



DATABLAU · DAM

Metadata management

Product module user manual

Version 9.1.2

Contents

01	Metadata management
02	Metadata management quick start
03	View basic lineage from metadata objects
04	Configure metadata harvesting
05	Find and maintain table and column metadata
06	Maintain metadata objects
07	Harvest custom metamodel data
08	Understand metamodels and built-in models
09	Design a custom metamodel
10	Import and validate custom metamodel data
11	Review model changes and metadata
12	Manage report and file metadata
13	Configure source account permissions
14	Map columns to data standards

Metadata management

Register, harvest, organize, and search metadata for databases, reports, files, APIs, and enterprise objects, and view basic lineage from those objects.

Metadata describes resources: their names, structures, sources, meanings, and ownership. Harvesting or editing descriptions does not read or change business records. See [Metamodels](#) for enterprise-specific structures.

Workspace	Pages	Purpose
Metadata	Overview, Datasets, Reports, Files	Query registered or harvested objects
Harvesting Management	Systems, Data Sources, Harvesting Management, Metamodels	Prepare and execute harvesting
Lineage Management	Lineage Spaces, Preprocessing, Namespaces	Manage basic metadata lineage associations

Prepare sources and plugins, select a harvesting type, execute, then query the appropriate area. File inputs such as dictionaries, DDL, and APIs create structured objects rather than Files entries.

Choose the source and result location

Content	Entry point	Results
Relational/nonrelational databases and platforms	Data Source	Datasets
Supported FineBI, FineReport, SmartBI reports	Report	Reports
Custom reports created/imported in DAM or SQLink	Manual report maintenance	Reports
DDM dictionaries and DDL	File	Parsed objects in Datasets, not raw files in Files
Server files	File	Files
API and metamodel objects	Applicable file, source, or metamodel path	Datasets

Use the decision table in [Harvesting](#) if unsure. Refer to DDM documentation for model design, comparison, and synchronization. After model changes, harvest again and verify metadata, standards, and lineage.

Choose a reading path

Start with [Quick start](#), then choose a task.

Goal	Guide
Systems and connections	Application systems and sources
Harvest databases, reports, files, custom objects	Harvesting
Least-privilege source account	Account permissions
Search, edit, export, bulk update	Tables and columns
Maintain objects after changes	Object maintenance , Model change review
Standardize semantics and values	Standard mappings
Reports and files	Report and file metadata
Upstream and downstream	Basic lineage
Custom structures	Metamodel design
Built-in and custom models	Metamodel concepts
Custom object instances	Metamodel harvesting
Import and lineage synchronization	Metamodel import

View the metadata overview

The overview shows systems, harvesting, standards, rules, and reports. Edit mode changes widgets and layout. Background jobs calculate statistics; do not replace result verification with recently saved overview counts.

Connection coverage is systems with data sources divided by all systems. Standard coverage is mapped columns divided by all columns. Standards validation coverage is fully conforming mapped columns divided by all mapped columns. System data source counts include only harvested sources.

Understand the SQLink boundary

Metadata lineage supports upstream/downstream inspection. A Lineage Management license is different from SQLINK. See [Data lineage](#) for maps, scenarios, advanced parsing, and impact analysis.

Metadata management quick start

Harvest one source, verify one table, and map one column to a standard. Prepare a system, source account, and accessible standards.

1. Prepare a source

Configure [Systems and data sources](#), test the connection, and choose a few known tables for verification.

2. Configure and run harvesting

Select the source and schema in [Harvesting](#), review scope, create a task, and inspect run history. Saving configuration does not complete harvesting.

3. Check tables and columns

Find the table in Datasets. Follow [Metadata maintenance](#) to verify columns, types, lengths, and properties, and add Chinese names and descriptions.

4. Map a standard

For a column with confirmed meaning, use [Standard mappings](#). Verify the association in table details and the standard's reference information.

