connectivity with the Tableau server. The authentication token is automatically refreshed in the case of an HTTP 401 Unauthorized error. If the token is refreshed, the log file includes the following line: <code>Refreshing token</code>. If a refreshed token doesn't resolve the issue, the problem could be that there are too many concurrent HTTP calls, using the same token. Try using the optional <code>concurrencyLevel</code> property in your lineage harvester configuration file, to indicate that you want HTTP requests to be run in a synchronous manner, instead of in parallel.
The lineage harvester does not connect to hosts using a proxy server.	Technical lineage supports proxy server authentication. For complete details, including the necessary commands, see Connecting to a proxy server.
You get a TCP timeout error.	To avoid TCP timeout errors, try configuring the Linux TCP keepalive setting: 1. Edit your <code>/etc/sysctl.conf</code> file: <pre># vi /etc/sysctl.conf</pre> 2. Add the following settings: <pre>net.ipv4.tcp_keepalive_time = 60 net.ipv4.tcp_keepalive_intvl = 10 net.ipv4.tcp_keepalive_probes = 6</pre> 3. To load the settings, run the following command: <pre># sysctl -p</pre>

Problem	Solution
The designation "UNDEFINED" is shown in the technical lineage.	If you are using a Tableau <source ID> configuration file , ensure that you have specified a value for the relevant <code>col-libraSystemName</code> property.

Problem	Solution
You get the following error message or a similar certificate error: <pre>Source '<data source name>' failed with exception: javax.net.ssl.SSLHandshakeException: General SSLEngine problem</pre>	This message appears when the proxy server sends an unexpected certificate to the lineage harvester or when the default Java TrustStore is empty or outdated. First update Java and rerun the lineage harvester to see if that resolves the issue. If the same error message is shown, try the following: On Windows Note - In the following example commands, we refer to the techlin-gcp-us instance. You should refer to the correct Collibra Data Lineage service instance in the geographic location of your Collibra Data Intelligence Cloud environment. - Tableau is used as an example in the following steps. You can replace <code>tableau</code> in the example commands with your data source. - Run the following command to extract the certificate from the Tableau server: <pre>keytool -printcert -rfc -sslserver techlin-gcp-us.collibra.com:443 > tableau-cert.crt</pre> Tip Replace the URL <code>techlin-gcp-us.collibra.com</code> with the URL for your Tableau server, which you specify in the lineage harvester configuration file. This will create a file named <code>tableau-cert.crt</code> in the folder where you run this command. - Run the following command to find the location of your <code>JAVA_HOME</code>: <pre>echo %JAVA_HOME%</pre> » The location path will be something like the following: <pre>C:\Program Files\Java\jdk-17.0.2</pre> - Use the location path of your <code>JAVA_HOME</code> in the following command, to import the <code>tableau-cert.crt</code> file into the <code>cacerts</code> file found above. <pre>keytool -importcert -file tableau-cert.crt -alias "TableauProdServerCert" -keystore "C:\Program Files\Java\jdk-17.0.2\cacerts"</pre>

Problem	Solution
	Note You can specify a different alias, if you want. 4. Run the following command: <pre>keytool -list -keystore "C:\Program Files\Java\jdk-17.0.2\lib\security\cacerts" findstr "Tableau"</pre> 5. Enter the keystore password. Tip The password is typically <code>changeit</code>. » A list of all certificates that match the Tableau string in the "C:\Program Files\Java\jdk-17.0.2\cacerts" file is shown. Tip In the list of certificates, look for the one that you imported in step 3. If it's listed, it means the "C:\Program Files\Java\jdk-17.0.2\cacerts" file has the certificate needed to validate the Tableau server. 6. Run the following command to have the lineage harvester use the cacerts file that you just updated. <pre>set JAVA_OPTS=-Djavax.net.ssl.trustStore="C:\Program Files\Java\jdk-17.0.2\lib\security\cacerts" -Djavax.net.ssl.trustStorePassword="changeit"</pre> 7. Run the following command to test the synchronization: <pre>./lineage-harvester.bat full-sync -s tableau</pre> On Linux

Problem	Solution
	Note • In the following example commands, we refer to the techlin-gcp-us instance. You should refer to the correct Collibra Data Lineage service instance in the geographic location of your Collibra Data Intelligence Cloud environment. • If you want to add an existing certificate to the Java TrustStore, instead of creating a new Keystore, replace "<your keystore name>" in steps 2 and 3, with the path to the <code>cacerts</code> file in your Java installation, for example <code>%JAVA_HOME%\jre\lib\cacerts</code>. 1. Use the following command to get a certificate from the corresponding techlin-gcp-us.com site, which is part of the CollibraData Lineage infrastructure: <pre>openssl x509 -in <(openssl s_client - connect techlin-gcp-us.collibra.com:443 - prexit 2>/dev/null) -out techlin-gcp-us.crt</pre> Tip If you already have a correctly formatted certificate on the server, you can skip this step. 2. Add the certificate to the Java TrustStore: <pre>keytool -importcert -file techlin-gcp- us.crt -alias techlin-gcp-us -keystore <your keystore name> -storepass changeit</pre> 3. Run the lineage harvester and use the new TrustStore using the following parameter: <pre>-Djavax.net.ssl.trustStore=<your keystore name></pre> Example To synchronize your data sources again, run the following command: <pre>./bin/lineage-harvester full-sync - Djavax.net.ssl.trustStore=mykeystore</pre>

Problem	Solution
You get an external system ID mapping error.	The error message looks similar to the following: <pre data-bbox="619 427 1362 1061">PROCESSING ERROR: "syn- cer.domain.DgcSyncError: java.lang.Ex- ception: Unexpected DGC job status: ERROR Error message: { "type" : "MESSAGE", "message" : "A mapping for the external system id 'd0f3a21a2324fa117112409b- dea6ade7' and resource '59cf9293-fca1- 4f78-99ab-31c150a23626' already exists." } Caused by: java.lang.Exception: Unex- pected DGC job status: ERROR Error message: { "type" : "MESSAGE", "message" : "A mapping for the external system id 'd0f3a21a2324fa117112409b- dea6ade7' and resource '59cf9293-fca1- 4f78-99ab-31c150a23626' already exists." }"</pre> Please create a support ticket and provide your answers to the following two questions. Note Refer to the example error message above and replace the IDs of the external system and the mapped resource with those in the error message you received. - What is the asset type of the mapped asset? In this example, the asset with ID 59cf9293-fca1-4f78-99ab-31c150a23626? Tip To view the asset type, go to the following URL: <your-collibra-platform-url>/asset/59cf9293-fca1-4f78-99ab-31c150a23626 - What is the mapping definition?

Problem	Solution
	Tip To view the mapping definition, go to the following URL: <your-colibra-platform-url>/rest/2.0/mappings/externalSystem/d0f3a21a2324fa117112409bdea6ade7/mappedResource/59cf9293-fca1-4f78-99ab-31c150a23626

Problem	Solution						
If you are using Collibra Data Intelligence Cloud 2021.11 or older, you have to add all Tableau attributes in the operating model to a scope and create a scoped assignment before you ingest Tableau via the lineage harvester.	Show me how to add attributes to a scope and create a scoped assignment Prerequisites - You are using Collibra 2021.11 or older. - You have a global role that has the System administration global permission. Steps - On the main menu, click , and then click  Settings. - The Collibra settings page opens. - In the tab pane, click Scopes. - Above the table, to the right, click Add. - The Create Scope dialog box appears. - Enter the required information. <table border="1"> <thead> <tr> <th>Field</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Name</td><td>The name of the scope.</td></tr> <tr> <td>Description</td><td>The description of the scope, for example to add extra details.</td></tr> </tbody> </table> - Click Save. - Open a Tableau asset type: - On the main menu, click , and then click  Settings. - The Collibra settings page opens. - In the tab pane, click Asset Types. - The asset type table appears. - In the overview of asset types, click an asset type. - The Asset type editor opens. Important You need to do this for each of the following asset types: - Tableau Project - Tableau Site - Tableau Workbook	Field	Description	Name	The name of the scope.	Description	The description of the scope, for example to add extra details.
Field	Description						
Name	The name of the scope.						
Description	The description of the scope, for example to add extra details.						

Problem

Solution

◦ Tableau Data Attribute

◦ Tableau Server

There are other Tableau asset types, but they do not require the scoped assignment.

7. In the tab pane, click **Add assignment**.

» The **Select scope for this assignment** dialog box appears.

8. Select the custom scope that you have created for Tableau assets.

Note You can only add one scope at a time.

9. Click **Add assignment**.

» The settings of the global assignment are copied into the selected scope.

Warning After you've created the scoped assignment, do not change the assignment itself. The sole purpose of the scoped assignment is to ingest read-only attributes for which you normally need a system user.

Note If you ingest Tableau metadata in a Collibra version 2021.09 or older, you must also manually create two new relation types and add them to the [Tableau <source ID> configuration file](#).

In Collibra, the name of a Tableau Data Attribute name is an asterisk (*).

☐ Name	Status	Asset Type
☐ *	Candidate	Tableau Data Attribute
☐ *	Candidate	Tableau Data Attribute
☐ *	Candidate	Tableau Data Attribute

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

Problem	Solution
You get the following error message: Can't show all data because the timeout limit PT1M has been exceeded. Use pagination, additional filtering, or both in the query, and try again.	To resolve this error, try lowering the relevant paging option value (or values) for the failing query. Consider the following failure summary: <pre> harvester.AbstractHarvester - Failure summary: Source 'tabemeat' failed with exception: harvester.error.TableauHarvesterError: Failed to process Tableau source: Failed to execute metadata query. Errors: [{"mes- sage":"Can't show all data because the timeout limit PT1M has been exceeded. Use pagination, additional filtering, or both in the query, and try again."}]. Query: query worksheets(\$cursor: String, \$pageSize: Int, \$nestedPageSize: Int) { sheetsConnection(first: \$pageSize, after: \$cursor) { nodes { id name luid </pre> In this example, the query name is worksheets: <pre> Query: query worksheets(\$cursor: String, \$pageSize: Int, \$nestedPageSize: Int) </pre> Try lowering one or both of the paging option values for worksheets: • <code>worksheetsPageSize</code> (default value is 100) • <code>worksheetsFieldsPageSize</code> (default value is 100) 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, </pre>

Problem	Solution																
	<pre> "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> - dashboardsPageSize: The number of dashboards per page. </td></tr> <tr> <td>Worksheet</td><td> - worksheetsPageSize: The number of worksheets per page. - worksheetsFieldsPageSize: The number of worksheet fields per page. </td></tr> <tr> <td>Database</td><td> - databasesPageSize: The number of databases per page. </td></tr> <tr> <td>Table</td><td> - tablesPageSize: The number of tables per page. - tablesColumnsPageSize: The number of table columns per page. </td></tr> <tr> <td>Table columns</td><td> - tableColumnsPageSize: The number of table columns per page. </td></tr> <tr> <td>Parameter</td><td> - parametersPageSize: The number of parameters per page. </td></tr> <tr> <td>Users</td><td> - usersPageSize: The number of users per page. </td></tr> </table>	Metadata type	Setting and description	Dashboard	dashboardsPageSize: The number of dashboards per page.	Worksheet	- worksheetsPageSize: The number of worksheets per page. - worksheetsFieldsPageSize: The number of worksheet fields per page.	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.
Metadata type	Setting and description																
Dashboard	dashboardsPageSize: The number of dashboards per page.																
Worksheet	- worksheetsPageSize: The number of worksheets per page. - worksheetsFieldsPageSize: The number of worksheet fields per page.																
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.																

Problem	Solution	
	Metadata type	Setting and description
	Data source	• <code>datasourcesPageSize</code>: The number of data sources per page. • <code>datasourcesFieldsPageSize</code>: The number of data source fields per page. • <code>columnsLimit</code>: The number of data source field columns per page. • <code>fieldsLimit</code> : The number of referenced data source fields per page.
	Data source field	• <code>datasourceFieldsPageSize</code>: The number of data source fields per page. • <code>columnsLimit</code>: The number of data source field columns per page. • <code>fieldsLimit</code> : The number of referenced data source fields per page.

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.

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.

Power BI general troubleshooting

The following messages and issues can appear when you run the lineage harvester, view a technical lineage or upload the new relations to Data Catalog via Colibra Data Lineage.

Problem	Solution
A proxy is blocking the connection.	Get more information about the proxy by running the following command: <pre>bin/lineage-harvester.sh -Djavax.net.debug=all 2>&1 tee debug.log</pre> Then fix your proxy settings.

Problem	Solution
There is an issue with the token request.	Run the following command: <pre>curl -d 'client_id=<enter client id>&grant_type=client_credentials&client_secret=<client secret>' \ 2-X GET \ 3"https://login.partner.microsoftonline.cn/<tenant domain>/oauth2/v2.0/token"</pre> If the command runs successful, the token is not the problem. If the command does not run successfully, an error message provides more information about the issue.
Access to a Microsoft API is denied.	Something is wrong in the Power BI or Azure configuration. Run each of the APIs mentioned in the Metadata scanning - Power BI page of the Microsoft documentation. Check the limitations mentioned on the same page, to know if something is contributing to your situation.

Working with the lineage harvester

The lineage harvester is a connectivity tool that allows you to create a technical lineage.

You can create a technical lineage for different data sources by using the lineage harvester. The information in this section applies to all supported [the JDBC data sources and ETL tools](#). This information also shows how to delete the technical lineage for a data source. If an error occurs, you can use the troubleshooting section to fix the errors.

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.

Tip

For information about ingesting metadata from the following BI tools and creating a technical lineage via the lineage harvester, go to the dedicated sections:

- Tableau:
 - [Via the Data Catalog user interface](#).
 - [Via the lineage harvester](#).
- [Power BI \(deprecated\)](#)
- [Power BI](#)
- [Looker](#)
- [MicroStrategy](#)
- [MicroStrategy \(NEW\)\(Beta\)](#)
- [SQL Server Reporting Services and Power BI Report Server](#)

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

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

Creating a Technical lineage via the lineage harvester

The following table shows which steps you have to take to create a technical lineage for [the JDBC data sources and ETL tools](#) by using the lineage harvester.

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

Tip

For information about ingesting metadata from the following BI tools and creating a technical lineage via the lineage harvester, go to the dedicated sections:

- Tableau:
 - [Via the Data Catalog user interface.](#)
 - [Via the lineage harvester.](#)
- [Power BI \(deprecated\)](#)
- [Power BI](#)
- [Looker](#)
- [MicroStrategy](#)
- [MicroStrategy \(NEW\)\(Beta\)](#)
- [SQL Server Reporting Services and Power BI Report Server](#)

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

Step	What?	Description
1	Prepare Data Catalog physical data layer	Before you create a technical lineage, you prepare Data Catalog's physical data layer. This is necessary to automatically stitch assets in Data Catalog and the data elements in the data source for which you want to create a technical lineage. By preparing Data Catalog's physical data layer, you create assets of the following types: • System • Database • Schema • Table Note If you don't prepare the Data Catalog physical data layer, you can still create a technical lineage. However, stitching will not be performed.
2	Set up 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. You can download the lineage harvester from the Colibra Community Downloads page.
3	If you want to create technical lineage for Azure Data Factory, complete the tasks in Azure Data Factory prerequisites .	The lineage harvester uses Azure APIs to get the information necessary to build technical lineage from Azure Data Factory. Use the Azure Data Factory prerequisites topic to register Azure Data Factory in the Azure Portal and assign the necessary permissions and access.

Step	What?	Description
4	Prepare the configuration file	Prepare a configuration file to determine for which data sources you want to create a technical lineage. The lineage harvester uses the configuration file to extract information from data sources for which you want to create a technical lineage. Tip Use the configuration file generator to create an example configuration file with the properties of your choosing. You can copy this example to your configuration file and replace the values of the properties to match your data source information. When you have prepared the configuration file, you can use specific commands to perform different actions on the data sources that are defined in your configuration file. For example, you use the <code>full-sync</code> command to upload the source code from the data sources in the configuration file to the Collibra, where they are analyzed and processed and where the technical lineage is created. Tip • If you want to use SQL files from a previously loaded data source, you have to download the SQL files of a data source to the lineage harvester. • If you want to use a data source in an external directory, for example Informatica PowerCenter, SQL Server Integration Services or IBM InfoSphere DataStage, you have to prepare the external directory folder. • If you want to use a JSON file to create a custom technical lineage, you have to prepare the JSON file.
5	Run the lineage harvester.	After you prepared the lineage harvester configuration file, you can run the lineage harvester.
6	View the technical lineage.	After you created the technical lineage, you can go to a Power BI Column, Looker Look, Column or Table asset page and click the Technical lineage tab to view the technical lineage. You can use the Browse tab pane to search for different data objects and trace their dependencies or use the Settings tab pane to edit or export the technical lineage and see the logs created by the lineage harvester.

Prepare the Data Catalog physical data layer for technical lineage

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 via 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 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.

Note This topic 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.

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 [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 Jobserver 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.

- 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
- You have the permissions to retrieve the metadata of the following database components through the JDBC Driver Database Metadata methods:
 - Schemas
 - Tables
 - Columns

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.

- 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 **System**.
 - » The **Create Asset** dialog box appears.

- d. Enter the required information.

Field	Description
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.
Name	A name to identify 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.

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.

- 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.

d. Enter the required information.

Field	Description
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.
Name	A name to identify 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.

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.

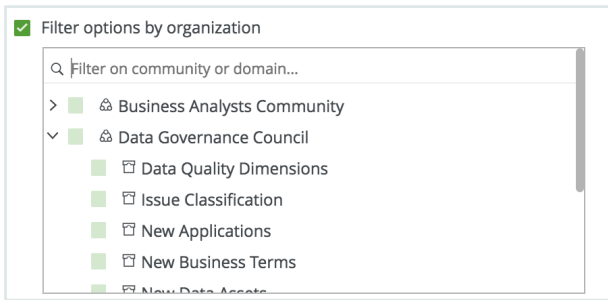
a. In the tab pane, click **Add Characteristic**.

» The **Add a characteristic** dialog box appears.

b. Click **Relations**.c. Search for and click **groups Technology asset**.

» The **Add groups Technology asset** dialog box appears.

d. Enter the required information.

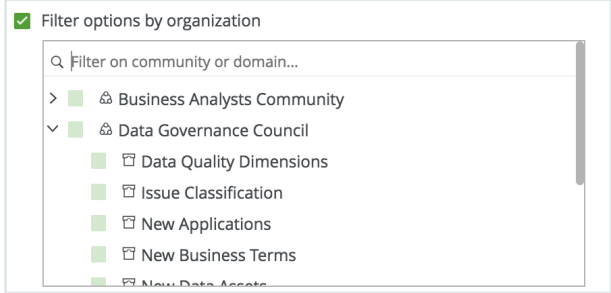
Option	Description
Assets	The name of the database.
Filter suggested assets by organization	Option to filter the suggestions based on selected communities and domains. If this option is selected, the organization tree appears. You can then filter and select domains and communities. 
Start date	Optionally enter the date on which the relation between the assets becomes applicable. Leave this field empty to create a permanent relation.
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**.

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

- 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.
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. Leave this field empty to create a permanent relation.
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.

The lineage harvester also uses [scheduled jobs](#) to automate the technical lineage process.

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.

Lineage harvester system requirements

You need to meet the system requirements to be able to [install](#) and run the [lineage harvester](#).

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 [Technical lineage general troubleshooting](#).

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

Prerequisites

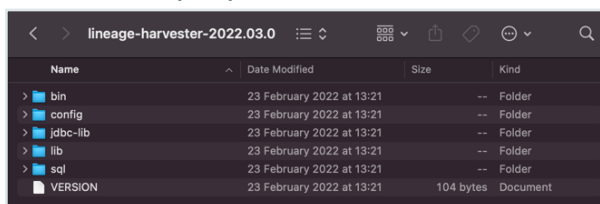
- You have purchased Collibra Data Lineage.
- You meet the [minimum system requirements](#).
- You have added Firewall rules so that the lineage harvester can connect to:
 - The host names of all databases in the lineage harvester configuration file.
 - All [Collibra Data Lineage service instances](#) 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)

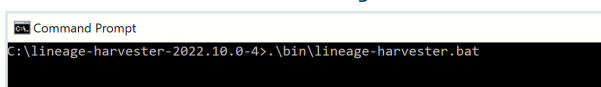
Note The lineage harvester connects to different instances based on your geographic location and cloud provider. If your location or cloud provider changes, the lineage harvester rescans all your data sources. You have to allow all Collibra Data Lineage service instances in your geographic location. 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

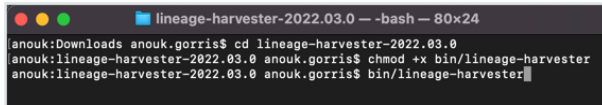
1. [Download](#) the newest lineage harvester.
2. Unzip the archive.
 - » You can now access the lineage harvester folder. The lineage harvester folder name is unique per version.



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`

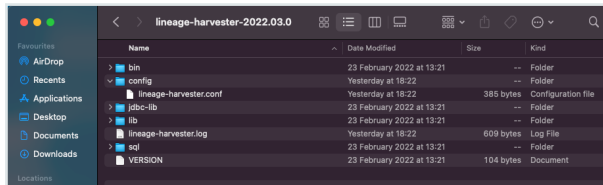


```

lineage-harvester-2022.03.0 --bash -- 80x24
anouk:Downloads anouk.gorris$ cd lineage-harvester-2022.03.0
anouk:lineage-harvester-2022.03.0 anouk.gorris$ chmod +x bin/lineage-harvester
anouk:lineage-harvester-2022.03.0 anouk.gorris$ 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.

Command	Description
<code>full-sync</code>	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. 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.
<pre>load-sources</pre>	Downloads all your data sources in a separate ZIP file, per data source, to the lineage harvester output folder.

Command	Description
<code>-s <ID of data source></code>	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: <pre>Source ID <ID of data source> (from edge: false true) (useSystemName: false true).</pre> Source ID <ID of data source> The source ID of your data source. <pre>from edge: false true</pre> 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. <pre>useSystemName: false true</pre> 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.

Azure Data Factory prerequisites

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](#)
- [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.

Properties	Description
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.
resourceGroupName	The name of a resource group with the Reader role for the Azure Data Factory instance.
subscriptionId	The subscription ID of the resource group.
password	The secret value for the application ID.

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.
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
- The Directory ID

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.

Tip These IDs are the values you will use for the `applicationId` and `tenantDomain` properties, respectively, in your Azure Data Factory configuration file.

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.

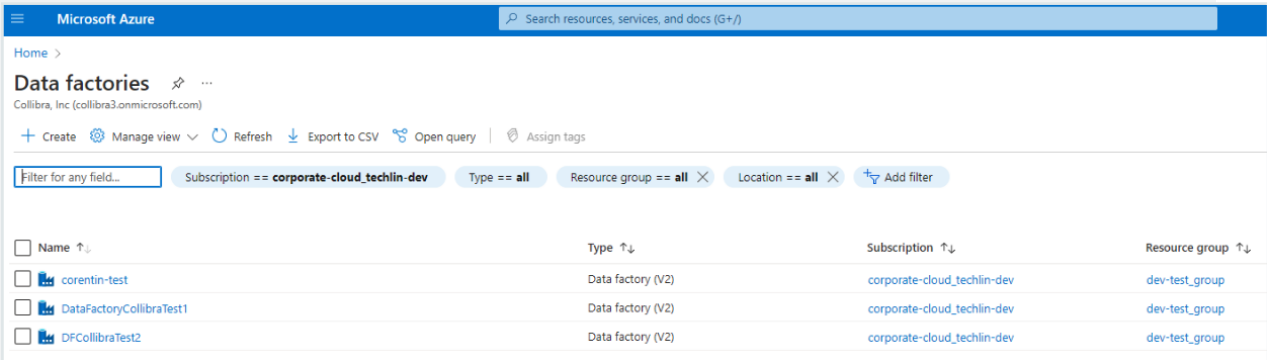
Important Make note of the authentication secret. For security purposes, It will not be available later. If you lose the authentication secret, you will need to create a new one.

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

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.

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.



Create a resource group and add your Azure Data Factory instance

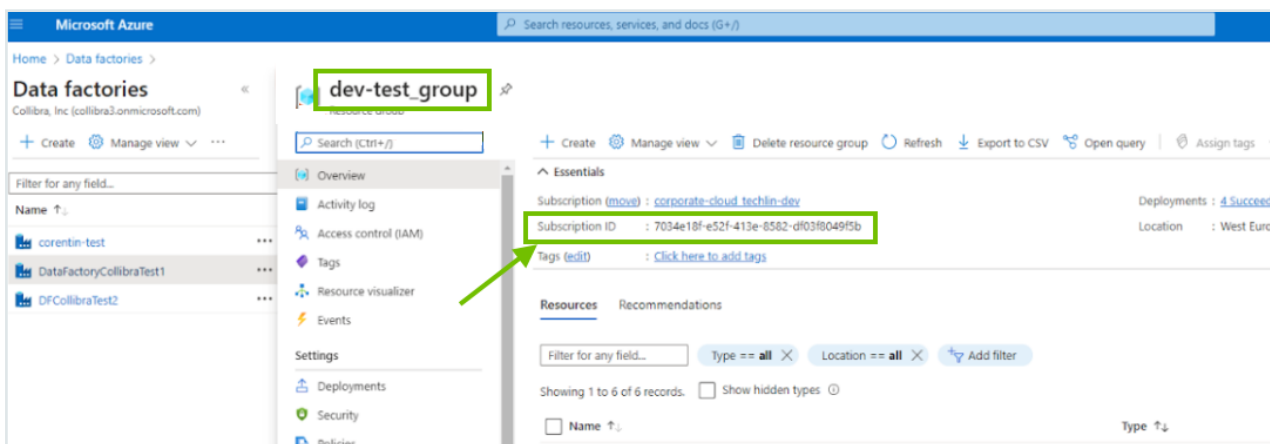
1. Go to the Group Management page for your Azure Data Factory instance.
2. Follow the Microsoft Azure instructions on how to [create a resource group](#), and refer to the following table for help with the various settings:

Setting	Description
Group Name	The name of the new resource group that you are creating. Make a note of this name. You will need it later. Tip The resource group name is the value you will use for the <code>resourceGroupName</code> property, in your Azure Data Factory configuration file.
Group Type	The type of resource group. Select Security.

Setting	Description
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.

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.

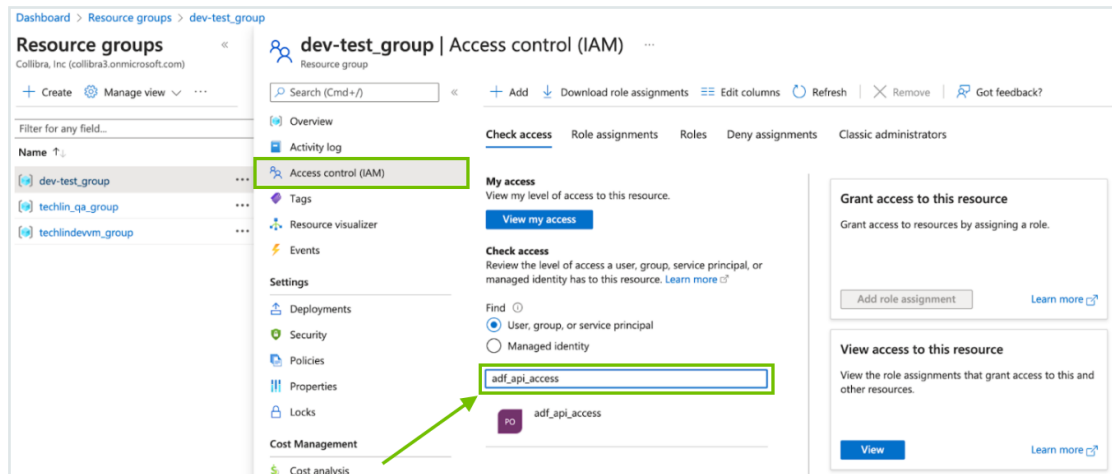


Tip The subscription ID is the value you will use for the `subscriptionId` property, in your Azure Data Factory configuration file.

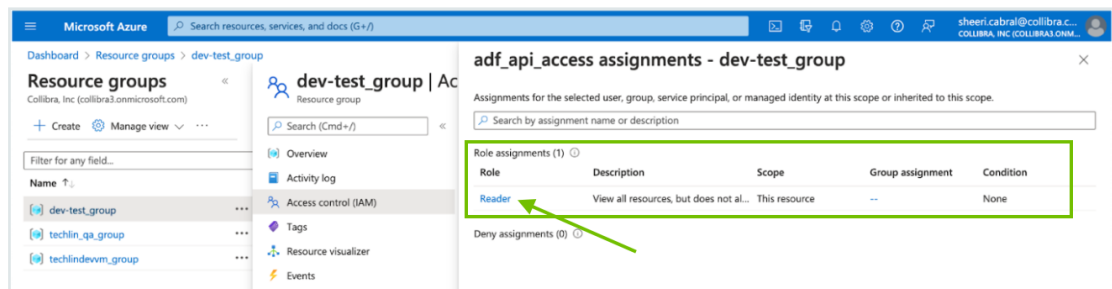
Assign read-only permissions to the resource group

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

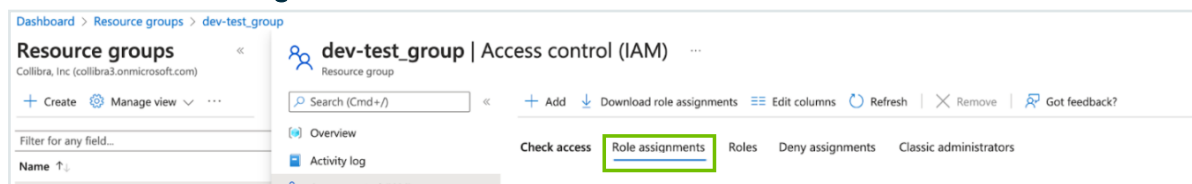
1. Check to see which permissions the resource group has.
 - a. On the Resource groups page, click **Access control (IAM)**.
 - b. In the **Check access** search box, type the name of the resource group.



- c. In the search results, click on the resource 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 resource 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.

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.
- Technical lineage supports the following means of authentication:
 - For all data sources, except for [external directories](#): username and password.
 - Google BigQuery data sources: username and password or a service account key file. For more information, see the [Google BigQuery documentation](#).
 - Snowflake: username and password or key pair authentication.
 - No other authentication methods are supported.
- Comments in the lineage harvester configuration file are not supported.

Tip

For information about ingesting metadata from the following BI tools and creating a technical lineage via the lineage harvester, go to the dedicated sections:

- Tableau:
 - [Via the Data Catalog user interface](#).
 - [Via the lineage harvester](#).
- Power BI (deprecated)
- Power BI
- Looker
- MicroStrategy
- MicroStrategy (NEW)(Beta)
- SQL Server Reporting Services and Power BI Report Server

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

Before you begin

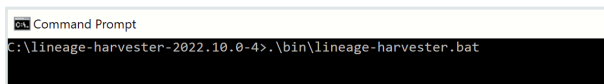
- [Set up the latest lineage harvester](#).
- [Prepare the Data Catalog physical data layer for technical lineage](#).
- If you want to use a previously loaded data source, [download](#) the SQL files of the data source to the lineage harvester.
- If you want to use an external directory, [prepare](#) a folder with data objects from the external directory.

Requirements and permissions

- 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 BI Data Catalog 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 to create an empty lineage harvester configuration file by entering the following command:
 - **Windows:** `.\bin\lineage-harvester.bat`



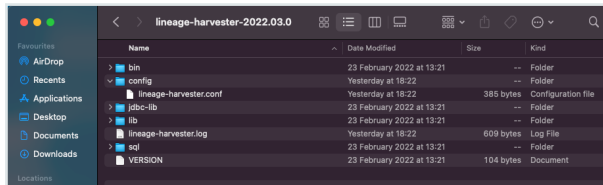
- For other operating systems: `chmod +x bin/lineage-harvester` and then `bin/lineage-harvester`

```

lineage-harvester-2022.03.0 -- -bash -- 80x24
anouk:Downloads anouk.gorris$ cd lineage-harvester-2022.03.0
anouk:lineage-harvester-2022.03.0 anouk.gorris$ chmod +x bin/lineage-harvester
anouk:lineage-harvester-2022.03.0 anouk.gorris$ bin/lineage-harvester

```

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



- Optional: If you use more than one lineage harvesters on different servers, repeat Step 1 to create an empty lineage harvester configuration file for each lineage harvester on each server.
- Open the configuration file and enter the values for each property. If you create multiple configuration files, ensure that each configuration file contains configurations of unique data sources. If the configuration of a data source duplicates in different configuration files, only one configuration takes effect.

Tip Use the [configuration file generator](#) to create an example configuration file with the properties of your choosing. You can copy this example to your configuration file and replace the values of the properties to match your data source information.

Properties	Description
general	This section describes the connection between Collibra 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 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. By default, the <code>useCollibraSystemName</code> property is set to <code>false</code>. ◦ If you keep the <code>useCollibraSystemName</code> property set to <code>false</code>, the lineage harvester ignores the <code>collibraSystemName</code> property in the rest of the configuration file. ◦ If you set the <code>useCollibraSystemName</code> property to <code>true</code>, the lineage harvester reads the value in the <code>collibraSystemName</code> property in all sections of the configuration file. It also reads the <code>collibraSystemName</code> property in the following files: ■ The Informatica <source ID> configuration file Important You must prepare a <source ID> configuration file regardless of whether the <code>useCollibraSystemName</code> property in your lineage harvester configuration files is set to <code>true</code> or <code>false</code>. ■ The IBM DataStage or SQL Server Integration Services connection definition configuration files. ■ The Informatica Intelligent Cloud Services <source ID> configuration file. Important You must prepare a <source ID> configuration file regardless of whether the <code>useCollibraSystemName</code> property

Properties	Description
	in your lineage harvester configuration files is set to <code>true</code> or <code>false</code>. 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. Indicates whether or not you intend to use a Matillion <source ID> configuration file to specify the system name of a data source. This is useful if you have multiple databases with the same name, or if you want to group a number of databases under one system. By default, this property is set to <code>false</code>. If you set this property to <code>true</code>, you must prepare a Matillion <source ID> configuration file. 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>col-</code>

Properties	Description
	<code>libraSystemName</code> properties in the configuration file. This is the default value. <code>true</code> The lineage harvester reads the system and server names that you specify on the <code>col-libraSystemName</code> properties in all sections of the configuration file. Only specify this value when you have multiple databases with the same name. Note Specify this property with the value of <code>true</code> only when you have multiple databases with the same name.
<code>sources</code>	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.
<SQL directory properties>	This configuration section contains the required information of one individual SQL directory with <code>connection type</code> "Folder".
<code>id</code>	The unique ID of the data source. For example, <code>my_first_data_source</code> .
<code>type</code>	The kind of data source. In this case, the value has to be <code>SqlDirectory</code> .
<code>path</code>	The full path to the folder where you added SQL files , for example, <code>C:\path\to\config\dir</code> .
<code>mask</code>	The pattern of the file names in the directory. By default, this is <code>*</code> .

Properties	Description
<code>recursive</code>	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.

Properties	Description
	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 an SAP HANA data source. ◦ <i>hana-cviews</i>, for getting lineage from calculated views in an SAP HANA data source. Important ■ The <i>hana-cviews</i> dialect is supported for SAP HANA (on-premises). It is not supported for SAP HANA Cloud. ■ 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 <i>hana</i> dialect, and in the other, specify the <i>hana-cviews</i> dialect. ◦ <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. If you want to use a Spark SQL data source, make sure that you have an AWS host.

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 the SQL directory and Automatic stitching for technical lineage. Important HiveQL, MySQL and Teradata data sources don't have schemas. Therefore, HiveQL, MySQL and Teradata databases are stored in Data Catalog and technical lineage as Schema assets. The technical lineage Browse tab pane shows the following names: ◦ For HiveQL and Teradata: ■ 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. ◦ For MySQL:

Properties	Description
	■ The database name is the name that you enter for the <code>database</code> property.
<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.
<code>schema</code>	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.
<code>verbose</code>	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>.

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.
<External directories>	This configuration section contains the required information to connect to the following data sources: ◦ Informatica PowerCenter ◦ SQL Server Integration Services (SSIS). ◦ IBM InfoSphere DataStage Note Make sure that you have prepared a local folder with the Informatica objects, SSIS files or DataStage files for which you want to create a technical lineage.
<code>collibraSystemName</code>	The name of the data source's system or server. If the <code>useCollibraSystemName</code> property is set to <code>true</code>, you must prepare a configuration file to provide the system information.
<code>id</code>	The unique ID of your data source. For example, <i>my_informatica</i>.
<code>type</code>	The kind of data source. In this case, the value has to be <i>ExternalDirectory</i>.

Properties	Description
dirType	The type of external directory. The value has to be one of the following: ◦ <code>infa</code>, for an Informatica PowerCenter data source. ◦ <code>ssis</code>, for a SQL Server Integration Service data source. ◦ <code>datastage</code>, for a IBM InfoSphere DataStage source.
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>.
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
<Azure Data Factory>	This configuration section contains the required information to enable the lineage harvester to collect and process Azure Data Factory objects.
id	The unique ID that identifies the data source on a Collibra Data Lineage service instance, for example, <i>my_adf</i> . If you want to create an Azure Data Factory <source ID> file , the <source ID> file name must be the same as the source ID.
type	The type of data source. The value must be <i>AzureDataFactory</i> .
collibraSystemName	The system or server name of the data source. This property is optional. Use this property with the <code>useCollibraSystemName</code> property 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.
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.

Properties	Description
subscriptionId	The subscription ID of the resource group.
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.

Properties	Description
<Informatica Intelligent Cloud Services Data Integration>	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", "collibraSystemName" : "iics-devel- opment", "loginUrl" : "https://dm- us.in- formaticaintelligentcloud.com", "username" : "login-iics" "objects" : [{ "path" : "Default/Sales", "type" : "Project" }, { "path" : "My Pro- ject/Statistics", "type" : "Project" }] } { "type" : "IICS", "id" : "iics_source-2", "collibraSystemName" : "iics-devel- opment", "loginUrl" : "https://dm- us.in- formaticaintelligentcloud.com", "username" : "login-iics" "objects" : [{ </pre>

Properties	Description
	<pre> "path" : "Finance/Task_Flows", "type" : "Folder" }, { "path" : "Common/Task_Flows/tf_ CalendarDimension", "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	The unique ID that is used to identify the data source on the Collibra Data Lineage service. For example, my_data_integration.
collibraSystemName	The name of the Informatica server or system. Important You must prepare a <source ID> configuration file to provide this system information. This is true regardless of whether the useCollibraSystemName property is set to <i>true</i> or <i>false</i>.
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 export. Each object requires a path and a type, for example: <pre data-bbox="746 423 1428 909">"objects": [{ "path" : "Sales", "type" : "Project" }, { "path" : "Finance/Task_Flows", "type" : "Folder" }, { "path" : "Common/Task_Flows/tf_ CalendarDimension", "type" : "Taskflow" }]</pre> The following section provides information to identify and access Data Integration objects. Tip For more information about the objects that you can export and the required information, see the Informatica documentation.
path	The path to the object, which is relative to the Explore directory in IICS, for example, <i>Sales</i>.
type	The type of the object. For example, Taskflow. IICS scanner's starting point is a Taskflow. Therefore the only meaningful types to export are: Taskflow, Project and Folder. Note 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 <code>/root/child/child2/paramfile.txt</code>, 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
<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.
<Matillion>	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.
<code>id</code>	The unique ID that is used to identify the data source on the Collibra Data Lineage service instance. For example, <code>my_matillion_data_integration</code> .
<code>type</code>	The kind of data source. In this case, the value has to be <code>Matillion</code> .

Properties	Description
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.
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.
dialect	The dialect of the database. You can enter one of the following values: ◦ <code>redshift</code>, for an Amazon Redshift data source. ◦ <code>snowflake</code>, for a Snowflake data source.
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.

Properties	Description
<code>collibraSystemName</code>	Regardless of the value set for the <code>useCollibraSystemName</code> property, the following is true: ◦ You must include this property in your configuration file. ◦ You can leave this property empty. ◦ Any value that you give is ignored. If the <code>useCollibraSystemName</code> property is set to <code>true</code>, you must prepare a Matillion <source-ID> configuration file. In that case, the <code>CollibraSystemName</code> property in the <code><source ID></code> configuration file is taken into account. Note This is a legacy property that will be deprecated in a future release.
<code>auth</code>	The section contains the authentication details for signing in to Matillion.
<code>type</code>	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.
<code>username</code>	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.

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.
<database properties>	This configuration section contains the required information of one individual data source with <code>connection type</code> "JDBC".
<code>id</code>	The unique ID of your data source. For example, <code>my_second_data_source</code> .
<code>type</code>	The kind of data source. In this case, the value has to be <code>Database</code> .
<code>username</code>	The username that you use to sign in to your data source.

Properties	Description
dialect	The dialect of the database. Tip You can enter one of the following values: ◦ <i>azure</i>, for an Azure SQL Server data source. ◦ <i>db2</i>, for an IBM DB2 data source. ◦ <i>hana</i>, for an SAP HANA data source. ◦ <i>hana-cviews</i>, for getting lineage from calculated views in an SAP HANA data source. Important ■ The <i>hana-cviews</i> dialect is supported for SAP HANA (on-premises). It is not supported for SAP HANA Cloud. ■ 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 <i>hana</i> dialect, and in the other, specify the <i>hana-cviews</i> dialect. ◦ <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>spark</i>, for a Spark SQL data source. ◦ <i>sybase</i>, for a Sybase data source. ◦ <i>teradata</i>, for a Teradata data source. If you want to use a Spark SQL data source, make sure that you have an AWS host.

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, MySQL and Teradata data sources don't have schemas. Therefore, HiveQL, MySQL and Teradata databases are stored in Data Catalog and technical lineage as Schema assets. The technical lineage Browse tab pane shows the following names: - For HiveQL and Teradata: - 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. - For MySQL: - The database name is the name that you enter for the <code>database</code> property. Workaround for Oracle When ingesting an Oracle data source, the value of the <code>databaseNames</code> property in your configuration file must be either the Oracle SID or service name, depending on whether you set the <code>connectAsServiceName</code> property to <code>true</code> or <code>false</code>. This means that the database in the technical lineage will have the name of the Oracle SID or service name. However, if the database asset in Data Catalog reflects the true name of the

Properties	Description
	database, stitching will break. To resolve this issue and preserve stitching, you need to rename the database asset in Data Catalog to match the value you put for the <code>databaseNames</code> property. This is a known issue that we will fix in a future version of Collibra. Tip To avoid this workaround, you can use the <code>"type": "DatabaseOracle"</code> and related properties in your configuration file. That 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.

Properties	Description
externalDbName	This property can be considered a means of database mapping, to help preserve stitching. It is relevant only for HiveQL, MySQL and Teradata data sources, specifically because they are schema-less data sources. You can add the key/value pair to the configuration file, as follows: "externalDbName": "CDATA" 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> Now, because HiveQL is schema-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.
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.
<code><Oracle></code>	This configuration section contains the required information for an Oracle database. Tip We recommend the <code>"type"</code>: <code>"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"</code>: <code>"Database"</code> configuration to ingest Oracle databases.
<code>id</code>	The unique ID of your Oracle database. For example, <code>my_oracle_db</code> .
<code>type</code>	The kind of data source. In this case, the value has to be <code>DatabaseOracle</code> .
<code>hostname</code>	The name of your database host.

Properties	Description
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": "auto- mation_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: <pre> { "id": "oracle2",</pre>

Properties	Description
	<pre>"type": "DatabaseOracle", "hostname": "host_url", "username": "user1", "collibraSystemName": "auto- mation_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": "auto- mation_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: <pre> { "id": "oracle4",</pre>

Properties	Description
	<pre>"type": "DatabaseOracle", "hostname": "host_url", "username": "user1", "collibraSystemName": "auto- mation_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
<code>databaseNames</code>	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
<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.

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.
<code><Google BigQuery database></code>	This configuration section contains the required information for a Google BigQuery database.
<code>id</code>	The unique ID of your data source. For example, <code>my_third_data_source</code> .
<code>type</code>	The kind of data source. In this case, the value has to be <code>DatabaseBigQuery</code> .
<code>projectIDs</code>	The IDs of your Google BigQuery project. You can add multiple projects. For example, <code>["first-project", "second-project", "third-project"]</code>. 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.

Properties	Description
region	The location of your BigQuery data. This is the region that you specified when you create a data set. 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.
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
<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.
<Snowflake>	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.
<code>id</code>	The unique ID that identifies the data source on a Collibra Data Lineage service instance, for example, <code>my_snowflake_2</code>. If you use the SQL-API ingestion method and want to create a Snowflake <source ID> file, <source ID> file name must be the same as the source ID.
<code>type</code>	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	The system or server name of the data source. This property is optional. Use this property with the <code>useCollibraSystemName</code> property 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.
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.

Properties	Description
type	The authentication method. Specify one of the following values. The values are case-sensitive. <code>Basic</code> The username and password authentication method. Specify the <code>auth.username</code> property if you use this authentication method. <code>KeyPair</code> 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.
usePassword	The private key file password. This property is required if you use the key pair authentication method. Specify one of the following values: <code>true</code> The password is required. <code>false</code> The password is not required. This is the default value.

Properties	Description
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, <pre>["MyFirstSnowflakeExternalDatabase", "MySecondSnowflakeExternalDatabase"].</pre>
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, <pre>["MyFirstSnowflakeSchema", "MySecondSnowflakeSchema"].</pre>
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 <code>ACCESS_HISTORY</code> 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>.
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
<SQL files in the lineage harvester output folder>	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.
type	The kind of data source. In this case, the value has to be <code>LoadedSource</code> .
id	The unique ID of the data source that you uploaded to the lineage harvester folder. 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.

4. Save the configuration file.

Example

```
{
  "general" : {
    "catalog" : {
      "url" : "https://<organization>.collibra.com",
      "username" : "<your-collibra-username>"
    },
    "useCollibraSystemName" : false
  },
  "sources" : [
    {
      "type" : "AzureDataFactory",
      "id": "adf_source",
      "collibraSystemName": "this_is_ignored",
      "tenantDomain": "my_server.onmicrosoft.com",
      "loginFlow": {
        "type": "ServicePrincipal",
        "applicationId": "00000000-1111-2222-3333-444444444444"
      },
      "subscriptionId" : "99999999-8888-7777-6666-555555555555",
      "resourceGroupName" : "adf_read_only_resource_group"
    }
  ]
}
```

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 more easily 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" : [
    {
      "collibraSystemName" : "adf-system-name",
      "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
    }
    {
        "collibraSystemName" : "datastage-system-name",
        "id" : "datastage_source",
        "type" : "ExternalDirectory",
        "dirType" : "DATASTAGE",
        "path" : "/path/to/the/datastage/folder/",
        "mask" : "*",
        "recursive" : false,
        "deleteRawMetadataAfterProcessing": false
    }
    {
        "collibraSystemName" : "infa-system-name",
        "id" : "informatica_source",
        "type" : "ExternalDirectory",
        "dirType" : "INFA",
        "path" : "/path/to/the/informatica/folder/",
        "mask" : "*",
        "recursive" : false,
        "deleteRawMetadataAfterProcessing": false
    }
    {
        "collibraSystemName" : "ssis-system-name",
        "id" : "datastage_source",
        "type" : "ExternalDirectory",
        "dirType" : "SSIS",
        "path" : "/path/to/the/ssis/folder/",
        "mask" : "*",
        "recursive" : false,
        "deleteRawMetadataAfterProcessing": false
    }
    {
        "type" : "IICS",
        "id" : "iics_source",
        "collibraSystemName" : "iics-development",
        "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",
        "dialect" : "redshift",
        "startTimestamp" : 1594080796911,
        "collibraSystemName": "Matillion-system",
        "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    }
    {
        "type" : "ExistingLineage",
        "id" : "MyPowerBISourceID",
        "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",
    "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"
    }
    {
        "type": "Microstrategy",
        "id": "microstrategy-batch",
        "domainId": "<domain-resource-id>",
        "username": "mstr",
        "hostname": "remote.postgres.com",
        "port": 5432,
        "databaseName": "poc_metadata",
        "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",
        "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
  } ]
}

```

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 <source ID> 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 <source ID> configuration file](#).

```

{
  "collibraSystemNames": {
    "databases": [
      {
        "hostName": "database-hostname",
        "collibraSystemName": "public"
      }
    ],
    "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"
      }
    ]
  }
},

```

```

"databaseMapping": {
  "<hostname:port>": "<actual database name>"
},
"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](#).

```

{
  "general": {
    "catalog": {
      "url": "https://<organization>.collibra.com",
      "userName": "<your-collibra-username>"
    },
    "useCollibraSystemName": false
  },
  "sources": [{
    "id": "looker-id",
    "type": "Looker",
    "lookerUrl": "https://<instance-name>.api.looker.com",
    "clientId": "looker-api-user-name",
    "clientSecret": "looker-api-userkey",
    "domainId": "domain-resource-id",
    "deleteRawMetadataAfterProcessing": true
  }]
}

```

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 source 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": {
    "dialect": "mssql",
    "collibraSystemName": "mssql-system-name"
  },
  "found_dbname=testsid;found_hostname=*": {
    "dbname": "oracle-database-name",
    "schema": "oracle-schema-name",
    "dialect": "oracle",
    "collibraSystemName": "oracle-system-name"
  }
}

```

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 the 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](#).

Prepare an external directory folder for the lineage harvester

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

If the external directory files do not have the necessary information, for example a database and a schema, to stitch the data sources, you have to provide the connection definitions manually via a JSON configuration file. This is required at each connection, regardless of whether the `useCollibraSystemName` property in the [lineage harvester configuration file](#) is set to `true` or `false`.

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 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.

If your folder contains previous versions of the parameter files, objects might be duplicated across different file versions. Collibra Data Lineage ignores any duplicated objects and issues an error message. 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 duplicated objects, export all objects and parameter files at the same time.

Note

- 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](#).
- 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)

3. In the local folder, create a folder named *techlin-param* and put the parameter files in the *techlin-param* folder.
4. Optionally, create a source ID configuration file with connection definitions and system names:

Tip If you previously created a technical lineage for Informatica PowerCenter with connection definitions, the `connection_definitions.conf` file will still be taken into account.

- a. Create a new JSON file in the lineage harvester **config** folder.
- b. 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 `informatica-source-1`. As a result, the name of your JSON file should be *informatica-source-1.conf*.

- c. For each data source, add the following content to the JSON file:

Property	Description
<code>connectionDefinitions</code>	This section contains the connection properties to a source in Informatica PowerCenter.
<code><connectionName></code>	The type of your source or target data source. This section contains the connection properties to a source or target in Informatica PowerCenter.
<code>dbname</code>	The name of your source or target database.
<code>schema</code>	The name of your source or target schema.

Property	Description
dialect	The dialect of the referenced database.

Property	Description
	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 an SAP HANA data source. ■ <i>hana-cviews</i>, for getting lineage from calculated views in an SAP HANA data source. Important ■ The <i>hana-cviews</i> dialect is supported for SAP HANA (on-premises). It is not supported for SAP HANA Cloud. ■ 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 <i>hana</i> dialect, and in the other, specify the <i>hana-cviews</i> dialect. ■ <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.

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 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 definitions JSON file. For example, if the parameter file contains `$DBCConnection_dwh=DWH_EXPORT` then you add the following connection definitions to the JSON file:

```
{
  "DWH_EXPORT":
    { "dbname": "DWH", "schema": "DBO" }
}
```

5. Add a new section for Informatica PowerCenter to the lineage harvester [configuration file](#).

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.

4. Optionally, configure the connection definitions:

Tip If the `useCollibraSystemName` in the [lineage harvester configuration file](#) is set to `true`, you must provide the `connection_definitions.conf` file.

- a. Create a new JSON file in the local folder.
- b. Name the JSON file *connection_definitions.conf*.
- c. For each supported data source, specify the relevant translations.

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.

Property	Description
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: ■ <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 an SAP HANA data source. ■ <i>hana-cviews</i>, for getting lineage from calculated views in an SAP HANA data source. Important ■ The <i>hana-cviews</i> dialect is supported for SAP HANA (on-premises). It is not supported for SAP HANA Cloud. ■ 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 <i>hana</i> dialect, and in the other, specify the <i>hana-cviews</i> dialect. ■ <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.

Property	Description
collibraSystemName	The name of the referenced data source's system or server. This property is only required when you set the useCollibraSystemName property in the lineage harvester configuration file to <code>true</code>. If this property is set to <code>false</code>, you can remove the collibraSystemName property or enter an empty string. Note 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 the "useCollibraSystemName" property is: ■ <code>false</code>, system or server names in table references in analyzed SQL code are now 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 full 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 "collibraSystemName" field is used as the default system or server name.

5. Add a section for SQL Server Integration Services to the lineage harvester [configuration file](#).

Example of the connection_definitions.conf 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 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.

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.

7. Optionally, configure the connection definitions:

- Create a new JSON file in the local folder.
- Name the JSON file *connection_definitions.conf*.
- For each data source, specify the relevant translations:

Property	Description
OdbcDataSources	Open Database Connectivity data sources in IBM InfoSphere DataStage for which you want to create a technical lineage.
<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.

Property	Description
	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 an SAP HANA data source. ■ <i>hana-cviews</i>, for getting lineage from calculated views in an SAP HANA data source. Important ■ The <i>hana-cviews</i> dialect is supported for SAP HANA (on-premises). It is not supported for SAP HANA Cloud. ■ 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 <i>hana</i> dialect, and in the other, specify the <i>hana-cviews</i> dialect. ■ <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.

Property	Description
collibraSystemName	The name of the data source's system or server. This property is only required when you set the useCollibraSystemName property in the lineage harvester configuration file to <code>true</code>. If this property is set to <code>false</code>, you can remove the collibraSystemName property or enter an empty string. Note Specify this property with the same name as the full 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.
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 <code>admin@database-name</code>. 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.
dialect	The dialect of the referenced database.

Property	Description
collibraSystemName	The name of the data source's system or server. This property is only required when you set the useCollibraSystemName property in the lineage harvester configuration file to <code>true</code>. If this property is set to <code>false</code>, you can remove the collibraSystemName property or enter an empty string. Note Specify this property with the same name as the full 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.
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.

Property	Description
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.

Property

Description

```

{
  "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"
    }
  ]
}

```

Property	Description

Tip Click to copy the example to your clipboard.Example of the connection_ definitions.conf file

8. Add a section for IBM InfoSphere DataStage to the lineage harvester [configuration file](#).

Example of the connection_definitions.conf 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"
  }
]
}

```

What's next

You can now [prepare](#) the rest lineage harvester configuration file and run it to create a technical lineage for Informatica PowerCenter, SQL Server Integration Services, IBM InfoSphere DataStage and, optionally, other data sources.

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

Note For more information about the scope, see the overview of [supported data sources](#).

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.

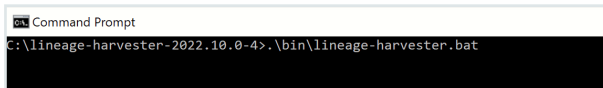
- You have the necessary permissions to all **database objects** that the lineage harvester accesses.

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.

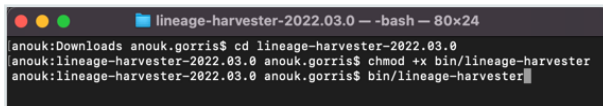
Steps

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

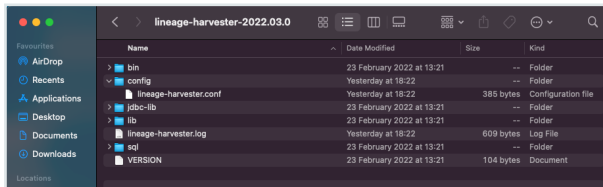
- **Windows:** `.\bin\lineage-harvester.bat`



- **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 Azure Data Factory <source ID> configuration file

You can create the Azure Data Factory <source ID> configuration file to define connections and mappings.

Example of the <source ID> configuration file

```
{
  "found_dbname=databasename1;found_hostname=server-name.on-microsoft.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 the lineage harvester. 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 lineage harvester 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 a database. If you don't specify a value for this property, "DEFAULT" is shown in the technical lineage harvester. 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- e=databasename1;found_host- name=*;found_ schema=schema1": { "dbname": "Customers", "schema": "mssql-schema- name", "dialect": "mssql", "collibraSystemName": "Cus- tomers-Europe" }, "found_dbname- e=databasename2;found_host- name=server- name.onmicrosoft.com;found_ schema=schema2": { "dbname": "Customers", "schema": "oracle-schema- name", "dialect": "oracle", "collibraSystemName": "Cus- tomers-USA" }, </pre> Warning The value of this property must exactly match (including for case-sensitivity) the name of your System asset in Collibra.	Yes

Property	Description	Mandatory?
	Important If you are using a <source ID> configuration file for the purpose of providing the true system name of an ODBC database in Azure Data Factory, 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, you must specify a Collibra system name in the <source ID> configuration file.	

4. Save the <source ID> configuration file.

Prepare Informatica Intelligent Cloud Services <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

- Your user role must have privileges to export assets.
- You must have read permission on all assets that you want to export.

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.	Yes
<code>collibraSystemName</code>	The system or server name of the connection.	Yes

Property	Description	Required?
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.	Yes
schemaName	The name of your schema.	Yes
dialect	The dialect of the connection. Specify this property to properly extract and parse queries that are related to this connection. You can enter one of the following values: ◦ <code>bigquery</code> ◦ <code>db2</code> ◦ <code>hana</code> ◦ <code>hive</code> ◦ <code>greenplum</code> ◦ <code>mssql</code> ◦ <code>mysql</code> ◦ <code>netezza</code> ◦ <code>oracle</code> ◦ <code>postgres</code> ◦ <code>redshift</code> ◦ <code>snowflake</code> ◦ <code>spark</code> ◦ <code>teradata</code>	No

4. Save the configuration file.

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"
    }
  ]
}
```

Prepare Matillion <source ID> configuration file

You use the [lineage harvester configuration file](#) to access Matillion data objects. The lineage harvester processes the data objects to create a technical lineage. However, if the `useCollibraSystemName` property in the lineage harvester configuration file is set to `true`, you also have to provide a <source ID> configuration file to define the system name for all sources and targets in the Matillion integration.

This is useful if you have multiple databases with the same name and want to distinguish between them in the technical lineage harvester by specifying the system or server specific to each.

Note To preserve stitching, you need a System asset in Data Catalog of the same name of each system or server you specify in your <source ID> configuration file.

Prerequisites

- The `useCollibraSystemName` in the lineage harvester configuration file is set to `true`.
- You have Admin permission on all objects that you want to harvest.

Steps

1. Create a new JSON configuration 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 `matillion-source-1`, then the name of your JSON file should be *matillion-source-1.conf*.

Important Your JSON file must have the file extension *.conf*.

3. For each Matillion connection, you can add the following 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. 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".											
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										

Property	Description	Mandatory?
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
dialect	Select one of the following dialects for your data source 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 a database. Specify this property to ensure mapping between the Matillion source names and the System assets in your Collibra 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", "dialect": "mssql", "collibraSystemName": "Customers-Europe" }, "found_dbname=testsid;found_hostname=*": { "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.	Yes

Property	Description	Mandatory?
	Important If you are using a <source ID> configuration file for the purpose of providing the true system name of an ODBC database in Matillion, 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, you must specify a Collibra system name in the <source ID> configuration file.	

4. Save the configuration file.

Example of the <source-ID>.conf file

```
{
  "found_dbname=dbtest;found_hostname=test": {
    "dialect": "mssql",
    "collibraSystemName": "mssql-system-name"
  },
  "found_dbname=testsid;found_hostname=*": {
    "dbname": "oracle-database-name",
    "schema": "oracle-schema-name",
    "dialect": "oracle",
    "collibraSystemName": "oracle-system-name"
  }
}
```

Prepare Snowflake SQL-API <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
}
```

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-snowflake`, the name of your JSON file must be **`my-snowflake.conf`**.

3. For each database in Snowflake, add the following content to the JSON file:

Property	Description	Mandatory?
<code>displaySampleQueries</code>	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

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 manage which data objects, for example columns and tables, are ingested in the technical lineage. By doing this, you can decide:

- 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.

You can manage technical lineage ingestion by creating a customized SQL file. In the SQL file, you can exclude data objects or change queries that are used to extract data from the database.

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 specify 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](#).

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"
```

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
```

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

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.

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 troubleshooting guide](#) or [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 assets from [registered data sources](#).
- 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 troubleshooting guide](#) or [Collibra Support Portal](#) to fix the error.

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 troubleshooting guide](#) or [Collibra Support Portal](#) to fix the error.

Working with technical lineage via Edge

You can create a technical lineage for different data sources by using technical lineage via Edge. Use this information to create a technical lineage for [the JDBC data sources and ETL tools](#) or a custom technical lineage. This information also shows how to delete the technical lineage for a data source. If an error occurs, you can use the troubleshooting section to fix the errors.

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

For information about ingesting metadata from the following BI tools and creating a technical lineage via Edge, go to the dedicated sections:

- [Tableau](#)
- [Power BI](#)

Create technical lineage via Edge

This topic provides an overview of the necessary steps to create a technical lineage for the JDBC data sources and ETL tools or a custom technical lineage on 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](#).

Before you begin

- Use Collibra Data Intelligence Cloud 2023.03 or later.
- [Create an Edge site](#) in Collibra Data Intelligence Cloud.
- [Install an Edge site](#).
- [Create a JDBC connection](#).
- [Register the data source via Edge](#). Before you register the data source, ensure that you [add the Catalog JDBC ingestion capability](#), so that Collibra Data Lineage can stitch the data objects in your technical lineage to the assets in Data Catalog.

Requirements and permissions

- 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

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

- Permissions required for the data source that you selected:

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.jobs.create
- bigquery.readsessions.create
- bigquery.tables.getData
- 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 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.

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

Note The following permissions are the same, 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. The username of the user must be specified in [the JDBC connection](#) that you use to access Snowflake.

Tip 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 added the Matillion certificate to a Java truststore.
- You have at least a Matillion Enterprise license.

Steps

What's next?

[View the technical lineage.](#)

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 menu, 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 troubleshooting guide](#) or [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` 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 menu, 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 [Creating a Technical lineage via the lineage harvester](#) or [Create 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 troubleshooting guide](#) or [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 troubleshooting guide](#) or [Collibra Support Portal](#) to fix the error.

Troubleshooting for technical lineage via Edge

In this topic

- [Retrieve your Edge Site Id and Job Id](#)
- [Message "Source 'source_name' was never processed with the current useSystemName flag"](#)
- [Message "Failed to load artifacts message"](#)
- [Message "A UNIQUE constraint failed"](#)
- [Message "Failed to fetch lineage API key because of a client error"](#)
- [Message "MountVolume.NewMounter initialization failed" does not exist"](#)

Retrieve your Edge Site Id and Job Id

If you report an error with JDBC Technical lineage running on Edge, the Customer Support team can ask you for the Edge Site Id and Job Id. The team needs this information to access details about the error.

To retrieve the Job Id, see [View the summary of an technical lineage synchronization](#).

To retrieve the Site Id:

1. Go to **Settings**.
2. In the Edge section, click **Sites**.

3. Click the name of the Edge site.
 - » The Edge site Id is available in the ID field.

Message "Source 'source_name' was never processed with the current useSystemName flag"

Description	Solution
This error occurs when a technical lineage capability was synchronized with the following values set differently on Edge and for the lineage harvester: - The value of the Collibra system name setting on Edge. - The value of the useCol-libraSystemName property in the lineage harvester configuration file. Both values must be the same even if you use technical lineage via Edge and the lineage harvester for different data sources.	Complete the following steps: - Ensure that the value for the Collibra system name setting is the same with the value of the useCol-libraSystemName property in the lineage harvester configuration file. - Synchronize the technical lineage capability again.

Message "Failed to load artifacts message"

Description	Solution
If the Technical Lineage synchronization activity was not successful and you see error "failed to load artifacts" in the Lineage harvester synchronization dialog, it means the Technical Lineage capability could not be loaded in Edge.	Report this error and the Job Id to the Customer Support team for further investigation.

Message "A UNIQUE constraint failed"

Message code	Description	Solution
MSG-LIN-2501	A UNIQUE constraint failed.	When a technical lineage capability was being synchronized, synchronization processing failed because two capabilities were added for one BI tool data source with two different source IDs. To resolve this issue, complete the following steps: 1. If you do not have a lineage harvester installed, install one. 2. Enter the <code>list-sources</code> command and review the listed data sources to identify the data source that was added twice. 3. Take any of the following actions: ◦ If the technical lineage capability with the source ID that you want to remove still exists on Edge: i. On Edge, edit the technical lineage capability for the identified data source that you want to exclude by clearing the Active check box. ii. Synchronize the technical lineage capability for your data source again. ◦ If the technical lineage capability no longer exists on Edge: i. Enter the <code>ignore-source</code> command with the source ID that you want to remove. ii. Enter the full-sync command to synchronize the technical lineage again.

Message "Failed to fetch lineage API key because of a client error"

Message code	Description	Solution
MSG-LIN-3001	The DGC user name and DGC user password are not defined or incorrect.	Complete the following steps: • Verify the Edge technical lineage settings. • Synchronize the technical lineage again.

Message "MountVolume.NewMounter initialization failed" does not exist"

Description	Solution
You get the following message: <pre>MountVolume.NewMounter initialization failed for volume \"pv-shared-folder-2d53d256-fd6b- 4be8-a732-fe0f1c98704e-edge\" : path \"\\var\\lib\\edge\\storage\\dir\" does not exist</pre> When a technical lineage capability that uses a Shared Storage connection was being synchronized, the Shared Storage connection folder with the name of <code>dir</code> did not exist.	Complete the following steps: • Create a folder on the Edge site server. The folder path must be relative to <code>/var/lib/edge/storage/</code>. • When you create the Shared Storage connection, specify the folder name. • Synchronize the technical lineage again. For more information, go to Create technical lineage via Edge.

Working with custom technical lineage

You can create a custom technical lineage to include metadata of [data sources](#) that the lineage harvester does not support or add functionality that is not supported.

To create a custom technical lineage, define the custom technical lineage in a JSON file and refer to the JSON file in the lineage harvester configuration file. The lineage harvester generates a technical lineage based on your definition in the JSON file. You can create the following custom technical lineages:

- A simple custom technical lineage, which defines a basic object hierarchy and creates a lineage between two or more data objects.
- An advanced custom technical lineage, which contains a simple custom technical lineage and uses separate source code files that define lineage transformations to create the lineage.

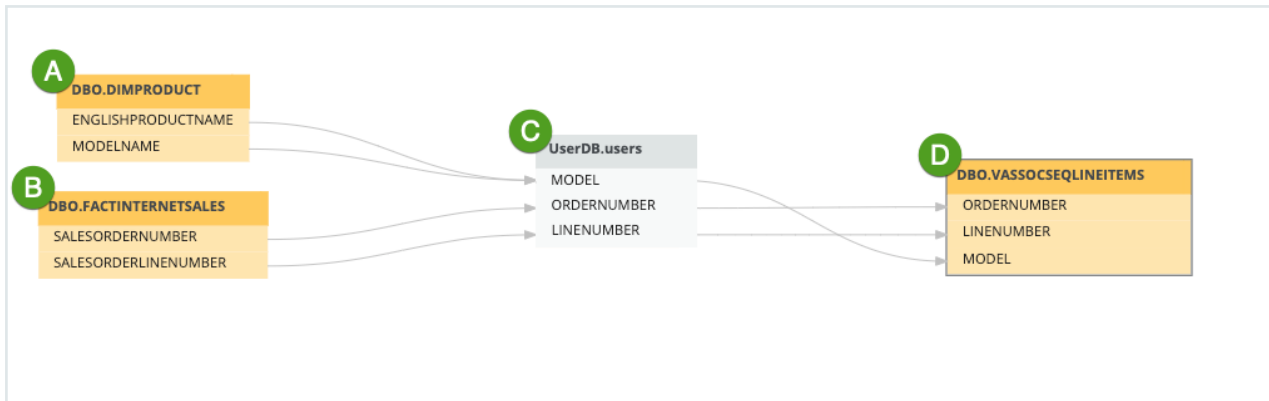
You can use the custom technical lineage as your only lineage source. You can also combine custom technical lineage with other lineage sources. For example, you can configure the lineage harvester to collect data objects from Oracle, Tableau and the custom technical lineage definition in the JSON file.

For steps to create a custom technical lineage by using the lineage harvester, go to [Create custom technical lineage](#).

For steps to create a custom technical lineage on Edge, go to [Create technical lineage via Edge](#).

Example

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. You can create a JSON file that contains the metadata of system C and generate the following technical lineage graph.



Creating custom technical lineage via lineage harvester

This topic is an overview of steps to create a custom technical lineage.

Before you begin

- [Set up the latest lineage harvester.](#)
- To stitch the data objects of your data sources with Data Catalog assets, [prepare the Data Catalog physical data layer for technical lineage](#). When you prepare the Data Catalog physical data layer, you must register your data sources in Data Catalog and use a structure that matches the structure of ingested assets in Data Catalog.
- [Determine whether you want to create a simple or advanced custom technical lineage.](#)

Requirements and permissions

- A [global role](#) with the following [global permissions](#):
 - Manage all resources
 - System administration
- 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

Steps

1. [Create a custom technical lineage JSON file.](#)
2. [Configure the lineage harvester for the custom technical lineage.](#)
3. [Run the lineage harvester.](#)

Create a custom technical lineage JSON file

To create a custom technical lineage, create a JSON file that defines the custom technical lineage, refer to the JSON file in the lineage harvester configuration file, and run the lineage harvester.

Steps

1. Create a local folder.
2. Create a JSON file in the local folder and name the JSON file **lineage.json**.
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 [Working with custom technical lineage](#).
4. Specify the JSON file to define a simple or an advanced custom technical lineage. For details about the JSON file, go to [Custom technical lineage JSON file](#) and [Custom technical lineage JSON file examples](#).

What's next

[Configure the lineage harvester](#) and refer to this JSON file in the lineage harvester configuration file.

Custom technical lineage JSON file

In the **lineage.json** file, you can define a basic data object hierarchy, a lineage between two or more data objects and transformations that 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 transformation definitions in a source code file.

If you want to create a simple custom technical lineage, specify the `tree` and `lineages` sections. You can add the transformation code in the `lineages` section.

If you want to create an advanced custom technical lineage, specify the `tree`, `lineages` and `codebase_files` sections. Add references to transformation code in source code files in the `codebase_files` section.

Transformation code in both simple and advanced custom technical lineages is displayed at the bottom part of the Collibra 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 the 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.
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.
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 lineage.json 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.

Properties	Description
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>. Note If the <code>useCollibraSystemName</code> property in the lineage harvester configuration file for custom technical lineage 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>, the system data object is not used for stitching.
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.
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.
<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: system > database > schema > table > column. This example shows data objects that cannot be stitched: dashboard > report > column. Note If the <code>useCollibraSystemName</code> property in the lineage harvester configuration file for custom technical lineage 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>, the system data object is not used for stitching.
mapping	Yes Simple custom technical lineage only	The mapping name. This property specifies a name for the transformation code.

Properties	Required	Description
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.
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 directory 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.

Properties	Required	Description
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 pos_start property of the mapping_refs property in the codebase_files section.
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
<code><mapping></code>	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 <code>lineage</code> section.
<code>pos_start</code>	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 <code>lineage</code> section.
<code>pos_len</code>	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": 10</code> 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 sample JSON files that define a simple custom technical lineage and an advanced custom technical lineage, see [Custom technical lineage JSON file example](#).

Custom technical lineage JSON file examples

This topic shows sample `lineage.json` files that create a simple custom technical lineage and an advanced custom technical lineage.

Each sample 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

Sample JSON file for a simple custom technical lineage

In the following sample, 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.

To show the transformation code at the bottom of the Collibra technical lineage graph that uses a simple custom technical lineage, 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"
}
]
}

```

Sample JSON file for an advanced custom technical lineage

In the following sample, 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.

```

{
  "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
            }
        }
    }
}

```



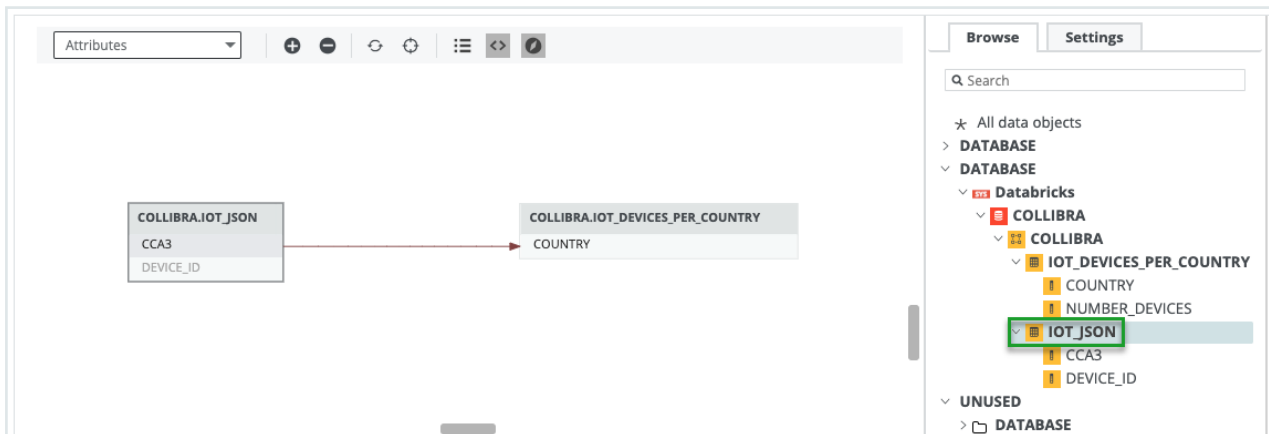
```

    }
  }
}

```

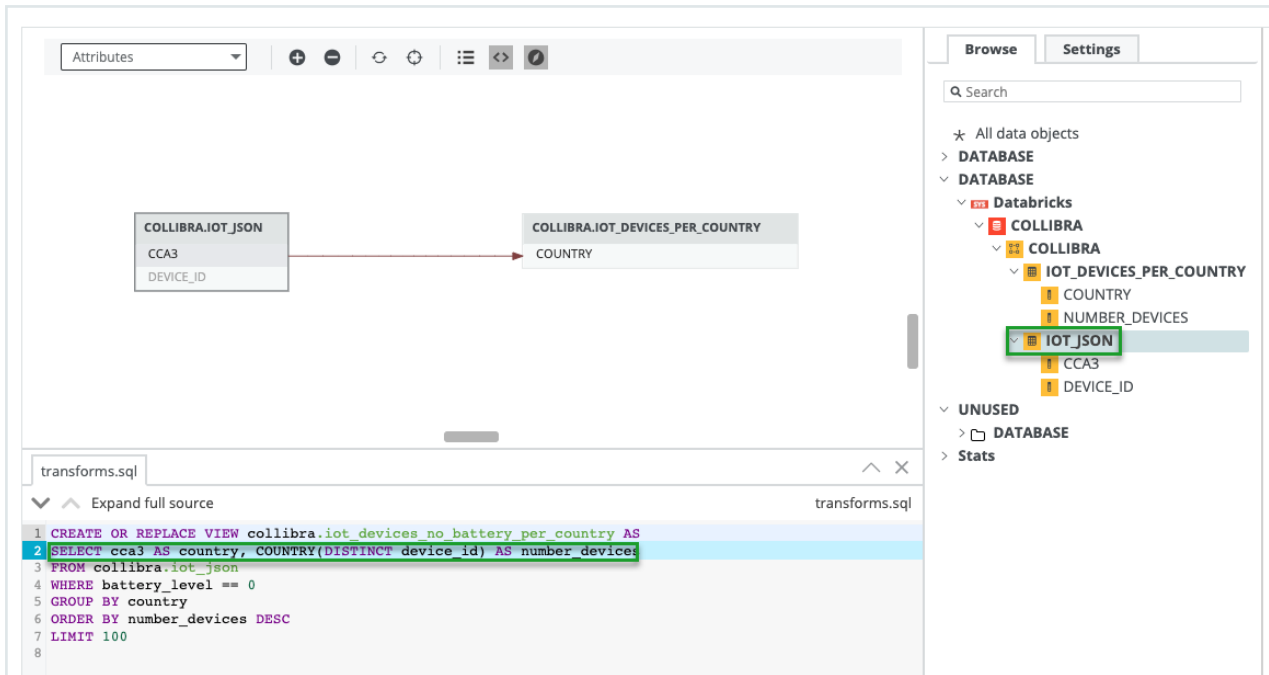
Sample technical lineage graphs

Both sample **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 sample **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.



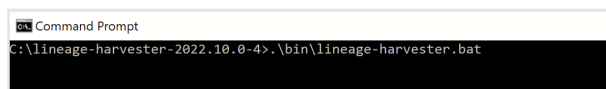
Configure the lineage harvester for a custom technical lineage

To create a custom technical lineage, create a JSON file that defines the custom technical lineage, refer to the JSON file in the lineage harvester configuration file, and run the lineage harvester.

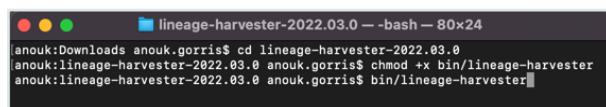
Steps

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

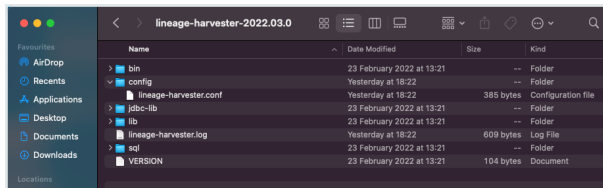
- **Windows:** `.\bin\lineage-harvester.bat`



- **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. Specify the lineage harvester configuration file and save the configuration file. For details about the configuration file, see [Lineage harvester configuration file for the custom technical lineage](#).

What's next

[Run the lineage harvester.](#)

Lineage harvester configuration file for the custom technical lineage

The lineage harvester uses this lineage harvester configuration file to extract data from the metadata of the data sources that you want to process.

When you run the lineage harvester for the first time, it creates an empty lineage harvester configuration file. You can manually add properties and values to the configuration file.

If you want to create the technical lineage for multiple data sources, use the [configuration file generator](#) to create an example configuration file with different data sources, and update the example to match your data source information.

Requirements and restrictions

- In the configuration file, you must use UTF-8 or ISO-8859-1 characters, with the exception of SQL files, which can only be UTF-8 encoded.
- Comments in the lineage harvester configuration file are not supported.
- Technical lineage supports the username and password authentication method for the custom technical lineage.

Format

```
{
  "general" : {
    "catalog" : {
      "url" : "",
      "username" : "",
    },
    "useCollibraSystemName" : false|true
  },
  "sources" : [ {
    "type" : "ExternalDirectory",
    "id" : "",
    "dirType" : "custom-lineage",
    "collibraSystemName" : "",
    "path" : "",
    "deleteRawMetadataAfterProcessing": false|true
  } ]
}
```

Properties	Description
general	Describes the connection between Collibra Data Lineage and Data Catalog.
catalog	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. Specify the public URL of your Collibra environment. Other URLs are not accepted.
username	The username that you use to sign in Collibra.

Properties	Description
<code>useCollibraSystemName</code>	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. Specify one of the following values: <code>false</code> The lineage harvester does not stitch the system or server name of your data source to the System asset in Data Catalog. This is the default value. <code>true</code> The lineage harvester reads the system or server names that you specify for the system data object in the <code>tree</code> and <code>lineage</code> sections in the custom technical lineage JSON file and stitches the names to the System assets in Data Catalog. Only specify this value when you have multiple databases with the same name.
<code>sources</code>	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.
<code>type</code>	The kind of data source. The value must be <code>ExternalDirectory</code>.
<code>id</code>	The unique ID of your custom technical lineage. This property identifies the metadata that the lineage harvester processes. Specify this property with an unique string, for example, <code>MyCustomLineage</code>.

Properties	Description
dirType	The type of external directory. The value is <code>custom-lineage</code> .
collibraSystemName	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 the <code>tree</code> and <code>lineage</code> sections in the custom technical lineage JSON file.
path	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.
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.

Example

```
{
  "general" : {
    "catalog" : {
      "url" : "https://companydomain.collibra.com",
      "username" : "my-Collibra-username",
    },
    "useCollibraSystemName" : false
  },
  "sources" : [{
    "id": "MyCustomLineage",
    "type": "ExternalDirectory",
    "dirType": "custom-lineage",
    "path": "/path/to/custom-lineage/dir/",
    "collibraSystemName": "MySystemName"
  }
  ]
}
```

Run the lineage harvester

To create a custom technical lineage, create a JSON file that defines the custom technical lineage, refer to the JSON file in the lineage harvester configuration file, and run the lineage harvester.

Before you begin

If you use a proxy server, connect to the proxy server. For more information, go to [Connecting to a proxy server](#).

Steps

1. Start the lineage harvester by entering the `full-sync` command.
 - To process data from all data sources in the configuration file, enter the following command:

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, enter the following command:

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"
```

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

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.

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 troubleshooting guide](#) or [Collibra Support Portal](#) to fix the error.

If you want to synchronize the data sources on fixed times, you can use [scheduled jobs](#).

Working with Tableau

Tableau is business intelligence software that helps people see and understand their data. Integrating Tableau in Collibra Data Intelligence Cloud enables you to see metadata from Tableau Server and Tableau Online in CollibraData Catalog.

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](#)

Tableau terminology	343
Tableau asset types and domain types	345
Tableau operating model	346
Technical lineage for Tableau	363
Supported data sources in Tableau	382
Set up Tableau	384
Prepare a domain for Tableau ingestion	392
Prepare the Data Catalog physical data layer for Tableau stitching	394
Soft delete of Tableau assets	398
Tableau hostname, schema, and system name mapping (Beta)	399
Technical lineage for Tableau via Edge	401
Technical lineage for Tableau via the lineage harvester	419

Tableau terminology

The following table shows the Tableau terminology and corresponding asset types and terminology in Collibra Data Intelligence Cloud.

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
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

Tableau asset types and domain types

The [Tableau integration](#) of Collibra Data Intelligence Cloud uses a specific subset of [asset types](#) and [domain types](#). All of these come out of the box with your software.

The following table shows the asset and domain types that are used for the Tableau integration. Above each asset type you can see the parent asset types in the breadcrumbs.

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

Asset type	Description	Domain type
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
Technology Asset ▸ Server ▸ BI Server ▸ Tableau Server	A visual analytics platform for creating interactive dashboards and rich visualisations	BI Catalog

Tableau operating model

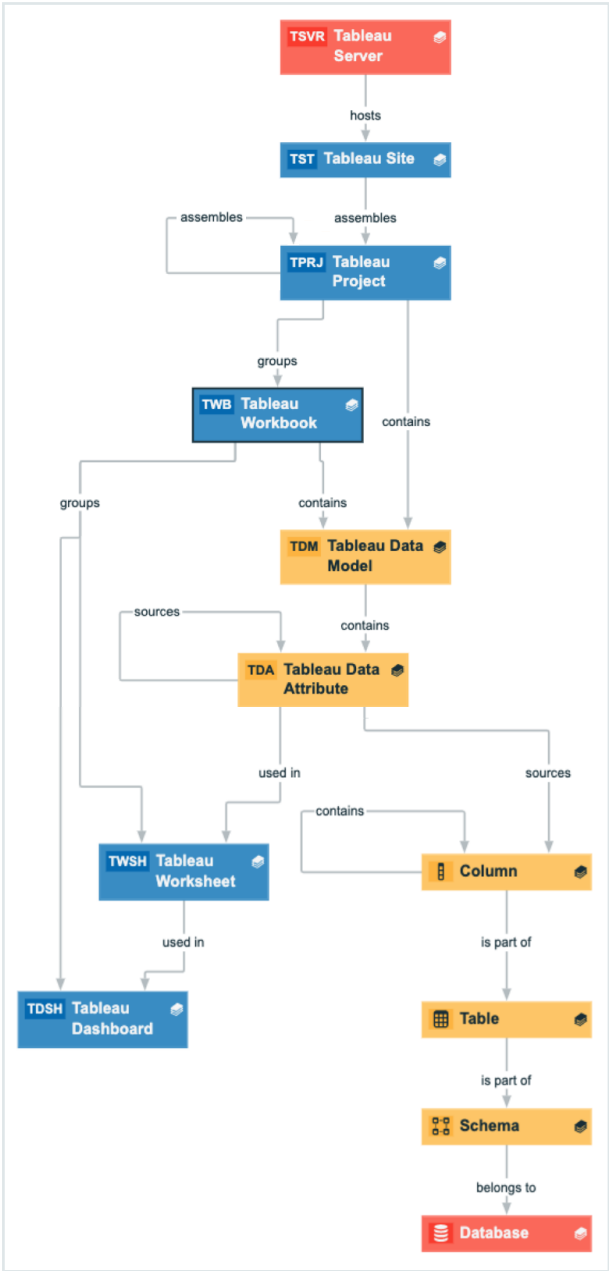
Synchronizing means refreshing the assets that are currently in Data Catalog as a result of a previous ingestion or synchronization job. After synchronizing Tableau, the assets in Data Catalog accurately reflect the metadata as it exists at the time of synchronization.

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	BI Folder assembles / Is assembled in BI Folder	00000000-0000-0000-0000-120000000001
	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

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>.	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>.	00000000-0000-0000-0000-200000000001
	Report Image	00000000-0000-0000-0000-000000000262
	URL: The link to the data in Tableau	00000000-0000-0000-0000-000000000258
	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.	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>.	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
	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 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
	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>.	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

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.

Example of ingested Tableau metadata

The following image shows an example structure after synchronizing Tableau.

The screenshot shows the 'Business Analysts Community' interface. The left sidebar contains navigation links: Overview, Organization, Comments, Responsibilities, Assets, History, and Files. The main content area is titled 'Description' and shows a table of ingested Tableau metadata.

Name	Description	Domain Type	Owner	Stakeholder	Business Steward
Business Analysts Community					
New Tableau			Admin Istrator		
New Tableau		BI Catalog			
Schemas	Community containing all inge...				
Tableau			Admin Istrator		
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. In the tab pane, click  **Diagram**.
 - » The diagram appears 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"
    },
    {
      "id": "Tableau Project",
```

```

    "type":{
      "id":"000000000-0000-0000-0000-110000000001"
    },
    "layoutRegion":"context"
  },
  {
    "id":"Tableau Site",
    "type":{
      "id":"000000000-0000-0000-0000-110000000000"
    },
    "layoutRegion":"context"
  },
  {
    "id":"Tableau Server",
    "type":{
      "id":"000000000-0000-0000-0000-110000000005"
    },
    "layoutRegion":"context"
  },
  {
    "id":"Tableau Data Attribute",
    "type":{
      "id":"000000000-0000-0000-0000-110000000010"
    },
    "layoutRegion":"context"
  },
  {
    "id":"Column",
    "type":{
      "id":"000000000-0000-0000-0000-0000000031008"
    },
    "layoutRegion":"context"
  },
  {
    "id":"Table",
    "type":{
      "id":"000000000-0000-0000-0000-0000000031007"
    },
    "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":"boxing",
        "type":{
            "id":"000000000-0000-0000-0000-1200000000002"
        },
        "roleDirection":true
    },
    {
        "from":"Tableau Site",
        "to":"Tableau Project",
        "label":"",
        "style":"boxing",
        "type":{
            "id":"000000000-0000-0000-0000-1200000000001"
        },
        "roleDirection":true
    },
    {
        "from":"Tableau Server",
        "to":"Tableau Site",
        "label":"",
        "style":"boxing",
        "type":{
            "id":"000000000-0000-0000-0000-1200000000000"
        },
        "roleDirection":true
    },
    {
        "from":"Tableau Data Model",
        "to":"Tableau Data Attribute",
        "label":"",
        "style":"boxing",
        "type":{
            "id":"000000000-0000-0000-0000-0000000007196"
        },
        "roleDirection":true
    },
    {
        "from":"Tableau Data Attribute",
        "to":"Tableau Data Attribute",
        "label":"",

```

```

    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-0000000007069"
    },
    "roleDirection": false
  },
  {
    "from": "Tableau Workbook",
    "to": "Tableau Data Model",
    "label": "",
    "style": "boxing",
    "type": {
      "id": "000000000-0000-0000-0000-1200000000020"
    },
    "roleDirection": true
  },
  {
    "from": "Tableau Project",
    "to": "Tableau Data Model",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-1200000000014"
    },
    "roleDirection": true
  },
  {
    "from": "Column",
    "to": "Column",
    "label": "",
    "style": "boxing",
    "type": {
      "id": "000000000-0000-0000-0000-0000000007042"
    },
    "roleDirection": false
  },
  {
    "from": "Column",
    "to": "Table",
    "label": "",
    "style": "boxed",
    "type": {
      "id": "000000000-0000-0000-0000-0000000007042"
    },
    "roleDirection": true
  },
  {
    "from": "Table",
    "to": "Schema",
    "label": "",

```

```

        "style": "boxed",
        "type": {
            "id": "00000000-0000-0000-0000-0000000007043"
        },
        "roleDirection": false
    },
    {
        "from": "Schema",
        "to": "Database",
        "label": "",
        "style": "boxed",
        "type": {
            "id": "00000000-0000-0000-0000-0000000007024"
        },
        "roleDirection": false
    },
    {
        "from": "Tableau Data Attribute",
        "to": "Tableau Worksheet",
        "label": "",
        "style": "arrow",
        "type": {
            "id": "00000000-0000-0000-0000-1200000000021"
        },
        "roleDirection": false
    },
    {
        "from": "Tableau Workbook",
        "to": "Tableau Worksheet",
        "label": "",
        "style": "boxing",
        "type": {
            "id": "00000000-0000-0000-0000-1200000000004"
        },
        "roleDirection": true
    },
    {
        "from": "Tableau Workbook",
        "to": "Tableau Dashboard",
        "label": "",
        "style": "boxing",
        "type": {
            "id": "00000000-0000-0000-0000-1200000000004"
        },
        "roleDirection": true
    },
    {
        "from": "Tableau Worksheet",
        "to": "Tableau Dashboard",
        "label": "",

```

```

        "style":"arrow",
        "type":{
          "id":"000000000-0000-0000-0000-1200000000007"
        },
        "roleDirection":false
      },
      {
        "from":"Tableau Data Attribute",
        "to":"Column",
        "label":"",
        "style":"arrow",
        "type":{
          "id":"000000000-0000-0000-0000-0000000007069"
        },
        "roleDirection":false
      },
      {
        "from":"Tableau Project",
        "to":"Tableau Project",
        "label":"",
        "style":"boxed",
        "type":{
          "id":"000000000-0000-0000-0000-1200000000001"
        },
        "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
    }
  }
}

```

Technical lineage for Tableau

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

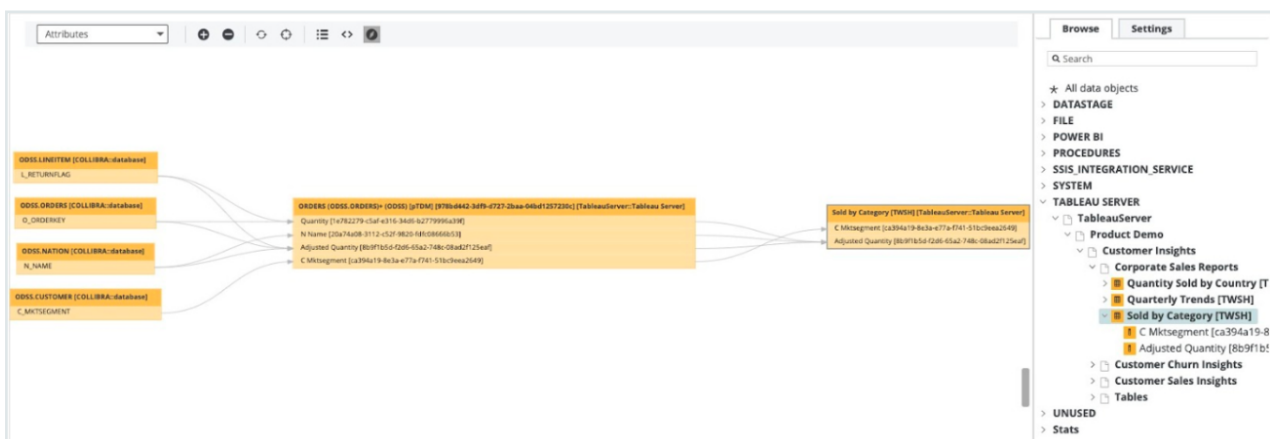
The [technical lineage graph](#) shows relations of the type "Data Element sources / targets Data Element" between Tableau assets and other data objects in the data flow, for example between a Column asset and a Tableau Data Attribute asset. These relations are created during the Tableau ingestion process 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*.



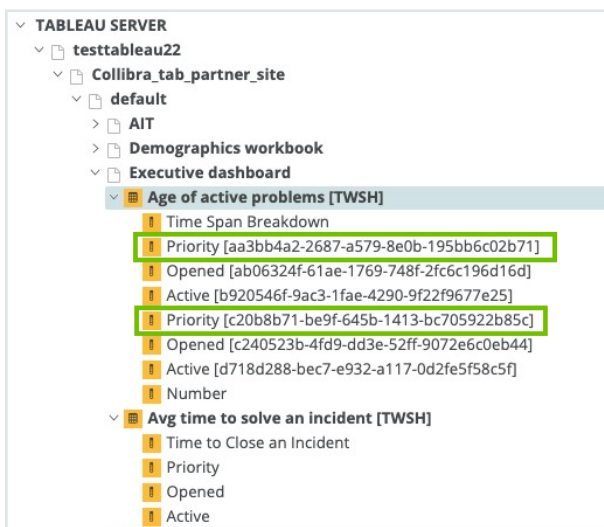
UUIDs in the Tableau technical lineage

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.

```

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%.

Error codes

The `Errors` summary represents a summary of all errors per Tableau site. Currently, processing stops after the first error is identified during the processing of the site. In a future version of Collibra, processing will continue after the first error.

Sources		Stitching						
Selection	Source ID	Scanner type		Success rate	Done	Parsing Error	Analyze Error	Last sync
☐	tableau	TABLEAU		100 %	1	0	0	2023-01-04
☒				0 %	0	0	0	None UTC
Full-text search		Q		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.

Element	Description																				
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																	
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 <code>TABLE_ID</code> is skipped Indicates that the parent database for a table with <code>TABLE_ID</code> was not found in the previously processed databases. Possible causes: • The <code>database</code> property is not present in the JSON file. • The <code>database</code> property is empty: <code>"database": {}</code>. • The <code>DATABASE_ID</code> is not present for the <code>id</code> 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 <code>project</code> property is not present in the JSON file. • The <code>project</code> property is empty: <code>"project": {}</code>. • The <code>PROJECT_ID</code> is not present for the <code>id</code> property. Example for a workbook: <pre> { "workbooks": { "workbook": [{ "project": { "id": "PROJECT_ID" }, "id": "WORKBOOK_ID", ... }] } } </pre> Example for a published datasource: To identify the <code>PROJECT_ID</code>, first find the <code>DATASOURCE_LUID</code> 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": { "nodes": [{ "id": "DASHBOARD_ID", </pre>

Group name	Name	Description
		<pre>"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": "Embed- dedDatasource", "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": "Data- sourceField", "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: 1. 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> 2. 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": [{ "upstreamColumnsCon- nection": { "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>
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.