Select Business Glossary

Business Glossary

Directory

基础标准

制造业

零售业

保险业

银行业

Business Glossary,English name,Reference

Select Business Department

Query

Business Glossary Name	Business Glossary Code	Theme	Business Department
风险预警信号	CCUST_RISK_PRE_SIGN	制造业	金融科技部
☐ 证件类型	CCUST_CERT_TYPE	银行业	数据管理部
主机	DB-00000000003	制造业	金融科技部
机构名称	STD_BK_ORG_NAME	银行业	零售金融部
业务概率	STD_BK_TX_PROB	银行业	零售金融部
交易金额	STD_BK_TX_AMT	银行业	零售金融部
交易笔数	STD_BK_TX_CNT	银行业	零售金融部
业务标志	STD_BK_BIZ_FLAG	银行业	零售金融部
交易日期	STD_BK_TX_DATE	银行业	零售金融部
城市名称	STD_BK_CITY_NAME	银行业	零售金融部

Total 642

20/page

1 / 33 Page

Close

Select

Verify candidate names, codes, and meanings before selecting. Example objects illustrate the interface. Click to open the full image.

5. Inspect relationships and schedule updates

If parsed lineage exists, view [Basic lineage](#). Otherwise prepare lineage sources; an empty graph does not establish no dependencies. Schedule recurring harvesting after verifying initial scope and properties.

Completion criteria

- Correct tables and columns are searchable under the source.
- Structure matches the source and descriptions explain purpose.
- Standard meanings match and references work in both directions.
- Run history is traceable and intervals meet requirements.

Next: **Data standards** or **Data quality**.

View basic lineage from metadata objects

Start with a table, column, or report to understand its sources and destinations. Available relationships depend on harvesting, lineage parsing, and object matching; names alone do not establish lineage.

View lineage

1. Find the object in Datasets or Reports.
2. Open the lineage relationships or analysis area in details.
3. Verify its source and name, then inspect upstream and downstream nodes and edges.
4. Follow upstream paths to trace origins or downstream paths to assess change impact.
5. Open important nodes in metadata details to verify physical identity and meaning.

Column and table lineage have different granularity. Table relationships alone do not establish how a column is calculated.

If no relationships appear

Check harvesting, lineage file parsing, namespace and schema matching, then resource synchronization. An empty graph can indicate incomplete preparation rather than no business dependencies.

Understand the SQLink boundary

Lineage management and end-to-end lineage viewed from metadata belong to metadata lineage. Lineage maps and application scenarios in the standalone Data Lineage section belong to [SQLink](#) and require the corresponding capabilities.

Administrators maintain harvesting, spaces, namespaces, and preprocessing. See [Lineage administration](#). SQLink analysis is not a default metadata capability.

Configure metadata harvesting

Harvesting defines the source, method, and scope for retrieving metadata. New configurations have four categories: Data Source, Report, File, and Metamodel. Connections, plugins, configurations, and execution are separate steps. Saving a connection or configuration does not produce searchable metadata.