Group name	Name	Description
Configuration	Invalid Collibra system names	Raised during the processing of the <code>collibraSystemNames</code> 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 <code>hostnameMapping</code> 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"

Supported data sources in Tableau

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

Data source	Version	Support for technical lineage	Support for stitching
MySQL	5.7, 8 and newer	Yes	Yes
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

Set up Tableau

Before you start the Tableau integration in Data Catalog, make sure that the lineage harvester can reach the Tableau metadata. Perform these tasks before you start the actual Tableau ingestion 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.

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 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 [Publishing data separately or embedded in workbooks](#).

In Collibra, 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.

Tableau roles and permissions

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.

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 your Tableau Online or Tableau Server is licensed with the Data Management add-on.
- 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.

Here are a few tested configurations for the Explorer role:

Combination of accessibility factors	You can access...
• Data Management add-on: Yes • Project leader: Yes • Derived permissions: No	• All Tableau data objects. • External assets.
• Data Management add-on: No • Project leader: Yes • Derived permissions: Yes	• All Tableau data objects. • External assets for which you have derived permissions.
• Data Management add-on: No • Project leader: Yes • Derived permissions: No	• All Tableau data objects only.
• Data Management add-on: No • 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.

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

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 in Tableau

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 in Tableau

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

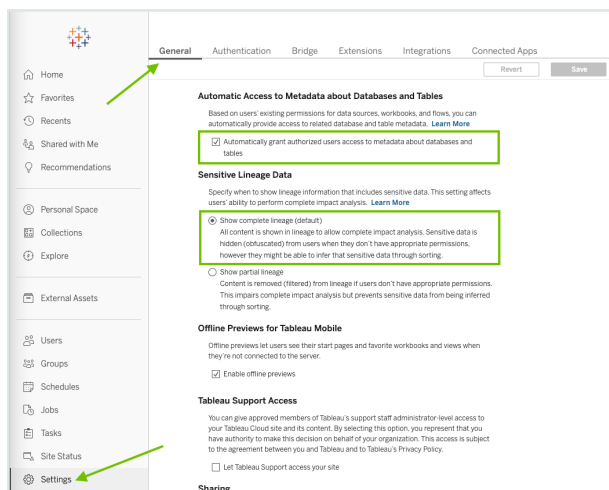
- You have at least a View permission on the Tableau projects, workbooks and data sources you want to ingest.
- You have an Administrator role or you have the Explorer role with a sufficient combination of accessibility factors, as previously described in [Explorer role](#).

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



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.

For complete information on ingestion results based on your Tableau permissions, see [Tableau ingestion results](#).

Tableau ingestion results

The following tables show the ingestion results based on Tableau permissions.

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.

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 a domain for Tableau ingestion

Before you can ingest Tableau metadata, you have to designate a domain for storing the new Tableau 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.

Warning If you are using Collibra Data Intelligence Cloud 2021.11 or older, you have to add all Tableau attributes in the [operating model](#) to a scope and create a scoped assignment before you ingest Tableau via the lineage harvester. For complete information and step-by-step instruction, see [Tableau general troubleshooting](#).

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 a domain type from the list.

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
Type	The domain type of the domain you are creating. In this case, you need to select <i>BI Catalog</i> .
Community	The community under which the domain will be located.
Name	The name of the new domain or domains. Tip You can create multiple domains in one go. To do this, press Enter 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=00000000-0000-0000-0000-0000000040001`. 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 [Prepare the lineage harvester configuration file for Tableau](#).

Warning If you are using Collibra 2021.11 or older, you have to add all Tableau attributes in the [operating model](#) to a scope and create a scoped assignment before you ingest Tableau via the lineage harvester. For complete information and step-by-step instruction, see [Tableau general troubleshooting](#).

What's next?

[Prepare](#) the Data Catalog physical data layer.

Prepare the Data Catalog physical data layer for Tableau stitching

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 via 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 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.

Note 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.

For more information, see [Automatic stitching](#) and [Prepare the lineage harvester configuration file for Tableau](#).

Important In the global [assignment](#) of each asset type included in the TableauPower BI [operating model](#), ensure that none of the characteristics that are in the operating model have a maximum cardinality of “0”. If the maximum cardinality is set to “0” for any such characteristics, ingestion will fail.

Prerequisites

- You have a [global role](#) with the Catalog [global permission](#), for example, Catalog Author.
- You have a role with the following [resource permissions](#) on the **Schema** community:
 - Asset: add
 - Attribute: add
 - Domain: add
 - Attachment: add

Steps

1. [Register](#) a database as data source.
 - » After registration, the assets of the following asset types are created in Data Catalog:
 - Schema
 - Table
 - Column
2. Create a Database asset.

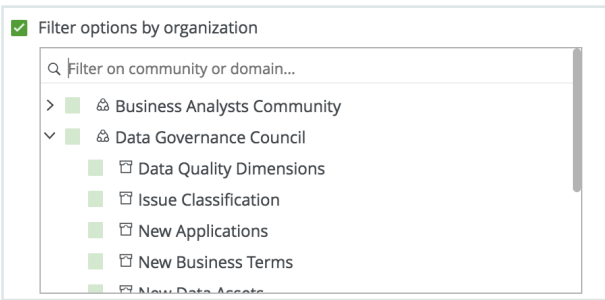
Tip We strongly recommend to use the name as your original data source, so that the name of the Database asset matches [Tableau's naming convention](#).

1. Open the product for which you want to create an asset (for example, **Business Glossary**).
2. On the main toolbar, click **+**.
3. On the **Assets** tab, click **Database**.
 - » The **Create Asset** dialog box appears.
4. Enter the required information.

Field	Description
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.
Name	A name to identify 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.

5. 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 Database asset and the Schema asset using the Technology Asset has / belongs to Schema relation type.
 - 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.
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. Leave this field empty to create a permanent relation.
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**.

4. Check that the following relations are created for all Column assets that you want to stitch to Tableau assets:
 - Schema contains / is part of Table
 - Column is part of / contains Table

What's next?

Set up the lineage harvester for Tableau ingestion.

Soft delete of Tableau 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.

Tableau hostname, schema, and system name mapping (Beta)

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.

Note This feature has been validated by several customers. It is in beta, however, because it represents a significant change in your [Tableau <source ID> configuration file](#). The beta period gives you time to adopt the new feature, while we gather more feedback about its functionality.

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.

Important

- This section replaces the following sections, and should not be used in combination with either of the following sections:
 - The `databaseMapping` section, if included.
 - The `databases` sub-section of the `collibraSystemNames` section, if included.

When the beta period ends and the `hostnameMapping` section is generally available, the `databaseMapping` section and the `databases` sub-section of the `collibraSystemNames` section will be deprecated.

- If you use the `hostnameMapping` section, you can still use the `collibraSystemName` property in conjunction with the `files`, `connectors` or `cloudfiles` sub-sections.

For descriptions of these properties, go to [Prepare a Tableau <source ID> configuration file](#).

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 Collibra 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",
    "collibraSystemName": "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",
  }
}
```

Technical lineage for Tableau via Edge

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.

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 assets from [registered data sources](#).
- Between ingested assets from BI sources and Data Catalog assets from registered data sources.

Before Collibra 2023.02, if you wanted to collect source code from Tableau and create these new relations, you had to prepare a configuration file and use the [lineage harvester](#). With Collibra 2023.02, you can now integrate Tableau by configuring the lineage harvester as a capability on Edge. Benefits include:

- Seamless integration with Data Catalog.
- The Edge User Interface (UI), instead of Command Line Interface (CLI).
- Tableau connection from Edge, instead of lineage harvester drivers.

You can still use the lineage harvester as a standalone app.

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.

Creating a technical lineage for Tableau via Edge

This topic includes an overview of the tasks to complete to create a technical lineage for Tableau via Edge.

Before you begin

- [Create](#) an Edge site in Collibra Data Intelligence Cloud.
- Optionally, [configure](#) a forward proxy.
- [Install](#) an Edge site.
- [Enable and configure](#) technical lineage via Edge.

Steps

1. [Prepare](#) a domain for Tableau ingestion.
2. [Create](#) a connection to Tableau.
3. [Add](#) the Technical lineage for Tableau capability to the Edge site.

Warning If you are switching between the lineage harvester and Edge, the value in the **Source ID** field in your Edge capability must exactly match the value of the `id` property in your [lineage harvester configuration file](#).

4. [Synchronize](#) the Tableau technical lineage

What's next?

- View the domain in which you ingested the assets.
- [View](#) the technical lineage.

Prepare a domain for Tableau ingestion

During Tableau integration, Tableau assets are ingested in the domain you specify in the **Domain ID** field in the [Technical Lineage for Tableau capability](#).

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. In the list, 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
Type	The domain type of the domain you are creating. In this case, <i>BI Catalog</i> .
Community	The community under which the domain will be located.
Name	The name of the new domain or domains.

5. Click **Create**.

6. Open your domain.

7. Copy the reference ID of the 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 ID in the **Domain ID** field in the Technical Lineage for Tableau capability.

Warning Collibra Data Lineage creates all Tableau assets in a single BI Catalog domain. We highly recommend that you do not move these assets to another domain. If you move assets to another domain, they will be deleted and recreated in the initial BI Catalog domain when you [synchronize Tableau](#). As a consequence, all manually added data of those assets is lost.

Create a connection to Tableau

To retrieve data from Tableau, you have to connect to Tableau via the Edge site.

Prerequisites

- You have a [global role](#) that has the **System administration global permission**.
- You have a [global role](#) that has the Manage Edge sites [global permission](#).
- You have a [resource role](#) with the Configure external system [resource permission](#), for example, Owner.
- You have [registered Tableau](#).

- If you connect to Tableau Online, you have a Tableau user with at least Viewer rights.
- If you connect to Tableau Server, you have a Tableau user with access to at least one site.
- You have the necessary [Tableau permissions](#).

Steps

1. Open an Edge site.
 - a. In the main menu, click **Settings**.
 - » The **System** tab page appears.
 - b. In the tab pane, click **Edge**.
 - » The Edge sites overview appears.
 - c. In the Edge site overview, click the name of an Edge site.
 - » The Edge site page appears.
2. In the **Connections** section, click **Create Connection**.
 - » The **Connection settings** page appears.

3. Enter the required information.

Field	Description	Required
Connection settings	This section contains general information about the connection.	
Name	The name of the connection. The name can be anything, as long as it is unique. Tip The name that you provide here is the name you have to select in the Tableau connection field, when adding the Technical Lineage for Tableau capability to the Edge site.	✓ Yes
Description	The description of the connection.	✗ No
Connection provider	The type of connection. Select Tableau connection.	✓ Yes
Connection parameters	This section contains connection authentication information.	
URL	The URL of your Tableau server.	✓ Yes
Authentication type	The authentication type for your connection to the Tableau server.	✓ Yes
Username/Token name	◦ If you selected authentication type <i>username</i>, enter the username of the Tableau user. ◦ If you selected authentication type <i>token</i>, enter the personal access token name of the Tableau user.	✓ Yes
Password/Token secret	◦ If you selected authentication type <i>username</i>, enter the password of the Tableau user. ◦ If you selected authentication type <i>token</i>, enter the personal access token secret of the Tableau user.	✓ Yes

4. Click **Create**.

What's next?

Add the Technical Lineage for Tableau capability to the Edge site.

Add the Technical Lineage for Tableau capability to the Edge site

After you [create](#) a connection to Tableau, you can add the Technical Lineage for Tableau capability.

Requirements and permissions

A [global role](#) that has the following [global permissions](#):

- System administration
- Manage connections and capabilities, for example Edge integration engineer.
- Register profiling information

Steps

1. Open an Edge site.
 - a. On the main menu, 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 table, click the name of the Edge site whose status is **Healthy**.
 - » The Edge site page opens.
2. In the **Capabilities** section, click **Add capability**.
 - » The **Add capability** page appears.
3. Enter the required information.

Field	Description	Required
Capability	This section contains general information about the capability.	

Field	Description	Required
Name	The name of the Edge capability.	✓ Yes
Description	The description of the Edge capability.	✗ No
Capability template	The capability template for Tableau. Select Technical Lineage for Tableau .	✓ Yes
Main Properties	The required information for creating a technical lineage.	
Source ID	The name of the data source. You can give this any name, as long as it is unique. Warning If you are switching between the lineage harvester and Edge, the value in this field must exactly match the value of the <code>id</code> property in your lineage harvester configuration file.	✓ Yes
Tableau connection	The Tableau connection that you created for ingestion in Data Catalog. Tip Select the name that you provided in the Name field when you created a connection to Tableau.	✓ Yes
Domain ID	The unique reference ID of the domain in Collibra Data Intelligence Cloud in which you want to ingest the Tableau assets. How do I find a domain reference ID? Open the relevant domain in Collibra. The URL looks like: <a href="https://<yourcollibrainstance>/domain/22258f64-40b6-4b16-9c08-c95f8ec0da26?view=00000000-0000-0000-0000-000000040001">https://<yourcollibrainstance>/domain/22258f64-40b6-4b16-9c08-c95f8ec0da26?view=00000000-0000-0000-0000-000000040001. In this example, the reference ID is in bold.	✓ Yes

Field	Description	Required
REST only	Indication whether or not you want to use both the Tableau REST API and Tableau Metadata API to harvest Tableau metadata. ◦ Cleared (default): The lineage harvester will use the REST API and Metadata API to harvest Tableau metadata. ◦ Selected: The lineage harvester will only use the REST API to harvest Tableau metadata. Note This field must be cleared, 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.	✗ No
Exclude images	Indication whether or not you want to excluding the downloading of images. ◦ Cleared: Images are downloaded. ◦ Selected (default): Images are not downloaded. 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.	✗ No

Field	Description	Required
Site ID	The site IDs of the Tableau sites that you want to include in the ingestion process. If you leave this field empty, Collibra Data Lineage will ingest the default Tableau site. To ingest from multiple Tableau sites, enter each site ID in a separate Site ID field. To ingest the default Tableau site, enter <i>"Default"</i> or leave the field empty. This field is not case sensitive. Warning If you enter <i>"Default"</i>, you must include the double quotation marks. The site IDs of any other Tableau sites must not be enclosed in double quotation marks. If the formatting of the site IDs does not conform to this detail, ingestion will fail. Example Let's say that you want to ingest from the default Tableau site and a site named "ABC". - In the Site ID field, enter <i>"Default"</i> for the default Tableau site. - Click Add property. » An empty Site ID field is added. - In the new Site ID field, enter the ID of the Tableau site ABC. Site ID Default Site ID ABC Add property	✕ No

Field	Description	Required
	Tip Ensure that you specify the correct value. The correct value 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 /site/ 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 Profits, on a site named Sales, 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 Sales does not figure in the URL.	
Site Name	The site name, or names, of the Tableau sites you specified in the Site ID field. If you don't provide a site ID in the Site ID field, in which case the default Tableau site is ingested, leave this field blank.	✗ No
Concurrency level	This field 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.	✗ No

Field	Description	Required
Source configuration	This field allows you to provide JSON code for database mapping, domain mapping and filtering. Tip If you previously integrated Tableau via the lineage harvester, you can copy and paste in this field the JSON code from your Tableau <source ID> configuration file. Show me some examples Database mapping: <pre>"databaseMapping": { "<hostname:port>": "<actual database name>" },</pre> Domain mapping: <pre>"domainMapping": { "<Site-1>": "reference-id-of-Domain-1", "<Site-1> > <Project-Default>": "reference-id-of-Domain-2" }</pre> Filtering: <pre>"filters": { "projects":{"site_name2 > project_name2": "domain-reference-id2", "site_name3 > project_name3 > sub-project_name": "domain-reference-id2"} }</pre>	✕ No

Field	Description	Required
Custom Properties	This section identifies to which Collibra Data Lineage service instance you want to upload the Tableau metadata. Warning This applies only to US government customers.	
techlinKey	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 to US government customers.	✗ No
techlinHost	The URL of the Collibra Data Lineage service instance to which you want to upload Tableau metadata, for example "techlin-gcp-eu.collibra.com". Warning This applies only to US government customers.	✗ No
Advanced Properties	This section contains the advanced properties for creating a technical lineage.	
Delete Raw Metadata After Processing	Technical lineage via Edge harvests raw metadata from specified data sources and uploads it in a ZIP file to a Collibra Data Lineage service instance. This option indicates whether the raw metadata should be deleted from the Collibra Data Lineage service instance after the metadata that is targeted for ingestion in Data Catalog is processed. Select this option to indicate that the raw source metadata is deleted after processing. Clear the checkbox to keep the raw source metadata after processing. In this case, it is stored in the Collibra infrastructure. Note Selecting this option can negatively impact performance.	

Field	Description	Required
Analyze Only	This option determines whether to only load and analyze the source data on the Collibra Data Lineage service instances. When you select this option, the technical lineage of the data source is not created during the synchronization of the capability. Selecting this option is equivalent to entering the load-sources and analyze command with a source specified when you use the lineage harvester. This option is not enabled by default. Use case You can use this option to control when to start full synchronization of data sources. For example if you have three data sources, A, B, and C, you can synchronize the data sources as follows: - During weekdays, synchronize data sources A and B with the Analyze Only checkbox selected in the capabilities. Collibra Data Lineage only loads and analyzes data sources A and B without synchronizing the technical lineage. - On the weekend, synchronize data source C without selecting the Analyze Only checkbox in the capability. Collibra Data Lineage synchronizes the technical lineage for all data sources including A, B, and C.	✗ No
Active	The option determines whether to include or remove the technical lineage of the data source. Select this option to include the technical lineage of this data source. Clear the checkbox to exclude the technical lineage of this data source.	✗ No

Field	Description	Required								
Paging	This section allows you to customize 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. 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, "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><tr><td>Database</td><td> ◦ <code>databasesPageSize</code>: The number of databases 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.	Database	◦ <code>databasesPageSize</code>: The number of databases per page.	✕ No
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.									
Database	◦ <code>databasesPageSize</code>: The number of databases per page.									

Field	Description	Required												
	<table><tr><th>Metadata type</th><th>Setting and description</th></tr><tr><td>Table</td><td> ◦ <code>tablesPageSize</code>: The number of tables per page. ◦ <code>tablesColumnsPageSize</code>: The number of table columns per page. </td></tr><tr><td>Table columns</td><td> ◦ <code>tableColumnsPageSize</code>: The number of table columns per page. </td></tr><tr><td>Users</td><td> ◦ <code>usersPageSize</code>: The number of users per page. </td></tr><tr><td>Data source</td><td> ◦ <code>datasourcesPageSize</code>: The number of data sources per page. ◦ <code>datasourcesFieldsPageSize</code>: The number of data source fields per page. ◦ <code>columnsLimit</code>: The number of data source field columns per page. ◦ <code>fieldsLimit</code> : The number of referenced data source fields per page. </td></tr><tr><td>Data source field</td><td> ◦ <code>datasourceFieldsPageSize</code>: The number of data source fields per page. ◦ <code>columnsLimit</code>: The number of data source field columns per page. ◦ <code>fieldsLimit</code> : The number of referenced data source fields per page. </td></tr></table>	Metadata type	Setting and description	Table	◦ <code>tablesPageSize</code>: The number of tables per page. ◦ <code>tablesColumnsPageSize</code>: The number of table columns per page.	Table columns	◦ <code>tableColumnsPageSize</code>: The number of table columns per page.	Users	◦ <code>usersPageSize</code>: The number of users per page.	Data source	◦ <code>datasourcesPageSize</code>: The number of data sources per page. ◦ <code>datasourcesFieldsPageSize</code>: The number of data source fields per page. ◦ <code>columnsLimit</code>: The number of data source field columns per page. ◦ <code>fieldsLimit</code> : The number of referenced data source fields per page.	Data source field	◦ <code>datasourceFieldsPageSize</code>: The number of data source fields per page. ◦ <code>columnsLimit</code>: The number of data source field columns per page. ◦ <code>fieldsLimit</code> : The number of referenced data source fields per page.	
	Metadata type	Setting and description												
	Table	◦ <code>tablesPageSize</code>: The number of tables per page. ◦ <code>tablesColumnsPageSize</code>: The number of table columns per page.												
	Table columns	◦ <code>tableColumnsPageSize</code>: The number of table columns per page.												
	Users	◦ <code>usersPageSize</code>: The number of users per page.												
	Data source	◦ <code>datasourcesPageSize</code>: The number of data sources per page. ◦ <code>datasourcesFieldsPageSize</code>: The number of data source fields per page. ◦ <code>columnsLimit</code>: The number of data source field columns per page. ◦ <code>fieldsLimit</code> : The number of referenced data source fields per page.												
Data source field	◦ <code>datasourceFieldsPageSize</code>: The number of data source fields per page. ◦ <code>columnsLimit</code>: The number of data source field columns per page. ◦ <code>fieldsLimit</code> : The number of referenced data source fields per page.													

4. Click **Create**.

» The Edge site connects to your data source.

What's next?

[Synchronize](#) the Tableau technical lineage

Synchronize a Tableau technical lineage

You can synchronize your technical lineage manually or automatically by adding a synchronization schedule.

When you synchronize a technical lineage capability, a full synchronization is run, which is equivalent to the `full-sync` command with the `-s` argument specified when you use the lineage harvester. You can synchronize as many technical lineage capabilities in parallel as you want. If a full synchronization is in progress, the other capabilities are put in a queue until a new full synchronization starts automatically.

Prerequisites

- 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

Manually synchronize technical lineage for Tableau

1. On the main menu, click , and then click  **Catalog**.
 - » The Catalog Home opens.
2. On the main toolbar, click .
 - » The **Create** dialog box appears.
3. In the **Register with Edge** section of the **Create** dialog box, click **Register a data source**.
 - » The **Register content** page opens.
4. In the **Connection name** column, locate the Tableau connection that you used when you added the Tableau capability. Then, click the capability name in the **Data**

sources/Capabilities column of the Tableau connection.

- » The technical lineage capability configuration page opens.
- 5. On the **Configuration Section** tab pane, click **Synchronize now**.
 - » A notification indicates synchronization has started.
 - » The synchronization job is started. The tables and columns from the selected schema are ingested based on the configured table rules.

Add a synchronization schedule

1. On the main menu, click , and then click  **Catalog**.
 - » The Catalog Home opens.
2. On the main toolbar, click .
 - » The **Create** dialog box appears.
3. In the **Register with Edge** section of the **Create** dialog box, click **Register a data source**.
 - » The **Register content** page opens.
4. In the **Connection name** column, locate the Tableau connection that you used when you added the Tableau capability. Then, click the capability name in the **Data sources/Capabilities** column of the Tableau connection.
 - » The technical lineage capability configuration page opens.
5. On the **Synchronization schedule** tab pane, click **Add schedule**.
6. Enter the required information and click **Save**:

Field	Description
Repeat	The interval when you want to synchronize automatically. The possible values are: Daily , Weekly , Monthly , and Cron expression .
Cron	The Quartz Cron expression that determines when the synchronization takes place. This field is only visible if you select Cron expression in the Repeat field.
Every	The day on which you want to synchronize, for example, Sunday. This field is only visible if you select Weekly in the Repeat field.

Field	Description
Every first	The day of the month on which you want to synchronize, for example, Tuesday. This field is only visible if you select <code>Monthly</code> in the Repeat field.
At	The time at which you want to synchronize automatically, for example, 14:00. This field is only visible if you select <code>Daily</code> , <code>Weekly</code> , or <code>Monthly</code> in the Repeat field.
Time zone	The time zone for the schedule.

What's next?

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 troubleshooting guide](#) or [Collibra Support Portal](#) to fix the errors.

Technical lineage for Tableau via the lineage harvester

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

- [Via Edge](#).
- [Via the Data Catalog user interface](#).
- Via the lineage harvester, as described in this section of the documentation.

This topic describes the features and limitations of using the [lineage harvester](#) to create Tableau assets and data in Data Catalog.

The Tableau operating model associated with this integration method is only available in Collibra versions 2021.10 and newer.

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 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](#).

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.

Features

Feature	Integration via Data Catalog UI	Integration via the lineage harvester
Catalog ingestion	✓	✓
Technical lineage		✓
Automatic stitching		✓
Embedded data source connectivity		✓
Custom SQL parsing		✓
On-prem credential storage		✓

Feature	Integration via Data Catalog UI	Integration via the lineage harvester
Ingestion via Explorer role (with the Data Management Add-On)		✓

Limitations

Currently, there are still a few limitations to integrating Tableau metadata via the lineage harvester:

- You ingest Tableau via the lineage harvester, instead of via the Data Catalog UI. All changes to the Tableau ingestion must be configured in the lineage harvester configuration file, instead of Data Catalog.
- The **Stitch** button is part of the legacy Tableau integration (via Data Catalog) and does not work when integrating Tableau via the lineage harvester, as stitching is done automatically via this method.
- We partially support 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”, which has the value “false”.
- Data fields are ingested with their actual names. Labels and aliases are not returned by the APIs.

Tableau integration steps

The Tableau integration enables you to harvest Tableau metadata and create new Tableau assets in Data Catalog. Collibra Data Intelligence Cloud analyzes and processes the metadata and presents it as specific [asset types](#), retaining their original names.

Steps

1. [Set up](#) Tableau.

2. [Prepare](#) a domain for Tableau ingestion.
If you already have a domain reserved for ingestion of the new Tableau assets, you can skip this step.
3. [Prepare](#) the Data Catalog physical data layer.
4. [Download](#) and [install](#) the lineage harvester.
5. [Prepare](#) the lineage harvester configuration file.
6. Optionally, [Prepare](#) a <source ID> configuration file.

Tip Hostname mapping (Beta) will replace database mapping and the `collibraSystemName` section for databases in a future Collibra version. For complete information and examples of hostname mapping, go to [Tableau hostname mapping \(Beta\)](#).

7. Start the lineage harvester again in the console and run the following command:
 - for Windows: `.\bin\lineage-harvester.bat full-sync`
 - for other operating systems: `./bin/lineage-harvester full-sync`
 When prompted, enter the password or client secret to connect to your Collibra Data Intelligence Cloud and Tableau environment. The passwords are encrypted and stored in `/config/pwd.conf`.

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.

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 Tableau 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 Tableau](#). As a consequence, any manually added data of those assets is lost.

Lineage harvester system requirements

You need to meet the system requirements to be able to [install](#) and run the [lineage harvester](#).

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 [Technical lineage general troubleshooting](#).

- 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 the 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.

Tableau ingestion workflow

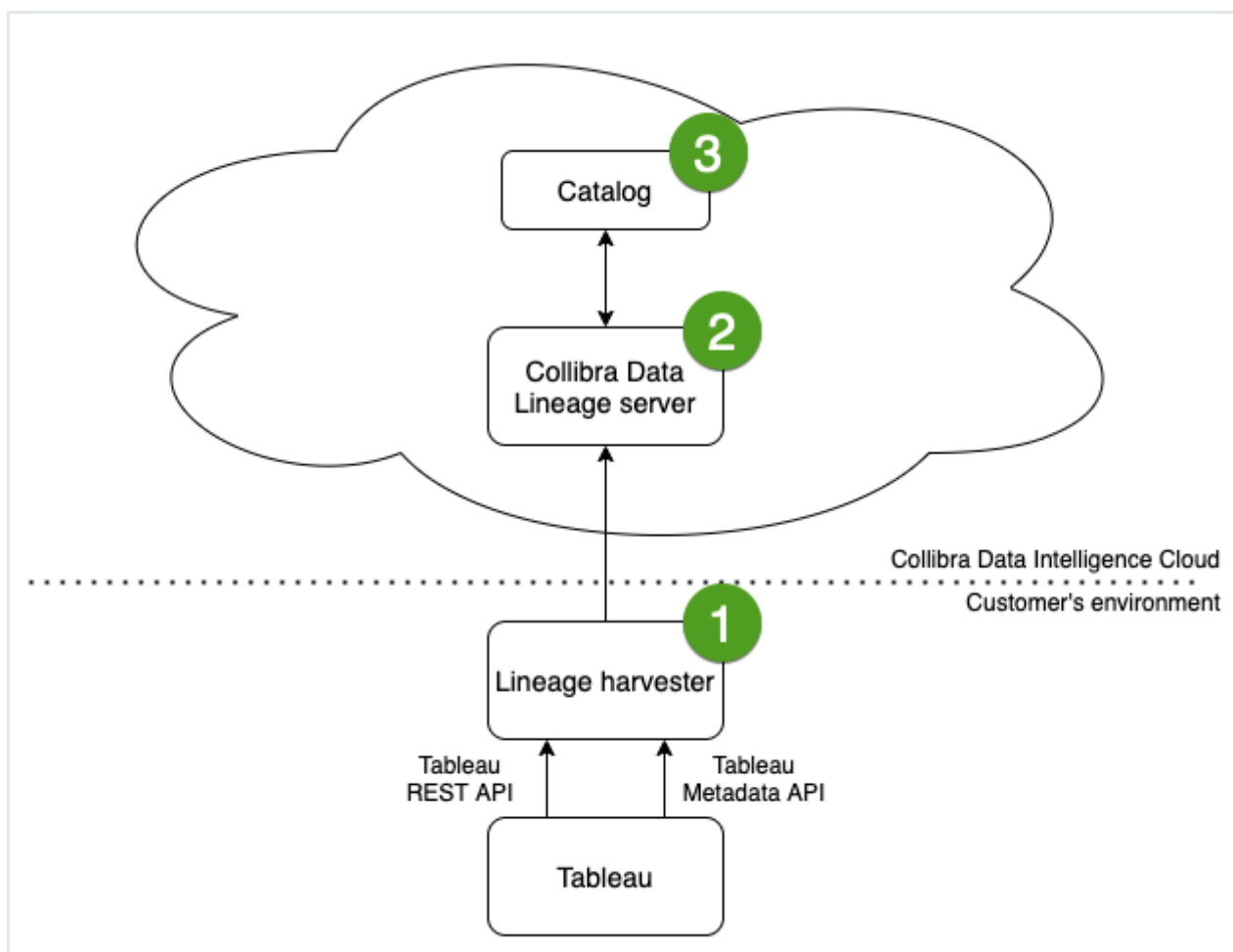
You run the lineage harvester to start the Tableau ingestion workflow. When you initiate Tableau ingestion, each workflow component performs the following actions:

1. The lineage harvester:
 - Communicates with Tableau.
 - Harvests the Tableau metadata that will be ingested to Data Catalog.
 - Sends the Tableau metadata to the Collibra Data Lineage service.
2. The Collibra Data Lineage service:
 - Analyzes the Tableau metadata.
 - Creates new assets and relations.

- Stitches existing assets in Data Catalog to Tableau assets.
- Imports new Tableau assets and their relations in Data Catalog.

3. Data Catalog:

- Shows new Tableau assets
- Shows a Technical lineage for Tableau assets.
- Shows stitching results between Tableau Data Attribute assets and Column assets.



Note This is the recommended workflow. If you do not want to use the Tableau Metadata API, you can disable it via the [configuration file](#).

Install the lineage harvester for Tableau integration

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

Tip

- Install the lineage harvester close to your data source or on the same server.
- The lineage harvester uses port 443.

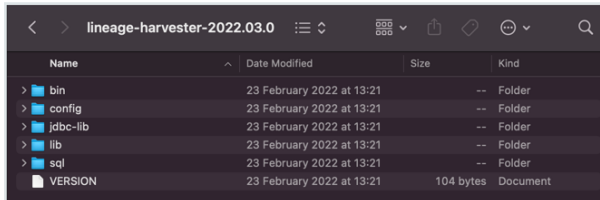
Prerequisites

- You have purchased Collibra Data Lineage.
- 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.

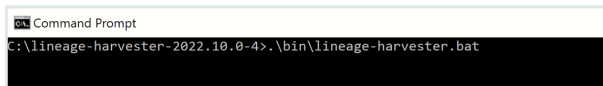
Steps

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.

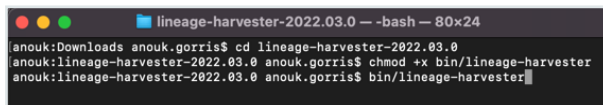


3. Start the lineage harvester to create an empty lineage harvester configuration file by entering the following command:

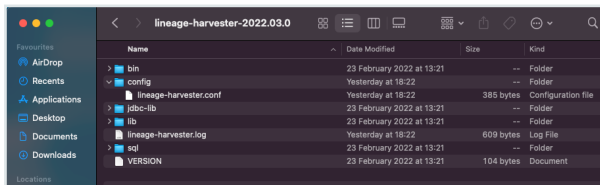
- **Windows:** `.\bin\lineage-harvester.bat`



- **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.

Prepare the lineage harvester configuration file for Tableau

You have to prepare a configuration file before you run the lineage harvester. The lineage harvester collects your Tableau metadata and sends it to the Collibra Data Lineage service, where it is processed and analyzed. Collibra Data Intelligence Cloud then imports the Tableau assets and relations to Data Catalog.

Warning If you are switching between the lineage harvester and Edge, the value of the `id` property in your lineage harvester configuration file must exactly match the value in the **Source ID** field in your [Edge capability](#).

Example configuration file

```
{
  "general": {
    "catalog": {
      "url": "https://<organization>.collibra.com",
      "username": "<your-collibra-username>"
    },
    "useCollibraSystemName": false
  },
  "sources": [
    {
      "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,
      "concurrencyLevel": 1,
      "deleteRawMetadataAfterProcessing": true,
      "paging": {
        "pagination-setting": 100,
        "pagination-setting-2": 100
      }
    }
  ]
}
```


Before you begin

- [Set up the latest lineage harvester.](#)
- [Test your connectivity with the Tableau server.](#)
- [Prepare the Data Catalog physical data layer for technical lineage.](#)
- [Create one or more BI Data Catalog domains in which you want to ingest the Tableau assets.](#)

Requirements and permissions

- Collibra Data Intelligence Cloud.
- 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 BI Data Catalog domain:
 - Asset: add
 - Attribute: add
 - Domain: add
 - Attachment: add

Steps

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

Tip Watch a video on how to do this:

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 to US government customers.

Properties	Description
url	The URL of the Collibra Data Lineage service instance. “url”: “https://techlin-gov.collibra.com” Warning This 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 applies only to US 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>.
sources	This section contains all of the Tableau connection properties.
type	The kind of data source. In this case, the value has to be <i>Tableau</i> .

Properties	Description
id	The unique ID to identify the Tableau metadata that was uploaded to the Collibra Data Lineage. 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. Tip This value can be anything as long as it is a unique. The lineage harvester uses the ID to identify a batch of data on the Collibra Data Lineage service.
url	The link to the data in Tableau.

Properties	Description
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.
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
siteIds	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 Profits, on a site named Sales, 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 Sales 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>siteIds</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: https://<yourcollibrainstance>/domain/22258f64-40b6-4b16-9c08-c95f8ec0da26?view=00000000-0000-0000-0000-0000000040001. In this example, the reference ID is in bold.

Properties	Description
excludelImages	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 filters 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.

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.

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><tr><td>Database</td><td> ◦ <code>databasesPageSize</code>: The number of databases 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.	Database	◦ <code>databasesPageSize</code>: The number of databases 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.								
Database	◦ <code>databasesPageSize</code>: The number of databases per page.								

Properties	Description														
	<table> <tr> <th>Metadata type</th><th>Setting and description</th></tr> <tr> <td>Table</td><td> ◦ <code>tablesPageSize</code>: The number of tables per page. ◦ <code>tablesColumnsPageSize</code>: The number of table columns per page. </td></tr> <tr> <td>Table columns</td><td> ◦ <code>tableColumnsPageSize</code>: The number of table columns per page. </td></tr> <tr> <td>Parameter</td><td> ◦ <code>parametersPageSize</code>: The number of parameters per page. </td></tr> <tr> <td>Users</td><td> ◦ <code>usersPageSize</code>: The number of users per page. </td></tr> <tr> <td>Data source</td><td> ◦ <code>datasourcesPageSize</code>: The number of data sources per page. ◦ <code>datasourcesFieldsPageSize</code>: The number of data source fields per page. ◦ <code>columnsLimit</code>: The number of data source field columns per page. ◦ <code>fieldsLimit</code>: The number of referenced data source fields per page. </td></tr> <tr> <td>Data source field</td><td> ◦ <code>datasourceFieldsPageSize</code>: The number of data source fields per page. ◦ <code>columnsLimit</code>: The number of data source field columns per page. ◦ <code>fieldsLimit</code>: The number of referenced data source fields per page. </td></tr> </table>	Metadata type	Setting and description	Table	◦ <code>tablesPageSize</code>: The number of tables per page. ◦ <code>tablesColumnsPageSize</code>: The number of table columns per page.	Table columns	◦ <code>tableColumnsPageSize</code>: The number of table columns per page.	Parameter	◦ <code>parametersPageSize</code>: The number of parameters per page.	Users	◦ <code>usersPageSize</code>: The number of users per page.	Data source	◦ <code>datasourcesPageSize</code>: The number of data sources per page. ◦ <code>datasourcesFieldsPageSize</code>: The number of data source fields per page. ◦ <code>columnsLimit</code>: The number of data source field columns per page. ◦ <code>fieldsLimit</code>: The number of referenced data source fields per page.	Data source field	◦ <code>datasourceFieldsPageSize</code>: The number of data source fields per page. ◦ <code>columnsLimit</code>: The number of data source field columns per page. ◦ <code>fieldsLimit</code>: The number of referenced data source fields per page.
Metadata type	Setting and description														
Table	◦ <code>tablesPageSize</code>: The number of tables per page. ◦ <code>tablesColumnsPageSize</code>: The number of table columns per page.														
Table columns	◦ <code>tableColumnsPageSize</code>: The number of table columns per page.														
Parameter	◦ <code>parametersPageSize</code>: The number of parameters per page.														
Users	◦ <code>usersPageSize</code>: The number of users per page.														
Data source	◦ <code>datasourcesPageSize</code>: The number of data sources per page. ◦ <code>datasourcesFieldsPageSize</code>: The number of data source fields per page. ◦ <code>columnsLimit</code>: The number of data source field columns per page. ◦ <code>fieldsLimit</code>: The number of referenced data source fields per page.														
Data source field	◦ <code>datasourceFieldsPageSize</code>: The number of data source fields per page. ◦ <code>columnsLimit</code>: The number of data source field columns per page. ◦ <code>fieldsLimit</code>: The number of referenced data source fields per page.														

2. Save the configuration file.

What's next?

Prepare a <source ID> configuration file.

Prepare a Tableau <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 databases in Tableau.
- Map a Tableau technical database name to the real database name, to preserve stitching. See the [databaseMapping](#) property.

Tip Try out the new [hostnameMapping feature \(beta\)](#), to map database, schema, or system names that were returned by the Tableau APIs to the actual names of the assets in Data Catalog. When the beta period ends and the `hostnameMapping` is generally available, the [databaseMapping](#) section and the [databases](#) sub-section of the `collibraSystemNames` section will be deprecated.

- 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

```

{
  "collibraSystemNames": {
    "databases": [
      {
        "hostName": "database-hostname",
        "collibraSystemName": "public"
      }
    ],
    "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"
      }
    ]
  },
  "databaseMapping": {
    "<hostname:port>": "<actual database name>"
  },
  "filters": {
    "sites": {
      "site_name": "domain_id"
    },
    "projects": {
      "site_name2 > project_name2": "domain-reference-
id2",
      "site_name3 > project_name3 > subproject_name":
"domain-reference-id2"
    }
  }
}

```

```

{
  "collibraSystemNames": {
    "databases": [

```

```

    {
      "hostName": "database-hostname",
      "collibraSystemName": "public"
    }
  ],
  "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"
    }
  ]
},
"databaseMapping": {
  "<hostname:port>": "<actual database name>"
},
"domainMapping": {
  "<Site-1>": "reference-id-of-Domain-1",
  "<Site-1> > <Project-Default>": "reference-id-of-Domain-2"
}
}

```

Example <source ID> configuration file

```

{
  "collibraSystemNames": {
    "databases": [
      {
        "hostName": "database-hostname",
        "collibraSystemName": "public"
      }
    ],
    "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"
  }
],
"databaseMapping": {
  "<hostname:port>": "<actual database name>"
},
"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.

Property	Description	
hostnameMapping (Beta)	This section allows you to map Tableau technical database, server and schema names to the respective real names, to preserve stitching. Important - This section replaces the following sections, and should not be used in combination with either of the following sections: - The <code>databaseMapping</code> section, if included. - The <code>databases</code> sub-section of the <code>collibraSystemNames</code> section, if included. When the beta period ends and the <code>hostnameMapping</code> section is generally available, the <code>databaseMapping</code> section and the <code>databases</code> sub-section of the <code>collibraSystemNames</code> section will be deprecated. - 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.	No
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 (found_dbname), on which server a database is running (found_hostname), and optionally, the name of the schema (found_schema).	No
dbname	The name of the database of a supported data source in Tableau.	No

Property	Description	
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.	No
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.	No

Property	Description	
collibraSystemName	The system or server name of the database. Warning The value of this property must exactly match the name of your System asset in Collibra.	No
databases	This section contains connection information to one or more databases in Tableau. Tip ◦ If you do not have databases in Tableau, you can remove this section. ◦ The values that you specify for this property are not case-sensitive.	
hostName	The host name of the database.	

Property	Description
collibraSystemName	The system 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 <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> "collibraSystemNames": { "databases": [{ "hostName": "database- hostname-1", "collibraSystemName": "Cus- tomers-Europe" }, { "hostName": "database-host- name-2", "collibraSystemName": "Cus- tomers-USA" }], </pre> Warning The value of this property must exactly match (including for case-sensitivity) the name of your System asset in Collibra.

Property	Description
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.
filePath	The full path to the file. For example, the path to a JSON file.
collibraSystemName	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.
collibraSystemName	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.
collibraSystemName	The system name of the cloud file.

Property	Description
databaseMapping	The Tableau API returns a technical database name based on the hostname, instead of the actual database name, which breaks stitching. The values that you specify for this property are not case-sensitive. This property allows you to map a Tableau technical database name to the real database name, for example: <pre>"databaseMapping": { "<hostname:port>": "<actual database name>" }</pre> Including the port, as shown in the example, is optional.

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 <code><source ID></code> 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 > Data bricks":"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.
<code>site_name > 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. 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.

Property	Description
	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 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: <pre>"Collibra_tab_partner_site > JB_Test_2812": "d224a1a5-43b4-43b2-8df0-ddf8f2726b82"</pre>
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.

What's next?

See [Overview Tableau integration steps](#).

Schedule Tableau ingestion jobs

You can use [Task Scheduler](#) on Windows or [Crontab](#) on Mac and Linux to make the lineage harvester run scheduled jobs. In a scheduled job, the lineage harvester uploads the Tableau metadata information to Collibra.

Collibra automatically creates new Tableau assets and stitches the Tableau assets to existing data sources in Data Catalog at specific times, dates or intervals, using the information in your configuration file.

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 Tableau](#). As a consequence, any manually added data of those assets is lost.

Warning Relations that were manually created between Tableau assets and other assets via a relation type in the [Tableau operating model](#), are deleted after a refresh of the Tableau metadata.

Migrating Tableau assets to the new Tableau operating model

A key feature of the Collibra Data Intelligence Cloud 2022.02 release was the ability to ingest Tableau metadata in Collibra Data Catalog and synchronize the metadata using the [lineage harvester](#). However, this new integration method was only available to customers who did not need to migrate existing Tableau assets to the new operating model. A migration script now eliminates that limitation.

In this section, we provide an overview of:

- How to integrate Tableau metadata via the lineage harvester.
- How to use the lineage harvester to migrate your existing Tableau assets to the new operating model.

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.
Migration script	A specific set of lineage harvester commands used to migrate your custom asset types, attribute types and relation types that were created as part of Tableau integration v1. Note You need lineage harvester version 2022.03.0-5 or newer. We recommend that you use the newest lineage harvester.

Methodology

The following is our methodology for migrating Tableau integration v1 metadata to the new operating model. For greater detail see [Overview: Tableau integration v2 and migration](#).

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 the harvester integration, 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.
2	Integrate the same Tableau metadata, but this time via Tableau integration v2.	After successful integration, you will have: • A single BI Catalog domain in Collibra with custom Tableau integration v1 assets and their custom attributes and relations. • A single BI Catalog domain in Collibra with Tableau integration v2 assets. Important The new Tableau operating model is only available in Collibra versions 2021.10 and newer.
3	Run the migration script.	The full name of each Tableau integration v1 asset is compared to the full name of the same assets from the Tableau integration v2. When the names match, all of the custom characteristics of the v1 assets are saved to the respective v2 assets. Assets of custom v1 asset types are recreated in the specified domain. Specifically: • The following elements are migrated: ◦ Your custom v1 asset types, attribute types and relation types. ◦ All assets of your custom v1 asset types. ◦ The custom attributes and relations of your custom v1 assets. ◦ Any tags that you added to your v1 assets. • The following elements are ignored during the migration: ◦ All assets of out-of-the-box v1 asset types: ▪ Their custom attributes and relations, however, are migrated and saved to their respective v2 assets. ▪ With the exception of Tableau Data Entity, Tableau Report Attribute and Tableau View assets, which are also ignored, but so too are the attributes and relations of such assets. ◦ Any attribute types and relation types that are included in the operating model.

No.	Step	Details
4	Verify the migration results.	Compare your Tableau integration v2 assets to the respective Tableau integration v1 assets. Look to see that the metadata that you manually added to your integration v1 assets has been added to your integration v2 assets.
5	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.

Overview: Tableau integration v2 and migration

The Tableau integration v2 enables you to harvest Tableau metadata and create new Tableau assets in Data Catalog. Collibra Data Intelligence Cloud analyzes and processes the metadata and presents it as specific [asset types](#), retaining their original names.

Steps

The following table shows the steps and prerequisites required to ingest metadata in Collibra via lineage harvester (Tableau integration v2) and run the migration script.

Note

- This overview assumes that you have already ingested Tableau assets via Tableau integration v1.
- In the commands that you enter to run the migration, you need to specify which custom asset types, attribute types and relation types you want to migrate.

Step	What?	Description	Prerequisites
1	Set up Tableau.	Before you start the Tableau integration in Data Catalog, make sure that the lineage harvester can reach the Tableau metadata. Perform these tasks before you start the actual Tableau ingestion 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.	You have a Tableau subscription.
2	Create a new domain.	Before you can ingest Tableau metadata, you have to create a new domain or choose an existing domain to store the new Tableau assets. Warning If you are using Collibra Data Intelligence Cloud 2021.11 or older, you have to add all Tableau attributes in the operating model to a scope and create a scoped assignment before you ingest Tableau via the lineage harvester. For complete information and step-by-step instruction, see Tableau general troubleshooting.	You have a resource role with the following resource permissions: Domain: Add

Step	What?	Description	Prerequisites
3	Prepare the physical data layer.	You prepare Data Catalog's physical data layer to enable Data Catalog to automatically stitch the Tableau assets to existing assets in Data Catalog.	- You have a global role with the Catalog global permission, for example, Catalog Author. - You have set up the JDBC driver of your source data, for example Snowflake. - You have a resource role with the following resource permissions on the Schema community: - Asset > add - Attribute > add - Domain > add - Attachment > add - You have the permissions to retrieve the metadata of the following database components through the JDBC Driver Database Metadata methods: - Schemas - Tables - Columns
4	Download and install the lineage harvester	You use the lineage harvester to trigger the creation of Tableau assets, their relations and a technical lineage in Data Catalog. You can download the lineage harvester from the Colibra Product Resource Downloads page. For a list of lineage harvester installation requirements, see About the lineage harvester installation.	Your environment meets the system requirements to install and use the lineage harvester.

Step	What?	Description	Prerequisites
5	Prepare the lineage harvester configuration file and run the lineage harvester.	You create a lineage harvester configuration file with Tableau connection information and run the lineage harvester to import the results of the Tableau integration and the technical lineage for Tableau into Data Catalog. As a result, you now have a duplicate of your Tableau metadata in Collibra.	• You have downloaded the lineage harvester version 2022.03 or newer. • Your environment meets the system requirements to install and run the lineage harvester. • 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. • You have a global role with the Data Stewardship Manager global permission. • 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

Step	What?	Description	Prerequisites
6	Run the migration script	The migration script is triggered by a lineage harvester command. You then use arguments to migrate your customized asset types and custom attribute types and relation types. Note You need lineage harvester version 2022.03.0-5 or newer. We recommend that you use the newest lineage harvester.	Same prerequisites as for the previous step.
7	Verify the migration results	Compare your Tableau integration v2 assets to the respective Tableau integration v1 assets. Look to see that the metadata that you manually added to your integration v1 assets has been added to your integration v2 assets.	None
8	Delete your Tableau integration v1 metadata.	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.	- You have a global role with the Catalog global permission, for example, Catalog Author. - You have a resource role with the following resource permission on the community level in which you created the BI Data Catalog domain: - Asset: Remove - Domain: Remove

Naming convention

When you synchronize Tableau, Collibra 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.

Run the migration script

The migration script is triggered by a lineage harvester command. You then use arguments to migrate your customized asset types and custom attribute types and relation types.

Prerequisites

- You have Collibra Data Intelligence Cloud 2022.01 or newer.
- You have downloaded lineage harvester version 2022.03 or newer and you have the necessary system requirements to run it.
- You have a [global role](#) that has the Manage all resources [global permission](#).
- 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](#).
- You have a [global role](#) with the Data Stewardship Manager [global permission](#).
- You have 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
- You have [tested](#) your connectivity with the Tableau server.

Steps

1. Run the following command to start the lineage harvester and trigger the migration:
 - **Windows:** `.\bin\lineage-harvester migrate-tableau <v1_tableau_server_asset_id> <v2_source_id>`
 - **for other operating systems:** `./bin/lineage-harvester migrate-tableau <v1_tableau_server_asset_id> <v2_source_id>`

2. Use the following arguments to migrate:

- Customized asset types: `-a <customAssetTypeId>`
- Custom attribute types: `-t <customAttributeTypeId>`
- Custom relation types: `-r <customRelationTypeId>`

Tip You can migrate multiple asset types, attribute types and relation types by repeating the relevant command. In the following example, two asset types are migrated, one after the other, by repeating the `-a` command, followed by the relevant ID of each asset type.

Example

```
./bin/lineage-harvester migrate-tableau 7cc9f692-bbe4-
467f-8ffb-f43545465fcf testtableau22 \
-a asd13io2-sda2-sdi2-jsd9-asdoi124io12 \
-a ard86co4-sea5-sc4r-hk39-kjsv9she3hs9 \
-t 3ffafa8e-029c-4d01-a3c9-1c36e43c2655 \
-r d0086c90-98e6-4782-b07a-40fcb43845a3
```

What's next?

- The following elements are migrated:
 - Your custom v1 asset types, attribute types and relation types.
 - All assets of your custom v1 asset types.
 - The custom attributes and relations of your custom v1 assets.
 - Any tags that you added to your v1 assets.
- The following elements are ignored during the migration:
 - All assets of out-of-the-box v1 asset types:
 - Their custom attributes and relations, however, are migrated and saved to their respective v2 assets.
 - With the exception of Tableau Data Entity, Tableau Report Attribute and Tableau View assets, which are also ignored, but so too are the attributes and relations of such assets.
 - Any attribute types and relation types that are included in the operating model.

Tip You can check the progress of the migration in [Activities](#).

To refresh the Tableau integration v2 metadata, you can run the lineage harvester again using the `full-sync` command, or [schedule jobs](#) to run them automatically.

Soft deletion of your Tableau integration v1 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. You can either manually delete the assets or use a lineage harvester argument to perform a soft delete of the assets. Technically speaking, the soft delete does not delete the assets from your Collibra environment; rather, it changes the status of the assets to Obsolete. You can then create an [asset filter](#) to view all assets with the status Obsolete, and then manually delete them.

Prerequisites

- You have Collibra Data Intelligence Cloud 2022.01 or newer.
- You have downloaded lineage harvester version 2022.03 or newer and you have the necessary system requirements to run it.
- You have a [global role](#) that has the Manage all resources [global permission](#).
- 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](#).
- You have a [global role](#) with the Data Stewardship Manager [global permission](#).
- You have a [resource role](#) with the following [resource permission](#) on the community level in which you created the BI Data Catalog domain:
 - Asset: Update Status

Steps

1. Run the following command to start the lineage harvester and trigger the migration:
 - **Windows:** `.\bin\lineage-harvester migrate-tableau --delete <v1_tableau_server_asset_id> <v2_source_id>`
 - **for other operating systems:** `./bin/lineage-harvester migrate-tableau --delete <v1_tableau_server_asset_id> <v2_source_id>`

Example

```
./bin/lineage-harvester migrate-tableau --delete 7cc9f692-  
bbe4-467f-8ffb-f43545465fcf testtableau22
```

Tip You can check the progress of the migration in [Activities](#).

Working with Power BI service

Power BI service is a cloud business intelligence software that helps you see and understand your data. You can ingest Power BI metadata in Data Catalog and create a technical lineage.

The Power BI service integration in Collibra Data Intelligence Cloud is not the same as the Power BI Report Server integration. If you want to ingest Power BI Report Server metadata in Collibra Data Intelligence Cloud, please read the Power BI Report Server section in the [Documentation Center](#).

If you want to ingest Power BI metadata in Data Catalog, you have to purchase the Power BI connector and lineage feature.

Note There are currently two supported methods for integrating Power BI metadata in Data Catalog:

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

Power BI integration [via the Power BI harvester](#) is deprecated and no longer supported. If you previously used this integration method, you can use the lineage harvester to [migrate](#) your existing Power BI assets to the lineage harvester integration method.

Features

Collibra Data Lineage currently supports two means by which to integrate Power BI in Data Catalog. The following table shows the features specific to the two integration methods.

Feature	Integration via the Power BI harvester (deprecated)	Integration via the lineage harvester
Catalog ingestion	✓	✓
Technical lineage	✓	✓

Feature	Integration via the Power BI harvester (deprecated)	Integration via the lineage harvester
Automatic stitching	✓	✓
Uses only one harvester		✓
No Windows dependency		✓
Uses new Power BI APIs		✓
Workspace filtering for high-volume data		✓
Mapping to target domains		✓
Data flow support		✓

Tip After we have successfully switched to the new Power BI APIs, we will have more opportunities to improve the integration. You can add your ideas for product enhancements and new features in the [Collibra Integrations Ideation Portal](#).

Power BI terminology

Before you ingest [Power BI](#), read more about the Power BI terminology and how it maps with the Collibra Data Intelligence Cloud asset types.

Note For more information, see the [Power BI documentation](#).

Power BI term	Description	Asset type in Collibra
Capacity	A resource that hosts Power BI Workspaces.	Power BI Capacity

Power BI term	Description	Asset type in Collibra
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
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

Power BI asset and domain types

The [Power BI](#) integration in Collibra Data Intelligence Cloud uses a specific subset of packaged [asset types](#) and [domain types](#).

The following table contains the asset and domain types that are used for the Power BI integration. You can see the parent asset types in the breadcrumbs above each asset type.

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

Asset type	Description	Domain type
Data Asset › Data Structure › Data Entity › BI Data Entity › Power BI Table	A table in a Power BI Data Model.	BI Catalog
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 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

Power BI operating model

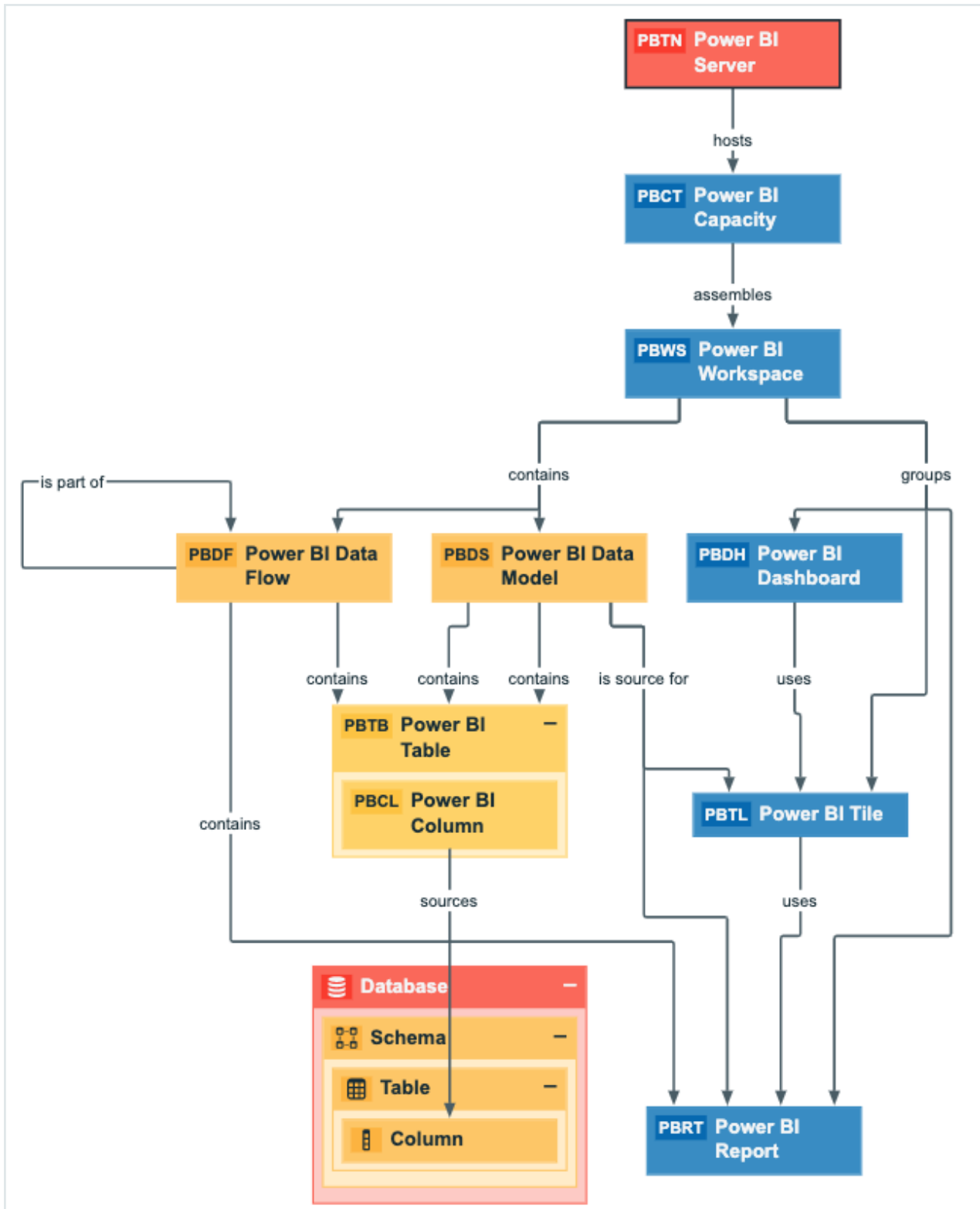
The lineage harvester collects Power BI metadata and sends it to the [Collibra Data Lineage service instances](#). Collibra processes the metadata and creates new Power BI assets and relations in Data Catalog. You can see them on the asset page overview or visualize them in a [diagram](#) or in a [technical 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.

Power BI metadata overview

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.	00000000-0000-0000-0000-0000000000258
	Data asset is source / Source for BI Report	00000000-0000-0000-0000-1200000000013
	Report uses / used in Report	00000000-0000-0000-0000-1200000000007
	Report related to / impacted by Business Asset	00000000-0000-0000-0000-1200000000006
Power BI Data Flow Resource ID: 00000000-0000-0000-0000-1000000000010	Full name	
	Display name	
	Description	00000000-0000-0000-0000-0000000003114
	BI Folder contains / contained in Data Asset	00000000-0000-0000-0000-1200000000014
	Data Entity is part of / contains Data Model	00000000-0000-0000-0000-0000000007046

Asset type	Synchronized metadata	Resource ID
Power BI Data Model Resource ID: 00000000-0000-0000-0000-100000000007	Full name	
	Display name	
	Owner in source The only harvested metadata are email addresses.	00000000-0000-0000-0000-200000000001
	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 / source 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
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 email addresses.	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 / source 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
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
Power BI Table Resource ID: 00000000-0000-0000-0000-100000000009	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
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 / source 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
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 assembles / 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 were 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. In the tab pane, click  **Diagram**.
 - » The diagram appears 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": "000000000-0000-0000-0000-100000000001"
      },
      "fields": []
    },
    {
      "id": "Power BI Capacity",
      "type": {
        "id": "000000000-0000-0000-0000-100000000002"
      }
    },
    {
      "id": "Power BI Workspace",
      "type": {
        "id": "000000000-0000-0000-0000-100000000003"
      }
    },
    {
      "id": "Power BI Dashboard",
      "type": {
        "id": "000000000-0000-0000-0000-100000000004"
      }
    },
    {
      "id": "Power BI Report",
      "type": {
        "id": "000000000-0000-0000-0000-100000000006"
      }
    },
    {
      "id": "Power BI Tile",
      "type": {
        "id": "000000000-0000-0000-0000-100000000005"
      }
    },
    {
      "id": "Power BI Data Model",
```

```

    "type": {
      "id": "000000000-0000-0000-0000-100000000007"
    }
  },
  {
    "id": "Power BI Data Flow",
    "type": {
      "id": "000000000-0000-0000-0000-100000000010"
    }
  },
  {
    "id": "Power BI Table",
    "type": {
      "id": "000000000-0000-0000-0000-100000000009"
    }
  },
  {
    "id": "Power BI Column",
    "type": {
      "id": "000000000-0000-0000-0000-100000000008"
    }
  },
  {
    "id": "Column",
    "type": {
      "id": "000000000-0000-0000-0000-000000031008"
    }
  },
  {
    "id": "Table",
    "type": {
      "id": "000000000-0000-0000-0000-000000031007"
    }
  },
  {
    "id": "Schema",
    "type": {
      "id": "000000000-0000-0000-0001-000400000002"
    }
  },
  {
    "id": "Database",
    "type": {
      "id": "000000000-0000-0000-0000-000000031006"
    }
  }
],
"edges": [
  {
    "from": "Power BI Server",

```

```

    "to": "Power BI Capacity",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-120000000000"
    },
    "roleDirection": true
  },
  {
    "from": "Power BI Capacity",
    "to": "Power BI Workspace",
    "label": "",
    "style": "boxing",
    "type": {
      "id": "000000000-0000-0000-0000-120000000001"
    },
    "roleDirection": true
  },
  {
    "from": "Power BI Workspace",
    "to": "Power BI Dashboard",
    "label": "",
    "style": "boxing",
    "type": {
      "id": "000000000-0000-0000-0000-120000000002"
    },
    "roleDirection": true
  },
  {
    "from": "Power BI Workspace",
    "to": "Power BI Report",
    "label": "",
    "style": "boxing",
    "type": {
      "id": "000000000-0000-0000-0000-120000000002"
    },
    "roleDirection": true
  },
  {
    "from": "Power BI Workspace",
    "to": "Power BI Tile",
    "label": "",
    "style": "boxing",
    "type": {
      "id": "000000000-0000-0000-0000-120000000002"
    },
    "roleDirection": true
  },
  {
    "from": "Power BI Workspace",

```

```

    "to": "Power BI Data Model",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-1200000000014"
    },
    "roleDirection": true
  },
  {
    "from": "Power BI Workspace",
    "to": "Power BI Data Flow",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-1200000000014"
    },
    "roleDirection": true
  },
  {
    "from": "Power BI Dashboard",
    "to": "Power BI Tile",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-1200000000007"
    },
    "roleDirection": true
  },
  {
    "from": "Power BI Data Model",
    "to": "Power BI Tile",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-1200000000013"
    },
    "roleDirection": true
  },
  {
    "from": "Power BI Data Model",
    "to": "Power BI Report",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-1200000000013"
    },
    "roleDirection": true
  },
  {
    "from": "Power BI Data Model",

```

```

    "to": "Power BI Table",
    "label": "",
    "style": "boxing",
    "type": {
      "id": "000000000-0000-0000-0000-0000000007196"
    },
    "roleDirection": true
  },
  {
    "from": "Power BI Data Flow",
    "to": "Power BI Data Flow",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-0000000007046"
    },
    "roleDirection": true
  },
  {
    "from": "Power BI Data Flow",
    "to": "Power BI Table",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-1200000000014"
    },
    "roleDirection": true
  },
  {
    "from": "Power BI Tile",
    "to": "Power BI Report",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-1200000000007"
    },
    "roleDirection": true
  },
  {
    "from": "Power BI Table",
    "to": "Power BI Column",
    "label": "",
    "style": "boxing",
    "type": {
      "id": "000000000-0000-0000-0000-0000000007047"
    },
    "roleDirection": true
  },
  {
    "from": "Power BI Data Flow",

```



```

    "to": "Power BI Report",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-0000000007196"
    },
    "roleDirection": true
  },
  {
    "from": "Power BI Data Model",
    "to": "Power BI Table",
    "label": "",
    "style": "boxing",
    "type": {
      "id": "000000000-0000-0000-0000-0000000007046"
    },
    "roleDirection": false
  },
  {
    "from": "Power BI Column",
    "to": "Column",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-0000000007069"
    },
    "roleDirection": false
  },
  {
    "from": "Column",
    "to": "Table",
    "label": "",
    "style": "boxed",
    "type": {
      "id": "000000000-0000-0000-0000-0000000007042"
    },
    "roleDirection": true
  },
  {
    "from": "Table",
    "to": "Schema",
    "label": "",
    "style": "boxed",
    "type": {
      "id": "000000000-0000-0000-0000-0000000007043"
    },
    "roleDirection": false
  },
  {
    "from": "Schema",

```

```

        "to": "Database",
        "label": "",
        "style": "boxed",
        "type": {
            "id": "000000000-0000-0000-0000-0000000007024"
        },
        "roleDirection": false
    }
],
"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
    }
}
}

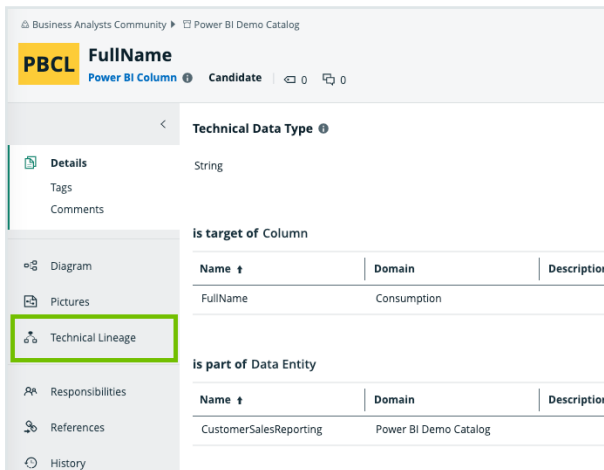
```

Technical lineage for Power BI

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.



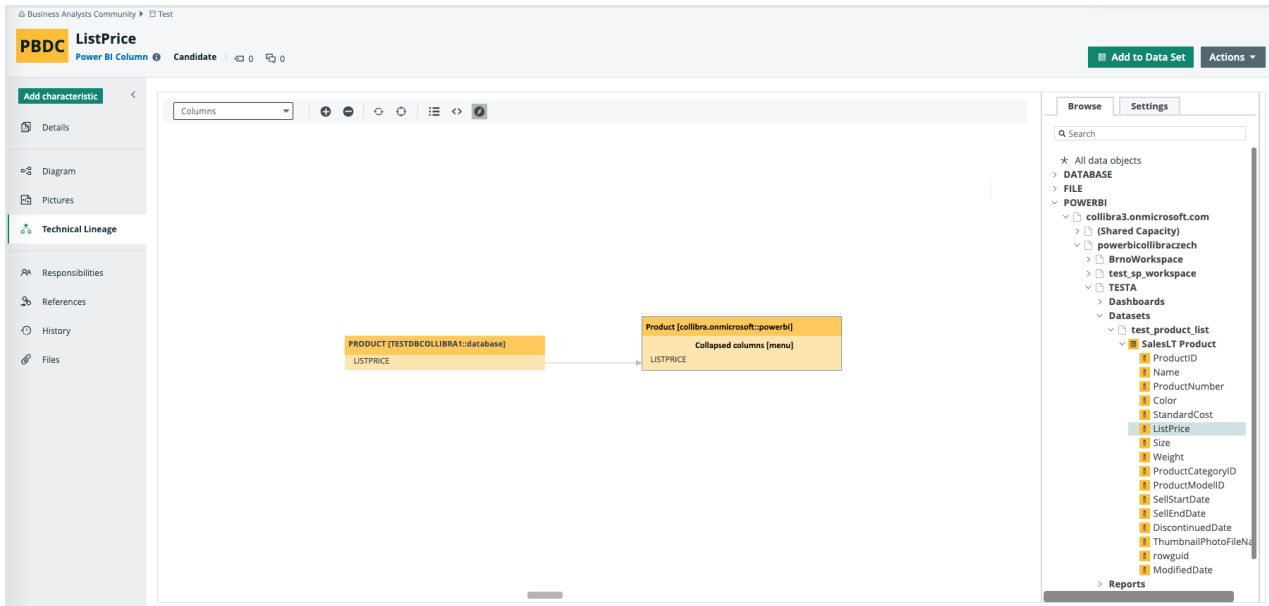
Note If you ingest Power BI for the first time or if you change your geolocation or cloud provider, you have to restart the DGC service before you can see your technical lineage.

Technical lineage graph

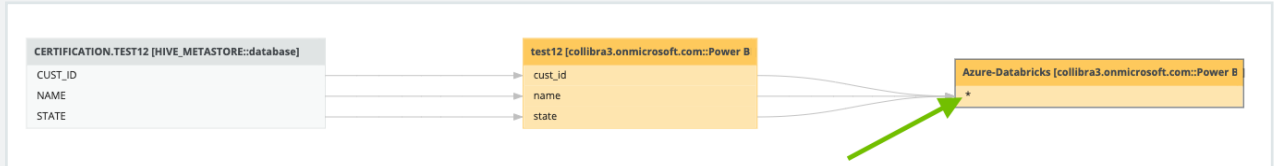
The technical lineage graph shows relations of the type "Data Element targets / sources Data Element" between BI assets and other data objects in the data flow, for example Column assets or Power BI Column assets. These relations are created during the Power BI ingestion process as a result of [automatic 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, 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.



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 error returned an error: (404) Not Found

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](#)

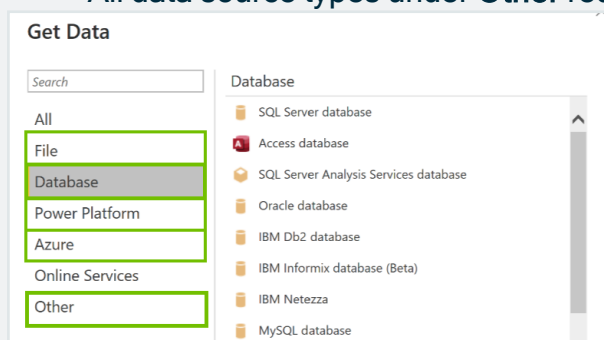
this metadata to data sources registered in Data Catalog. It also creates a [technical lineage](#) that shows where metadata is used and how it transforms.

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](#).

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.
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

Power BI data source	Connection type	Certified ?
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
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.

Power BI data source	Connection type	Certified ?
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
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.

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 [lineage harvester 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",
    "colibraSystemName": "oracle-system-name"
  }
}
```

Property	Description
found_dbname=<substitute database name>;found_hostname=<server name>	found_dbname 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. found_hostname 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.
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.

Property	Description
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
- 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
- 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:

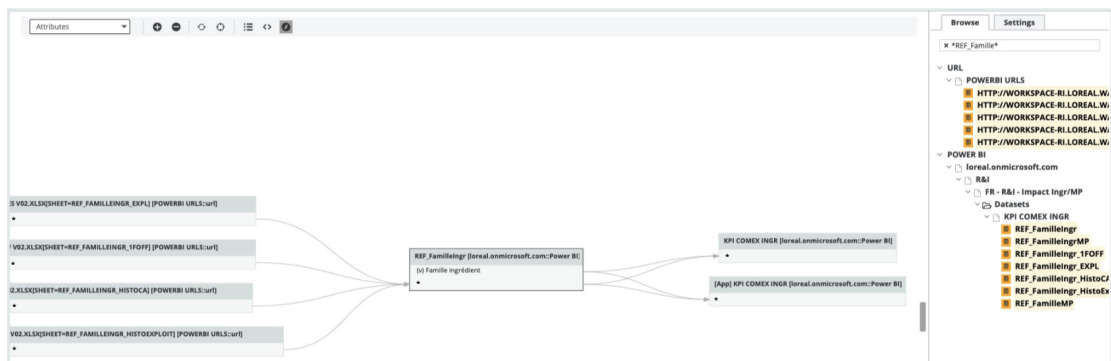
- workspace ID
- dataflow ID
- entity (Power BI Table) ID

- Replacer.ReplaceText
- Replacer.ReplaceValue
- Table.AddColumn
- Table.AddIndexColumn
- Table.Combine

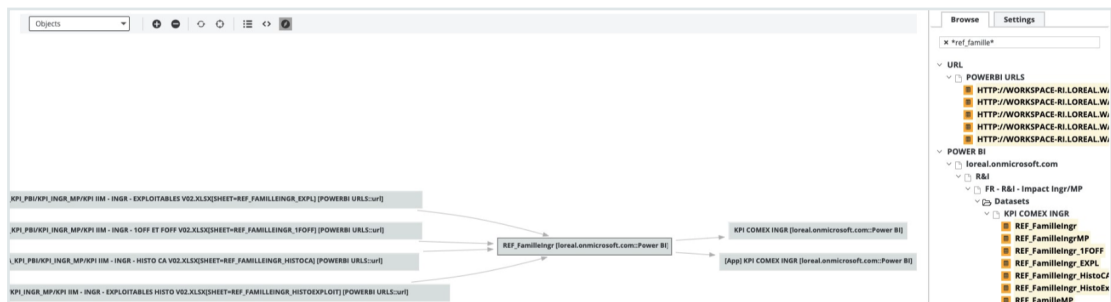
Additional information

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 (public beta)

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.

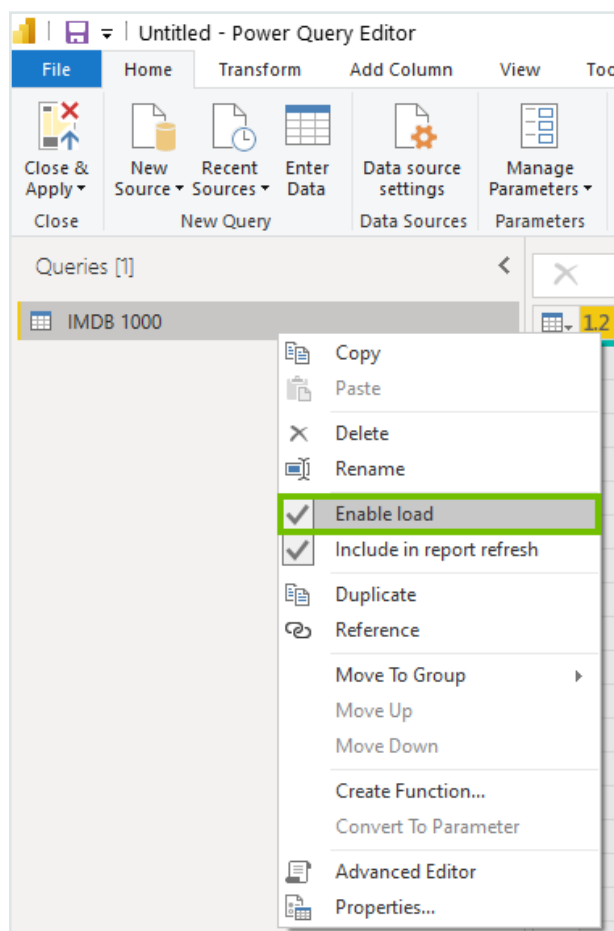
Parameters of unsupported Power Query M functions are not supported. For the lists of supported and unsupported Power Query M functions, see [Supported Power Query M functions](#). Likewise, global parameters are not supported. Global parameters are parameters that are not specific to a Power Query M 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.

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, 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.

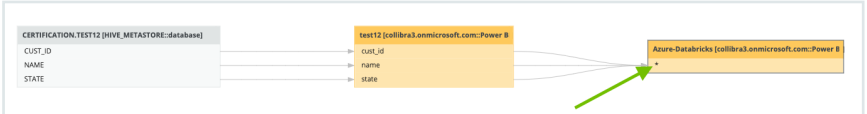
Power BI datamarts are currently not supported.

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.
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.  The diagram illustrates technical lineage. On the left, a box labeled 'CERTIFICATION.TEST12 [HIVE_METASTORE:database]' contains attributes 'CUST_ID', 'NAME', and 'STATE'. Arrows point from these attributes to a central box labeled 'test12 [collibra3.onmicrosoft.com:Power BI]' which contains 'cust_id', 'name', and 'state'. From this central box, three arrows point to a final box on the right labeled 'Azure-Databricks [collibra3.onmicrosoft.com:Power BI]'. The first arrow points to an asterisk (*) in the final box, which is highlighted with a green arrow. The other two arrows point to 'name' and 'state' in the final box.
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.

Known issue	Description
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. Colibra 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.
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 Colibra Integrations Ideation Portal.

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 and tables that are derived from DAX are shown in the technical lineage. However, the Colibra 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".

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 menu, 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.
 - » The **Asset type** editor opens.
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*, to make 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 1, 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**.

Power BI prerequisites

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](#).
- 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, 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 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>" -F
"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>" -F
"scope=https://analysis.windows.net/powerbi/api/.default" -
F grant_type=password
```

Note 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 [Technical lineage 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 [Technical lineage 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 Colibra, Inc
<code>Tenant.ReadWrite.All</code>	Delegated	Read and write all content in tenant	Yes	✔ Granted for Colibra, Inc

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

- Enable the **Allow service principals to use read-only admin APIs** option.
- 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.

- Enable the **Enhance admin APIs responses with detailed metadata** option.
- Enable the **Enhance admin APIs responses with DAX and mashup expressions** option.
- Apply the option to specific security groups.
- 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.

- Enable the **Enhance admin APIs responses with detailed metadata** option.
- Enable the **Enhance admin APIs responses with DAX and mashup expressions** option.

Power BI workspaces

Power BI workspaces represent the most used metadata in Power BI. It contains 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. Consider the following:

- 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.

Tip Use the [Power BI <source ID>_filter configuration file](#) to filter on Power BI workspaces.

Note 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.

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 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: ◦ <code>capacityNames</code> ◦ <code>capacityIds</code> ◦ <code>workspaceNames</code> ◦ <code>excludeWorkspaceNames</code> ◦ <code>excludeWorkspaceIds</code> • You can use wildcards with the following properties: ◦ <code>capacityNames</code> ◦ <code>workspaceNames</code> ◦ <code>excludeWorkspaceNames</code> 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", </pre>

Filtering method	Description
	<pre> "decription":"Domain for BICatalogJBTest1", "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 practice: 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.

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.

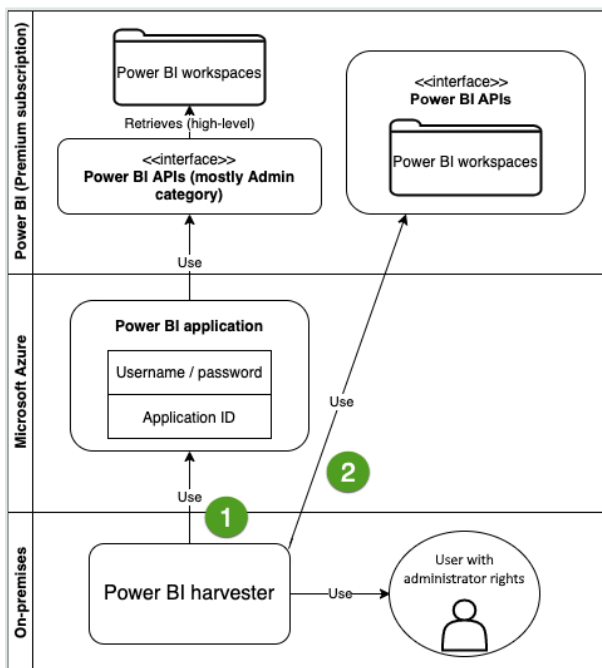
The metadata harvesting process

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

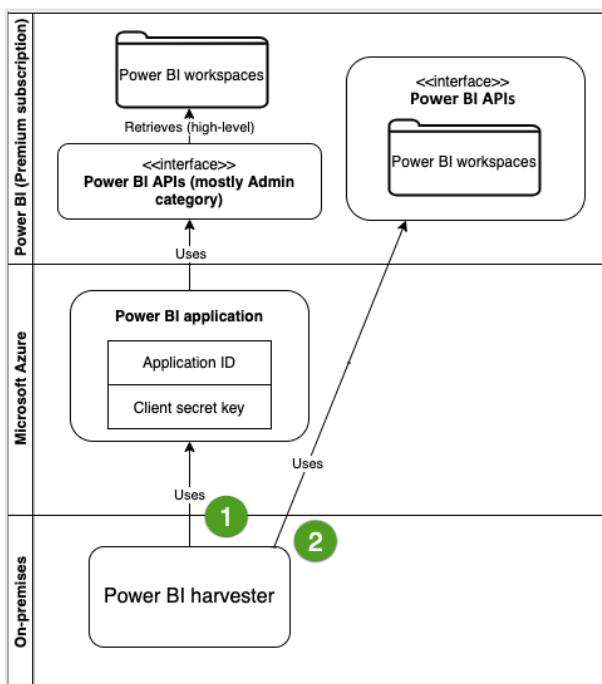


Step	Description
1	The lineage harvester uses the username, password and application ID to access the Power BI APIs. These APIs retrieve basic Power BI metadata, for example metadata in the Power BI tenant or server and reports.
2	The lineage harvester uses Power BI API calls to retrieve more specific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

Prepare a domain for Power BI ingestion

Before you ingest Power BI metadata, you have to designate a domain for storing the new [Power BI assets](#). You can choose an existing domain or create one or more new domains. You will include the domain reference ID (or IDs) in the [lineage harvester configuration file](#). Collibra uses the domains to ingest all Power BI assets during the [Power BI integration](#) process.

The following procedure guides you through the creation of 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 a domain type from the list.

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
Type	The domain type of the domain you are creating. In this case, you need to select <i>BI Catalog</i> .
Community	The community under which the domain will be located.
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 reference 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 reference ID is in bold.

8. Paste a reference ID in the `domainId` property in your [lineage harvester configuration file](#). This is your default domain.9. Optionally, if you want to ingest the contents of specific workspaces into specific domains in Collibra, paste the relevant domain reference IDs in the `filters` section of your [Power BI <source ID> configuration file](#).

What's next?

[Prepare](#) the Data Catalog physical data layer.

Soft delete of Power BI assets

When you integrate PowerBI, 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 a specific workspace, and then run the lineage harvester, the status of the corresponding assets in Data Catalog changes to "Missing from source". If you then add back the workspace and run the lineage harvester, the status of the assets will revert to the status they 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.

Technical lineage for Power BI via Edge

Note There are currently two supported methods for integrating Power BI metadata in Data Catalog:

- [Via the lineage harvester](#)
- Via Edge, as described in this topic.

Power BI integration [via the Power BI harvester](#) is deprecated and no longer supported.

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.

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 assets from [registered data sources](#).
- Between ingested assets from BI sources and Data Catalog assets from registered data sources.

Before Collibra 2023.02, if you wanted to collect source code from Power BI and create these new relations, you had to prepare a configuration file and use the [lineage harvester](#). With Collibra 2023.02, you can now integrate Power BI by configuring the lineage harvester as a capability on Edge. Benefits include:

- Seamless integration with Data Catalog.
- The Edge User Interface (UI), instead of Command Line Interface (CLI).
- Power BI connection from Edge, instead of lineage harvester drivers.

You can still use the lineage harvester as a standalone app.

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.

Creating a Technical lineage for Power BI via Edge

This topic includes an overview of the tasks to complete to create a technical lineage for Power BI via Edge.

Before you begin

- [Create](#) an Edge site in Collibra Data Intelligence Cloud.
- Optionally, [configure](#) a forward proxy.
- [Install](#) an Edge site.
- [Enable and configure](#) technical lineage via Edge.

Steps

1. [Prepare](#) a domain for Power BI ingestion.
2. [Create](#) a connection to Power BI.
3. [Add](#) the Technical lineage for Power BI capability to the Edge site.
4. [Synchronize](#) the Power BI technical lineage.

What's next?

- View the domain in which you ingested the assets.
- [View](#) the technical lineage.

Prepare a domain for Power BI ingestion

During Power BI integration, Power BI assets are ingested in the domain you specify in the **Domain ID** field in the [Technical Lineage for Power BI capability](#).

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. In the list, 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
Type	The domain type of the domain you are creating. In this case, <i>BI Catalog</i> .
Community	The community under which the domain will be located.
Name	The name of the new domain or domains.

5. Click **Create**.
6. Open your domain.
7. Copy the reference ID of the 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 ID in the **Domain ID** field in the Technical Lineage for Power BI capability.

Warning Collibra Data Lineage creates all Power BI assets in a single BI Catalog domain. We highly recommend that you do not move these assets to another domain. If you move assets to another domain, they will be deleted and recreated in the initial BI Catalog domain when you synchronize Power BI. As a consequence, all manually added data of those assets is lost.

Create a connection to Power BI

To retrieve data from Power BI, you have to connect to Power BI via the Edge site.

Prerequisites

- You have a [global role](#) that has the **System administration** [global permission](#).
- You have a [global role](#) that has the Manage Edge sites [global permission](#).
- You have a [resource role](#) with the Configure external system [resource permission](#), for example, Owner.
- You have the necessary Power BI permissions.

Steps

1. Open an Edge site.
 - a. In the main menu, click **Settings**.
 - » The **System** tab page appears.
 - b. In the tab pane, click **Edge**.
 - » The Edge sites overview appears.
 - c. In the Edge site overview, click the name of an Edge site.
 - » The Edge site page appears.
2. In the **Connections** section, click **Create Connection**.
 - » The **Connection settings** page appears.

3. Enter the required information.

Field	Description	Required
Connection settings	This section contains general information about the connection.	
Name	The name of the connection. The name can be anything, as long as it is unique. Tip The name that you provide here is the name you have to select in the Power BI connection field, when adding the Technical Lineage for Power BI capability to the Edge site.	✓ Yes
Description	The description of the connection.	✗ No
Connection provider	The type of connection. Select Power BI connection.	✓ Yes
Connection parameters	This section contains connection authentication information.	
Tenant Domain	The domain associated with the Microsoft Azure tenant. This domain is either a default domain or a custom domain. For example, collibrapowerbi.onmicrosoft.com. Note Usually, you can find a list of Power BI tenant or server domains in your Azure Active Directory or in the top right menu.	✓ Yes
Authentication type	The authentication type for your connection to Power BI. Enter one of the following: ◦ <i>Service Principal</i> ◦ <i>Resource Owner Password Credentials</i>	✓ Yes
Application ID	The unique ID of the Microsoft Azure Application (client) ID.	✓ Yes

Field	Description	Required
Username	The email address of your Azure Active Directory user. This field only applies if you entered <i>Resource Owner Password Credentials</i> in the Authentication type field.	✗ No
Password/Secret key	Your password (if you entered <i>Resource Owner Password Credentials</i> in the Authentication type field) or your secret key (if you entered <i>Service Principal</i> in the Authentication type field), for your Azure Active Directory user.	✓ Yes

4. Click **Create**.

What's next?

[Add](#) the Technical Lineage for Power BI capability to the Edge site.

Add the Technical Lineage for Power BI capability to the Edge site

After you [create](#) a connection to Power BI, you can add the Technical Lineage for Power BI capability.

Requirements and permissions

A [global role](#) that has the following [global permissions](#):

- System administration
- Manage connections and capabilities, for example Edge integration engineer.
- Register profiling information

Steps

1. Open an Edge site.
 - a. On the main menu, 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 table, click the name of the Edge site whose status is **Healthy**.
 - » The Edge site page opens.
2. In the **Capabilities** section, click **Add capability**.
 - » The **Add capability** page appears.
3. Enter the required information.

Field	Description	Required
Capability	This section contains general information about the capability.	
Name	The name of the Edge capability.	✓ Yes
Description	The description of the Edge capability.	✗ No
Capability template	The capability template. The value that you select in this field determines which sections appear on the page. In the drop-down list, select Technical Lineage for Power BI.	✓ Yes
Main Properties	The required custom properties are based on the selected capability template.	
Source ID	The connection to the data source .	✓ Yes
Power BI connection	The Power BI connection that you created for ingestion in Data Catalog. Tip Select the name that you provided in the Name field when you created a connection to Power BI.	✓ Yes
API URL	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.	✗ No

Field	Description	Required
Scope	Optional property that is intended only for customers with a different scope, such as Chinese tenants. Example 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.	scope
Domain ID	The unique reference ID of the domain in Collibra Data Intelligence Cloud in which you want to ingest the Power BI assets. How do I find a domain reference ID? Open the relevant domain in Collibra. The URL looks like: <a href="https://<yourcollibrainstance>/domain/22258f64-40b6-4b16-9c08-c95f8ec0da26?view=00000000-0000-0000-0000-0000000040001">https://<yourcollibrainstance>/domain/22258f64-40b6-4b16-9c08-c95f8ec0da26?view=00000000-0000-0000-0000-0000000040001. In this example, the reference ID is in bold.	✓ Yes

Field	Description	Required
Source configuration	This field allows you to provide JSON code for database mapping, workspace filtering and specifying the name of a System asset in Collibra. Tip If you previously integrated Power BI via the lineage harvester, you can copy and paste in this field the JSON code from your Power BI <source ID> configuration file. - Specify the name of a database, on which server the database is running, and optionally, the name of the schema. - 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 <code>useCollibraSystemName</code> in the lineage harvester configuration file is set to <code>true</code>, use the <code>collibraSystemName</code> 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.	✗ No
Custom Properties	This section identifies to which Collibra Data Lineage service instance you want to upload the Power BI metadata.	

Field	Description	Required
techlinKey	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 to US government customers.	✕ No
techlinHost	The URL of the Collibra Data Lineage service instance to which you want to upload Power BI metadata, for example "techlin-gcp-eu.collibra.com". Warning This applies only to US government customers.	✕ No
Advanced Properties	This section contains the advanced properties for creating a technical lineage.	
Delete Raw Metadata After Processing	Metadata is harvested and uploaded in a ZIP file to a Collibra Data Lineage service instance, for processing. Use this optional property to specify whether or not the raw metadata should be deleted after it has been processed. If you select this option, the raw metadata is deleted after processing. If you don't select this option, it is stored in an Amazon S3 bucket.	

Field	Description	Required
Analyze Only	This option determines whether to only load and analyze the source data on the Collibra Data Lineage service instances. When you select this option, the technical lineage of the data source is not created during the synchronization of the capability. Selecting this option is equivalent to entering the load-sources and analyze command with a source specified when you use the lineage harvester. This option is not enabled by default. Use case You can use this option to control when to start full synchronization of data sources. For example if you have three data sources, A, B, and C, you can synchronize the data sources as follows: - During weekdays, synchronize data sources A and B with the Analyze Only checkbox selected in the capabilities. Collibra Data Lineage only loads and analyzes data sources A and B without synchronizing the technical lineage. - On the weekend, synchronize data source C without selecting the Analyze Only checkbox in the capability. Collibra Data Lineage synchronizes the technical lineage for all data sources including A, B, and C.	✕ No
Active	The option determines whether to include or remove the technical lineage of the data source. Select this option to include the technical lineage of this data source. Clear the checkbox to exclude the technical lineage of this data source.	✕ No

4. Click **Create**.

» The Edge site connects to your data source.

What's next?

[Synchronize](#) the Power BI technical lineage.

Synchronize a Power BI technical lineage

You can synchronize your technical lineage manually or automatically by adding a synchronization schedule.

When you synchronize a technical lineage capability, a full synchronization is run, which is equivalent to the `full-sync command` with the `-s` argument specified when you use the lineage harvester. You can synchronize as many technical lineage capabilities in parallel as you want. If a full synchronization is in progress, the other capabilities are put in a queue until a new full synchronization starts automatically.

Prerequisites

- 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

Manually synchronize technical lineage for Power BI

1. On the main menu, click , and then click  **Catalog**.
 - » The Catalog Home opens.
2. On the main toolbar, click .
 - » The **Create** dialog box appears.
3. In the **Register with Edge** section of the **Create** dialog box, click **Register a data source**.
 - » The **Register content** page opens.
4. In the **Connection name** column, locate the Power BI connection that you used when you added the Power BI capability. Then, click the capability name in the **Data**

sources/Capabilities column of the Power BI connection.

» The technical lineage capability configuration page opens.

5. On the **Configuration Section** tab pane, click **Synchronize now**.

» A notification indicates synchronization has started.

» The synchronization job is started. The tables and columns from the selected schema are ingested based on the configured table rules.

Add a synchronization schedule

1. On the main menu, click , and then click  **Catalog**.
 - » The Catalog Home opens.
2. On the main toolbar, click .
 - » The **Create** dialog box appears.
3. In the **Register with Edge** section of the **Create** dialog box, click **Register a data source**.
 - » The **Register content** page opens.
4. In the **Connection name** column, locate the Power BI connection that you used when you added the Power BI capability. Then, click the capability name in the **Data sources/Capabilities** column of the Power BI connection.
 - » The technical lineage capability configuration page opens.
5. On the **Synchronization schedule** tab pane, click **Add schedule**.
6. Enter the required information and click **Save**:

Field	Description
Repeat	The interval when you want to synchronize automatically. The possible values are: Daily , Weekly , Monthly , and Cron expression .
Cron	The Quartz Cron expression that determines when the synchronization takes place. This field is only visible if you select Cron expression in the Repeat field.
Every	The day on which you want to synchronize, for example, Sunday. This field is only visible if you select Weekly in the Repeat field.

Field	Description
Every first	The day of the month on which you want to synchronize, for example, Tuesday. This field is only visible if you select <code>Monthly</code> in the Repeat field.
At	The time at which you want to synchronize automatically, for example, 14:00. This field is only visible if you select <code>Daily</code> , <code>Weekly</code> , or <code>Monthly</code> in the Repeat field.
Time zone	The time zone for the schedule.

What's next?

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 troubleshooting guide](#) or [Collibra Support Portal](#) to fix the errors.

Technical lineage for Power BI via the lineage harvester

Note There are currently two supported methods for integrating Power BI metadata in Data Catalog:

- [Via Edge](#).
- Via the lineage harvester, as described in this topic.

Power BI integration [via the Power BI harvester](#) is deprecated and no longer supported.

Power BI integration steps

The Power BI integration enables you to harvest Power BI metadata and create new Power BI assets in Data Catalog. Colibra analyzes and processes the BI metadata and presents it as specific [asset types](#), retaining their original names.

Steps

1. [Complete](#) the tasks in Power BI and Microsoft Azure.
2. [Prepare](#) a domain for Power BI ingestion.
3. [Prepare](#) the Data Catalog physical data layer.
4. Optionally, assign the attribute type State to the global assignment of the Power BI Workspace asset type.

For complete information, see [Power BI workspaces](#).

5. [Download](#) and [install](#) the lineage harvester.
6. [Prepare](#) the lineage harvester configuration file.
7. Optionally, [Prepare](#) the Power BI <source ID> configuration file.
8. 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](#).

9. Start the lineage harvester again in the console and run the following command:

- for Windows: `.\bin\lineage-harvester.bat full-sync`
- for other operating systems: `./bin/lineage-harvester full-sync`

When prompted, enter the password or client secret to connect to your Colibra Data Intelligence Cloud and Tableau environment. The passwords are encrypted and stored in `/config/pwd.conf`.

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.

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 [technical lineage](#).

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.

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.

Lineage harvester system requirements

You need to meet the system requirements to be able to [install](#) and run the [lineage harvester](#).

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 [Technical lineage general troubleshooting](#).

- 20 GB free disk space

Important If you have more than 50,000 actively used workspaces, we recommend increasing the heap size to at least 6 GB.

Certificate requirements

Ensure that the Power BI certificate exists in the Java TrustStore. For complete information, see [Add the Power BI certificate to the Java TrustStore](#).

Network requirements

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

You need the following minimum network requirements:

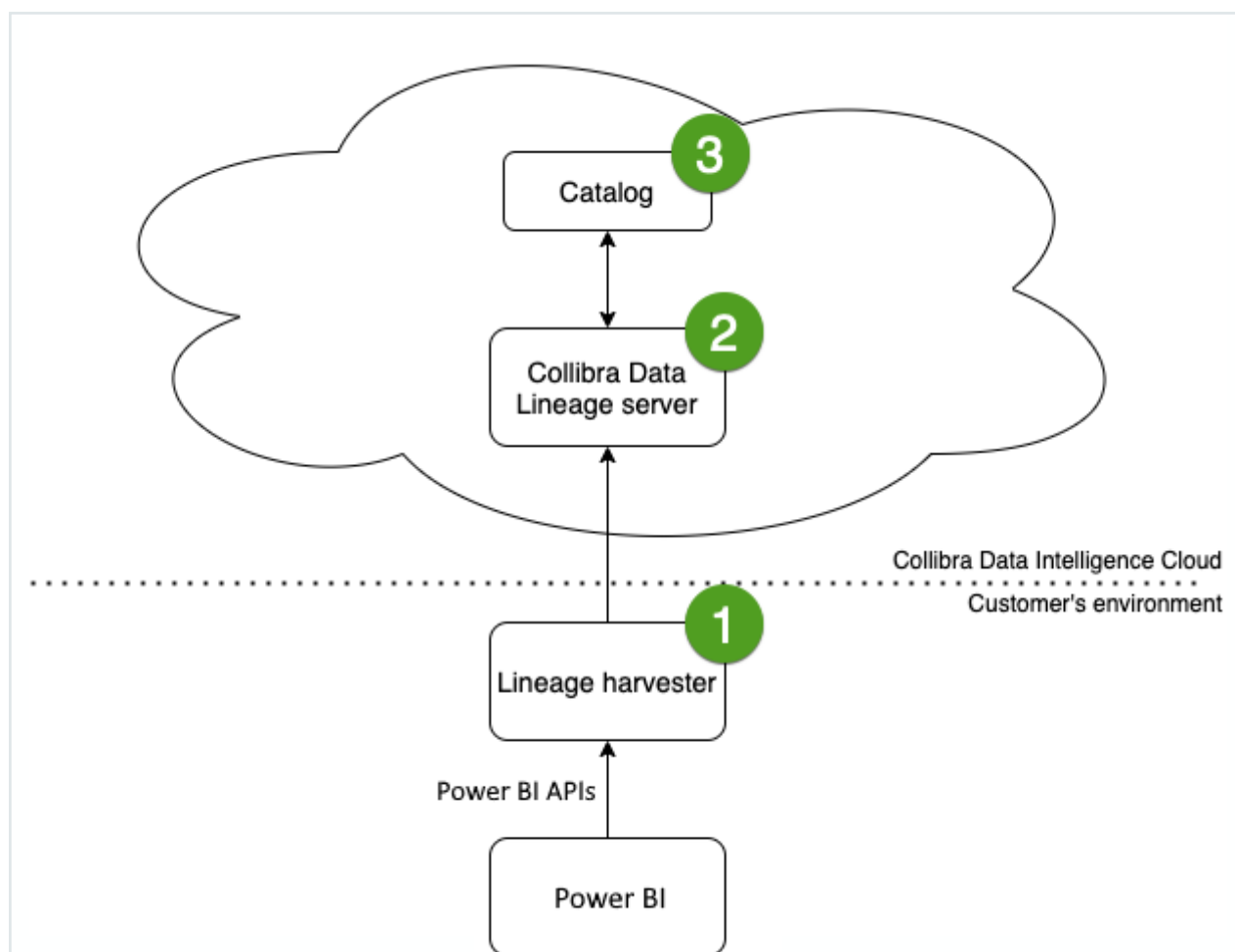
- Firewall rules so that the lineage harvester can connect to:
 - Collibra Data Intelligence Cloud.
 - All [Collibra Data Lineage service instances](#) in your geographic location:
 - 15.222.200.199 (techlin-aws-ca.collibra.com)
 - 18.198.89.106 (techlin-aws-eu.collibra.com)
 - 13.228.38.245 (techlin-aws-sg.collibra.com)
 - 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 Power BI harvester rescans all your Power BI metadata. 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 Power BI harvester cannot connect to other Collibra Data Lineage service instances.

Power BI ingestion workflow

You run the lineage harvester to start the Power BI ingestion workflow. When you initiate Power BI ingestion, each workflow component performs the following actions:

1. The lineage harvester:
 - Communicates with Power BI.
 - Harvests the Power BI metadata that will be ingested to Data Catalog.
 - Sends the Power BI metadata to the Collibra Data Lineage service.
2. The Collibra Data Lineage service:
 - Analyzes the Power BI metadata.
 - Creates new assets and relations.
 - Stitches existing assets in Data Catalog to Power BI assets.
 - Imports new Power BI assets and their relations in Data Catalog.
3. Data Catalog:
 - Shows the new [Power BI assets](#).
 - Shows a Technical lineage tab on Power BI Column pages.
 - Shows stitching results between Power BI Column assets and Column assets.



Install the lineage harvester for Power BI integration

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

Tip

- Install the lineage harvester close to your data source or on the same server.
- The lineage harvester uses port 443.

Prerequisites

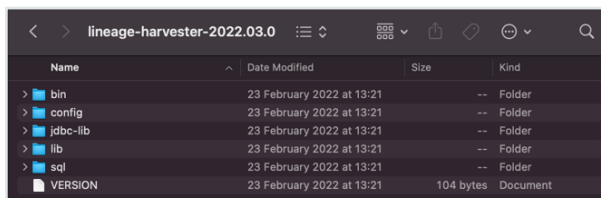
- You have purchased the Power BI metadata connector and lineage feature.
- You have Collibra Data Intelligence Cloud 2020.11 or newer.
- You meet the [minimum system requirements](#).

- 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.

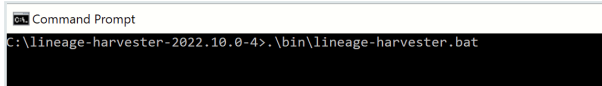
Steps

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.

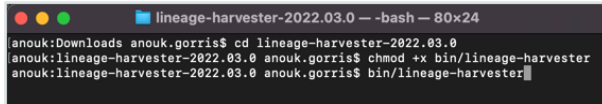


3. Start the lineage harvester to create an empty lineage harvester configuration file by entering the following command:

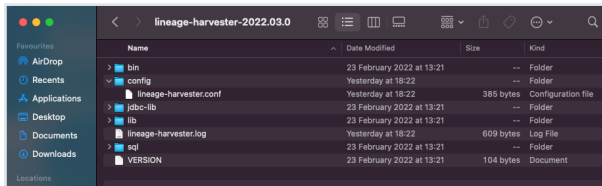
- **Windows:** `.\bin\lineage-harvester.bat`



- **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.

Prepare the lineage harvester configuration file for Power BI

You have to prepare a technical lineage configuration file and run the [lineage harvester](#) to retrieve metadata from Power BI and send it to the Collibra Data Lineage service to be scanned, processed and analyzed.

Example configuration file