Metadata

Dashboard

Dataset

BI Report

File

Management

Application

Data Source

Management

Meta Model

Lineage Management

Lineage Space

Pre-Process Manage

Namespace Management

Management

Name	Type	System	creator	Owner	Create Time	
11	label	反洗钱管理系统(AML)	王伟(weiwang)	-	2026-10-13 16:43:52	
JGBS	Data Dictionary(Physical)	监管数据报送系统(JGBS)	杨毅(yiyang)	杨毅(yiyang)	2026-10-11 10:42:56	
测试	API	反洗钱管理系统(AML)	潘广宗(guangzongp...)	-	2026-10-09 16:08:36	
> 反洗钱-采集	MySQL	反洗钱管理系统(AML)	潘广宗(guangzongp...)	-	2026-09-15 17:46:48	
> MC	MaxCompute	治理平台(dam)	潘广宗(guangzongp...)	-	2026-09-11 13:32:53	
测试数据源-默认采集	Kafka	反洗钱管理系统(AML)	超级管理员(admin)	-	2026-09-10 14:42:02	
> codex-governance-mysql-20260908	MySQL	反洗钱管理系统(AML)	冷鹏(pengleng)	-	2026-09-08 09:28:43	
814test	Data Dictionary(Physical)	汽车制造(汽车制造)	崔家名(jiamingcui)	崔家名(jiamingcui)	2026-08-14 15:22:58	
> 经分	MySQL	反洗钱管理系统(AML)	王鹏(pengwang)	-	2026-07-13 15:24:07	
订单管理	Data Dictionary(Physical)	承保系统(UWS)	杨毅(yiyang)	杨毅(yiyang)	2026-07-02 18:08:38	
标签类资产-默认采集	label	承保系统(UWS)	潘广宗(guangzongp...)	潘广宗(guangzongp...)	2026-06-08 16:57:55	
线索类资产-默认采集	clue	承保系统(UWS)	潘广宗(guangzongp...)	潘广宗(guangzongp...)	2026-06-08 16:37:43	
模型类资产-默认采集	Model	承保系统(UWS)	潘广宗(guangzongp...)	潘广宗(guangzongp...)	2026-06-08 15:10:08	
> 车企数仓	MySQL	汽车制造(汽车制造)	潘广宗(guangzongp...)	-	2026-05-07 18:16:19	
> 承保系统	MySQL	承保系统(UWS)	超级管理员(admin)	-	2026-04-16 11:35:14	
SmartBI	SMARTBI		超级管理员(admin)	-	2026-04-03 09:23:45	
FINEBI	FINE		超级管理员(admin)	-	2026-04-02 09:42:30	

New collection

datasource

report

file

Meta model

Names, users, and counts in the screenshot are examples from the verification environment.

Choose a harvesting path

Choose by the objects you want to manage, not simply by whether the input is a file.

Source or target	Entry point	Prerequisites	Results
Relational database	Data Source	System, connection, driver, source and reverse engineering plugins, database and schema scope	Tables, columns, views, procedures, and functions in Datasets
HBase, MongoDB, or another database platform	Data Source	Matching connection, driver, and plugins; supported objects depend on the plugin	Technical objects in Datasets
Custom source	Data Source	Published metamodel, matching source and reverse engineering plugins, template or adapter	Metamodel objects and hierarchies in Datasets
FineBI, FineReport, SmartBI, or another supported BI tool	Report	Supported report type, service address, credentials, report folders	Reports, information items, and associated objects in Reports
Custom report	Create/import in DAM Reports, or import in SQLink Report Management	Report template, object codes, system, relationships	DAM Reports
DDM physical data dictionary	File > Data Dictionary (Physical)	DDM export with the correct template	Parsed tables and columns in Datasets
DDM logical data dictionary	File > Data Dictionary (Logical)	DDM export with the correct template	Parsed entities and attributes in Datasets
DDL file	File > Parse DDL	Matching dialect, namespace, and file	Parsed structural objects in Datasets
API metadata	File > API Metadata	Built-in REST API metamodel; JSON file or HTTP access settings	Services, endpoints, request/response parameters in Datasets
File assets on a server	File > File Storage Server	Access method, path, credentials, scope	Files for querying and downloading in Files
Other enterprise objects	Metamodel, or File > Metamodel File	Designed and published metamodel, template or plugin	Instances and relationships in Datasets

A file can be a transport format or the asset itself. Dictionaries, DDL, API JSON, and metamodel templates create structured metadata. Only File Storage Server harvesting primarily creates entries in Files.

Check plugins

Plugins under **System Management > Data Source Management** determine selectable types, connections, and object recognition. Administrators should check them before configuration.

Plugin	Purpose	Typical missing or incompatible behavior
Data source	Source types, connections, and integration	Missing source type or failed connection
Reverse engineering	Convert source structures to DAM objects	Connection works but objects are not recognized correctly
Lineage parser	Parse transformations from SQL, ETL, BI, and other files	Objects exist but lineage parsing fails or is incomplete

Drivers do not replace plugins, and lineage parsers do not replace reverse engineering plugins. See [Drivers and plugins](#) for installation, replacement, and verification.

Prepare connections and accounts

Configure [Application systems and data sources](#). Confirm database types, drivers, and plugins, then prepare a dedicated account using [Source account permissions](#). Connection success proves login only. Test a few known objects to verify catalog, schema, and definition access.

Harvest data sources

Data Source covers relational, nonrelational, and custom metamodel sources. Installed plugins determine available types and objects.

1. Go to **Metadata Management > Harvesting Management** and create a data source configuration.
2. Enter its name and select a source using system and data source filters.
3. Click **Test**, then continue after success.
4. Select databases, schemas, and details; review connection, sampling, harvesting settings, and allowlists.
5. Confirm to create models and default tasks for selected schemas.
6. Configure task intervals and inspect execution history.
7. In **Datasets**, verify the source and harvested tables, columns, procedures, functions, or metamodel objects.

Start with a few known objects, compare counts, names, types, lengths, and relationships with the source, then expand. This is implementation guidance, not a product limit.

Setting	Check
Configuration name	Distinguishes source and purpose
System and source	Correct ownership and environment; test before scoping
Database and schema	Explicit intended scope rather than selecting all by default
Connection, sampling, harvesting	Structural access does not authorize business data access
Allowlist	Required tables are included; inspect logs for omissions
Interval	Verify the first result before scheduling updates and inspect actual triggers

Harvest reports

For supported BI tools such as FineBI, FineReport, and SmartBI, create a Report configuration, select the tool, enter its service connection and folders, test, and confirm. Use the selected tool's own connection fields and authentication.

For custom reports without a supported connection, create or import them in DAM Reports, or import through SQLink Report Management. All appear in DAM Reports. Manual properties do not automatically create complete database lineage.

Use [Report management](#) to verify folders, properties, information items, and table relationships. A successful connection does not guarantee complete relationships.

Harvest file-based sources

Choose the subtype according to input and target objects.

[Return](#) | [Management / New](#)

* Collection Manage

Name

Please enter a collection name

* System

Select

▼

* Database

Data Dictionary (Physical)

^

* Name Space

Data Dictionary (Physical)

* Parse engine

Data Dictionary (Logical)

Collection Type

Metamodel file

* File

File Storage Server

Parse DDL

API metadata



Drag the file here, or [Click on the upload](#)

Physical and logical data dictionaries

Data Dictionary (Physical) and **Data Dictionary (Logical)** accept DDM exports for tables/columns and entities/attributes respectively. Preserve template columns, identifiers, and hierarchy. Query parsed objects in Datasets, not the original Excel file in Files.

DDL files

Select **Parse DDL** and the matching database or dialect. The parser extracts structures from table, view, procedure, and function definitions. This supports sources without direct database access, but reflects only the supplied script. Have its provider confirm completeness and consistency with production. Results appear in Datasets.

API metadata

Harvest JSON files or HTTP APIs through the REST API metamodel. Verify mappings for services, endpoints, and request/response parameters first. Results are metamodel objects in Datasets, not downloadable file entries.

File storage servers

Configure server access, path, and scope, then execute. Query, view, and download file objects in Files subject to permissions. This subtype maps directly to Files.

Harvest metamodel objects

A metamodel defines object types, properties, and relationships; harvesting supplies instances. Use it for APIs, queues, object storage, and enterprise-specific objects beyond built-in database models.

Complete design and publication in [Metamodel concepts](#) and [Metamodel design](#). Choose a custom plugin, DAM template import, or **File > Metamodel File** according to the solution. Root, identifier, namespace, and template rules differ by path. See [Custom metamodel harvesting](#) and [Metamodel import](#).

Run and verify results

- Check status and logs for the intended run.
- Query Datasets, Reports, or Files using the decision table.
- For source, dictionary, DDL, API, and metamodel inputs, verify source, object type, schema/namespace, and counts in Datasets.
- For reports, verify folders, reports, information items, and table relationships.
- For file servers, verify names, paths, sources, and download access.
- Compare sample tables, columns, and views with source attributes.
- Inspect missing objects, failures, and allowlist exclusions.
- Configure recurring [Task scheduling](#) after verification.