```
{
  "general": {
    "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
  } ]
}

```

Before you begin

- [Set up the latest lineage harvester.](#)
- [Create](#) one or more BI Catalog domains in which you want to ingest the Power BI assets.
- Manually refresh your datasets in Power BI.

Requirements and permissions

- 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 BI Data Catalog domain:
 - Asset: add
 - Attribute: add
 - Domain: add
 - Attachment: add

Steps

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

Tip Watch a video on how to do this:

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 to US government customers.

Properties	Description
url	The URL of the Collibra Data Lineage service instance. "url": "https://techlin-gov.collibra.com" Warning This 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 applies only to US 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
<code>useCollibraSystemName</code>	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.
<code>sources</code>	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.

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	The unique ID to identify the Power BI service metadata that was uploaded to the Collibra Data Lineage service. 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.
tenantDomain	The Power BI tenant domain is the domain associated with the Microsoft Azure tenant. This domain is either a default domain or a custom domain. You can specify this property with the URL, such as <i>collibrapowerbi.onmicrosoft.com</i> or tenant ID, such as <i>e**b****-****-****-****-1b**d****4663</i>. Note Usually, you can find a list of Power BI tenant or server domains in your Azure Active Directory or in the top right menu.
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. ◦ If you use service principle: The value should be <code>ServicePrincipal</code>. ◦ If you use username and password: The value should be <code>ResourceOwnerPasswordCredentials</code>.

Properties	Description
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 HTTP1 streams, in case file-size limitations are resulting in timeout errors while using the default HTTP2 streams.
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
See an example.	
<pre> { "general": { "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 }] } </pre>	

2. Save the configuration file.

What's next?

[Prepare](#) the Power BI <source ID> configuration file.

Prepare Power BI <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:

- Specify the name of a database, on which server the database is running, and optionally, the name of the schema.

- 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.on-
microsoft.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. It allows you to 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). 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 of a supported data source in Power BI.	No
schema	The name of the default schema of a supported data source in Power BI. If the lineage harvester fails to find a specific schema, it uses the default schema.	No
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- e=databasename1;found_host- name=*;found_ schema=schema1": { "dbname": "Customers", "schema": "mssql-schema- name", "dialect": "mssql", "collibraSystemName": "Customers-Europe" }, "found_dbname- e=databasename2;found_host- name=server- name.on- microsoft.com;found_ schema=schema2": { "dbname": "Customers", "schema": "oracle-schema- name", "dialect": "oracle", "collibraSystemName": "Customers-USA" }, </pre>	Yes (unless you are using a <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.

What's next?

See [Overview Power BI integration steps](#).

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 there [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.

Note 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.

Command	Description
<code>full-sync</code>	Uploads all your data sources to the Collibra Data Lineage service instance where the data source metadata is processed and uploaded to Data Catalog.

Command	Description
<code>-s "<ID of data source>"</code>	Uploads only the data source with the specified ID. For example, <code>full-sync -s "myOracleDataSource"</code>. 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. 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.
<code>load-sources</code>	Downloads all your data sources in a separate ZIP file, per data source, to the lineage harvester output folder.
<code>-s <ID of data source></code>	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
<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.
<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>--timeout <seconds></code>	Determines the network timeout.
<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.

Migrating your existing Power BI assets via the lineage harvester

For those who have integrated Power BI in Data Catalog via the Power BI harvester, this topic provides instruction on how to migrate your existing Power BI assets to the [lineage harvester integration method](#).

Steps

The table below shows the steps and prerequisites required to integrate Power BI in Data Catalog. These steps are best practices, which means that some of them might be optional, but highly recommended.

Step	What?	Description	Prerequisites
1	Download and install the newest versions of: • The Power BI harvester. • The lineage harvester.	You need to run both harvesters, to synchronize the Power BI metadata via the original ingestion method, as described in Working with Power BI service. You can download both harvesters from the Collibra Product Resource Downloads page.	See: • Install the Power BI harvester. • Install the lineage harvester.
2	Prepare both configuration files.	Prepare the Power BI configuration file and the lineage harvester configuration file, for ingestion via the original method, and then run the harvesters. Empty configuration files are included in the respective folders, after installing each harvester.	See: • Prepare the Power BI configuration file. • Prepare the lineage harvester configuration file.
3	Update permissions.	Ensure that you have completed all of the prerequisite tasks ingesting Power BI metadata via the lineage harvester.	

Step	What?	Description	Prerequisites
4	Make random call to the Metadata API.	Make a call to the Metadata API, to ensure that you can communicate with the Power BI server. For example: <pre>GET https://api.powerbi.com/v1.0/myorg/admin/workspaces/modified</pre>	
5	Manually refresh your Power BI datasets.	Manually refresh your Power BI datasets to ensure that the data in your Power BI datasets is up-to-date. This is necessary because the Power BI APIs use cached results. For complete information, see: • The Microsoft documentation. • The Microsoft Power BI Blog.	See Power BI pre-requisites .
6	Edit the lineage harvester configuration file.	Edit the lineage harvester configuration file so that it includes the required properties for ingesting via the lineage harvester. See Prepare the lineage harvester configuration file for Power BI. Warning The value of the <code>id</code> property in your lineage harvester configuration file must be the same as the value of the <code>sourceId</code> property in your Power BI configuration file.	See Prepare the lineage harvester configuration file for Power BI .
7	Run the lineage harvester with the full-sync command	Start the lineage harvester again in the console and run the following command: • for Windows: <code>.\bin\lineage-harvester.bat full-sync</code> • for other operating systems: <code>./bin/lineage-harvester full-sync</code> This triggers the creation of Power BI assets, their relations and a technical lineage in Data Catalog. We highly recommend that you always install and use the newest lineage harvester.	See Prepare the lineage harvester configuration file for Power BI .

Migrate your existing Power BI assets to the new integration method

The following process allows you to migrate Power BI assets that you integrated via the Power BI harvester, to the [new integration method](#).

Prerequisites

- You have Collibra Data Intelligence Cloud 2020.11 or newer.
- You have downloaded the newest Power BI harvester and lineage harvester from the [Downloads page](#), and have the system requirements to install and use them.
- You have purchased the Power BI metadata connector and lineage feature.
- You meet the [minimum system requirements](#).
- You have completed all of the [prerequisite tasks](#).
 - You have registered Power BI in Microsoft Azure.
 - You have enabled the service principal option in the [Power BI Admin portal](#).
- You have added Firewall rules so that the Power BI harvester and lineage harvester can connect to the Collibra Data Lineage service instance with one of the following IP addresses:
 - 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)
- 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](#).
- You have a [global role](#) with the Data Stewardship Manager [global permission](#).

- 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
- Java Runtime Environment version 11 or newer or OpenJDK 11 or newer.
- You have a [global role](#) that has the Manage all resources [global permission](#).
- You have [created](#) a BI Catalog domain (or domains) in which you want to ingest the Power BI assets.
- For a full ingestion, we highly recommend to have a [Power BI Premium subscription](#).

Steps

1. Download and install:
 - The [Power BI harvester](#).
 - The [lineage harvester](#).
2. Prepare the [Power BI configuration file](#) and the [lineage harvester configuration file](#), and then run the harvesters.
3. Ensure that you have completed all of the [prerequisite tasks](#) to ingest Power BI metadata via the lineage harvester.
4. Make a random call to the Metadata API, to ensure that you can communicate with the Power BI server. For example:


```
GET https://api.powerbi.com/v1.0/myorg/admin/workspaces/modified
```
5. Manually refresh your Power BI datasets, to ensure that the data is up-to-date. For complete information, see:
 - The [Microsoft documentation](#).
 - The [Microsoft Power BI Blog](#).
6. Edit the lineage harvester configuration file so that it includes the required properties to ingest Power BI metadata via the lineage harvester.

Warning The value of the `id` property in your lineage harvester configuration file must be the same as the value of the `sourceId` property in your Power BI configuration file.

7. Start the lineage harvester again in the console and run the following command:
 - for Windows: `.\bin\lineage-harvester.bat full-sync`
 - for other operating systems: `./bin/lineage-harvester full-sync`

Working with Power BI service (deprecated)

Note Power BI integration via the Power BI harvester, as described in this section of the documentation, is deprecated. The development of new features and improvements is discontinued. For complete information on integrating Power BI via supported means and migrating existing Power BI assets, see the following topics:

- [Technical lineage for Power BI via the lineage harvester.](#)
- [Migrating your existing Power BI assets via the lineage harvester.](#)
- [Technical lineage for Power BI via Edge.](#)

Power BI service is a cloud business intelligence software that helps you see and understand your data. You can ingest Power BI metadata in Data Catalog and create a technical lineage.

The Power BI service integration in Collibra Data Intelligence Cloud is not the same as the Power BI Report Server integration. If you want to ingest Power BI Report Server metadata in Collibra Data Intelligence Cloud, please read the Power BI Report Server section of the user guide.

Note If you want to ingest Power BI metadata in Data Catalog, you have to purchase the Power BI connector and lineage feature.

Features

Collibra Data Lineage currently supports two methods by which to integrate Power BI in Data Catalog. The following table shows the features specific to the two integration methods.



Feature	Power BI via the Power BI harvester (deprecated)	Power BI via the lineage harvester
Catalog ingestion	✓	✓
Technical lineage	✓	✓
Automatic stitching	✓	✓
Uses only one harvester		✓
No Windows dependency		✓
Uses new Power BI APIs		✓
Workspace filtering for high-volume data		✓
Mapping to target domains		✓
Data flow support		✓

Power BI terminology (deprecated)

Before you ingest [Power BI](#), read more about the Power BI terminology and how it maps with the Collibra Data Intelligence Cloud asset types.

Note For more information, see the [Power BI documentation](#).

Power BI term	Description	Asset type in Collibra
Capacity	A resource that hosts Power BI Workspaces.	Power BI Capacity

Power BI term	Description	Asset type in Collibra
Dashboard	A collection of Power BI tiles with metrics from one or more Reports and Data Models.	Power BI Dashboard
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

Power BI operating model (deprecated)

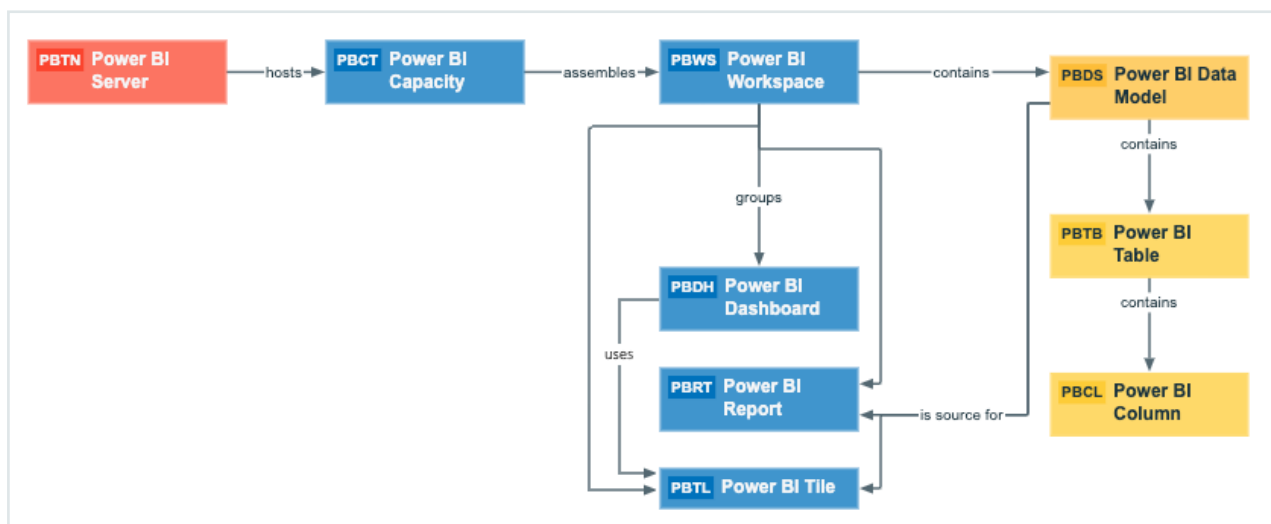
The [Power BI harvester](#) collects Power BI metadata and sends it to the [Collibra Data Lineage server](#). Collibra processes the metadata and creates new Power BI assets and relations in Data Catalog. You can see them on the asset page overview or visualize them in a [diagram](#) or in a [technical 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.
- Depending on your [Power BI subscription](#), it could be that not all asset types are created.
- Asset types are only created if you have all specific Power BI and Data Catalog [permissions](#).
- All Power BI asset types are created in the same domain.
- Relations that were manually created between Power BI assets and other assets in accordance with the relation types in the Power BI operating model, are deleted after a refresh of the Power BI metadata.

Power BI metadata overview

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, assuming you have the necessary [subscriptions](#) and [configurations](#) for a full ingestion. This table also shows the Resource IDs for each asset type and metadata.

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
	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	
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	Report uses / used in Report	00000000-0000-0000-0000-120000000007
	Report related to / impacted by Business Asset	00000000-0000-0000-0000-120000000006
Power BI Data Model Resource ID: 00000000-0000-0000-0000-100000000007	Full name	
	Display name	
	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 / source 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
Power BI Report Resource ID: 00000000-0000-0000-0000-100000000006	Full name	
	Display name	
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	Data Asset is source for / source 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
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
Power BI Table Resource ID: 00000000-0000-0000-0000-100000000009	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

Asset type	Synchronized metadata	Resource ID
Power BI Tile Resource ID: 00000000-0000-0000-0000-100000000005	Full name	
	Display name	
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	Data Asset is source for / source 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
Power BI Workspace Resource ID: 00000000-0000-0000-0000-100000000003	Full name	
	Display name	
	Description	00000000-0000-0000-0000-000000003114
	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
	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.

Recommended hierarchy within a domain

You can enable [hierarchies](#) for the domain (or domains) in which your Power BI assets were 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**.

The following image shows an example of a BI Catalog domain with hierarchies enabled.

BI Catalog Hierarchy ▾

> Delete Move

☐ Name	Status ↑	Asset Type
☐ 3f5befef-44a9-4ccb-8c92-603315fcdd70	Candidate	Power BI Server
☐ presalespowerbiresource	Candidate	Power BI Capacity
☐ Power BI Demo	Candidate	Power BI Workspace
☐ Colibra Connectivity Power BI Product Dashb...	Candidate	Power BI Dashboard
☐ Colibra Connectivity Power BI Sales Dashboard	Candidate	Power BI Dashboard
☐ This Year's Sales, Last Year's Sales	Candidate	Power BI Tile
☐ Retail Analysis Sample	Candidate	Power BI Data Model
☐ Retail Analysis Sample	Candidate	Power BI Report
☐ Retail Analysis Sample	Candidate	Power BI Data Model
☐ Product Cost Report	Candidate	Power BI Tile
☐ This Year's Sales	Candidate	Power BI Tile
☐ Customer Sales Report	Candidate	Power BI Report
☐ Customer Sales Report	Candidate	Power BI Data Model
☐ CustomerSalesReporting	Candidate	Power BI Table
☐ SalesAmount	Candidate	Power BI Column
☐ FullName	Candidate	Power BI Column
☐ OrderQuantity	Candidate	Power BI Column

Note In an asset view, like the one shown in the previous image, 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.

Power BI asset and domain types (deprecated)

The [Power BI](#) integration in Collibra Data Intelligence Cloud uses a specific subset of packaged [asset types](#) and [domain types](#).

The following table contains the asset and domain types that are used for the Power BI integration. You can see the parent asset types in the breadcrumbs above each asset type.

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

Asset type	Description	Domain type
Data Asset › Data Structure › Data Entity › BI Data Entity › Power BI Table	A table in a Power BI Data Model.	BI Catalog
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 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

Overview Power BI integration steps (deprecated)

The Power BI integration enables you to harvest Power BI metadata and create new Power BI assets in Data Catalog. Collibra analyzes and processes the BI metadata and presents it as specific [asset types](#), retaining their original names.

Tip To ingest Power BI metadata in Data Catalog, you need to run [two different harvesters](#): the [Power BI harvester](#) and the [lineage harvester](#). The order in which you run the harvesters is important. You first have to run the Power BI harvester to collect the metadata from your Power BI application and then run the lineage harvester to import new Power BI assets and their relations in Data Catalog. The [Power BI ingestion workflow](#) explains which roles the harvesters play in the Power BI ingestion process.

Steps

The table below shows the steps and prerequisites required to integrate Power BI in Data Catalog. These steps are best practices, which means that some of them might be optional, but highly recommended.

Step	What?	Description	Prerequisites
1	Set up a Power BI application.	Before you start the Power BI integration in Data Catalog, make sure that the Power BI harvester can reach the Power BI metadata. Perform these tasks before you start the actual Power BI ingestion process: • The authentication process. • The registration of your Power BI application in Microsoft Azure. • The Power BI roles and dedicated capacities for Power BI workspaces. • The required Power BI subscription. 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.	You have a Power BI subscription .

Step	What?	Description	Prerequisites
2	Create a new domain.	Before you can ingest Power BI metadata, you have to create a new domain or choose an existing domain to store the new Power BI assets.	You have a resource role with the following resource permissions: • Domain: Add
3	Optionally, assign the attribute type State to the global assignment of the Power BI Workspace asset type	On Power BI Workspace asset pages, you can include the attribute type State, to show the state of ingested Power BI workspaces. To do so, you have to edit the global assignment of the Power BI Workspace asset type and assign the attribute type State. If you delete a Power BI workspace, the workspace is maintained for a 90-day grace period. During the grace period, the workspace has the state Deleted. When you ingest Power BI metadata in Data Catalog, this deleted workspace is ingested. For complete information on Power BI workspaces and possible states, see the Microsoft Power BI documentation.	You have a global role that has the System administration global permission .
4	Ingest or import assets from supported JDBC data sources .	The Collibra Data Lineage server connects to Data Catalog and reads the full paths of existing assets. When the full path matches the full path of assets in Power BI, the Collibra Data Lineage server automatically stitches them.	Permissions depend on how you ingest or import the assets.

Step	What?	Description	Prerequisites
5	Download and install the Power BI harvester.	You use the Power BI harvester to collect metadata from Power BI and upload it to Collibra where the metadata is scanned, processed and analyzed. The installer file contains the following: • a config folder with an empty configuration file. • a bin folder. • a TXT file with more information about the configuration file. • a BAT file that you use to run the harvester.	• You have Collibra Data Intelligence Cloud 2020.11 or newer. • You have access to the Power BI harvester on the Downloads page. • Your environment meets the system requirements to install and use the Power BI harvester. • You have added Firewall rules so that the Power BI harvester can connect to the Collibra Data Lineage server with one of the following IP addresses: • 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)

Step	What?	Description	Prerequisites
			• 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) Important Ingestion results vary according to your Power BI subscription.

Step	What?	Description	Prerequisites
6	Prepare the Power BI configuration file and run the Power BI harvester.	You create a configuration file to provide the connection information that you need to connect your Power BI application to Collibra and to the domain in which you want to ingest the Power BI assets. You can access an empty configuration file in the Power BI harvester installation folder. When you have created and saved the configuration file, you can run the Power BI harvester which uploads the Power BI metadata to Collibra.	• You have access to the Power BI harvester on the Downloads page. • You have completed all prerequisite tasks. • You have a dedicated domain to ingest the Power BI assets. • 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. • You have a global role with the Data Stewardship Manager global permission. • 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 • Your environment meets the system requirements to run the Power BI harvester and the lineage harvester.

Step	What?	Description	Prerequisites
			Tip For a full ingestion, we highly recommend to have a Power BI Premium subscription.
7	Download and install the lineage harvester.	You use the lineage harvester to trigger the creation of Power BI assets, their relations and a technical lineage in Data Catalog. You can download the lineage harvester from the Collibra Product Resource Downloads page.	Your environment meets the system requirements to install and use the lineage harvester.

Step	What?	Description	Prerequisites
8	Prepare the lineage harvester configuration file and run the lineage harvester.	You create a lineage harvester configuration file with Power BI connection information and run the lineage harvester to import the results of the Power BI integration and the technical lineage for Power BI into Data Catalog. As a result, Collibra creates new Power BI assets in Data Catalog and imports relations between these assets. It also creates a technical lineage for Power BI assets and other data sources in the lineage harvester configuration file. Tip For more information about the lineage harvester, see the Collibra Data Lineage documentation.	- You have downloaded the lineage harvester version 1.2.1 or newer. - Your environment meets the system requirements to install and run the lineage harvester. - You have prepared a Power BI harvester configuration file. - 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. - 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

Step	What?	Description	Prerequisites
9	View the Power BI assets and technical lineage	After the Power BI metadata is ingested in Data Catalog, you can go to the domain where you ingested Power BI and see the list of ingested Power BI assets. These assets are automatically stitched to existing assets in Data Catalog. You can go to a Power BI Column asset page and click the Technical lineage lineage tab to view the technical lineage. 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. Warning When you run the harvesters, Collibra Data Lineage creates all Power BI assets in the same Data Catalog BI domain. We highly recommend that you do not move these assets to another domain. If you move assets to another domain, they will be deleted and recreated in the initial Data Catalog BI domain when you synchronize Power BI. As a consequence, all manually added data of those assets is lost.	You have a Data Catalog global role with the Technical lineage global permissions .

Note The order in which you run the [harvesters](#) is important. You first have to run the Power BI harvester to collect the metadata from your Power BI application and then run the lineage harvester to import new Power BI assets and their relations in Data Catalog.

Ingestion results based on Power BI subscriptions (deprecated)

You can only ingest Power BI metadata to which the Power BI user has access to. Your level of access is determined by your Power BI subscriptions.

The following table gives an overview of the minimum required subscriptions and the results in Data Catalog.

- Note
- The assets have the same names as their counterparts in Power BI. Full names and Display names cannot be changed in Data Catalog.
 - Depending on your [Power BI subscription](#), it could be that not all asset types are created.
 - Asset types are only created if you have all specific Power BI and Data Catalog [permissions](#).
 - All Power BI asset types are created in the same domain.
 - Relations that were manually created between Power BI assets and other assets in accordance with the relation types in the Power BI operating model, are deleted after a refresh of the Power BI metadata.

Minimum required subscription	Result in Data Catalog
Power BI Pro	Power BI Workspaces are not assigned to a dedicated capacity in Power BI. The following asset types are created in Data Catalog: • Power BI Server • Power BI Workspace • Power BI Dashboard • Power BI Tile • Power BI Report • Power BI Data Model Technical lineage is unavailable.

Minimum required subscription	Result in Data Catalog
Power BI Pro with Power BI Embedded Capacity subscription in Microsoft Azure	Power BI Workspaces are assigned to a embedded capacity in Azure. The following asset types are created in Data Catalog: • Power BI Server • Power BI Capacity • Power BI Workspace • Power BI Dashboard • Power BI Tile • Power BI Report • Power BI Data Model • Power BI Table • Power BI Column A technical lineage is created for all Power BI Column assets.
Premium	Power BI Workspaces are assigned to a dedicated capacity in Power BI. The following asset types are created in Data Catalog: • Power BI Server • Power BI Capacity • Power BI Workspace • Power BI Dashboard • Power BI Tile • Power BI Report • Power BI Data Model • Power BI Table • Power BI Column A technical lineage is created for all Power BI Column assets.

Minimum required subscription	Result in Data Catalog
Premium Per User	Power BI Workspaces are assigned to a dedicated capacity in Power BI. The following asset types are created in Data Catalog: • Power BI Server • Power BI Capacity • Power BI Workspace • Power BI Dashboard • Power BI Tile • Power BI Report • Power BI Data Model • Power BI Table • Power BI Column A technical lineage is created for all Power BI Column assets. Note The Power BI Premium Per User license is a new license type that is released for general availability, but is still in its preview period. For more information, see the Microsoft documentation.

Note We highly recommend you to have a Power BI Premium subscription. Power BI Premium also provides additional features and a better speed and performance.

Warning When you run the harvesters, Collibra Data Lineage creates all Power BI assets in the same Data Catalog BI domain. We highly recommend that you do not move these assets to another domain. If you move assets to another domain, they will be deleted and recreated in the initial Data Catalog BI domain when you synchronize Power BI. As a consequence, all manually added data of those assets is lost.

Power BI ingestion limitations (deprecated)

There are a few considerations and limitations that you must take into account when you use the [Power BI](#) metadata connector and lineage feature.

Supported subscriptions

Important Your [subscription](#) determines which Power BI metadata the Power BI harvester can collect.

You need one of the following subscriptions to ingest Power BI metadata in Data Catalog:

- Power BI Pro. To ensure a full ingestion, you also need a Power BI Embedded Capacity subscription in Microsoft Azure.
- Power BI Premium.
- Power BI Premium Per User.

Note The Power BI Premium Per User license is a new license type that is released for general availability, but is still in its preview period. For more information, see the [Microsoft documentation](#).

Other Power BI subscriptions are currently not supported.

Power BI metadata

The [Power BI 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.
- Power BI workspaces that are not part of a [dedicated capacity](#).
- Descriptions of most Power BI elements.
- [Power BI apps](#). They can be ingested as Power BI Reports, but there is no easy way to distinguish them from real Power BI reports.

The [Power BI harvester](#) cannot access metadata of the following Power BI elements:

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

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%.

Known issues

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

Known issue	Description
The Power BI harvester shows an Internal Server Error because of the Power BI workspace filter.	If you want to ingest a lot of Power BI data and you use the WorkspaceFilter in the Power BI configuration file, the Power BI API can go in timeout and, as a result, you get an Internal Server Error. If you get this error, we highly advise to not use the workspace filter. Note If the error message indicates that the issue is an internal server error, the problem is caused by the Power BI REST API, not the Power BI harvester itself.
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 Power BI 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
The IN operator is not supported.	Currently, the IN operator is not supported. As a result, you cannot use IN to filter the <code>workspaceFilter</code> property on specific Power BI workspace names in the Power BI harvester configuration file. For example, you want to filter on two Power BI workspace names. In the configuration file, you can enter the following value of the <code>workspaceFilter</code> property to ingest only workspaces with the name "workspace1" or "workspace2" <pre>"workspaceFilter": "name eq 'workspace1' or name eq 'workspace2'"</pre> However, you cannot ingest Power BI workspaces that have "workspace1" or "workspace2" in their name, because the IN operator is currently not supported: <pre>"name in ('workspace1', 'workspace2')"</pre> Tip For more syntax examples that you can use in the <code>workspaceFilter</code> property, see the README file attached to the Power BI harvester or see the Microsoft documentation.
Power BI assets that are moved to a different domain are deleted after synchronization.	When you run the harvesters, Collibra Data Lineage creates all Power BI assets in the same Data Catalog BI domain. We highly recommend that you do not move these assets to another domain. If you move assets to another domain, they will be deleted and recreated in the initial Data Catalog BI domain when you synchronize Power BI. As a consequence, all manually added data of those assets is lost.

Known issue	Description
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.
You are experiencing Power Query parsing errors when ingesting Power BI metadata.	The Power Query parser does not currently support parsing for: • The following functions: ◦ MicrosoftAzureConsumptionInsights.Tables ◦ Table.ExpandRecordColumn ◦ Dates Query • Transact-SQL statements

Supported data sources in Power BI (deprecated)

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.

Warning 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.

Power BI data source	Connection type	Technical lineage	Stitching to registered data sources in Data Catalog
Amazon Redshift	Import	Yes	Yes
Azure Databricks	Import	Yes	Yes
Google BigQuery	Import	Yes	Yes
ODBC	Import	Yes	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	Import	Yes	Yes
Snowflake	Import	Yes	Yes
SQL Server	Import	Yes	Yes
Sybase	Import	Yes	Yes

Note We cannot guarantee that other data sources in Power BI can be stitched successfully.

Providing ODBC database names in Power BI (deprecated)

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 [Power BI configuration file](#). If, for example, the value of the `sourceId` property in the Power BI 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>	found_dbname 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. found_hostname should be "ODBC", but you can also use an asterisk (*).

Property	Description
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 Power BI harvester fails to find a specific schema, it uses the default schema.
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> 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 useCollibraSystemName property in the Power BI configuration file to <code>true</code>. • Specify a Collibra system name in the <source ID> configuration file. However, if the useCollibraSystemName property is set to <code>true</code> in the Power BI configuration file, then you must specify a Collibra system name in the <source ID> configuration file.

For complete information on working with <source ID> configuration files, see [Power BI <source ID> configuration file](#).

Power BI prerequisites (deprecated)

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 [Power BI harvester](#) to reach your Power BI application and collect its metadata.

The tasks include the following:

- The [authentication process](#).
- The [registration](#) of your Power BI application in Microsoft Azure.
- The Power BI dedicated capacities and roles for [Power BI workspaces](#).

The [metadata harvesting process](#) explains in detail which prerequisites you need to enable the Power BI 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

Important Your [subscription](#) determines which Power BI metadata the Power BI harvester can collect.

You need one of the following subscriptions to ingest Power BI metadata in Data Catalog:

- Power BI Pro. To ensure a full ingestion, you also need a Power BI Embedded Capacity subscription in Microsoft Azure.
- Power BI Premium.

- Power BI Premium Per User.

Note The Power BI Premium Per User license is a new license type that is released for general availability, but is still in its preview period. For more information, see the [Microsoft documentation](#).

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

Authentication (deprecated)

You have to be authenticated to access Power BI metadata. Your authentication method determines how you [retrieve the metadata](#). The Power BI harvester supports two types of authentication:

- Username and password
- Service principal authentication

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

Note We recommend that you use the service principal authentication.

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, you need to be an Azure Active Directory user with a Power BI admin role in Power BI and have a Contributor role in the [Power BI workspaces](#) that you want to ingest into Data Catalog.

When you become an Azure Active Directory user, a new email address is created. You use this email address to sign in to Power BI.

The email address that is created in Microsoft Azure is the username that 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 [Power BI configuration file](#).

In the [Power BI Tenant settings in Power BI](#), you have to [enable](#) the **Allow XMLA endpoints and Analyze in Excel with on-premises datasets**. This setting has to be applied to the entire organization (default) or to the specific security group to which your [workspaces](#) belong.

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 lets an Azure Active Directory automatically access Power BI.

The Service Principal authentication relies on the Power BI Tenant ID and the Azure Active Directory application ID that you provide in the [configuration file](#). The password you need to access Power BI is the client secret key of the Azure Active Directory application.

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

- Create an Azure AD security group.
- Add the security group in the [Power BI Tenant settings in Power BI](#).
- In the [Power BI Admin portal](#), you also do the following :
 - [Enable](#) the **Allow service principals to use read-only Power BI admin APIs (preview)** option.
 - [Enable](#) the **Allow service principal to use Power BI APIs** option in the Developer settings.
 - Apply the option to specific security groups.
 - Enter the name of the security group to which you want to add the service principal.

- **Enable** the **Allow XMLA endpoints and Analyze in Excel with on-premises datasets**. This setting has to be applied to the entire organization (default) or to the specific security group to which your **workspaces** belong.

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

- Assign the Contributor role to the security group in the Power BI workspaces you want to ingest.

Tip Do not confuse the **Allow service principals to use read-only Power BI admin APIs (preview)** 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 (deprecated)

Before you set up the Power BI 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 may 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 [Power BI 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).
3. In the Azure portal, go to **Authentication** pane and do the following:
 - a. Go to the **Advanced settings** section.
 - b. Set the **Treat application as a public client** to *Yes*.

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](#)).
- » The user now has the following permissions:
 - Microsoft Graph
 - `User.Read`
 - 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`
 - `Tenant.Read.All`, with **Admin consent required** set to **Yes**.
 - `Workspace.Read.All`
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 Power BI admin APIs (preview)** option.
 - b. [Enable](#) the **Allow service principal to use Power BI APIs** option in the Developer settings.
 - c. Apply the option to specific security groups.
 - d. Enter the name of the security group to which you want to add the service principal.
 - e. [Enable](#) the **Allow XMLA endpoints and Analyze in Excel with on-premises datasets**.
 - f. Apply the integration setting to the entire organization (default) or to the specific security group to which your [workspaces](#) belong.

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):
 - a. **Enable** the **Allow XMLA endpoints and Analyze in Excel with on-premises datasets**.
 - b. Apply the integration setting to the entire organization (default) or to the specific security group to which your [workspaces](#) belong.

What's next?

You can add your [Power BI workspaces](#) to a dedicated capacity.

Power BI workspaces (deprecated)

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 Power BI harvester can access all metadata in your Power BI workspaces. Consider the following:

- 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.

Tip You can filter on Power BI workspaces in the [Power BI configuration file](#).

Roles and permissions

Depending on the authentication type that you want to use, you also require additional permissions in the Power BI workspaces to access the Power BI metadata.

- In case of [username / password authentication](#), the Azure Active Directory user with a Power BI admin role in Power BI must have the Contributor role in the Power BI

workspaces you want to ingest.

- In case of [Service Principal authentication](#), you have to add the Active Directory security group to which you added the Service Principal to your Power BI workspaces. The Power BI workspaces you want to ingest must have the Contributor role in the Power BI security group.

Ingesting 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.

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.

Viewing workspace states in Collibra

On Power BI Workspace asset pages, you can include the attribute type State, to show the state of ingested Power BI workspaces. 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](#).

The metadata harvesting process (deprecated)

Collibra uses two methods to harvest Power BI metadata: via REST API calls and via XMLA endpoints. The REST API retrieves basic metadata, and XMLA endpoints retrieve more specific metadata.

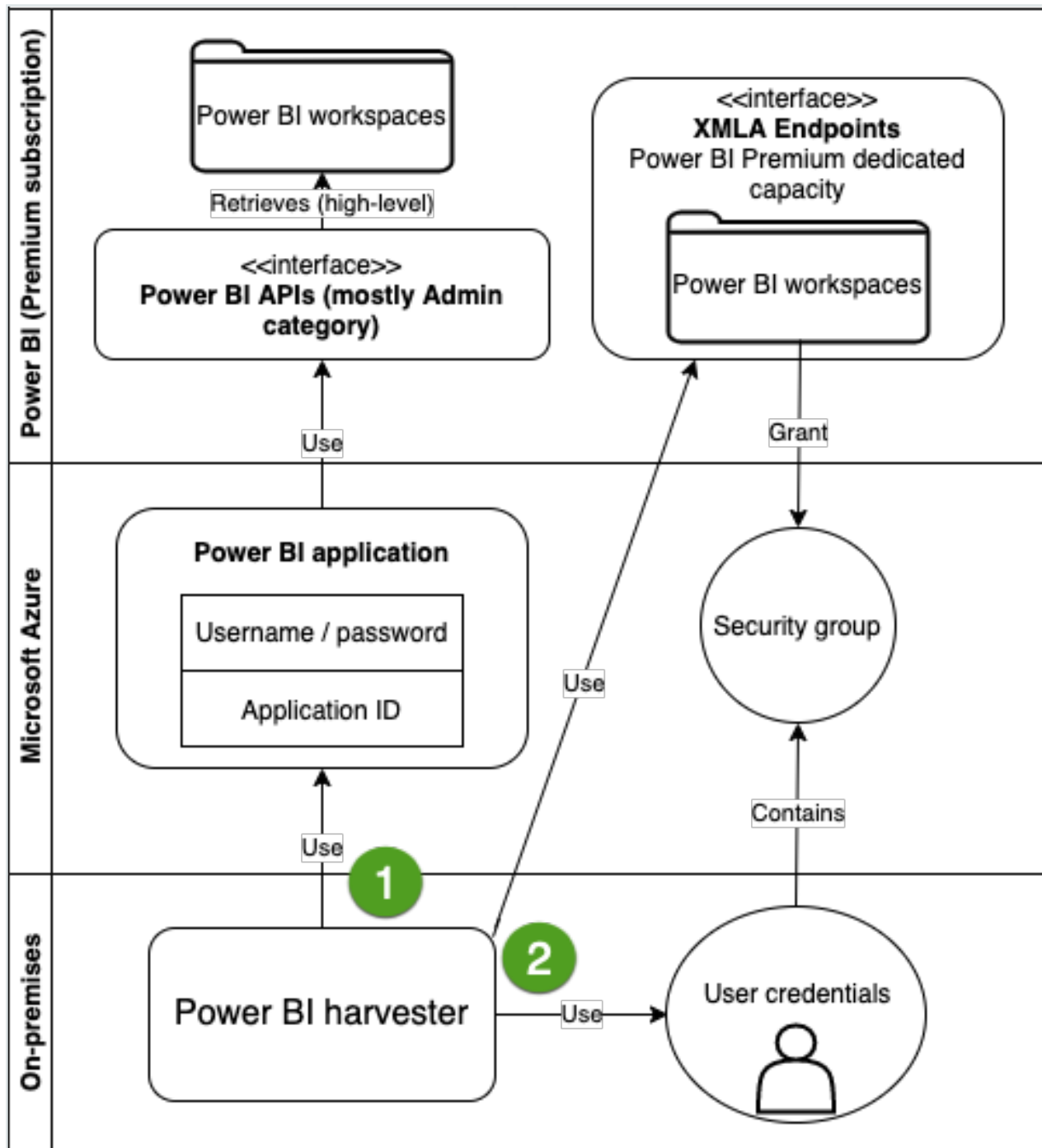
To enable the lineage harvester to access metadata in [Power BI workspaces](#), you must add the workspaces to a Power BI Premium dedicated capacity and have the correct configurations in [Microsoft Azure](#).

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

The following table shows which metadata the Power BI harvester retrieves and how.

Metadata about...	is retrieved using...
Reports	Microsoft Azure Admin Power BI REST API calls .
Data set columns and lineage	XMLA (Queries or M-queries) endpoints

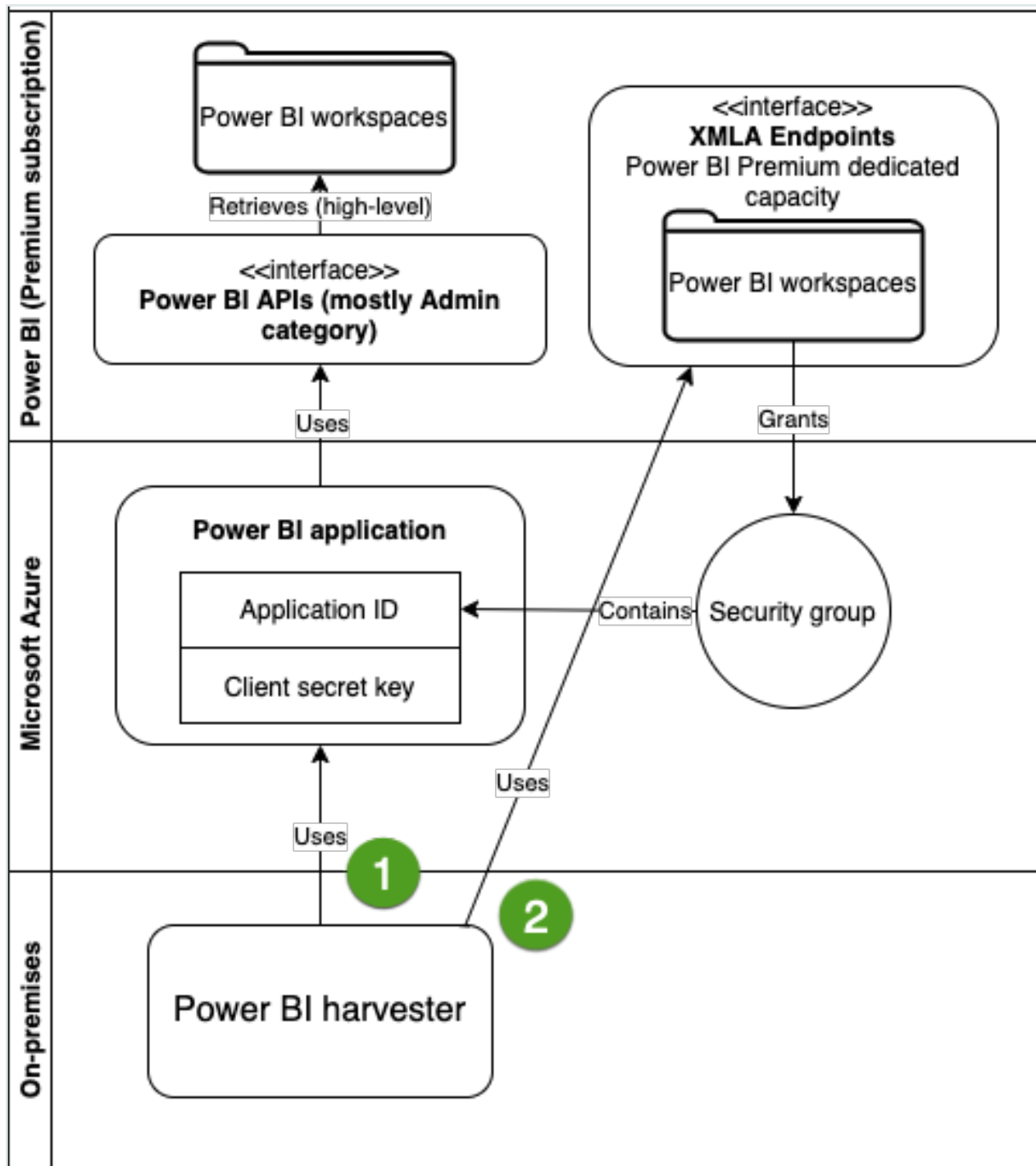
Overview of the metadata harvesting process with username / password authentication



Step	Retrieved via	Description
1	Power BI API calls	The Power BI 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	XMLA	You add the Azure Active Directory user with a Power BI admin role in Power BI to a security group and grant him the Contributor role in Power BI workspaces . You add the Power BI workspaces that you want to ingest to the same security group. As a result, the Power BI harvester uses XMLA endpoints to retrieve more specific metadata, for example Power BI columns and lineage. Specific metadata from Power BI workspaces is only harvested if you added the Power BI workspaces to the Power BI dedicated capacity and you have the necessary permissions to harvest the metadata..

Note Make sure that all necessary [dedicated capacities](#) are running and accessible to the Power BI harvester. If not, creating assets for Power BI data sets and your technical lineage may fail.

Overview of the metadata harvesting process with service principal authentication



Step	Retrieved via	Description
1	Power BI API calls	The Power BI 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	XMLA	You add the service Principal to a security group and grant it the Contributor role in the Power BI workspaces . As a result, the Power BI harvester uses XMLA endpoints to retrieve more specific metadata, for example in Power BI columns and lineage. Specific metadata from Power BI workspaces is only harvested if you add the Power BI workspaces to the dedicated capacity and you have the necessary permissions to harvest the metadata.

Note Make sure that all necessary [dedicated capacities](#) are running and accessible to the Power BI harvester. If not, creating assets for Power BI data sets and your technical lineage may fail.

Prepare a domain for Power BI ingestion (deprecated)

You can create a new domain for your [Power BI asset](#) and use the domain ID in the [Power BI harvester configuration file](#). As a result, Collibra uses this domain to ingest all Power BI assets during the [Power BI integration](#) process.

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 a domain type from the list.

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
Type	The domain type of the domain you are creating. In this case, you need to select <i>BI Catalog</i> .
Community	The community under which the domain will be located.
Name	The name of the new domain.

5. Click **Create**.
6. Open your domain.
7. 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.

8. Paste the domain ID in the Power BI [configuration file](#).

Warning When you run the harvesters, Collibra Data Lineage creates all Power BI assets in the same Data Catalog BI domain. We highly recommend that you do not move these assets to another domain. If you move assets to another domain, they will be deleted and recreated in the initial Data Catalog BI domain when you synchronize Power BI. As a consequence, all manually added data of those assets is lost.

Power BI and lineage harvester set-up (deprecated)

To ingest Power BI metadata in Data Catalog, you need to run two different harvesters:

- The [Power BI harvester](#)
- The [lineage harvester](#).

The order in which you run the harvesters is important. You first have to run the Power BI harvester to collect the metadata from your Power BI application and then run the lineage harvester to import new Power BI assets and their relations in Data Catalog. The [Power BI ingestion workflow](#) explains which roles the harvesters play in the Power BI ingestion process.

Note You need to purchase the Power BI metadata connector and lineage feature to access the Power BI and lineage harvesters.

Warning If you upgrade from Power BI harvester 1.0.0.0 to Power BI harvester 1.0.0.1 or newer, you have to follow an [upgrade procedure](#).

Power BI ingestion workflow (deprecated)

To ingest Power BI metadata into Data Catalog, you use two types of harvesters:

- A [Power BI harvester](#)
- A [lineage harvester](#)

Note The harvesters can run on the same or on different machines. However, the Power BI harvester must run on a Windows machine.

When the Power BI harvester initiates the Power BI integration, each workflow component performs the following actions:

1. The Power BI harvester:
 - Communicates with Power BI.
 - Harvests Power BI metadata for ingestion and lineage.
 - Sends the Power BI metadata to the Collibra cloud environment.

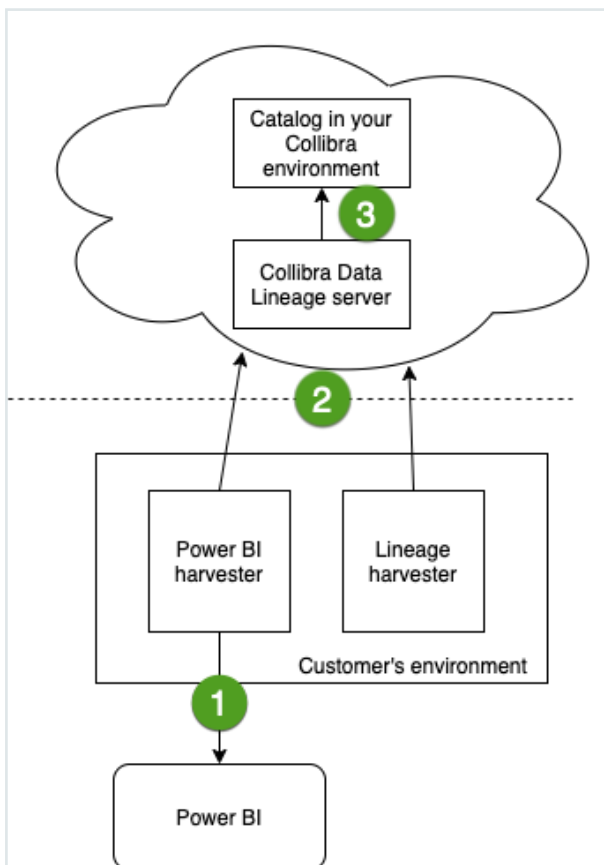
Note The Power BI harvester only harvests the metadata, it does not change it.

2. The lineage harvester:

- Triggers a new synchronization of the metadata in Collibra to create a technical lineage for Power BI and new relations between Power BI assets.
- Sends the Power BI ingestion results to Data Catalog.
- Sends the lineage results to Data Catalog.

3. Data Catalog (via the [Collibra Data Lineage server](#)):

- Shows the new [Power BI assets](#).
- Shows a [Technical lineage tab](#) on Power BI Column pages.



Collibra Data Lineage servers (deprecated)

A Collibra Data Lineage server processes and analyzes the [harvested metadata](#) and uploads it to Data Catalog. Collibra Data Lineage servers never process actual data.

Based on your geographical location and cloud provider, the [Power BI harvester](#) sends metadata to one of the following Collibra Data Lineage servers:

- 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)

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.

Set up the Power BI harvester (deprecated)

The Power BI harvester is a standalone console application that runs on a Windows machine. You use it to extract data from the [Power BI REST API](#) and [XMLA endpoints](#) and send it to the Collibra Data Lineage server in Collibra's cloud environment for analysis.

The [metadata harvesting process](#) explains in detail which [prerequisites](#) you need to enable the Power BI 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.

Power BI harvester system requirements (deprecated)

You need to meet the following system requirements to install and run the [Power BI harvester](#) on your Windows machine.

Note If you want to successfully ingest Power BI metadata into Data Catalog, you need to meet both the system requirements to run the Power BI harvester, and also the [system requirements](#) to run the [lineage harvester](#).

Software requirements

You need to meet the software requirements to install and run the Power BI harvester.

Minimum software requirements

You need the following minimum software requirements:

- Microsoft .NET Framework 4.7.2.
- One of the following:
 - Client operating system: Windows 7 SP1, 8.1 or 10, version 1607.
 - Server operating system: Windows Server 2008 R2 SP1.

Note .NET Framework 4.7.2 is available as a system update.

Recommended software requirements

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

- Microsoft .NET Framework 4.7.2 or higher.
- Client operating system: Windows 10 April 2018 update, version 1803 or newer.
- Server operating system: Windows Server 2016 version 1803 or newer.

Hardware requirements

You need to meet the hardware requirements to install and run the Power BI 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 you meet the following hardware requirements:

- 4 GB RAM
- 20 GB free disk space

Network requirements

You have firewalls rule to have access to:

- The Microsoft API.
- A Collibra Data Lineage server with IP address:
 - 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 Power BI harvester connects to different servers based on your geographic location and cloud provider. If your location or cloud provider changes, the Power BI harvester rescans all your Power BI metadata. You have to allow all Collibra Data Lineage servers in your geographic location. In addition, we highly recommend that you always allow the techlin-aws-us server as a backup, in case the Power BI harvester cannot connect to other Collibra Data Lineage servers.

Note The Power BI harvester uses port 443 (tcp only).

Install the Power BI harvester (deprecated)

Before you can use the Power BI harvester, you need to download it and install it on your Windows machine. You can download the Power BI harvester from the Collibra Product Resource Center downloads page.

Warning If you upgrade to Power BI harvester 1.0.0.1 or newer, you have to follow an [upgrade procedure](#).

Prerequisites

- You have Collibra Data Intelligence Cloud 2020.11 or newer.
- You have access to the Power BI harvester on the Downloads page.
- Your environment meets the [system requirements](#) to install and use the Power BI harvester.
- You have added Firewall rules so that the Power BI harvester can connect to the Collibra Data Lineage server with one of the following IP addresses:
 - 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)

Steps

1. [Download](#) the Power BI harvester.
2. Unzip the archive.
3. Open the Power BI harvester folder.
 - » The Power BI harvester folder contains two folders, a BAT file that you use to run the harvester and a TXT file with information about the configuration file.
 - » An empty [Power BI configuration file](#) is available in the **config** folder.

What's next?

You can now [prepare](#) the Power BI connection properties in the configuration file and run the Power BI harvester.

Note We highly recommend that you run the Power BI harvester via command line. This enables you to follow the metadata upload and see possible errors that may occur.

Prepare the Power BI configuration file (deprecated)

You create a configuration file for the [Power BI](#) metadata that you want to ingest. This configuration file is used by the [Power BI harvester](#) to retrieve metadata from Power BI and send it to Collibra to be scanned, processed and analyzed.

Prerequisites

- You have access to the Power BI harvester on the Downloads page.
- You have completed all [prerequisite tasks](#).
- You have a dedicated [domain](#) to ingest the Power BI assets.

- 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](#).
- You have a [global role](#) with the Data Stewardship Manager [global permission](#).
- 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
- Your environment meets the [system requirements](#) to run the Power BI harvester and the lineage harvester.

Tip For a full ingestion, we highly recommend to have a [Power BI Premium subscription](#).

Steps

1. In the Power BI harvester folder, open the empty configuration file.
2. Enter the values for each property.

Properties	Description	Mandatory
powerbi	This section contains information that is necessary to connect to your Power BI application.	Yes
tenantDomain	The Power BI tenant domain is the domain associated with the Microsoft Azure tenant. This domain is either a default domain or a custom domain. For example, <i>collibrapowerbi.onmicrosoft.com</i>. Note Usually, you can find a list of Power BI tenant or server domains in your Azure Active Directory or in the top right menu.	Yes
applicationId	The unique ID of the Microsoft Azure Application (client) ID .	Yes

Properties	Description	Mandatory
userName	The username that you use to access Power BI. Depending on your authentication type, the username should have a different value: - For username and password authentication, you enter the username that you use when you sign in to Power BI. - For Service Principal authentication, you leave this field empty. Tip If you cannot store your username in the configuration file for security or other reasons, delete this field and provide the username via command line or when prompted by the Power BI harvester.	No
password	The password or client secret key that you use to access Power BI. Depending on your authentication type, the password needs a different value: - For username and password authentication, you enter the password that you use when you sign in to Power BI. - For Service Principal authentication, you enter the Power BI application client secret key. In case the password is an empty string, leave this field empty. Tip If you cannot store your password in the configuration file for security or other reasons, delete this field and provide the password via command line or when prompted by the Power BI harvester.	No

Properties	Description	Mandatory
workspaceFilter	An option to exclude specific Power BI workspaces from the ingestion process. You can add multiple workspaces. For example "workspace1, workspace2, workspace3". If the workspaceFilter field remains empty or is deleted from the configuration file, all accessible Power BI workspaces are processed and ingested. Tip For more information about the query options to filter Power BI workspaces, see the Microsoft documentation. Be aware that the "IN" operator is currently not supported. Important If you use Power BI harvester older than version 1.1.0.0, the workspaceFilter property is named groupFilter. This change is backward compatible. However, if you download a new Power BI harvester, we highly recommend to update your configuration file.	No
techlin	This section contains information to identify your Power BI metadata on the Collibra Data Lineage server.	Yes
sourceId	The unique ID of your Power BI metadata. The lineage harvester uses this ID to locate the Power BI metadata on the Collibra Data Lineage server. Tip This value can be anything as long as it is a unique, human readable ID and the same as the value of the <code>Id</code> property in the lineage harvester configuration file. The Power BI and lineage harvesters use the ID to identify a batch of data on the Collibra Data Lineage server.	Yes
catalog	This section contains information that is necessary to connect to Data Catalog.	Yes

Properties	Description	Mandatory
domainId	The unique resource ID of the domain 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
url	The URL of your Collibra Data Intelligence Cloud instance. Note You can only enter the public URL of your Collibra Data Intelligence Cloud environment. Other URLs will not be accepted.	Yes
userName	The username that you use to sign in to Collibra. Tip If you cannot store your username in the configuration file for security or other reasons, delete this field and provide the username via command line or when prompted by the Power BI harvester.	No
password	The password that you use to sign in to Collibra. Tip If you cannot store your password in the configuration file for security or other reasons, delete this field and provide the password via command line or when prompted by the Power BI harvester.	No

Properties	Description	Mandatory
useCollibraSystemName	Indication whether 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 useCollibraSystemName property is set to <code>false</code>. If you want to use it, set it to <code>true</code>. ◦ If you set the useCollibraSystemName property to <code>true</code>, the Power BI harvester reads the <source-ID> configuration file and takes the value in the collibraSystemName property into account. ◦ If you set the useCollibraSystemName property to <code>false</code>, the Power BI harvester ignores the collibraSystemName property in the <source-ID> configuration file. Warning Unless you have multiple databases with the same name, we highly recommend that you keep the default value.	Yes

3. Save the configuration file.
4. Trigger the Power BI harvester to upload the Power BI metadata:
 - Run the following command line if your configuration file is in its default location: `.\powerbi-harvester.bat`
 - Launch the path to the Power BI configuration file if you moved the configuration file to a different location: `.\bin\powerbi-harvester.exe .\config\powerbi-harvester.conf`

Note We highly recommend that you run the Power BI harvester via command line. This enables you to follow the metadata upload and see possible errors that may occur.

5. If the Power BI harvester prompts for credentials, enter them or use [command line options](#) to provide them.

Note Credentials provided via command line overwrite the credentials in the configuration file.

» The Power BI harvester collects the Power BI metadata and sends it to the [Collibra Data Lineage server](#). Collibra scans and analyzes the metadata.

Tip If you want to [ingest multiple Power BI applications](#), create a new configuration file using a unique ID and repeat these steps. In the [lineage harvester configuration file](#), you can add multiple Power BI sections that each refer to a different ID.

Note If you are not able to run the Power BI harvester, go to the [troubleshooting](#) section to resolve your issues.

Example

This example shows a configuration file with the [username / password authentication method](#).