Troubleshoot common issues

Symptom	Checks
Connection failure	Network, parameters, credentials, login rights, driver
Connection succeeds but no metadata	Task execution, catalogs, schema, allowlist, scope, logs
Fewer objects than expected	Account visibility, definitions, filters, failed objects
Reports lack tables	Source relationship information and database harvesting
Successful file input but no Files entry	Dictionaries, DDL, API, and metamodel results belong in Datasets
Missing database type	Compatible source and reverse engineering plugins and driver
Duplicate or incorrect hierarchy	Publication, identifiers, parent references, path-specific rules
Scheduled task does not run	Enablement, interval, time template intersection

Find and maintain table and column metadata

Datasets contains structured metadata from relational and nonrelational sources, dictionaries, DDL, API files, and custom metamodels. Business users read definitions, data stewards maintain business properties, and technical users inspect structure and changes.

Know which objects appear in Datasets

Source	Typical objects
Relational database	Sources, databases, schemas, tables, columns, views, procedures, functions, indexes, constraints
HBase, MongoDB, and other nonrelational sources	Databases, collections, columns, and objects recognized by the plugin
DDM dictionaries and DDL	Parsed tables, columns, entities, attributes, views, procedures, functions
API metadata	Services, endpoints, parameters, relationships
Custom metamodel	Defined objects, properties, hierarchies, associations

Harvested or imported reports belong in Reports; file server objects belong in Files. See [Harvesting](#).

Use the **By Technical Department** view, all-metadata and all-source filters. Columns include physical name, Chinese name, asset type, system, schema, department, and creation time. Open details through row actions. Pagination and counts reflect the current query.

Metadata

Dashboard

Dataset

BI Report

File

Management

Application

Data Source

Management

Meta Model

Lineage Management

Lineage Space

Pre-Process Manage

Namespace Management

Department

Metadata

Database

机构管理/金融科技部

机构管理/数据管理部

机构管理/数据平台部

机构管理/信息技术部

Search keywords

Table/Entity

View

Column/Attribute

Stored Procedure

Function

package

Synonym

More filter

User Defined Property

Table Name	Alias	Asset Type	Application	Schema	Department	Create Time	Data Owner	Actions
C_CUST_ACCT	对公客户信息管...	Table	对公客户信息管理系...	ECIF	金融科技部	2026-10-12 10:58:07		良
YLBZCFZH00B		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良
YLBZCFPX0B		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良
NUJHYMXB		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良
NUJHTZQB		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良
NUJHGLQB		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良
NUJHTGGXB		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良
NUJHQB		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良
KHZH/YLSB		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良
XTJHRZQB		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良
ZZTZ/YLSB		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良
ZZTZCB		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良
XTJHQB		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良
YLBZTGLHTQB		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良
JKSZWTYXXMXB		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良
YLBZYWQB		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良
TZHQXB		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良
QTTZBQB		Table	监管数据报送系统 (J...	JGBS	金融科技部	2026-10-11 10:42:56		良

[Open the full image](#). Names and counts illustrate fields and navigation only.

Find an object

1. Go to **Metadata Management > Datasets** and select a system or source.
2. Search for a table, column, view, function, procedure, API, or metamodel object.
3. Narrow results by type and other available filters.
4. Verify source, schema, and physical name before opening similarly named objects.

For large inventories, choose a department or source before searching. Cross-check system, schema, and physical name. Counts represent filtered results, not all platform metadata.

View tables and columns

Table details include columns, samples, and lineage. Click a column name for details. Read business and technical properties together; the Chinese display name does not replace the physical identifier.

Edit supported properties and reopen to verify. Distinguish source properties from steward-maintained properties, and review history after subsequent harvesting. Viewing SQL for views, procedures, and functions requires role permission beyond object visibility.

View sample data

Sampling reads real data and requires permission. Distinguish database errors from empty results; check error details, read rights, and structure access. Sampling explains columns but does not replace full quality validation or marketplace delivery.

Hive sampling first reads structure, then the requested rows. Structure errors produce **Sampling Failed**; a successful empty query produces **No Data**. Diagnose these separately.

Compare change history

Select two records in **Change History** and compare table properties, columns, and indexes. Record additions, deletions, and modifications, then assess dependencies through [Basic lineage](#). This is object history, not software version history.

View object relationships

Knowledge Graph shows associations with tags, columns, systems, and other resources. Choose the correct resource and relationship meaning. Knowledge associations are not automatically transformation lineage.

Export and update in bulk

After technical harvesting, export and update business semantics: logical/Chinese names, descriptions, tags, ownership, and standard mappings. These update DAM governance properties, not database structures.

Content	Check before updating
Logical or Chinese name	Consistent terms; distinguish objects by source, schema, physical name
Description	Business meaning, scope, restrictions
Tags	Approved tags and classifications, without improvised synonyms
Standard mappings	Published standards with matching meaning, type, and value domain
Responsibility and other properties	Confirmed owners, departments, and maintenance boundaries

1. In the source ... menu, select **Export Metadata**, or export from the list.
2. Download all objects or select tables and **Download Selected Metadata**. Large exports may be split into a ZIP archive.
3. Maintain table and column semantics, preserving identifiers, physical names, sources, schemas, template columns, and worksheets.
4. Select **Update Metadata** in the source menu and upload the revised Excel dictionary.
5. Inspect results and failures. Ensure empty values did not overwrite properties and standard codes and tags resolved.
6. Verify changed objects. Test a few records before bulk updates.

Exports reflect filters and permissions and are not full environment backups. Later harvesting may change technical properties. Check configuration, history, and import results for semantic retention or conflicts.

Check and troubleshoot

Symptom	Checks
Missing object	System/source, type filter, harvesting, permissions
Hidden SQL	Role permission to view statements
Sampling failure	Database error, account read rights, connection, structure
Update has no effect	Correct source/object, preserved identifiers, import result
Same name, different content	System, source, schema, physical name

Related: [Standard mappings](#), [Harvesting](#).

Maintain metadata objects

Database and metamodel objects, reports, and files have distinct query and maintenance areas. Harvesting retrieves source information; manual maintenance adds business properties. Keep their provenance distinct.

Choose the correct query area

Object	Area
Sources, tables, columns, views, procedures, functions	Datasets
Structures parsed from dictionaries or DDL	Datasets
APIs and custom metamodel objects	Datasets
Harvested or imported reports	Reports
Files harvested from storage servers	Files

Search and maintain in bulk

In **Metadata > Datasets**, filter by department, type, source, physical/Chinese name, asset type, system, schema, and creation time. Verify source, database, schema, table, and harvesting time before editing descriptions or ownership.

Export filtered results and preserve identifiers. Add logical names, descriptions, tags, ownership, and standards in the template, then **Update Metadata**. Edit only supported columns; retain identifying columns and avoid overwriting technical properties with empty values. See [Export and bulk update](#).

Review metadata after model changes

Refer to DDM documentation for model design, comparison, and synchronization. DAM maintainers should:

1. Obtain the approved change list and confirm source and schema.
2. Harvest affected objects again and verify additions, deletions, and property changes.
3. Check affected standard mappings, lineage, and quality rules.
4. Assign owners to stale objects and broken relationships.

See [Model change review](#).

Maintain report and file metadata

Reports can be harvested from BI tools or imported/registered through DAM Reports or SQLink Report Management. Maintain names, systems, owners, and descriptions in DAM Reports. Files contains only file server objects. Check template columns and codes before import; record export filters.

For dependency analysis, first verify harvesting and correct system bindings. Unbound objects may show only their properties.

Verify results

- Counts match filters.
- Technical timestamps update and business maintenance history remains traceable.
- Standards, asset registration, and lineage links still work.
- Check asset, rule, and lineage references before deletion or retirement.

Harvest custom metamodel data

DAM and SQLink both map instances to model structures but use different sources, roots, and lineage preparation.

DAM: plugins and templates

Install plugins

1. An administrator opens **System Management > Plugin Management**.
2. Install matching source and reverse engineering plugins.
3. Verify enablement before configuring the source.

METAMODELDEMO supports template validation. Confirm the production plugin choice with the delivery team.

Define a source and task

1. Create a custom source in **Metadata > Data Sources**.
2. Associate the system and plugin, then save.
3. Create a task through standard harvesting. It primarily maps source to model; the configuration name normally identifies the schema.

Generate and complete the Excel template

1. Download the model-based template from source details.
2. Preserve prefilled root ID and Name. Name corresponds to the schema.
3. Enter child data and parent ID/Name in child sheets.
4. Preserve consistent identifiers through deeper levels.
5. In **Metadata > Datasets**, select the source and **Import Metadata**.