```
{
  "powerbi": {
    "tenantDomain": "<organization.onmicrosoft.com>",
    "applicationId": "<microsoft-azure-id>",
    "userName": "<your-power-bi-email-address>",
    "password": "<password-to-access-power-bi>",
    "workspaceFilter": "workspace-name1", "workspace-name2"
  },
  "techlin": {
    "sourceId" : "<unique-power-bi-ID>"
  },
  "catalog": {
    "domainId": "<your-catalog-domain>",
    "url": "<url-to-collibra>",
    "userName": "<my-collibra-username>",
    "password": "<my-collibra-password>"
  },
  "useCollibraSystemName": false
}
```

This example shows a configuration file with the [service principal authentication method](#).

```
{
  "powerbi": {
    "tenantDomain": "<organization.onmicrosoft.com>",
    "applicationId": "<microsoft-azure-id>",
    "userName": "",
    "password": "<secret-key>",
    "workspaceFilter": "<filter-workspace-name>"
  },
  "techlin": {
    "sourceId" : "<unique-power-bi-ID>"
  },
  "catalog": {
    "domainId": "<your-catalog-domain>",
    "url": "<url-to-collibra>",
    "userName": "<my-collibra-username>",
    "password": "<my-collibra-password>"
  },
  "useCollibraSystemName": false
}
```

Warning If you are ingesting a large amount of Power BI data and you use the workspace filter (workspaceFilter), the Power BI harvester might time out, resulting in an Internal Server Error. If you get this error, we highly advise you to not use the workspace filter. See the known issues in [Power BI ingestion limitations](#).

What's next?

You can now [download and install](#) the lineage harvester and [prepare](#) the lineage harvester configuration file. The lineage harvester triggers Collibra to create new Power BI assets, stitch them and show a technical lineage for them.

To refresh the Power BI metadata in Data Catalog, you can run the Power BI harvester and lineage harvester again or [schedule jobs](#) to run them automatically.

Prepare Power BI <source ID> configuration file (deprecated)

The Power BI harvester uses a [Power BI configuration file](#) to collect the Power BI data objects. It then sends the metadata to the [Collibra Data Lineage server](#). However, if the `useCollibraSystemName` in the Power BI configuration file is set to `true`, you also have to provide a specific <source ID> configuration file that defines 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.

Tip The name "<source ID>" refers to the value of the `sourceId` property in the Power BI configuration file.

Prerequisites

- The `useCollibraSystemName` in the [Power BI harvester configuration file](#) is set to `true`.

Note This is not a prerequisite if you are using a <source ID> configuration file for the purpose of [providing the true system names](#) of the ODBC databases in Power BI. In that case, you can set the `useCollibraSystemName` property in the [Power BI harvester configuration file](#) to `true`, but it is not mandatory.

Steps

1. Create a new JSON file in the Power BI harvester **config** folder.
2. Give the JSON file the same name as the value of the `sourceId` property in the Power BI configuration file.

Example The value of the `sourceId` property in the Power BI configuration file is `power-bi-source-1`. Therefore, the name of your JSON file should be *power-bi-source-1.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>	The database information of supported data sources in Power BI that is typically collected by the Power BI harvester. It describes on which server the database is running (found_hostname) and what the name of the database is (found_dbname). 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".											
dbname	The name of the database of a supported data source in Power BI.	No										
schema	The name of the default schema of a supported data source in Power BI. If the Power BI harvester fails to find a specific schema, it uses the default schema.	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
collibraSystemName	The system or server name of a database. Warning The value of this property must exactly match the name of your System asset in Collibra. Important If you are using a <source> 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 useCollibraSystemName property in the Power BI configuration file to <code>true</code>. ◦ Specify a Collibra system name in the <source ID> configuration file. However, if the useCollibraSystemName property is set to <code>true</code> in the Power BI configuration file, then you must specify a Collibra system name in the <source ID> configuration file.	Yes (unless you are using a <source ID> file to provide the true system names of ODBC databases in Power BI.)

4. Save the <source ID> configuration file.

Example of the <source ID>.conf file

```
{
  "found_dbname=databasename1;found_hostname=*": {
    "dbname": "mssql-database-name",
    "schema": "mssql-schema-name",
    "dialect": "mssql",
    "collibraSystemName": "mssql-system-name"
  },
  "found_dbname=databasename2;found_hostname=server-name.on-
microsoft.com": {
    "dbname": "oracle-database-name",
    "schema": "oracle-schema-name",
    "dialect": "oracle",
    "collibraSystemName": "oracle-system-name"
  }
}
```

Ingest multiple Power BI applications (deprecated)

You can ingest more than one Power BI application in Collibra. For each Power BI application, you create a separate [Power BI configuration file](#), and then add a section in the [lineage harvester configuration file](#).

Prerequisites

- You have access to the Power BI harvester on the Downloads page.
- You have completed all [prerequisite tasks](#).
- You have a dedicated [domain](#) to ingest the Power BI assets.
- 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](#).
- You have a [global role](#) with the Data Stewardship Manager [global permission](#).
- 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
- Your environment meets the [system requirements](#) to run the Power BI harvester and the lineage harvester.

Tip For a full ingestion, we highly recommend to have a [Power BI Premium subscription](#).

Steps

1. [Prepare the Power BI configuration file](#) for one Power BI application.
2. Run the Power BI harvester.
3. For each additional Power BI application, do the following:
 - a. Prepare a new configuration file with the information of the next Power BI application.
 - i. Optionally, [create a new domain](#) in Data Catalog to ingest the assets of this Power BI application.
 - ii. Enter a new source ID that is different from the source IDs of existing Power BI configuration files.
 - b. Run the Power BI harvester again.

Note Make sure that you refer to the path of this configuration file when you run the Power BI harvester.

- » The Power BI harvester collects the Power BI metadata of each Power BI application and sends it to the [Collibra Data Lineage server](#).
 - » Collibra scans and analyzes the metadata.
4. In the [lineage harvester configuration file](#), create a Power BI section for each Power BI application. Use the source ID of each Power BI configuration file as the ID of the Power BI section in the lineage harvester configuration file.
 5. Run the lineage harvester to ingest the Power BI metadata in Collibra.
 - » The Power BI metadata is ingested in the domain that you specified in the Power BI configuration file.

Example

You have two Power BI applications that you want to ingest. The first Power BI configuration file has source ID `power-bi-app-a`, the second Power BI configuration file has source ID `power-bi-app-b`. The lineage harvester configuration file contains two Power BI sections that each refer to a different source ID.

```
{
  "general": {
    "catalog" : {
      "url" : "https://companydomain.collibra.com",
      "username" : "my-Collibra-username"}
  },
  "sources" : [
    {
      "type" : "ExistingLineage",
      "id" : "power-bi-app-a"
    }
    {
      "type" : "ExistingLineage",
      "id" : "power-bi-app-b"
    }
  ]
}
```

What's next?

To refresh the Power BI metadata in Data Catalog, you can run the Power BI harvester and lineage harvester again or [schedule jobs](#) to run them automatically. You can schedule to synchronize Power BI applications at different times.

Command options and arguments (deprecated)

After creating a Power BI harvester [configuration file](#), you can use the command line to provide the Power BI harvester with additional information or perform specific actions.

Note Credentials provided via command line overwrite the credentials in the [configuration file](#).

Typical command options and arguments

The following table shows the most commonly used command options and arguments.

Command	Description
<code>--catalog-password "<Collibra password>"</code>	Your Collibra password. If you don't want to add your password in the Power BI harvester configuration file, you can provide it via command line. Note If you added an API key, the Data Catalog credentials will not be used.
<code>--catalog-user "<Collibra username>"</code>	Your Collibra username. If you don't want to add your password in the Power BI harvester configuration file, you can provide it via command line. Note If you added an API key, the Data Catalog credentials will not be used.
<code>--output-file <file path></code>	Save your harvested Power BI metadata to a specified directory. The output file is a ZIP file.
<code>--from-file <file path></code>	Upload Power BI metadata that was already harvested and saved to a specified file.
<code>--timeout <seconds></code>	Increase the timeout duration to specify a longer timeout for remote API calls. Example If you want the Power BI harvester to wait 15 minutes before canceling a remote API connection, you can use <code>--timeout 900</code>.

Set up the lineage harvester for Power BI ingestion (deprecated)

The lineage harvester is a software application that is needed to collect your Power BI metadata and send it to the Collibra Data Lineage server, where the metadata is

processed and a technical lineage and new [Power BI assets and relations](#) are created. Collibra Data Intelligence Cloud then import those assets and relations into Data Catalog.

For more information about the lineage harvester, read the [Collibra Data Lineage](#) documentation.

Note You need the lineage harvester 1.2.1 or newer to ingest Power BI metadata into Data Catalog.

Lineage harvester system requirements (deprecated)

You need to meet the system requirements to be able to [install](#) and run the [lineage harvester](#).

Software requirements

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

For Java Runtime Environment 16 or newer, or OpenJDK 16 or newer, set the `JAVA_OPTS` environment variable for the lineage harvester to function properly:

```
JAVA_OPTS='--illegal-access=deny'
```

Note To ingest Snowflake data sources, the minimum requirement is Java Runtime Environment version 16 or newer, or OpenJDK 16 or newer.

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 [Technical lineage general troubleshooting](#).

- 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:
 - Collibra Data Intelligence Cloud.
 - 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 Power BI 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 Power BI harvester rescans all your Power BI metadata. 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 Power BI harvester cannot connect to other Collibra Data Lineage service instances.

Install the lineage harvester for Power BI ingestion (deprecated)

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

Tip

- Install the lineage harvester close to your data source or on the same server.
- The lineage harvester uses port 443.

Prerequisites

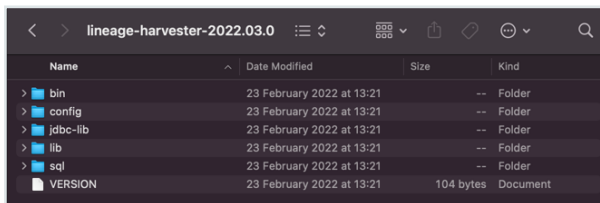
- You have purchased the Power BI metadata connector and lineage feature.
- You have Collibra Data Intelligence Cloud 2020.11 or newer.
- You meet the [minimum system requirements](#).
- You have added Firewall rules so that the lineage harvester can connect to:
 - The host names of all databases in 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.

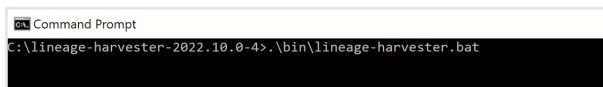
Steps

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.

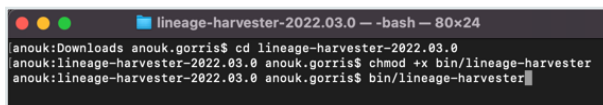


3. Start the lineage harvester to create an empty lineage harvester configuration file by entering the following command:

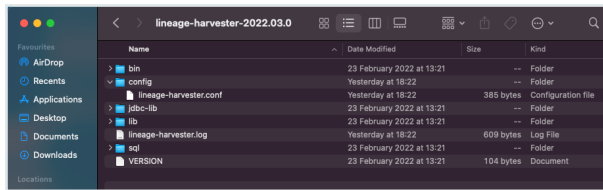
- **Windows:** `.\bin\lineage-harvester.bat`



- **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?

You can now [prepare](#) the lineage harvester configuration file.

Prepare the lineage harvester configuration file for Power BI (deprecated)

You have to prepare a technical lineage configuration file and run the [lineage harvester](#) to fetch the Power BI analysis results on the Collibra Data Lineage server and sent them as an import job to your Collibra Data Intelligence Cloud.

Note Comments in the lineage harvester configuration file are not supported.

Tip For more information, see [Collibra Data Lineage](#).

Prerequisites

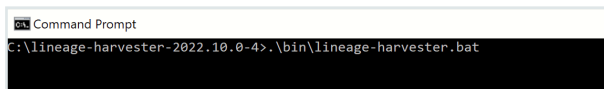
- You have prepared the [Power BI configuration file](#) and executed the Power BI harvester.
- You have a [global role](#) that has the Manage all resources [global permission](#).
- You have a [global role](#) with the Catalog [global permission](#), for example, Catalog Author.
- You have the Technical lineage global permission.
- You have [created a BI Catalog domain](#) in which you want to ingest the Power BI assets.

- 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
- You have **downloaded** lineage harvester version 2022.05 or newer. We highly recommend that you always install and use the newest lineage harvester.

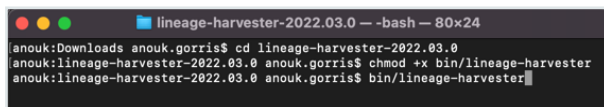
Steps

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

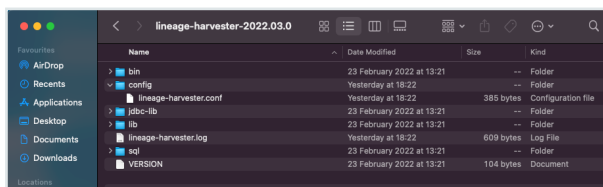
- **Windows:** `.\bin\lineage-harvester.bat`



- **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. Open the configuration file and enter the values for each property.

Properties	Description
general	This section describes the connection information between the lineage harvester and Data Catalog.
catalog	This section contains information that is necessary to connect to Data Catalog.

Properties	Description
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.
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.
type	The kind of data source. In this case, the value has to be <i>ExistingLineage</i> .
id	The unique ID to identify the Power BI service metadata that was uploaded to the Collibra Data Lineage server. The value has to be the same as the value you used in the <code>sourceId</code> property in the Power BI configuration file. Tip This value can be anything as long as it is a unique ID and the same as the value of the <code>sourceId</code> property in the Power BI configuration file. The Power BI and lineage harvesters use the ID to identify a batch of data on the Collibra Data Lineage server.

Tip If you want to [ingest multiple Power BI applications](#), create a separate [Power BI configuration file](#) for each Power BI application each with a unique source ID. Duplicate the Power BI section in the lineage harvester configuration file and enter the source ID in the ID property.

3. Save the configuration file.

4. Start the lineage harvester again in the console and run the following command:
 - for Windows: `.\bin\lineage-harvester.bat full-sync`
 - for other operating systems: `./bin/lineage-harvester full-sync`
5. When prompted, enter the passwords to connect to your Collibra Data Intelligence Cloud environment.
 - » The password is encrypted and stored in `/config/pwd.conf`

What's next?

The lineage harvester triggers Collibra to import [Power BI assets](#) and their relations and create a [technical lineage](#) for Power BI Column assets. Collibra also [stitches](#) the new Power BI assets to existing assets in Data Catalog.

To refresh the Power BI metadata in Data Catalog, you can run the Power BI harvester and lineage harvester again or [schedule jobs](#) to run them automatically.

Tip You can check the progress of the Power BI ingestion and technical lineage creation in [Activities](#). The **Results** field indicates how many relations were imported into Data Catalog.

Warning When you run the harvesters, Collibra Data Lineage creates all Power BI assets in the same Data Catalog BI domain. We highly recommend that you do not move these assets to another domain. If you move assets to another domain, they will be deleted and recreated in the initial Data Catalog BI domain when you synchronize Power BI. As a consequence, all manually added data of those assets is lost.

Power BI business logic (deprecated)

Power BI business users work with Power BI dashboards and reports to make business decisions. Collibra's [Power BI](#) connector and lineage feature offers business users several advantages:

- Easily find certified Power BI content.
- Shop for Power BI reports.
- Trace Power BI metadata to metadata of other data sources.

- Find where content is stored in Power BI.
- Get information about a Power BI Report in a single location.

Power BI asset pages

Depending on the [Power BI asset type](#), the asset page shows different [information ingested from Power BI](#). You can find a specific Power BI asset page using [Data Catalog search](#) or via the Data Catalog BI domain in which you ingested the Power BI metadata.

Details

Asset pages show attributes and relations to other assets. This information is synchronized with the Power BI service. However, you can add additional characteristics, tags or comments.

If you want to use a Power BI Data Model or a Power BI Report, you can add it to the Data Basket and check it out.

Example The following Power BI Report asset page shows in which Power BI Workspace the report is stored and which Power BI Data Set it uses. This asset has a clear description and is certified.

The screenshot shows the 'Power BI demo report' asset page. The page has a header with the PBRT logo, the report name, and a status bar indicating it is a 'Power BI Report' and 'Candidate'. There are buttons for 'Add to Data Basket' and 'Actions'. A left sidebar contains navigation options: 'Details' (selected), 'Tags (1)', 'Comments', 'Diagram', 'Pictures', 'Responsibilities', 'References', 'History', and 'Files'. The main content area shows the 'Description' (Power BI demo report is a report created for demo purposes.), 'Certified' status (checked), and a section 'is grouped into Business Dimension' with a table listing related assets. Below this is a 'source BI Data Set' section with another table. The 'Tags' section shows 'approved', and the 'Comments' section is empty.

Name ↑	Domain	Description
Power BI Workspace for demos	test	

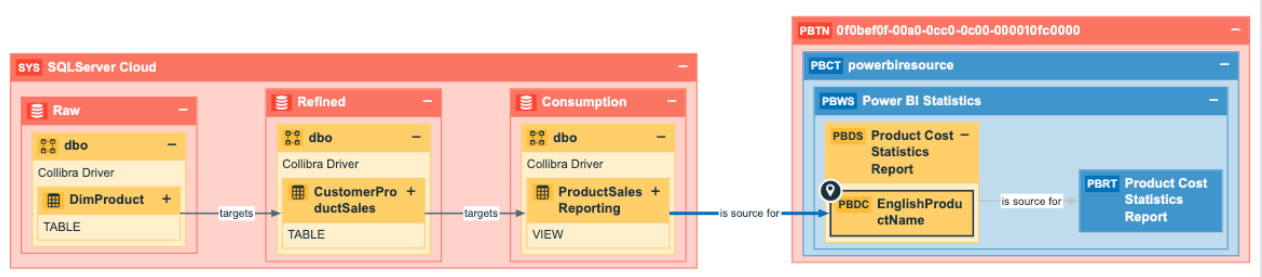
Name ↑	Domain	Description
Demo-data-set	test	

Business diagrams

The [business diagram](#) is a feature to show and interact with many assets and relations in an easy-to-read diagram. The business diagram helps you to quickly see to which other assets a specific asset is related. As such, the diagram can show a high-level presentation of a Power BI Report. This enables you, for example, to see:

- In which Power BI Workspace the Power BI Report is stored.
- In which Power BI Capacity the Power BI Workspace is stored.
- Which Power BI Data Model assets the Power BI Report uses.
- Which Table assets and Column assets from other data sources are the source of a Power BI Column asset.

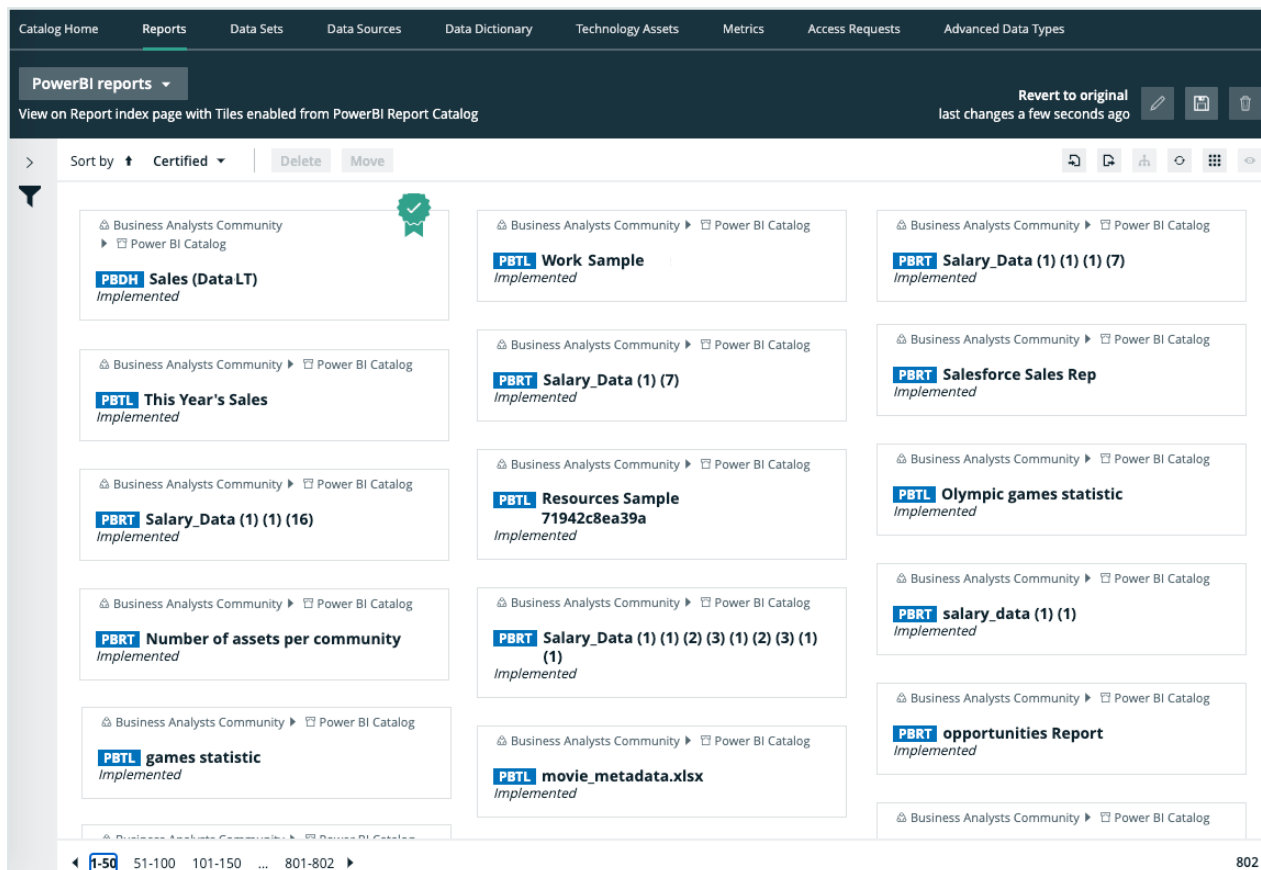
Example The following business diagram shows the *Product Cost Statistics Report* Power BI Report, which is stored in the *Power BI Statistics* Power BI Workspace. The report uses the *Product Cost - Statistics Report* Power BI Data Model. This data set contains data from the *SQL Server Cloud* source.



Report views

The Power BI connector and lineage feature enables you to find all ingested Power BI Reports and children of the Power BI Report asset type in a single location.

In the **Reports** tab page in Data Catalog you can see an overview of all Report assets and their children. Optionally, you can [create a view](#) with a filter to only show Power BI Reports. This is useful if you quickly want to find a report or if you want to know which reports are certified.



Technical lineage for Power BI service (deprecated)

When you ingest [Power BI](#) metadata in Data Catalog, you automatically create a technical lineage for Power BI Column assets. Each Power BI Column [asset page](#) has a Technical lineage tab page that shows the technical lineage of that Power BI Column asset.

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.

The screenshot shows the 'Power BI Demo Catalog' interface. The main header displays 'PBCL FullName' and 'Power BI Column'. Below this, a sidebar on the left contains navigation options: 'Details' (with sub-items 'Tags' and '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, there are two sections: 'Is target of Column' and 'Is part of Data Entity'. Each section contains a table with columns 'Name', 'Domain', and 'Description'.

Is target of Column		
Name ↑	Domain	Description
FullName	Consumption	

Is part of Data Entity		
Name ↑	Domain	Description
CustomerSalesReporting	Power BI Demo Catalog	

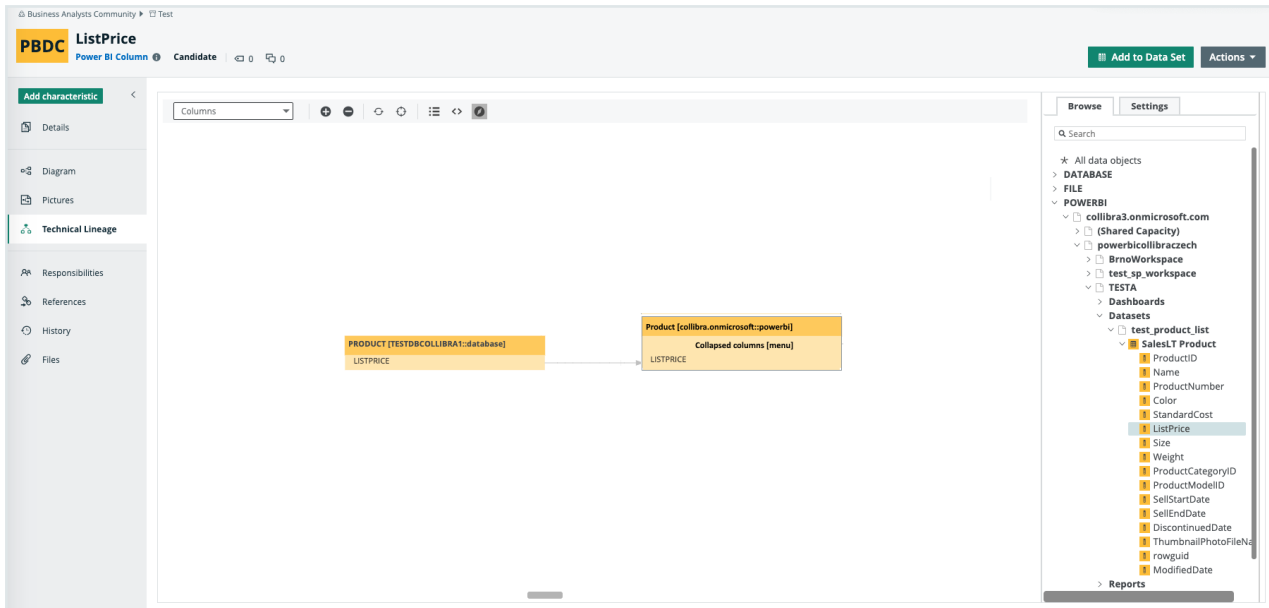
Technical lineage graph

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](#).

For more information about the technical lineage, see the [Collibra Data Lineage](#) section in the user guide.

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*.



Sources tab page

The Sources tab page shows the transformation details that were analyzed and processed on the [Collibra Data Lineage server](#) and the results of this analysis. The success rate of the analysis indicates how complete the technical lineage is. There are a few [limitations](#) that prevent the Collibra Data Lineage server from processing all Power BI metadata.

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 Sources tab page 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 server 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

Schedule jobs (deprecated)

You can use scheduled jobs to run the Power BI harvester and lineage harvester at specific times automatically.

Since you need both the Power BI harvester and the lineage harvester to successfully ingest Power BI metadata in Data Catalog, we highly recommend that you schedule the Power BI harvester job before you schedule the lineage harvester job.

Warning When you run the harvesters, Collibra Data Lineage creates all Power BI assets in the same Data Catalog BI domain. We highly recommend that you do not move these assets to another domain. If you move assets to another domain, they will be deleted and recreated in the initial Data Catalog BI domain when you synchronize Power BI. As a consequence, all manually added data of those assets is lost.

Schedule Power BI harvester jobs

You can use the Windows [Task Scheduler](#) to make the [Power BI harvester](#) run scheduled jobs periodically. In a scheduled job, the Power BI harvester automatically uploads Power BI metadata to Collibra.

Scheduled jobs only work if you add the correct credentials to the Power BI [configuration file](#) or if you use a tool to automatically provide the credentials each time the Power BI harvester job is scheduled.

Schedule lineage harvester jobs

You can use [Task Scheduler](#) on Windows or [Crontab](#) on Mac and Linux to make the [lineage harvester](#) run scheduled jobs periodically. In a scheduled job, the lineage harvester uploads Power BI metadata to your Collibra Data Intelligence Cloud environment and Data Catalog automatically creates new Power BI assets and relations at specific times, dates or intervals. Collibra also creates a [technical lineage](#) for Power BI Column assets.

Warning Relations that were manually created between Power BI assets and other assets via a relation type in the [Power BI operating model](#), are deleted after a refresh of the Power BI metadata.

Example You created a Power BI configuration file and added the required properties to the lineage harvester configuration file. You schedule the Power BI harvester job each Monday at 6 am and the lineage harvester job at 6 pm. As a result, your Power BI metadata is automatically refreshed on a weekly basis.

Harvesters upgrade (deprecated)

Each new [Power BI harvester](#) and [lineage harvester](#) adds features and enhancements to the previous version. We highly recommend that you always use the newest harvester available.

Upgrade to Power BI harvester 1.0.0.1 or newer and lineage harvester 1.3.0 or newer (deprecated)

The [Power BI harvester](#) 1.0.0.1 enables you to connect to a [Collibra Data Lineage server](#), based on your geolocation and cloud provider.

You only have to follow this upgrade procedure when you upgrade from Power BI harvester 1.0.0.0 to Power BI harvester 1.0.0.1 and newer or if the server's geolocation or cloud provider changes.

Tip We highly recommend that you always use the newest Power BI harvester and [lineage harvester](#).

Steps

1. If you have strict firewall rules, allow one of the following IP addresses, based on your Collibra geolocation and cloud provider:
 - 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 IP address 15.222.200.199 is only available for Power BI harvester 1.0.0.2 and lineage harvester 1.3.1 and newer.

2. Download Power BI harvester 1.0.0.1 or newer, from the Collibra Downloads page.
3. [Install](#) the new Power BI harvester.
4. Migrate your Power BI connection information in your old configuration file to the [configuration file](#) in the new Power BI harvester folder.
5. Trigger the Power BI harvester to upload the Power BI metadata to the Collibra Data Lineage server with the new IP address:
 - Run the following command line if your configuration file is in its default location: `.\powerbi-harvester.bat`
 - Launch the path to the Power BI configuration file if you moved the configuration file to a different location: `.\bin\powerbi-harvester.exe .\config\powerbi-harvester.conf`

Note We highly recommend that you run the Power BI harvester via command line. This enables you to follow the metadata upload and see possible errors that may occur.

6. Download lineage harvester 1.3.0 or newer, from the Collibra Downloads page.
7. [Install](#) the new lineage harvester.
8. Migrate the data sources in your old configuration file to the configuration file in the new lineage harvester folder.
9. Run the lineage harvester with the `full-sync` command.
 - » The lineage harvester uploads your data sources to the Collibra Data Lineage server with the new IP address.

10. Restart the DGC service in Collibra Console.

Tip For more information about Power BI and the Power BI harvester, see [Power BI](#).

What's next?

Collibra now synchronizes your [Power BI assets and relations](#). You can also access the [technical lineage](#) via a Power BI Column asset page.

Power BI troubleshooting (deprecated)

It is possible that you encounter problems during the Power BI ingestion process.

Note You can also encounter problems due to [Power BI ingestion limitations](#).

The following table lists possible problems and offer a solution.

Problem	Description
You have made a mistake in the Power BI harvester configuration file or the lineage harvester configuration file .	Make sure to check all properties and values before you run the Power BI harvester. If any of the values of the required configuration properties are missing, invalid or incorrect, the Power BI harvester or lineage harvester fails with an error or the Power BI ingestion will be incorrect. The Power BI harvester and lineage harvester should have the same value in the <code>url</code> and <code>(source)Id</code> property.
You don't have the correct Power BI permissions or not all prerequisites have been met before you start the Power BI integration process.	Make sure you have read and performed all prerequisites. The prerequisites are slightly different if you choose for username / password or service principal authentication .

Problem	Description
The Power BI harvester failed to retrieve Power BI capacities and shows status code "Unauthorized".	This is a common mistake when you use the service principal authentication method. To solve this issue, make sure that you have enabled the Allow service principals to use read-only Power BI admin APIs (preview) option in the Power BI Admin portal. Tip Do not confuse the Allow service principals to use read-only Power BI admin APIs (preview) option with the Allow service principal to use Power BI APIs option. You need to enable both options.
You have network or remote API issues.	Web services providing the API interfaces that the Power BI harvester uses may sometimes experience problems, or there may be problems with network access to these resources. If the Power BI harvester fails unexpectedly, check the following network resources and make sure they work properly: • Power BI REST API endpoints • XMLA endpoints • Technical lineage API Considering the nature of these remote resources, the cause of the problem can often be out of your control. Please wait until the issue is resolved or escalate the issue with the respective authority.
You cannot retrieve information for individual Power BI dashboards or data sets.	To retrieve metadata of individual Power BI dashboards or data sets, you require permissions to access them. However, sometimes the Power BI dashboards and data sets are in a problematic state or you cannot reach them due to Power BI-related issues. When you execute the Power BI harvester, a summary of all encountered problems is printed. To reduce the number of problems, you can use the group filters in the Power BI configuration file to restrict the set of harvested Power BI workspaces. Depending on the type of issue, you may need to solve them one by one.

Problem	Description
The Power BI harvester cannot retrieve certain workspaces in the <code>workspaceFilter</code> property.	Make sure the syntax in the <code>workspaceFilter</code> property in the configuration file is correct and you don't use the "IN" operator. Note Currently, the "IN" operator is not supported. As a result, you cannot use "IN" to filter on specific Power BI workspaces in the <code>workspaceFilter</code> property in the Power BI configuration file. For more information, see the Power BI limitations.
The Power BI harvester failed to connect to the Microsoft API.	Usually, this is a timeout issue. We highly recommend that you increase the timeout duration. Use the following command line option to set the timeout duration: <code>--timeout <seconds></code>. For example, if you want the Power BI harvester to wait 15 minutes for the connection, you can use <code>--timeout 900</code>.
You get an error message related to Usage Metrics.	If you see errors related to Usage Metrics, you can ignore them, because they do not cause Power BI ingestion to fail. Usage Metrics are reports that are automatically created in Power BI, but they do not represent any Power BI assets or technical lineage information.
The technical lineage is missing or incomplete.	If the technical lineage is missing, you must add your Power BI workspaces to a dedicated capacity to allow the Power BI harvester to extract data from XMLA endpoints. Harvesting metadata via Power BI REST API does not require the dedicated capacity. As a result, the Power BI harvester can only reach limited Power BI metadata and won't create a technical lineage. If the technical lineage is incomplete, certain aspects of the Power BI ingestion job may not be supported. Note You can only ingest new Power BI workspaces. This means that classic workspaces and My Workspace in Power BI is not supported. Also read the other limitations of the Power BI ingestion process to understand why technical lineage is missing or incomplete.

Problem	Description
Some Power BI metadata is missing in Data Catalog.	Do the following: • Use new Power BI workspaces if you want a full ingestion. • Add your Power BI workspaces to a dedicated capacity to allow the Power BI harvester to extract data from XMLA endpoints. • Grant the Power BI workspaces the Contributor role in the Power BI security group.
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.

Power BI ingestion tests

If you want to test the Power BI ingestion, we recommend that you use the `workspaceFilter` property in the Power BI [configuration file](#) to limit the Power BI ingestion to one or two [Power BI workspaces](#).

For more information about the query options to filter Power BI workspaces, see the [Microsoft documentation](#).

Example If you want to limit the Power BI ingestion to one Power BI workspace with the name `PowerBIWorkspace1`, you can set the `workspaceFilter` value to `"Name eq 'PowerBIWorkspace1'"`.

Power BI harvester messages (deprecated)

When something goes wrong during the [Power BI metadata harvesting process](#), the Power BI harvester logs show a message code that provides a link to more information. The message code indicates which part of the harvesting process failed or was skipped, and provides steps to resolve it.

Tip Make sure that you understand the [Power BI metadata harvesting process](#) and the typical [Power BI ingestion workflow](#).

Message code	Description
MSG-LIN-7000	This message is a reminder to follow all steps to ingest the Power BI metadata in Data Catalog. This message is always shown after the Power BI harvester successfully uploads the Power BI metadata to the Collibra Data Lineage server. Next, you have to create a lineage harvester configuration file and successfully run the lineage harvester to create Power BI assets and relations in Data Catalog.
MSG-LIN-7001	An unexpected problem occurred at the local machine. The error can be caused by an invalid path name, not enough storage space or other unexpected issues.
MSG-LIN-7002	There is a problem with the Power BI harvester configuration file or the source ID configuration file . Make sure that all information in the configuration file(s) and the path to the configuration file(s) is correct.
MSG-LIN-7003	The Power BI harvester could not retrieve tenant information, because the Microsoft API did not return a response that the Power BI harvester could process. To solve this problem, we recommend that you check your network settings and rerun the Power BI harvester. If the issue persists, please contact Collibra support or your customer success manager.
MSG-LIN-7004	The Power BI harvester could not communicate with the Collibra Data Lineage server, likely because of one of the following scenarios: • The remote API did not return a response that the Power BI harvester could process. • The API call returned a 401 (Unauthorized) error because an invalid user-Key token was used. To solve this problem, we recommend that you: • Ensure that the Power BI harvester is connecting to the correct Collibra Data Lineage server. • Check your network settings and rerun the Power BI harvester. If the issue persists, please contact Collibra support or your customer success manager.

Message code	Description
MSG-LIN-7005	The Power BI harvester could not retrieve Power BI metadata, because the Power BI service did not return a response that the Power BI harvester could process. To solve this problem, we recommend that you check your network settings and rerun the Power BI harvester. If the issue persists, please contact Collibra support or your customer success manager. Note If the error message indicates that the issue is an internal server error, the problem is caused by the Power BI REST API.
MSG-LIN-7006	The Power BI harvester could not communicate with a remote server, because the server did not return a response within an expected time interval and, as a result, the Power BI harvester aborted the process. To solve this problem, we recommend that you do the following: • Check your network settings. • Check the amount of metadata that is processed. If it is a large amount, use the <code>--timeout</code> command line option to specify a longer timeout for remote API calls. If the problem persists or the remote server does not respond within a reasonable time period, create a support ticket or contact your customer success manager.
MSG-LIN-7007	This problem occurs when the Power BI service returns inconsistent data. As a result, the Power BI harvester cannot successfully process the data to create a consistent result data set. The Power BI harvester uses multiple API calls to retrieve Power BI metadata. If something in the Power BI service changed during the harvesting process, the metadata can be inconsistent. We recommend to run the Power BI harvester again. If the issue persists, create a support ticket or contact your customer success manager.
MSG-LIN-7008	The Power BI harvester cannot access XMLA endpoints for some Power BI dedicated capacities with harvested workspaces, because the capacities are currently not running. As a result, the Collibra Data Lineage server cannot create a technical lineage for these workspaces. To solve this problem, check if the Power BI workspace is part of a running dedicated capacity and you meet the necessary prerequisites to access and export it.

Message code	Description
MSG-LIN-7009	The Power BI authentication failed. This problem can be caused by an error in the Power BI credentials. To solve this problem, check your Power BI login credentials in the Power BI harvester configuration file or reenter them via command line.
MSG-LIN-7010	The connection between the Power BI harvester and the Collibra Data Lineage server failed. This problem can be caused by an error in the Collibra Data Intelligence Cloud credentials or Collibra Data Intelligence Cloud host address. To solve this problem, check your Collibra Data Intelligence Cloud credentials in the Power BI harvester configuration file or reenter them via command line.
MSG-LIN-7011	The Power BI harvester could not retrieve the tenant domain information. To solve this problem, check that you have the correct tenant domain ID in the Power BI harvester configuration file.

Message code	Description
MSG-LIN-7012	The Power BI API failed. This can be caused by an error in the syntax of the <code>workspaceFilter</code> field in the Power BI harvester configuration file. Important The <code>workspace filter</code> operations use OData syntax and are processed by the Power BI service, not the Power BI harvester. Examples of supported workspace filter operations: • <code>name eq 'Workspace1' or name eq 'Workspace2'</code> only harvests workspaces with the specified names. • <code>not endswith(name, 'Test')</code> only harvests workspaces whose names don't end in <code>Test</code>. • <code>tolower(capacityId) eq '01234567-89ab-cdef-0123-456789abcdef'</code> only harvests workspaces hosted on the specified dedicated capacity. • <code>reports/any(d:contains(d/name, 'Sales'))</code> only harvests workspaces with reports whose names contain <code>Sales</code>. If you do not want to filter on specific workspaces, leave the <code>workspaceFilter</code> field in the Power BI harvester configuration file empty. Tip For more information about the query options to filter Power BI workspaces, see the Microsoft documentation. We cannot guarantee that other group filter operations work correctly. For example, the <code>IN</code> operator is currently not supported.
MSG-LIN-7013	You do not have the required permissions to harvest the Power BI metadata. Check that the user is a Power BI Administrator and that the Power BI application has all required permissions. To solve this problem, check that you have correctly registered your Power BI application in Microsoft Azure.
MSG-LIN-7014	You do not have the required permissions to harvest the Power BI metadata. Enable the Allow service principals to use read-only Power BI admin APIs (preview) option in the Power BI Admin Console. To solve this problem, check that you meet the prerequisites to use the service principal.

Message code	Description
MSG-LIN-7015	You do not have the required permissions to harvest the Power BI metadata. Enable the Allow service principal to use Power BI APIs option in the Power BI Admin Console. To solve this problem, check that you meet the prerequisites to use the service principal.
MSG-LIN-7016	The harvested Power BI workspaces are not assigned to a dedicated capacity. As a result, Data Catalog cannot ingest details about tables and columns and technical lineage are not available. Note You do not have to assign less important to a dedicated capacity, for example personal workspaces. However, if there are no workspaces on a dedicated capacity, the harvested Power BI metadata is very limited.
MSG-LIN-7017	The Power BI harvester could not access XMLA endpoints for any Power BI workspaces to retrieve detailed information about data sets. As a result, technical lineage is be available. To solve this issue, check that you meet the prerequisites to access XMLA endpoints for all Power BI workspaces that you want to ingest in Data Catalog.
MSG-LIN-7018	Batch processing failed at Collibra server. The harvested batch was uploaded to a Collibra Data Lineage server, but the server could not process the batch. Review the error message that accompanies this error code. It might identify a problem that you can resolve, for example if you used an unsupported version of the harvester. If the error message does not identify the problem or if you're unable to resolve it on your own, create a support ticket or contact your customer success manager.

Working with SSRS and PBRs

SQL Server Reporting Services (SSRS) and Power BI Report Server (PBRs) are server-based report generating applications, created by Microsoft, that help you see and understand your data.

PBRs is included in the licensing of the SQL Server Enterprise Edition or as a free extension of Power BI premium. SSRS and PBRs are closely related and both use the same API to communicate to the lineage harvester. As a result, Collibra created one operating model that contains data from both SSRS and PBRs. Whether you use SSRS, PBRs or both, you only need one integration in the lineage harvester configuration file.

Note

- Due to a bug in 2017 that is resolved by the newer APIs, we recommend using SQL Server 2019 or newer Reporting Services.
- While SSRS and PBRs use the same API, we can access less information from PBRs reports than from SSRS data. As a result, we do not support stitching and lineage information for PBRs reports.

SSRS and PBRs asset and domain types	677
SSRS and PBRs terminology	679
SSRS and PBRs operating model	680
Soft delete of SSRS assets	697
Technical lineage for SSRS and PBRs	698
SSRS and PBRs integration steps	700

SSRS and PBRs asset and domain types

The SQL Server Reporting Services (SSRS) and Power BI Report Server (PBRs) integration in Collibra Data Intelligence Cloud uses a specific subset of [asset types](#) and [domain types](#).

The following table shows the asset types and domain types that are used for the SSRS and PBRs integration. You can see the parent asset types in the breadcrumbs above each asset type.

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
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

Asset type	Description	Domain type
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

SSRS and PBRs terminology

The following table shows the SQL Server Reporting Services (SSRS) and Power BI Report Server (PBRs) terminology and how it maps to the Collibra Data Intelligence Cloud asset types.

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

Term	Description	Asset type in Colibra
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

SSRS and PBRS operating model

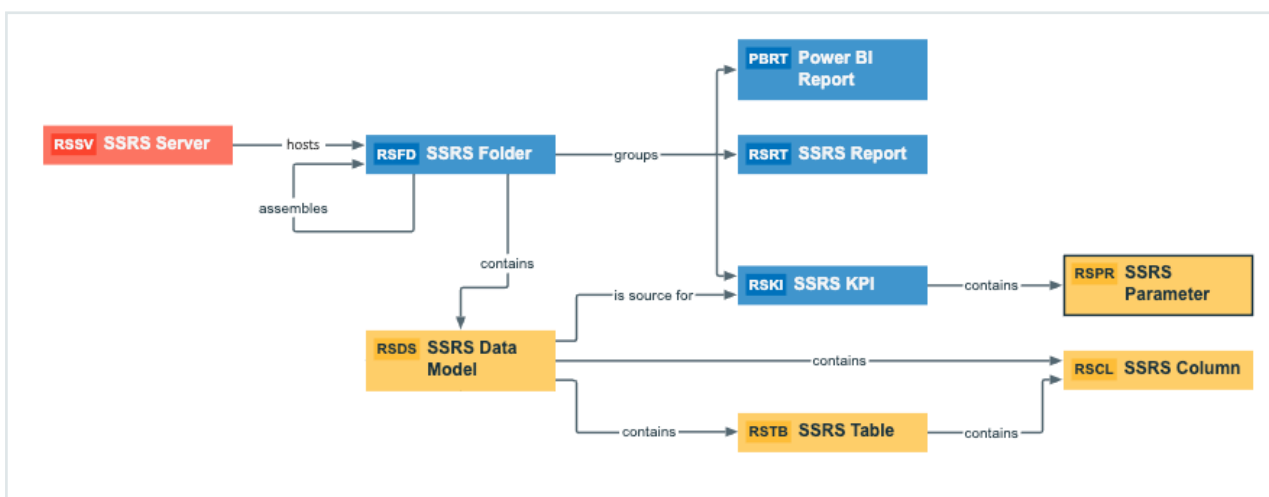
The lineage harvester collects SQL Server Reporting Services (SSRS) metadata and sends it to the Colibra Data Lineage service. Colibra processes the metadata and creates new SSRS assets and relations in Data Catalog. You can see them on the asset page overview or visualize them in a [diagram](#) or in 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.

SSRS and PBRs metadata overview

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 / source 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	00000000-0000-0000-0000-0000000003114
	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 / source 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	00000000-0000-0000-0000-0000000003114
	Business Asset represents / represented by Data Asset	00000000-0000-0000-0000-0000000007038
	Report Attribute contained in / contains Report	00000000-0000-0000-0000-0000000007058
	Report Attribute sourced from / is source of Data Attribute	00000000-0000-0000-0000-1200000000010

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 / source 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 (or domains) 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 Set **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. In the tab pane, click  **Diagram**.
 - » The diagram appears 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
    }
  }
}

```

Soft delete of SSRS assets

When you integrate SSRS-PBRS, 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 API considers the data object as new and creates another corresponding asset in Data Catalog. The result is two assets for the same data object; one with the status "Missing from source", one with the status "Candidate". As described in the following section, you can then run a workflow to delete the "Missing from source" assets, or you can manually delete them.

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.

Technical lineage for SSRS and PBRS

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

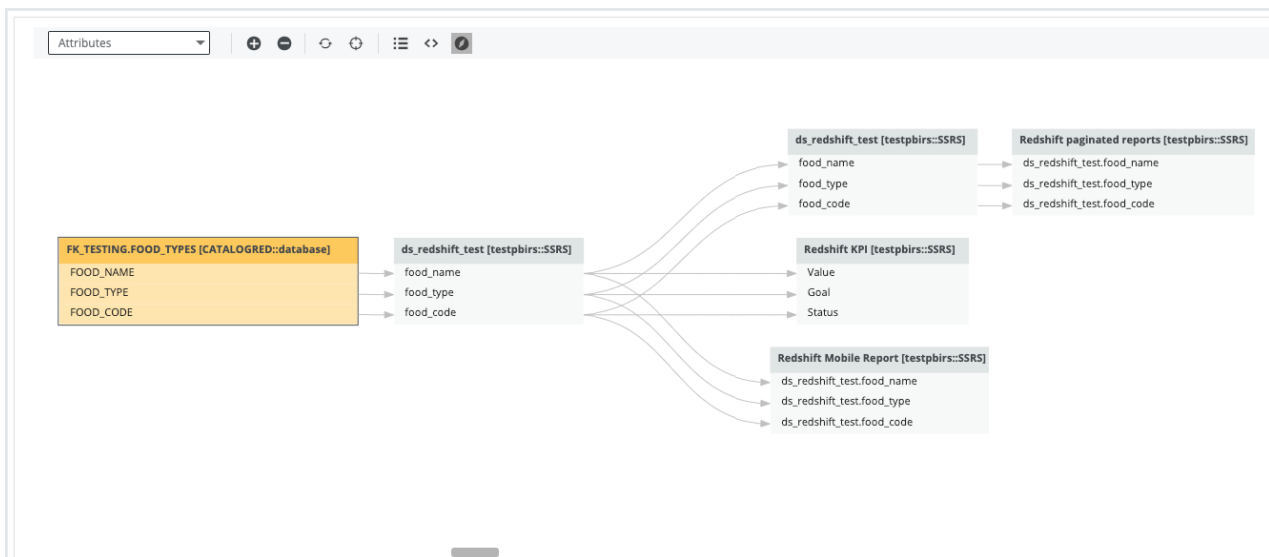
The technical lineage graph shows relations of the type "Column is source for / is target of Data Attribute" between BI assets and other data objects in the data flow, for example Column assets or Power BI Column assets. These relations are created during the ingestion process as a result of automatic 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%.

SSRS and PBRS integration steps

The SQL Server Reporting Services (SSRS) and Power BI Report Server (PBRS) integration in Collibra Data Intelligence Cloud enables you to harvest SSRS and PBRS metadata and create new SSRS and Power BI assets in Data Catalog. Collibra analyzes and processes the BI metadata and presents it as [assets of specific types](#), retaining their original names.

Note While SSRS and PBRS use the same API, Colibra Data Lineage can access less information from PBRS reports than from SSRS data. As a result, we do not support stitching and lineage information for PBRS reports.

Roles and permissions in SSRS or PBRS

To ingest SSRS and PBRS metadata into Data Catalog, the lineage harvester connects to the SSRS or PBRS web portal. You need a role with [user access to the server](#) from which you want to ingest:

- You have a system-level role, which is at least a System user role.
- You have an item-level role, which is at least a Content Manager role.

Note You can ingest SSRS and PBRS with any report server subscription. We highly recommend to install and use the latest version of SSRS and PBRS .

Steps

1. [Prepare](#) a domain for SSRS-PBRS ingestion.
If you already have a domain reserved for ingestion of the new SSRS assets, you can skip this step.
2. [Download](#) and [install](#) the lineage harvester.
3. [Prepare](#) the lineage harvester configuration file.
4. Optionally, [prepare](#) a <source ID> configuration file.
5. Start the lineage harvester again in the console and run the following command:
 - for Windows: `.\bin\lineage-harvester.bat full-sync`
 - for other operating systems: `./bin/lineage-harvester full-sync`

When prompted, enter the password or client secret to connect to your Colibra Data Intelligence Cloud and Tableau environment. The passwords are encrypted and stored in `/config/pwd.conf`.

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.

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 [SSRS-PBRs technical lineage](#).

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

Lineage harvester system requirements

You need to meet the system requirements to be able to [install](#) and run the [lineage harvester](#).

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 [Technical lineage general troubleshooting](#).

- 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 the 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.

Prepare a domain for SSRS and PBRS ingestion

Before you can ingest SSRS-PBRS metadata, you have to create a new domain or choose an existing domain to store the new SSRS assets. You specify the domain reference ID in the [lineage harvester configuration file](#). Collibra uses the domain to ingest all SSRS assets.

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 a domain type from the list.

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
Type	The domain type of the domain you are creating. In this case, you need to select <i>BI Catalog</i> .
Community	The community under which the domain will be located.
Name	The name of the new domain.

5. Click **Create**.
6. Open your domain.
7. 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=00000000-0000-0000-0000-000000040001`. In this example, the domain ID is in bold.

8. Paste the domain ID in the [lineage harvester configuration file](#).

Install the lineage harvester for SSRS-PRBS integration

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

Tip

- Install the lineage harvester close to your data source or on the same server.
- The lineage harvester uses port 443.

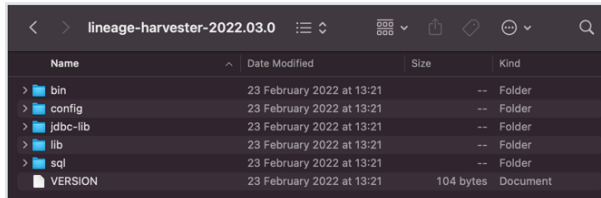
Prerequisites

- 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.

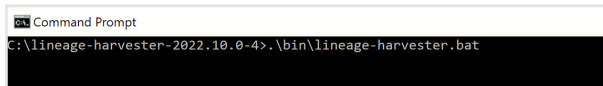
Steps

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.

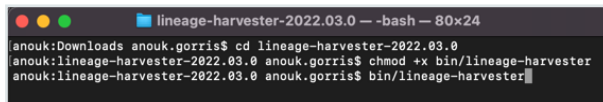


3. Start the lineage harvester to create an empty lineage harvester configuration file by entering the following command:

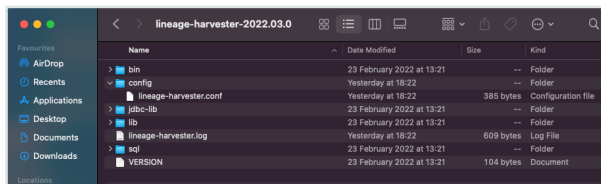
- **Windows:** `.\bin\lineage-harvester.bat`



- **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?

You can now **prepare** the lineage harvester configuration file.

Prepare the lineage harvester configuration file for SSRS and PBRs integration

You have to prepare a configuration file before you run the lineage harvester. The lineage harvester collects your SQL Server Reporting Services (SSRS) and Power BI Report Server (PBRs) metadata and sends it to Collibra Data Intelligence Cloud, where it is processed and analyzed. Collibra then imports the SSRS and PBRs assets and relations to Data Catalog.

Example configuration file

```
{
  "general": {
    "catalog": {
      "url": "https://<organization>.collibra.com",
      "username": "<your-collibra-username>"
    },
    "useCollibraSystemName": false
  },
  "sources": {
    "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": true
  }
}
```

Before you begin

- [Set up the latest lineage harvester.](#)
- [Prepare a domain in Data Catalog.](#)
- [Prepare a SSRS and PBRs <source ID> configuration file, if necessary.](#)

Requirements and permissions

- 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.
- Collibra Data Intelligence Cloud.
- 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 BI Data Catalog domain:
 - Asset: add
 - Attribute: add
 - Domain: add
 - Attachment: add

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 to US government customers.

Properties	Description
url	The URL of the Collibra Data Lineage service instance. "url": "https://techlin-gov.collibra.com" Warning This 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 applies only to US 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 all SSRS connection properties.

Properties	Description
id	The unique ID to identify the SSRSmetadata that was uploaded to the Collibra Data Lineage service. Tip This value can be anything as long as it is a unique. The lineage harvester uses the ID to identify a batch of data on the Collibra Data Lineage service. 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.
type	The kind of data source. In this case, the value has to be <i>SSRS</i> or <i>PBRS</i>. Note There is no difference between type SSRS or PBRS.
url	The URL to the server's web portal. By default, the URL is <code>http://<computer-name>/reports</code>. For example, "http://1.23.45.678/PowerBIReports".
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: <code>https://<yourcollibrainstance>/domain/22258f64-40b6-4b16-9c08-c95f8ec0da26?view=00000000-0000-0000-000000040001</code>. 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. 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: <code>["folder1", "/data-base/folder2", /folder3/*"]</code> Show me some examples <table> <tr> <th>Scenario</th><th>Configuration</th></tr> <tr> <td>Filter on all folders with the name Folder3, anywhere in the folder hierarchy.</td><td> <pre>["Folder3"]</pre> Note Reports in child folders of Folder3 are not included in the ingestion. As such: ◦ Reports in <code>/Folder1/Folder2/Folder3</code> are included in the ingestion. ◦ Reports in <code>/Folder3/ChildFolder</code> are not included in the ingestion. </td></tr> <tr> <td>Ingest two folders for which the folder names are unique.</td><td><pre>["Folder1", "Folder2"]</pre></td></tr> <tr> <td>Filter on a specific folder or folders, when the folder names are not unique.</td><td> In this case, specify the full paths to the folders, for example: <pre>["/Database1/Folder1", "/Database2/Database3/Folder2"]</pre> </td></tr> </table>	Scenario	Configuration	Filter on all folders with the name Folder3, anywhere in the folder hierarchy.	<pre>["Folder3"]</pre> Note Reports in child folders of Folder3 are not included in the ingestion. As such: ◦ Reports in <code>/Folder1/Folder2/Folder3</code> are included in the ingestion. ◦ Reports in <code>/Folder3/ChildFolder</code> are not included in the ingestion.	Ingest two folders for which the folder names are unique.	<pre>["Folder1", "Folder2"]</pre>	Filter on a specific folder or folders, when the folder names are not unique.	In this case, specify the full paths to the folders, for example: <pre>["/Database1/Folder1", "/Database2/Database3/Folder2"]</pre>
Scenario	Configuration								
Filter on all folders with the name Folder3, anywhere in the folder hierarchy.	<pre>["Folder3"]</pre> Note Reports in child folders of Folder3 are not included in the ingestion. As such: ◦ Reports in <code>/Folder1/Folder2/Folder3</code> are included in the ingestion. ◦ Reports in <code>/Folder3/ChildFolder</code> are not included in the ingestion.								
Ingest two folders for which the folder names are unique.	<pre>["Folder1", "Folder2"]</pre>								
Filter on a specific folder or folders, when the folder names are not unique.	In this case, specify the full paths to the folders, for example: <pre>["/Database1/Folder1", "/Database2/Database3/Folder2"]</pre>								

Properties	Description				
	<table> <tr> <th>Scenario</th><th>Configuration</th></tr> <tr> <td>Use a wildcard to ingest all child folders of a Folder1.</td><td> <pre>["/Folder1/*"]</pre> Note The reports in all child folders of Folder1 are ingested, but the reports in Folder1 itself are not ingested. </td></tr> </table> 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: <code>"folderFilter": ["*"]</code>. Tip For more information about connecting to a SSRS or PBRs folder, see the Microsoft documentation.	Scenario	Configuration	Use a wildcard to ingest all child folders of a Folder1.	<pre>["/Folder1/*"]</pre> Note The reports in all child folders of Folder1 are ingested, but the reports in Folder1 itself are not ingested.
Scenario	Configuration				
Use a wildcard to ingest all child folders of a Folder1.	<pre>["/Folder1/*"]</pre> Note The reports in all child folders of Folder1 are ingested, but the reports in Folder1 itself are not ingested.				
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.

Prepare a SSRS and PBRs <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 <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> "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.

What's next?

See [Overview SSRS and PBRS integration steps](#).

Schedule SSRS and PBRS ingestion jobs

You can use Task Scheduler on Windows or Crontab on Mac and Linux to enable the lineage harvester to run scheduled jobs. In a scheduled job, the lineage harvester uploads the SQL Server Reporting Services (SSRS) and Power BI Report Server (PBRS) metadata to Collibra.

Collibra automatically creates new assets and relations at specific times, dates or intervals, using the information in the lineage harvester [configuration file](#).

Example You created a lineage harvester configuration file with connection information to your SSRS or PBRS environment. You schedule the lineage harvester job to run each Sunday at 23:00. As a result, your SSRS and PBRS metadata is automatically refreshed on a weekly basis.

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

Warning Relations that were manually created between [SSRS and Power BI Report assets](#) and other assets via a relation type in the [operating model](#), are deleted after a refresh of the metadata.

Working with Looker

Looker is a business intelligence software that helps people see and understand their data.

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

Note When you ingest Looker metadata, you automatically create a technical lineage for Looker.

Looker terminology	723
Looker operating model	724
Looker asset and domain types	742
Looker business logic	743
Authentication	747
Technical lineage for Looker	748
Soft delete of Looker assets	749
Prepare a domain for Looker ingestion	750
Overview Looker integration steps	752
Schedule Looker ingestion jobs	772
Troubleshooting	773

Looker terminology

Before you ingest [Looker](#), read more about the Looker terminology and how it maps with the Collibra Data Intelligence Cloud asset types.

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

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
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

Looker operating model

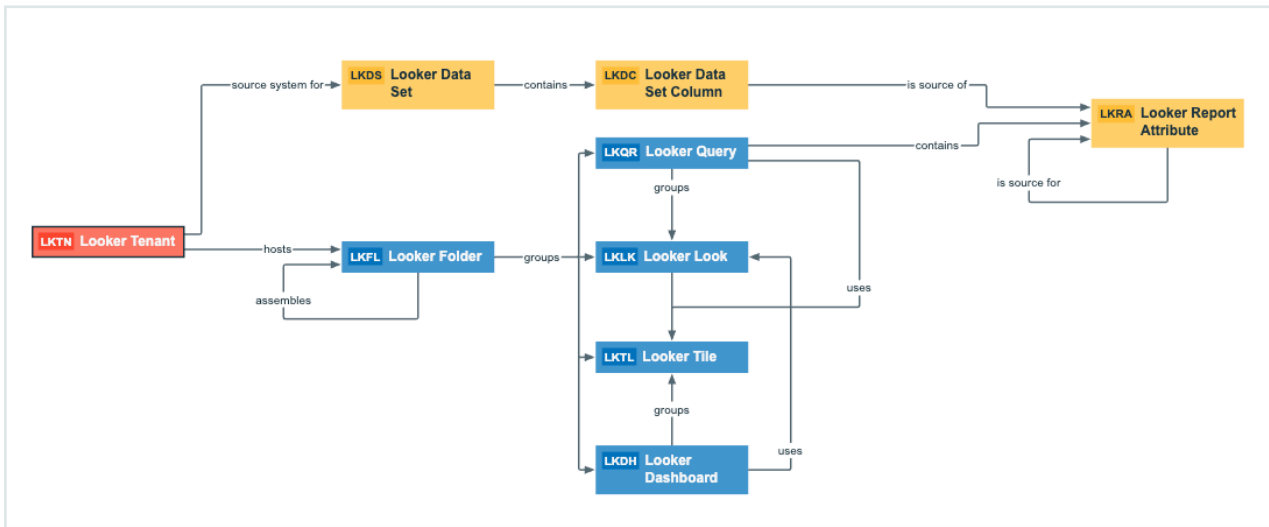
The Looker scanner collects Looker metadata and sends it to the Collibra Data Lineage service. Collibra processes the metadata and creates new Looker assets and relations in

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.

Looker metadata overview

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	

Asset type	Synchronized metadata	Resource ID
	Display name	
	Description	00000000-0000-0000-0000-000000003114
	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 / 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	

Asset type	Synchronized metadata	Resource ID
	Display name	
	Description	00000000-0000-0000-0000-000000003114
	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.

Business Analysts Community

Looker enablement

Type: BI Catalog ⓘ Edit Move Delete Auto hyperlinks

Overview

Assets

Responsibilities

History

Files

Default ▾

> Delete Move Validate

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 (or domains) 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. In the tab pane, click  **Diagram**.
 - » The diagram appears 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
  }
}
}

```

Looker asset and domain types

The [Looker](#) integration in Collibra Data Intelligence Cloud uses a specific subset of [asset types](#) and [domain types](#). All of these come out of the box with your software.

The following table contains the asset and domain types that are used for the Looker integration. Above each asset type you can see the parent asset types in the breadcrumbs.

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

Asset type	Description	Domain type
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
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

Looker business logic

Looker business users usually work with Looker dashboards and Looker looks to make business decisions. Collibra's Looker connector and lineage feature, offers business users several advantages:

- Easily find certified Looker content.
- Shop for Looker Looks.
- Find where content is stored in Looker.

- Get information about a Looker Look and other Looker report details 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.

Looker asset pages

Depending on the [Looker asset type](#), the asset page shows different [information ingested from Looker](#). You can find a specific Looker asset page using [Data Catalog search](#) or via the Data Catalog BI domain in which you ingested the Looker metadata.

Details

An asset page contains attributes and relations to other assets. This information is synchronized from Looker. However, you can add additional characteristics, tags or comments.

If you want to use a Looker Look, you can add it to the [Data Basket](#) and check it out.

Example The following Looker Look asset shows in which Looker Folder it is stored, in which Looker Dashboard it is shown, which Looker Tiles it uses and which Looker Queries it groups. This asset has a number of attributes that give more information about the Looker Look.

Business Analysts Community > **Looker Catalog**

LKLK Average_age
Looker Look Implemented

URL
<https://collibra.looker.com/looks/12>

Visits count
2

Favorites count
0

Document creation date
7/17/2019

Document modification date
7/17/2019

Document last accessed date
7/20/2020

used in Report

Name	Domain	Definition	Description
Web Analytics and Product Us...	Looker Catalog		

uses Report

Name	Domain	Definition	Description
Average_age	Looker Catalog		

Is grouped into Business Dimension

Name	Domain	Description
Shared	Looker Catalog	

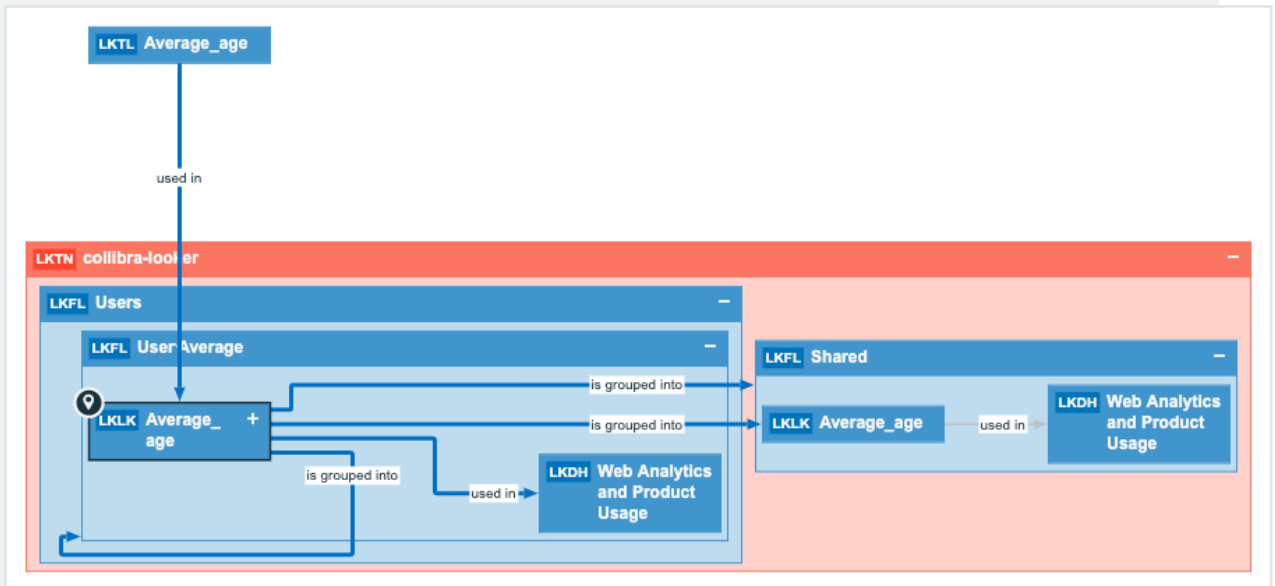
groups Report

Name	Domain	Definition	Description
Query 456 part 1	Looker Catalog		
Query 456 part 2	Looker Catalog		

Business diagrams

The **business diagram** is a feature to show and interact with many assets and relations in an easy-to-read diagram. The business diagram helps you to quickly see to which other assets a specific asset is related. As such, the diagram can show a high-level presentation of a Looker Look. This enables you to see how the Looker Look relates to other Looker assets.

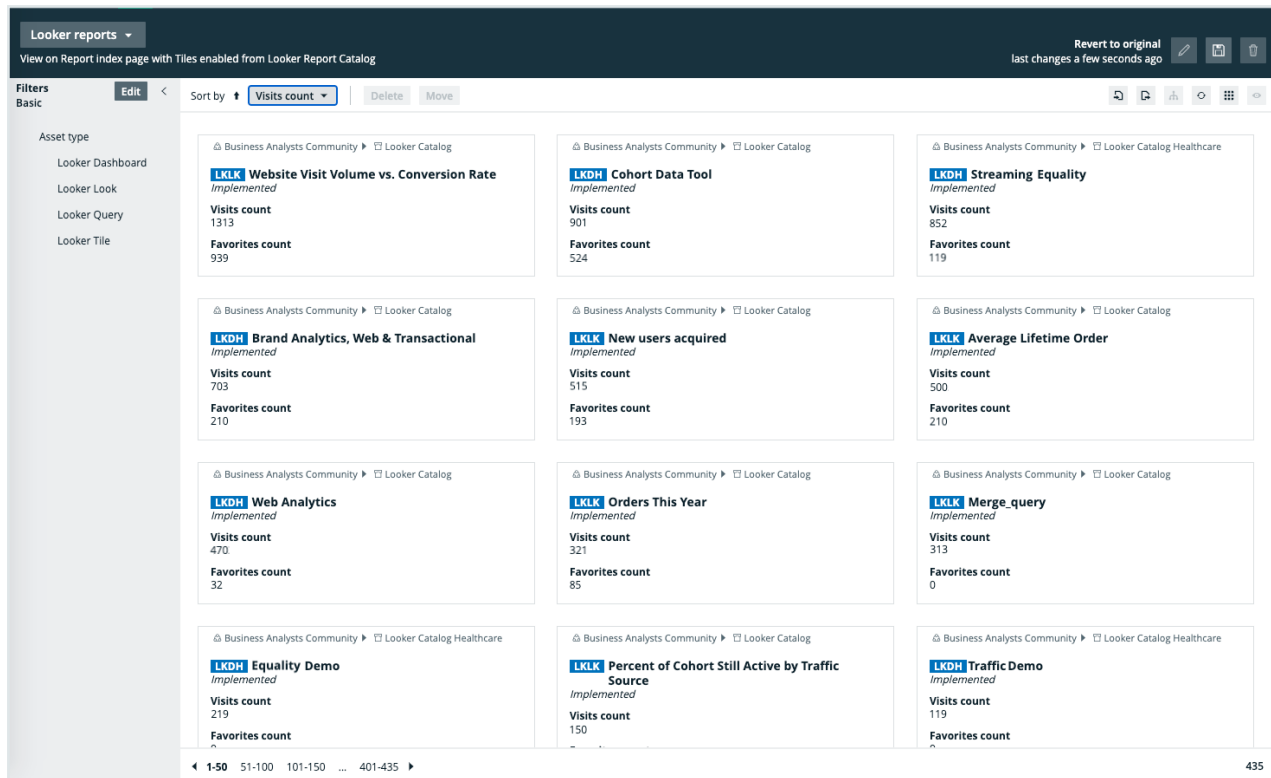
Example The following business diagram shows the *Average_age* Looker Look, which is stored in the *UserAverage* Looker Folder, but is also grouped into the *Shared* Looker Folder.



Report views

The Looker connector and lineage feature enables you to find all ingested Looker Look, Looker Dashboard, Looker Tile and Looker Query asset types in a single location.

In the **Reports** tab page in Data Catalog you can see an overview of all Report assets and their children. Optionally, you can [create a view](#) with a filter to only show Looker assets. This is useful if you quickly want to see all reports or if you want find specific reports for example certified reports or reports that are visited the most.



Authentication

Before you start the Looker integration, you have to enable Collibra to access your Looker metadata. 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.

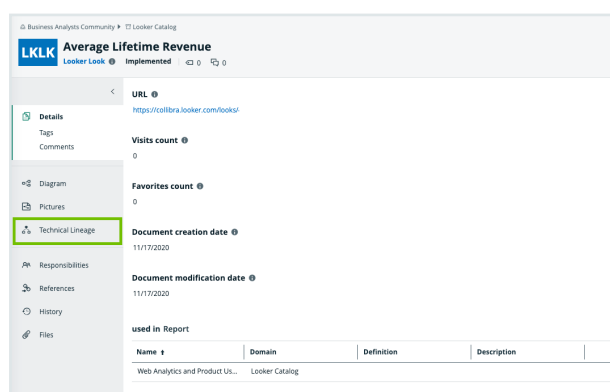
2. Create the [API3 credentials](#).
3. Use the API3 credentials in the [configuration file](#).

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](#).

Technical lineage for Looker

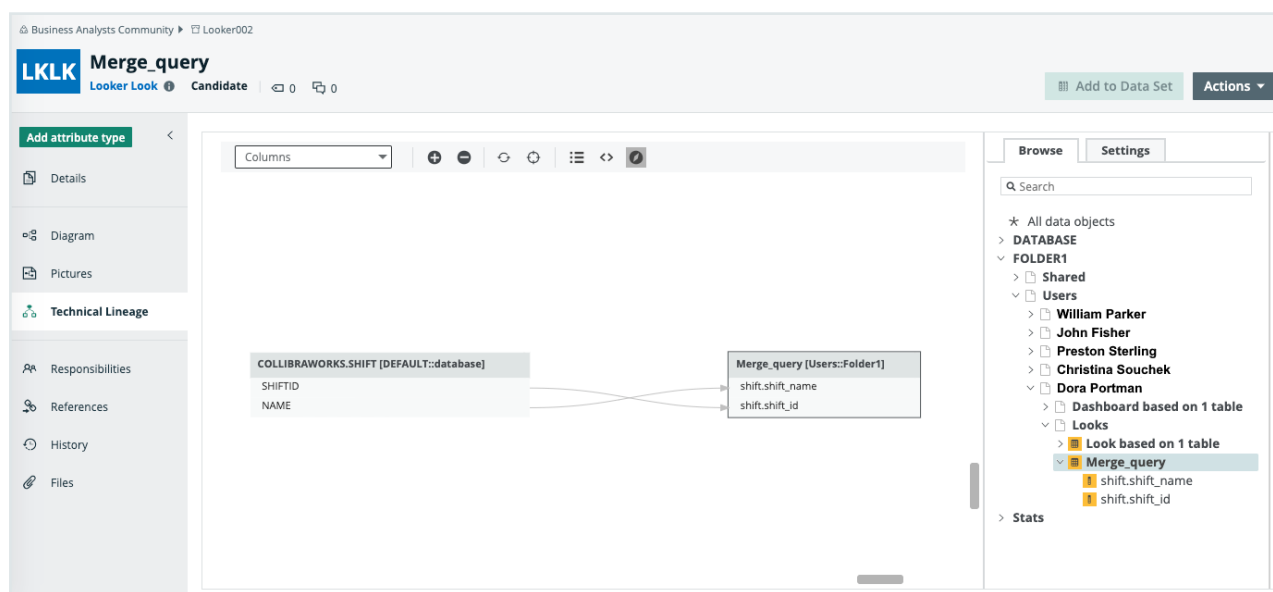
When you ingest Looker metadata, you automatically create a technical lineage for Looker Look assets. If you have the right [permissions](#) to view the technical lineage, you can go to a Looker Look asset page and click the Technical lineage tab, which allows you to access the technical lineage.



Note Due to the limitations of the Looker REST API, we 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. As a result, the technical lineage only shows Looker metadata as it exists on the Collibra Data Lineage service and not as assets in Data Catalog.

Example

The following technical lineage graph shows the technical lineage of Looker objects.



Soft delete of Looker assets

When you integrate Looker, 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 API considers the data object as new and creates another corresponding asset in Data Catalog. The result is two assets for the same data object; one with the status “Missing from source”, one with the status “Candidate”. As described in the following section, you can then run a workflow to delete the “Missing from source” assets, or you can manually delete them.

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.

Prepare a domain for Looker ingestion

Before you can ingest Looker metadata, you have to create a new domain or choose an existing domain to store the new Looker assets. You then specify:

- One domain reference ID in the [lineage harvester configuration file](#). This domain 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](#).

Collibra uses the specified domain (or domains) to ingest all Looker assets during the [Looker integration process](#).

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 a domain type from the list.

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
Type	The domain type of the domain you are creating. In this case, you need to select <i>BI Catalog</i> .
Community	The community under which the domain will be located.
Name	The name of the new domain or domains. Tip You can create multiple domains in one go. To do this, press Enter 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 reference 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 reference ID is in bold.

8. Paste a reference ID in the `domainId` property in your [lineage harvester configuration file](#). This is your default domain.

Note Metadata from Looker data sets is ingested in the default domain, even if you configure filtering.

9. 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](#).

What's next?

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

Overview Looker integration steps

The Looker integration enables you to harvest Looker metadata and create new Looker assets in Data Catalog. Collibra analyzes and processes the Looker metadata and presents it as specific [asset types](#), retaining their original names.

Important In the global [assignment](#) of each asset type included in the Power BI Looker [operating model](#), ensure that none of the characteristics that are in the operating model have a maximum cardinality of “0”. If the maximum cardinality is set to “0” for any such characteristics, ingestion will fail.

Steps

1. [Set up](#) Looker authentication.
2. [Prepare](#) a domain for Tableau ingestion.
3. [Download](#) and [install](#) the lineage harvester.
4. [Prepare](#) the lineage harvester configuration file.
5. Optionally, [Prepare](#) the <source ID> configuration file.
6. Start the lineage harvester again in the console and run the following command:
 - for Windows: `.\bin\lineage-harvester.bat full-sync`
 - for other operating systems: `./bin/lineage-harvester full-sync`

When prompted, enter the password or client secret to connect to your Colibra Data Intelligence Cloud and Tableau environment. The passwords are encrypted and stored in `/config/pwd.conf`.

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.

After the metadata is ingested in Data Catalog, you can go to the domain (or domains) that you specified in your lineage harvester configuration file and view the newly created assets.

You can also view the [Looker technical lineage](#).

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

Lineage harvester system requirements

You need to meet the system requirements to be able to [install](#) and run the [lineage harvester](#).

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 [Technical lineage general troubleshooting](#).

- 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:
 - Your Collibra Data Intelligence Cloud instance version 2020.12 or newer.
 - 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 for Looker integration

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

Tip

- Install the lineage harvester close to your data source or on the same server.
- The lineage harvester uses port 443.

Prerequisites

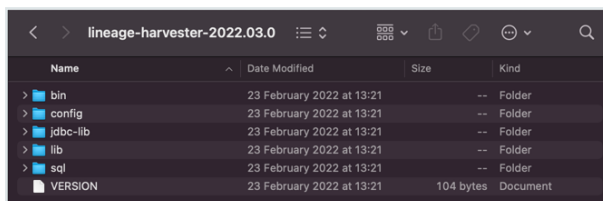
- You have purchased the Looker metadata connector and lineage feature.
- You have Collibra Data Intelligence Cloud 2020.12 or newer.
- You meet the [minimum system requirements](#).
- You have added Firewall rules so that the lineage harvester can connect to:
 - The host names of all databases in the 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

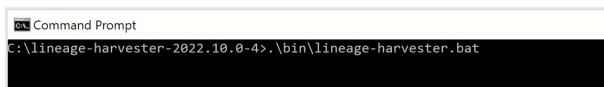
1. [Download](#) the newest lineage harvester.
2. Unzip the archive.

» You can now access the lineage harvester folder. The lineage harvester folder name is unique per version.

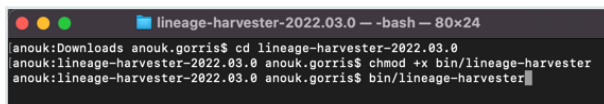


3. Start the lineage harvester to create an empty lineage harvester configuration file by entering the following command:

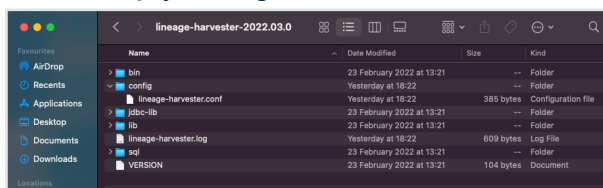
- Windows: `.\bin\lineage-harvester.bat`



- 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.

Prepare the lineage harvester configuration file for Looker

You have to prepare a configuration file before you run the [lineage harvester](#). The lineage harvester collects your Looker metadata and sends it to the [Collibra Data Lineage service](#)

[instance](#), where it is processed and analyzed. Collibra Data Intelligence Cloud then imports the Looker assets and relations to Data Catalog.

Example configuration file

```
{
  "general": {
    "catalog": {
      "url": "https://<organization>.collibra.com",
      "userName": "<your-collibra-username>"
    },
    "useCollibraSystemName": false
  },
  "sources": [{
    "id": "looker-id",
    "type": "Looker",
    "lookerUrl": "https://<instance-name>.api.looker.com",
    "clientId": "looker-api-user-name",
    "clientSecret": "looker-api-userkey",
    "domainId": "domain-resource-id",
    "deleteRawMetadataAfterProcessing": true
  }]
}
```

Before you begin

- [Set up the latest lineage harvester](#).
- [Create one or more BI Data Catalog](#) domains in which you want to ingest the Looker assets.

Requirements and permissions

- Collibra Data Intelligence Cloud.
- 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 BI Data Catalog domain:

- Asset: add
- Attribute: add
- Domain: add
- Attachment: add

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 to US government customers.
url	The URL of the Collibra Data Lineage service instance. "url": "https://techlin-gov.collibra.com" Warning This 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 applies only to US government customers.

Properties	Description
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.
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.

Properties	Description
<code>sources</code>	This section contains all Looker connection properties.
<code>id</code>	The unique ID of your Looker metadata. For example, <i>my_looker</i>. Tip This value can be anything as long as it is unique and human readable. The ID identifies the batch of Looker metadata on the Collibra Data Lineage service. 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.
<code>type</code>	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 /api-docs/. 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.
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>

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.

What's next?

[Prepare](#) the <source ID> configuration file.

Prepare Looker <source ID> 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":"<reference ID>",
      "description":"any-description",
      "folderNames":["Folder1", "Folder2"]
    },
    {
      "domainId":"<reference ID>",
      "description":"any-description",
      "folderNames":["Folder3", "Folder4"]
    },
    {
      "domainId":"<reference ID>",
      "description":"any-description",
      "folderIds":["123xxxx", "456xxxx"]
    }
  ]
}
```

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.	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.	