Correct missing parent IDs, duplicate names, required values, and hierarchy errors in logs. Test a few objects, tree navigation, and groups before bulk import.

SQLink: file harvesting and synchronization

1. Select **File** in SQLink harvesting and configure the metamodel.
2. Enter the lineage namespace. The root ID may be empty; child Name values can express hierarchy.
3. Execute and inspect status and object counts.
4. Filter the SQLink catalog by source and confirm synchronization to DAM Datasets.

Do not apply DAM's required root IDs to SQLink files or SQLink's empty-root rules to DAM Excel templates.

Display and incremental updates

Verify root, children, property groups, relationships, dataset filters, and asset registration in that order. Submit only changed objects for incremental updates and retain identifiers to prevent duplicates.

Add custom object lineage

1. Create an Excel lineage task in a SQLink space.
2. Download the relationship template and enter sources, targets, namespaces, and schemas. Many-to-many relationships are supported.
3. Turn off Chinese display in object lineage analysis to inspect namespace and schema.
4. Import, parse, and verify dependencies from metadata objects.
5. For an empty graph, check synchronization to the space's database resources; synchronize manually if needed.

Metamodel relationships describe business associations; lineage describes actual processing paths. Neither replaces the other.

Understand metamodels and built-in models

A metamodel defines object types, properties, and relationships. Metadata contains instances of those definitions. It is a structural design, not a single table or task.

Understand the four components

Component	Purpose	Example
Model	Overall structure for an asset class	Database, API, data product model
Object	Managed entity type	Database, table, endpoint, data product
Property	Object characteristic	Name, definition, owner, URL
Relationship	Containment, reference, dependency, association	Database contains table; metric derives from column

Objects are units in Datasets and the catalog. Properties determine details; relationships support tree navigation, links, and analysis.

Use built-in metamodels

Built-in models include dedicated harvesting and governance pages:

- Database model: databases, schemas, tables, columns, views, procedures, functions, indexes, primary and foreign keys.
- REST API model: services, endpoints, request/response parameters, relationships.

You do not need to redesign these objects or individually map columns to custom models. See [Compatibility](#) for supported sources.

Choose a built-in or custom model

Need	Path	Reason
Databases, warehouses, conventional columns	Built-in database model	Dedicated harvesting and display
OpenAPI/REST endpoints	Built-in API model	Reusable structures and harvesting
Regulatory submissions, devices, enterprise products	Custom model	Organization-specific structures
Add business descriptions to tables	Maintain properties or mappings	No new model required

Create custom models only when built-in models cannot express the object boundary. Assess impacts before changing core objects referenced by tasks, catalogs, or APIs.

Understand DAM and SQLink paths

DAM commonly uses plugins and Excel templates; SQLink emphasizes file harvesting and lineage. Do not interchange their root, namespace, and schema rules.

Next: [Design metamodels](#) and [Harvest custom objects](#).

Design a custom metamodel

Metamodels define structure; metadata records instances. Prefer built-in database and REST API models where suitable; customize for enterprise-specific objects.

Define four elements

Element	Meaning	Design question
Model	Overall asset structure	What scope is managed?
Object	Entity type	What hierarchy exists?
Property	Object characteristic	What information is recorded?
Relationship	Hierarchy, reference, association	How are objects connected?

Diagram objects and relationships first. Define shared identifiers, names, and descriptions. Avoid speculative unused properties.

Create a model

1. Open **Metamodels** and create a model.
2. Enter a stable, meaningful code, name, and description.
3. Open the editor. ID and GUID are system-generated and read-only.
4. Review data and structure areas, then create objects, properties, and relationships.

Edits save immediately. Assess production impacts on harvesting, templates, and data before changing a model; closing the page does not cancel edits.