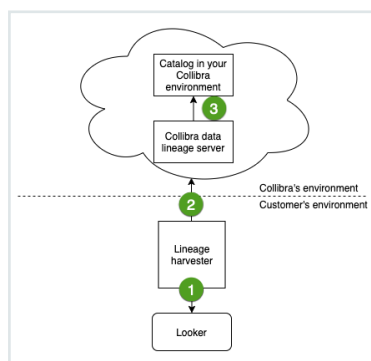
4. Save the <source ID> configuration file.

What's next?

See [Overview Looker integration steps](#).

Looker ingestion workflow

You run the lineage harvester to start the Looker ingestion workflow. When you initiate Looker ingestion, each workflow component performs the following actions:



1. The lineage

harvester:

- Communicate s with Looker.
- Harvests the Looker metadata that will be ingested into Data Catalog.
- Sends the Looker metadata to the [Collibra Data Lineage service instance](#).

2. Collibra:

- Analyzes the Looker metadata.
- Creates new assets and relations.
- Imports new Looker assets and their relations in Data Catalog.

3. Data Catalog:

- Shows new [Looker assets](#).
- Shows a [technical lineage](#) tab on

Looker Look
asset pages.

Schedule Looker ingestion jobs

You can use [Task Scheduler](#) on Windows or [Crontab](#) on Mac and Linux to make the [lineage harvester](#) run scheduled jobs. In a scheduled job, the lineage harvester uploads the Looker data source information to Collibra.

Collibra automatically creates new assets and relations of the type "Data Element targets / sources Data Element" at specific times, dates or intervals, using the information in your [configuration file](#).

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

Warning 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.

Example You created a configuration file with connection information to your Looker environment. You schedule the lineage harvester job to run each Sunday at 23:00. As a result, your Looker metadata is automatically refreshed on a weekly basis.

Troubleshooting

If the connection between the [lineage harvester](#) and your Looker instance fails, try signing in to Looker as an Admin user on the same machine that runs the lineage harvester. Open the [interactive API documentation](#). If you are not able to open the API page, try one of the following:

- Check that you have [network access](#) to the API URL.
- Check that you have the correct [credentials](#) to sign in to the interactive API documentation. If necessary, create new API3 keys and try again. If you are now able to access the interactive API documentation, use the new Client ID and Client Secret in the [configuration file](#).
- Sign in to the interactive API documentation with your API3 credentials and test the API calls. If your test is successful, compare API URL in the Request URL section to the `lookerUrl` value in the [configuration file](#).

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 `/api-docs/`.

Working with MicroStrategy

MicroStrategy Intelligence Server is business intelligence software that connects to data sources to create and store layers of objects in the MicroStrategy metadata. For more information on MicroStrategy, see the [MicroStrategy documentation](#).

Collibra Data Lineage currently supports two means by which to integrate MicroStrategy in Data Catalog. Both integration methods require the lineage harvester:

- Working with MicroStrategy, which is addressed in this section.
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 repository, see the [MicroStrategy repository documentation](#).

Important The operating model associated with this integration method remains valid; however, the new operating model is compatible only with the new integration method.

- [Working with MicroStrategy \(NEW\)\(Beta\)](#)
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.

MicroStrategy terminology	775
MicroStrategy asset and domain types	775
MicroStrategy operating model	777
Soft delete of MicroStrategy assets	785
MicroStrategy integration steps	787

MicroStrategy terminology

The following table shows the MicroStrategy terminology and how it maps to the Collibra Data Intelligence Cloud asset types.

MicroStrategy term	Description	Asset type in Collibra
Attribute	A detailed view of a MicroStrategy visualization, with findings and insights.	MicroStrategy Report Attribute
Column	A column in a MicroStrategy data model.	MicroStrategy Column
Dataset	A collection of data that is used to create MicroStrategy reports.	MicroStrategy Data Model
Dossier	A collection of MicroStrategy chapters and pages.	MicroStrategy Dossier
Folder	A collection of MicroStrategy reports and data models.	MicroStrategy Folder
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

MicroStrategy asset and domain types

The [MicroStrategy](#) integration in Collibra Data Intelligence Cloud uses a specific subset of [asset types](#) and [domain types](#).

The following table shows the asset and domain types that are used for the MicroStrategy integration. Above each asset type you can see the parent asset types in the breadcrumbs.

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
Data Asset › Data Element › Data Attribute › BI Data Attribute › MicroStrategy Data Attribute	A column in a MicroStrategy data model.	BI Catalog

Asset type	Description	Domain type
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

MicroStrategy operating model

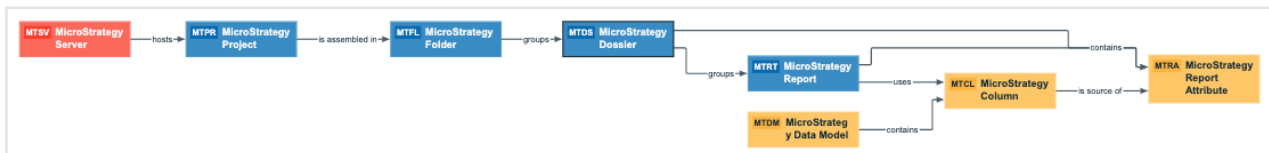
The harvester collects MicroStrategy metadata and sends it to the Collibra Data Lineage service. Collibra processes the metadata and creates new MicroStrategy assets and relations in Data Catalog. You can see them on the asset page overview or visualize them in a [diagram](#).

Note

- The assets have the same names as their counterparts in MicroStrategy. You cannot edit Full names and Names in Data Catalog.
- 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.

MicroStrategy metadata overview

The following image shows the relations between MicroStrategy asset types.



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.

Asset type	Harvested MicroStrategy metadata in Data Catalog	Resource ID
MicroStrategy Column Resource ID: 00000000-0000-0000-0000-100000000047	Description	00000000-0000-0000-0000-0000000003114
	BI Data Model contains / is part of BI Data Attribute	00000000-0000-0000-0000-0000000007196
	Report Attribute sourced from / is source of Data Attribute	00000000-0000-0000-0000-120000000010

Asset type	Harvested MicroStrategy metadata in Data Catalog	Resource ID
MicroStrategy Data Model Resource ID: 00000000-0000-0000-0000-100000000046	Certified (not available yet)	00000000-0000-0000-0001-000500000001
	Description	00000000-0000-0000-0000-000000003114
	BI Data Model contains / is part of BI Data Attribute	00000000-0000-0000-0000-000000007196
MicroStrategy Dossier Resource ID: 00000000-0000-0000-0000-100000000043	Certified (not available yet)	00000000-0000-0000-0001-000500000001
	Description	00000000-0000-0000-0000-000000003114
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	Report Attribute contained in / contains Report	00000000-0000-0000-0000-000000007058
	Report groups / is grouped into Report	00000000-0000-0000-0000-120000000004
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
	BI Folder contains / contained in Data Asset	00000000-0000-0000-0000-120000000014
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002

Asset type	Harvested MicroStrategy metadata in Data Catalog	Resource ID
MicroStrategy Project Resource ID: 00000000-0000-0000-0000-100000000041	Description	00000000-0000-0000-0000-000000003114
	Document creation date	00000000-0000-0000-0000-000000000260
	Document modification date	00000000-0000-0000-0000-000000000261
	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
MicroStrategy Report Resource ID: 00000000-0000-0000-0000-100000000044	Certified (not available yet)	00000000-0000-0000-0001-000500000001
	Description	00000000-0000-0000-0000-000000003114
	Business Dimension groups / is grouped into Report	00000000-0000-0000-0000-120000000002
	Report Attribute contained in / contains Report	00000000-0000-0000-0000-000000007058
	Report groups / is grouped into Report	00000000-0000-0000-0000-120000000004

Asset type	Harvested MicroStrategy metadata in Data Catalog	Resource ID
MicroStrategy Report Attribute Resource ID: 00000000-0000-0000-0000-100000000045	Description	00000000-0000-0000-0000-000000003114
	Role in Report	00000000-0000-0000-0000-000000000266
	BI Folder contains / contained in Data Asset	00000000-0000-0000-0000-120000000014
	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
MicroStrategy Server Resource ID: 00000000-0000-0000-0000-100000000040	Description	00000000-0000-0000-0000-000000003114
	Server hosts / is hosted in Business Dimension	00000000-0000-0000-0000-120000000000

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. In the tab pane, click  **Diagram**.
 - » The diagram appears 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 Dossier",
      "type": {
        "id": "00000000-0000-0000-0000-100000000043"
      }
    },
    {
      "id": "MicroStrategy Report",
      "type": {
        "id": "00000000-0000-0000-0000-100000000044"
      }
    },
    {
      "id": "MicroStrategy Column",
      "type": {
        "id": "00000000-0000-0000-0000-100000000047"
      }
    },
    {
      "id": "MicroStrategy Data Model",
      "type": {
        "id": "00000000-0000-0000-0000-100000000046"
      }
    },
    {
      "id": "MicroStrategy Folder",
      "type": {
        "id": "00000000-0000-0000-0000-100000000042"
      },
      "display": "expanded",
      "layoutRegion": "flow"
    },
    {
      "id": "MicroStrategy Project",
      "type": {
        "id": "00000000-0000-0000-0000-100000000041"
      }
    },
    {

```



```

      "id": "MicroStrategy Server",
      "type": {
        "id": "000000000-0000-0000-0000-1000000000040"
      }
    },
    {
      "id": "MicroStrategy Report Attribute",
      "type": {
        "id": "000000000-0000-0000-0000-1000000000045"
      }
    }
  ],
  "edges": [
    {
      "from": "MicroStrategy Dossier",
      "to": "MicroStrategy Report",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "000000000-0000-0000-0000-1200000000004"
      },
      "roleDirection": true
    },
    {
      "from": "MicroStrategy Data Model",
      "to": "MicroStrategy Column",
      "label": "",
      "style": "boxing",
      "type": {
        "id": "000000000-0000-0000-0000-0000000007196"
      },
      "roleDirection": true
    },
    {
      "from": "MicroStrategy Folder",
      "to": "MicroStrategy Dossier",
      "label": "",
      "style": "boxing",
      "type": {
        "id": "000000000-0000-0000-0000-1200000000002"
      },
      "roleDirection": true
    },
    {
      "from": "MicroStrategy Report",
      "to": "MicroStrategy Column",
      "label": "",
      "style": "arrow",
      "type": {
        "id": "000000000-0000-0000-0000-1200000000021"
      }
    }
  ]
}
```

```

    },
    "roleDirection": true
  },
  {
    "from": "MicroStrategy Project",
    "to": "MicroStrategy Folder",
    "label": "",
    "style": "boxing",
    "type": {
      "id": "000000000-0000-0000-0000-120000000001"
    },
    "roleDirection": false
  },
  {
    "from": "MicroStrategy Server",
    "to": "MicroStrategy Project",
    "label": "",
    "style": "boxing",
    "type": {
      "id": "000000000-0000-0000-0000-120000000000"
    },
    "roleDirection": true
  },
  {
    "from": "MicroStrategy Column",
    "to": "MicroStrategy Report Attribute",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-120000000010"
    },
    "roleDirection": false
  },
  {
    "from": "MicroStrategy Report",
    "to": "MicroStrategy Report Attribute",
    "label": "",
    "style": "boxing",
    "type": {
      "id": "000000000-0000-0000-0000-000000007058"
    },
    "roleDirection": false
  },
  {
    "from": "MicroStrategy Dossier",
    "to": "MicroStrategy Report Attribute",
    "label": "",
    "style": "arrow",
    "type": {
      "id": "000000000-0000-0000-0000-000000007058"
    }
  }

```

```

    },
    "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
  }
}
}

```

Soft delete of MicroStrategy assets

When you integrate MicroStrategy, 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 API considers the data object as new and creates another corresponding asset in Data Catalog. The result is two assets for the same data object; one with the status “Missing from source”, one with the status “Candidate”. As described in the following section, you can then run a workflow to delete the “Missing from source” assets, or you can manually delete them.

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 [Collibra Developer Portal](#) and deploy it in your Collibra 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.

MicroStrategy integration steps

The MicroStrategy integration in Collibra Data Intelligence Cloud enables you to harvest MicroStrategy Intelligence Server metadata and create new MicroStrategy assets in Data Catalog. Collibra analyzes and processes the BI metadata and presents it as [assets of specific types](#), retaining their original names.

Important In the global [assignment](#) of each asset type included in the MicroStrategy [operating model](#), ensure that none of the characteristics that are in the operating model have a maximum cardinality of “0”. If the maximum cardinality is set to “0” for any such characteristics, ingestion will fail.

Roles, privileges and permissions in MicroStrategy

To ingest MicroStrategy metadata in Data Catalog, the lineage harvester connects to the MicroStrategy Intelligence Server or a remote PostgreSQL or Microsoft SQL Server repository, depending on where you install the lineage harvester. You must have a role with user access to the relevant server and be able to access the metadata that is stored there.

Steps

1. [Prepare](#) a domain for MicroStrategy ingestion.
If you already have a domain reserved for ingestion of the new MicroStrategy assets, you can skip this step.
2. [Download](#) and [install](#) the lineage harvester.
3. [Prepare](#) the lineage harvester configuration file.
4. Start the lineage harvester again in the console and run the following command:
 - for Windows: `.\bin\lineage-harvester.bat full-sync`
 - for other operating systems: `./bin/lineage-harvester full-sync`

When prompted, enter the password or client secret to connect to your Collibra Data Intelligence Cloud and Tableau environment. The passwords are encrypted and stored in `/config/pwd.conf`.

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.

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.

Warning We strongly recommend that you not edit the full names of any Tableau 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 another domain. If you do, the assets will be deleted and recreated in the initial Data Catalog BI domain when you [synchronize](#) MicroStrategy. As a consequence, any manually added data of those assets is lost.

Lineage harvester system requirements

You need to meet the system requirements to be able to [install](#) and run the [lineage harvester](#).

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 [Technical lineage general troubleshooting](#).

- 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 the 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.

Prepare a domain for MicroStrategy ingestion

Before you can ingest MicroStrategy metadata, you have to create a new domain or choose an existing domain to store the new MicroStrategy assets. You specify the domain reference ID in the [lineage harvester configuration file](#). Collibra uses the domain to ingest all MicroStrategy assets.

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 a domain type from the list.

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
Type	The domain type of the domain you are creating. In this case, you need to select <i>BI Catalog</i> .
Community	The community under which the domain will be located.
Name	The name of the new domain.

5. Click **Create**.
6. Open your domain.
7. 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=00000000-0000-0000-0000-000000040001`. In this example, the domain ID is in bold.

8. Paste the domain ID in the [lineage harvester configuration file](#).

Install the lineage harvester

Before you can use the lineage harvester, you have to [download](#) it and install it.

Note We highly recommend that you always install and use the newest lineage harvester.

Where you choose to install the lineage harvester depends on the repository that the lineage harvester will access to harvest the metadata. You can install it either:

- [On the MicroStrategy server](#), to access the local PostgreSQL or Microsoft SQL Server repository.
- [Close to your data source](#), to access a remote PostgreSQL or Microsoft SQL Server repository.

Important

- If you have a MicroStrategy on-premises environment, you can install the lineage harvester on a server that has access to the MicroStrategy server or on the MicroStrategy server itself.
- If you use MicroStrategy cloud:
 - To connect to a remote PostgreSQL or Microsoft SQL Server repository, you have to install the lineage harvester where it can access the database.
 - To access the local PostgreSQL or Microsoft SQL Server repository, you have to install the lineage harvester on the MicroStrategy server.
- 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.

Install the lineage harvester on the MicroStrategy server

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](#).

Prerequisites

- You have Collibra Data Intelligence Cloud 2021.07 or newer.
- You have MicroStrategy 2021 or newer.
- If you intend to access a Microsoft SQL Server repository, you need the Admin permission for the repository.
- You meet the [minimum lineage harvester system requirements](#).
- Java Runtime Environment version 11 or newer or OpenJDK 11 or newer.

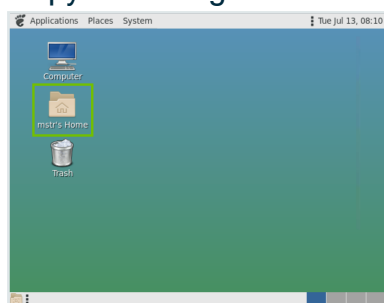
- You have added Firewall rules so that the lineage harvester can connect to:
 - The host names of all databases in the lineage harvester configuration file.
 - All [Collibra Data Lineage service instances](#) within your geographical location:
 - 15.222.200.199 (techlin-aws-ca.collibra.com)
 - 18.198.89.106 (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

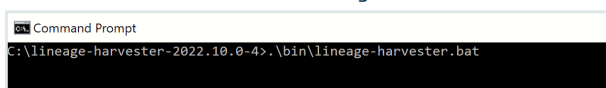
1. [Download](#) the lineage harvester.
2. Sign in to the MicroStrategy web portal.
3. Click **Remote Desktop Gateway**.
4. Sign in to **Apache Guacamole**.
5. Click **Platform Instance VNC**.

6. Copy the lineage harvester ZIP file to the Platform Instance VNC home directory.

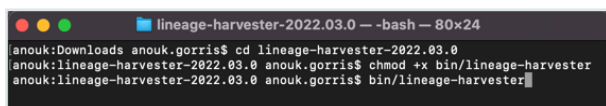


7. Unzip the archive.
 - » You can now access the lineage harvester folder.
8. Start the lineage harvester to create an empty lineage harvester configuration file by entering the following command:

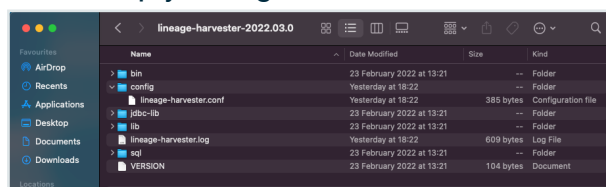
- Windows: `.\bin\lineage-harvester.bat`



- 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`.

Install the lineage harvester close to your data source

To access a remote PostgreSQL or remote Microsoft SQL Server repository, install the lineage harvester close to the data source.

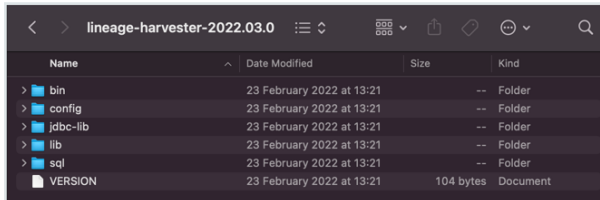
Prerequisites

- You have Collibra Data Intelligence Cloud 2021.07 or newer.
- You have MicroStrategy 2021 or newer.
- If you intend to access a Microsoft SQL Server repository, you need the Admin permission for the repository.
- You meet the [minimum lineage harvester system requirements](#).
- Java Runtime Environment version 11 or newer or OpenJDK 11 or newer.
- 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.

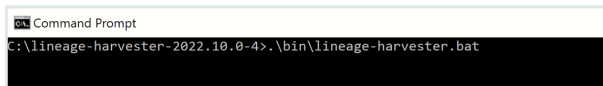
Steps

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.

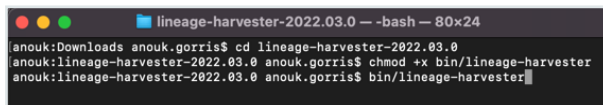


3. Start the lineage harvester to create an empty lineage harvester configuration file by entering the following command:

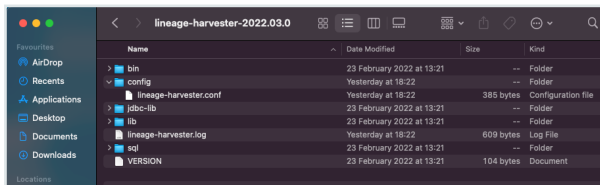
- Windows: `.\bin\lineage-harvester.bat`



- 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`.

Prepare the lineage harvester configuration file for MicroStrategy

You have to prepare a configuration file before you run the lineage harvester. The lineage harvester collects your MicroStrategy metadata and sends it to the [Collibra Data Lineage](#)

[service](#), where it is processed and analyzed. Collibra Data Intelligence Cloud then imports the [MicroStrategy assets and relations](#) to Data Catalog.

Example configuration file

```
{
  "general": {
    "catalog": {
      "url": "https://<organization>.collibra.com",
      "userName": "<your-collibra-username>"
    },
    "useCollibraSystemName": false
  },
  "sources": {
    "type": "Microstrategy",
    "id": "microstrategy-batch",
    "domainId": "<domain-resource-id>",
    "username": "mstr",
    "hostname": "remote.postgres.com",
    "port": 5432,
    "databaseName": "poc_metadata",
    "deleteRawMetadataAfterProcessing": true
  }
}
```

Before you begin

- [Set up the latest lineage harvester](#).
- Create a [BI Catalog domain](#) in which you want to ingest the MicroStrategy

Requirements and permissions

- Collibra Data Intelligence Cloud.
- 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 BI Data Catalog domain:

- Asset: add
- Attribute: add
- Domain: add
- Attachment: add
- Necessary permissions to all database objects that the lineage harvester accesses.
 - For 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 this configuration file must be the owner of the views in PostgreSQL.
 - For Microsoft SQL Server repository:
 - You need read access on the SYS schema and the View Definition Permission in your SQL Server.

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 to US government customers.
url	The URL of the Collibra Data Lineage service instance. "url": "https://techlin-gov.collibra.com" Warning This applies only to US government customers.

Properties	Description
<code>userKey</code>	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 to US government customers.
<code>catalog</code>	This section contains information that is necessary to connect to Data Catalog.
<code>url</code>	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.
<code>username</code>	The username that you use to sign in to Collibra.
<code>useCollibraSystemName</code>	By default, the <code>useCollibraSystemName</code> property is set to <code>false</code> . This property is not valid for MicroStrategy integration. We recommend that you leave this property set to <code>false</code> .
<code>sources</code>	This section contains all MicroStrategy connection properties.
<code>type</code>	The kind of data source. In this case, the value has to be <code>MicroStrategy</code> .

Properties	Description
id	The unique ID of your MicroStrategy metadata. 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. Tip This value can be anything as long as it is unique and human readable. The ID identifies the batch of MicroStrategy metadata on the Collibra Data Lineage service.
domainId	The unique reference ID of the domain in Collibra Data Intelligence Cloud in which you want to ingest the MicroStrategy assets .
username	The username that you use to sign in to MicroStrategy.
hostname	The endpoint that you use to access the PostgreSQL repository or remote data source, depending on where you installed the lineage harvester. For example <code>remote.postgres.com</code>.
port	The port number.
databaseName	Optionally, the name of your database. For example <code>poc_metadata</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.

Schedule MicroStrategy ingestion jobs

You can use [Task Scheduler](#) on Windows or [Crontab](#) on Mac and Linux to make the lineage harvester run scheduled jobs. In a scheduled job, the lineage harvester uploads the MicroStrategy data source information to Collibra.

Collibra automatically creates [new assets and relations](#) between the MicroStrategy assets at specific times, dates or intervals, using the information in your [configuration file](#).

Example You created a configuration file with connection information to your MicroStrategy environment. You schedule the lineage harvester job to run each Sunday at 23:00. As a result, your MicroStrategy metadata is automatically refreshed on a weekly basis.

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

Warning Relations that were manually created between MicroStrategy assets and other assets via a relation type in the [MicroStrategy operating model](#), are deleted after a refresh of the MicroStrategy metadata.

Working with MicroStrategy (NEW) (Beta)

The MicroStrategy integration registers MicroStrategy as a data source in Collibra and creates a technical lineage.

Collibra Data Lineage currently supports two means by which to integrate MicroStrategy in Data Catalog. Both integration methods require the lineage harvester:

- [Working with MicroStrategy](#)
 - Working with MicroStrategy (NEW, beta), which is addressed in this section.
- 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.

For more information on MicroStrategy, see the [MicroStrategy documentation](#).

MicroStrategy terminology (NEW)(Beta)	804
MicroStrategy asset and domain types (NEW)(Beta)	806
MicroStrategy operating model (NEW)(Beta)	808
Technical lineage for MicroStrategy (NEW)(Beta)	820
MicroStrategy integration steps (NEW)(Beta)	823

MicroStrategy terminology (NEW)(Beta)

The following table shows the MicroStrategy terminology and how it maps to the Collibra Data Intelligence Cloud asset types.

MicroStrategy data object	Description	Asset type in Colibra
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
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
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

MicroStrategy asset and domain types (NEW)(Beta)

The [MicroStrategy](#) integration in Collibra Data Intelligence Cloud uses a specific subset of [asset types](#) and [domain types](#).

The following table shows the asset and domain types that are used for the MicroStrategy integration. Above each asset type you can see the parent asset types in the breadcrumbs.

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

MicroStrategy operating model (NEW)(Beta)

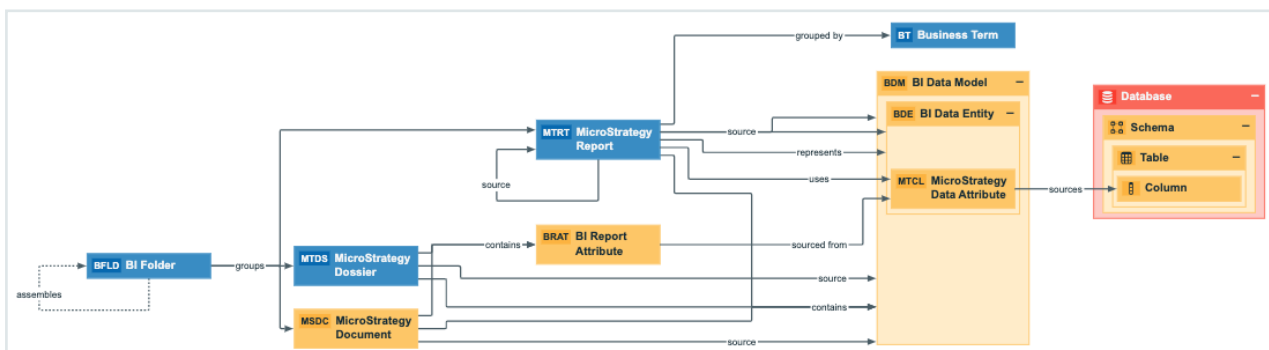
When you integrate MicroStrategy, Collibra Data Lineage creates new MicroStrategy assets and relations in Data Catalog. You can see them on the asset page overview or visualize them in a [diagram](#).

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.

MicroStrategy metadata overview

The following image shows the relations between MicroStrategy asset types.



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
	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
	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 / source 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	Description	00000000-0000-0000-0000-000000003114
	BI Folder contains / contained in Data Asset	00000000-0000-0000-0000-120000000014
	Data Model contains / is part of Data Entity	00000000-0000-0000-0000-000000007046
MicroStrategy Document Resource ID: 00000000-0000-0000-0000-100000000049	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	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	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	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 / source 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

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. In the tab pane, click  **Diagram**.
 - » The diagram appears 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
      }
    }
  }
}

```

Technical lineage for MicroStrategy (NEW) (Beta)

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:

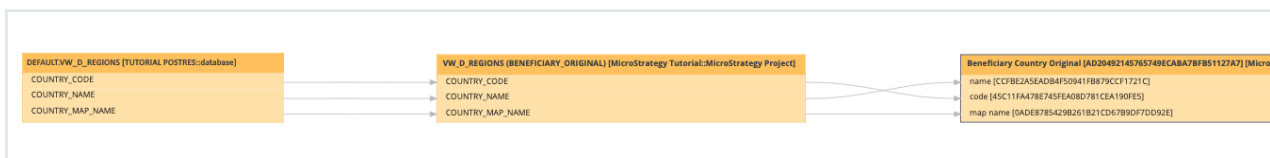
- Table
- Column
- MicroStrategy Data Attribute
- MicroStrategy Report

The **Technical lineage** tab is only shown if you have the Data Catalog global role with the Catalog and Technical lineage [global permissions](#).

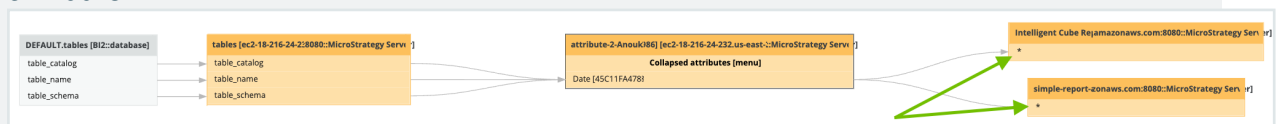
Note If you ingest MicroStrategy for the first time or if you change your geolocation or cloud provider, you have to restart the DGC service before you can see the technical lineage.

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](#).



Note 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.



UUIDs in the MicroStrategy technical lineage

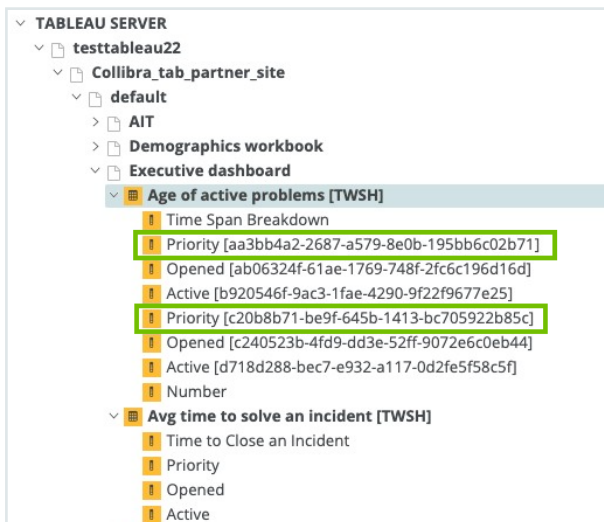
Collibra Data Lineage uses unique full names to create a technical lineage and stitch objects within the technical lineage. Full names in Collibra are constructed in accordance with the hierarchy of data objects in MicroStrategy, for example:

- 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 transformation details that were analyzed and processed on the [Collibra Data Lineage service instance](#) and the results of this analysis. The success rate of the analysis indicates how complete the technical lineage is.

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](#).

Sources		Stitching				
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 transformations	Full-text search		Filter by	All ▼

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

MicroStrategy integration steps (NEW)(Beta)

The MicroStrategy integration in Collibra Data Intelligence Cloud enables you to harvest MicroStrategy Intelligence Server metadata and create new MicroStrategy assets in Data Catalog. Collibra analyzes and processes the BI metadata and presents it as [assets of specific types](#), retaining their original names.

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.

Steps

1. [Prepare](#) a domain for MicroStrategy ingestion.
If you already have a domain reserved for ingestion of the new MicroStrategy assets, you can skip this step.
2. [Prepare](#) the Data Catalog physical data layer.
3. [Download](#) and [install](#) the lineage harvester.
4. [Prepare](#) the lineage harvester configuration file.
5. [Prepare](#) the <source ID> configuration file.
6. Start the lineage harvester again in the console and run the following command:

- for Windows: `.\bin\lineage-harvester.bat full-sync`
- for other operating systems: `./bin/lineage-harvester full-sync`

When prompted, enter the password or client secret to connect to your Collibra Data Intelligence Cloud and Tableau environment. The passwords are encrypted and stored in `/config/pwd.conf`.

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.

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 [MicroStrategy technical lineage](#).

Warning We strongly recommend that you not edit the full names of any MicroStrategy 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 another domain. If you do, the assets will be deleted and recreated in the initial Data Catalog BI domain when you [synchronize](#) MicroStrategy. As a consequence, any manually added data of those assets is lost.

Prepare a domain for MicroStrategy ingestion

Before you can ingest MicroStrategy metadata, you have to create a new domain or choose an existing domain to store the new MicroStrategy assets. You then have to specify the domain reference ID in the [lineage harvester configuration file](#). Collibra uses the domain to ingest all MicroStrategy assets.

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 a domain type from the list.

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
Type	The domain type of the domain you are creating. In this case, you need to select <i>BI Catalog</i> .
Community	The community under which the domain will be located.
Name	The name of the new domain.

5. Click **Create**.

6. Open your domain.
7. 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.

8. Paste the domain ID in the [lineage harvester configuration file](#).

Install the lineage harvester

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

Tip

- Install the lineage harvester close to your data source or on the same server.
- The lineage harvester uses port 443.

Prerequisites

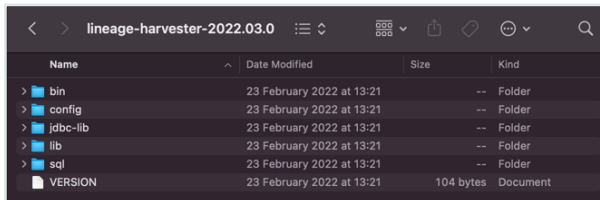
- You have Collibra Data Intelligence Cloud 2021.07 or newer.
- You have MicroStrategy 2021 or newer.
- If you intend to access a Microsoft SQL Server repository, you need the Admin permission for the repository.
- You meet the [minimum lineage harvester system requirements](#).
- Java Runtime Environment version 11 or newer or OpenJDK 11 or newer.
- 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.

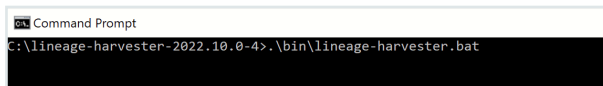
Steps

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.

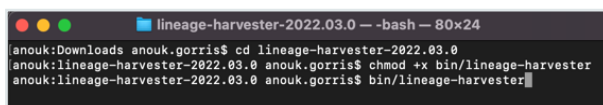


3. Start the lineage harvester to create an empty lineage harvester configuration file by entering the following command:

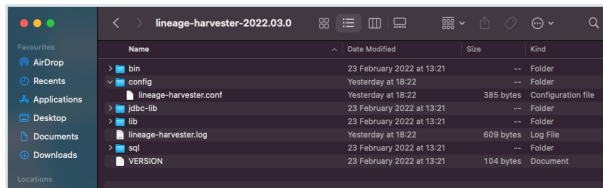
- **Windows:** `.\bin\lineage-harvester.bat`



- **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.

Prepare the Data Catalog physical data layer for MicroStrategy stitching

Before you can stitch data objects to the assets in Collibra Data Intelligence Cloud, you must prepare the Data Catalog physical data layer to create assets and the database > schema > table > column hierarchy.

If you set the `useCollibraSystemName` property to `true` in your [lineage harvester configuration file](#), you also have to create a System asset. For more information, see [Automatic stitching](#).

Important In the global [assignment](#) of each asset type included in the MicroStrategy [operating model](#), ensure that none of the characteristics that are in the operating model have a maximum cardinality of “0”. If the maximum cardinality is set to “0” for any such characteristics, ingestion will fail.

Prerequisites

- You have a [global role](#) with the Catalog [global permission](#), for example, Catalog Author.
- You have a role with the following [resource permissions](#) on the **Schema** community:
 - Asset: add
 - Attribute: add

- Domain: add
- Attachment: add

Note These permissions are always necessary when [registering a data source](#).

Steps

1. Create a System asset:

Note Carry out this step only if you have set the `useCollibraSystemName` property to `true` in your lineage harvester configuration file.

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

Field	Description
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.
Name	A name to identify 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. **Register** a database as data source. You can register a database or an SQL directory as **data source**.
 - » After registration, the assets of the following asset types are created in Data Catalog:
 - Schema
 - Table
 - Column

Tip The full name of your Schema asset must match the exact name of the schema in the data source that you register in the configuration file.

3. Create a Database asset:

Tip The full name of your Database asset must match the exact name of the database or project, in case of Google BigQuery, that you register in the configuration file.

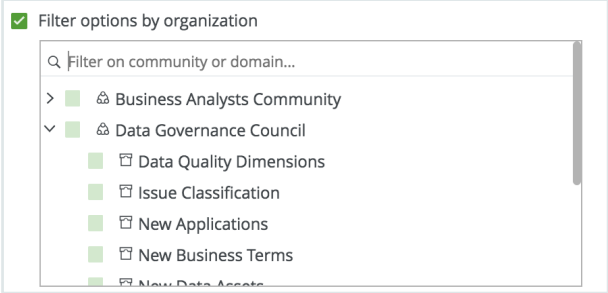
- a. Open the product for which you want to create an asset (for example, **Business Glossary**).
- b. On the main toolbar, click **+**.
 - » The **Create** dialog box appears.
- c. On the **Assets** tab, click Database.
 - » The **Create Asset** dialog box appears.
- d. Enter the required information.

Field	Description
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
Name	A name to identify 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.
4. Create a relation between the System asset and the Database asset using the "Technology Asset groups / is grouped by Technology Asset" relation type.
 - a. In the tab pane, click **Add Characteristic**.
 - » The **Add a characteristic** dialog box appears.
 - b. Click **Relations**.
 - c. Search for and click **groups Technology asset**.
 - » The **Add groups Technology asset** dialog box appears.

d. Enter the required information.

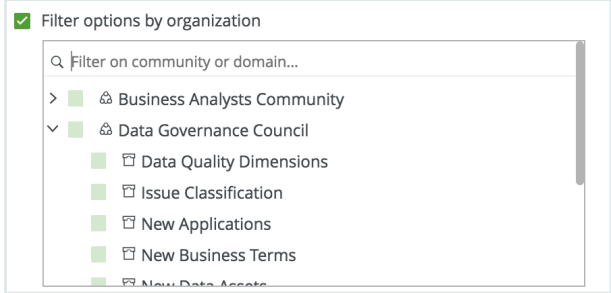
Option	Description
Assets	The name of the database.
Filter suggested assets by organization	Option to filter the suggestions based on selected communities and domains. If this option is selected, the organization tree appears. You can then filter and select domains and communities. 
Start date	Optionally enter the date on which the relation between the assets becomes applicable. Leave this field empty to create a permanent relation.
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**.

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

- 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.
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. Leave this field empty to create a permanent relation.
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, you can go to the [asset page](#) of a MicroStrategy Report or MicroStrategy Data Attribute asset and view the technical lineage.

The lineage harvester also uses [scheduled jobs](#) to automate the technical lineage process.

Prepare the lineage harvester configuration file for MicroStrategy (NEW)(Beta)

Before you run the lineage harvester, you have to prepare the lineage harvester configuration file to provide the connection information that you need to connect your MicroStrategy server and remote data source to the [Collibra Data Lineage service instance](#) and [domain](#) in which you want to ingest the MicroStrategy assets.

Example configuration file

```
{
  "general":{
    "techlin": {
      "url": "https://techlin-gcp-dev.collibra.com",
      "userKey": "xxxxxxxxxx"
    },
    "catalog":{
      "url":"https://infra-xxxxx.collibra.com",
      "username":"Admin"
    },
    "useCollibraSystemName": false
  },
  "sources":[
    {
      "id": "mstr",
      "type": "MSTR_V2",
      "url": "http://ec2-18-216-24-232.us-east-2.com-pute.amazonaws.com:8080",
      "username": "Administrator",
      "maxParallelRequests": 5,
      "appUrlSuffix": "MicroStrategy/servlet/mstrWeb"
    }
  ]
}
```

Before you begin

- [Install](#) the newest lineage harvester.
- [Prepare a BI Catalog domain](#) in which you want to ingest MicroStrategy metadata.
- [Prepare the Data Catalog physical data layer](#).

Requirements and permissions

- Collibra Data Intelligence Cloud.
- 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 BI Data Catalog domain:
 - Asset: add
 - Attribute: add
 - Domain: add
 - Attachment: add
- Necessary permissions to all [database objects](#) that the lineage harvester accesses.
- In MicroStrategy:
 - Admin API permissions.
 - 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.

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 to US government customers.

Properties	Description
url	The URL of the Collibra Data Lineage service instance. "url": "https://techlin-gov.collibra.com" Warning This 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 applies only to US 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 Power BI <source-ID> configuration file.
sources	This section contains all MicroStrategy connection properties.

Properties	Description
id	The unique ID of your MicroStrategy metadata. 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. Tip This value can be anything as long as it is unique and human readable. The ID identifies the batch of MicroStrategy metadata on the Collibra Data Lineage service.
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.
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
<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.
<code>appUrlSuffix</code>	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: <code>"appUrlSuffix": "MicroStrategy/servlet/mstrWeb"</code> - For .NET, use: <code>"appUrlSuffix": "MicroStrategy/asp/Main.aspx"</code>

2. Save the configuration file.

What's next?

Prepare your <source ID> configuration file.

Prepare the MicroStrategy <source ID> configuration file (NEW)(Beta)

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.
- Optionally, specify from which MicroStrategy projects you want to ingest metadata.
- 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.

Tip "<source ID>" refers to the value of the `Id` property in the lineage harvester configuration file.

Example

```
{
  "default_domain_id": "1a0a942e-e3a7-45a1-83e8-ade30b1cabla",
  "filters": [
    {
      "projectIds": [],
      "projectNames":
["Customers", "Research", "Marketing"]
    }
  ],
  "datasourceMapping": [
    {
      "found_datasource": "REDSHIFT",
      "found_project": "*",
      "mapping": {
        "dbname": "RD_pearl",
        "schema": "Default_North",
        "dialect": "spark",
        "collibraSystemName": "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`.

3. For each database in MicroStrategy, 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	Mandatory
default_domain_id	The domain in which you want the corresponding assets of MicroStrategy metadata to be ingested.	Yes

Property	Description	Mandatory
filters	This section allows you to specify from which MicroStrategy projects you want to harvest metadata. All metadata is ingested into the default domain, as specified via the <code>default_domain_id</code> property. If you don't want to filter on projects, don't include this section in your <source ID> configuration file.	No
projectIds	The IDs of the MicroStrategy projects from which you want to ingest metadata.	No
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.	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_data- source": "Customers", "found_pro- ject": "Project_A", "mapping": { "dbname": "Customers", "col- libraSystemName": "Cus- tomers_North" } } </pre>	Yes

Property	Description	Mandatory
	<pre> }, { "found_data- source": "Customers", "found_pro- ject": "Project_B", "mapping": { "dbname": "Customers", "col- libraSystemName": "Cus- tomers_South" } }]</pre> Warning The values of this property must exactly match the name of your System asset in Collibra.	

4. Save the <source ID> configuration file.

What's next?

Run the lineage harvester. For the required commands, see [Overview MicroStrategy integration steps \(NEW, beta\)](#).

Run the lineage harvester

After you have prepared the lineage harvester configuration file and the <source ID> configuration file, you can run the lineage harvester. The lineage harvester collects your MicroStrategy metadata and sends it to the Collibra Data Lineage service instance, where it is processed and analyzed. Collibra Data Intelligence Cloud then imports the [MicroStrategy assets and relations](#) to Data Catalog.

Before you begin

- [Install](#) the newest lineage harvester.
- Prepare a [BI Catalog domain](#) in which you want to ingest MicroStrategy metadata.
- Prepare the [Data Catalog physical data layer](#).
- Prepare the [<source ID> configuration file](#).

Requirements and permissions

- 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

Steps

1. Start the lineage harvester in the console and run the following command:
 - for Windows: `.\bin\lineage-harvester.bat full-sync`
 - for other operating systems: `./bin/lineage-harvester full-sync`
2. When prompted, enter the password or client secret to connect to your Collibra Data Intelligence Cloud and MicroStrategy environment.
 - » The passwords are encrypted and stored in `/config/pwd.conf`

What's next?

- Go to the domain in which you ingested the assets, to see the ingestion results.
- [View](#) the technical lineage.

Schedule MicroStrategy ingestion jobs

You can use [Task Scheduler](#) on Windows or [Crontab](#) on Mac and Linux to make the lineage harvester run scheduled jobs. In a scheduled job, the lineage harvester uploads the MicroStrategy data source information to Collibra.

Collibra automatically creates [new assets and relations](#) between the MicroStrategy assets at specific times, dates or intervals, using the information in your [configuration file](#).

Example You created a configuration file with connection information to your MicroStrategy environment. You schedule the lineage harvester job to run each Sunday at 23:00. As a result, your MicroStrategy metadata is automatically refreshed on a weekly basis.

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

Warning Relations that were manually created between MicroStrategy assets and other assets via a relation type in the [MicroStrategy operating model](#), are deleted after a refresh of the MicroStrategy metadata.