Complete basic information

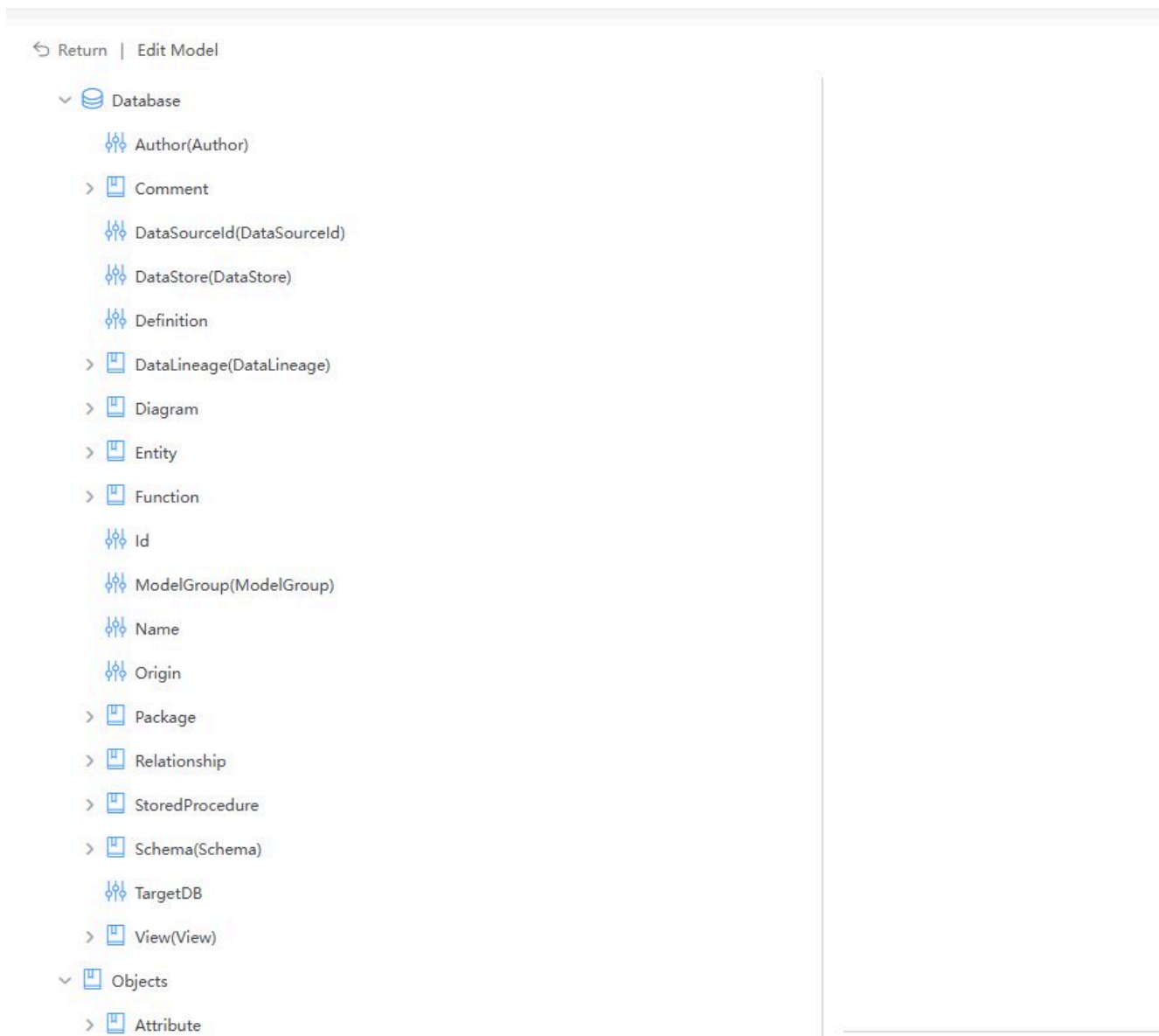
Field	Purpose	Guidance
Code	Stable system identifier	Meaningful English code; avoid later changes
Name/alias	Display and business identification	Describe the managed object; avoid temporary project abbreviations
Description	Goals and scope	Systems, team, exclusions
ID/GUID	Internal identity	System-managed

A code identifies one structure. For parallel structures, create another model or plan a migration before replacing production definitions.

Assemble the structure

1. Create objects with names, aliases, descriptions, and icons.
2. Enable asset registration only for types intended for the catalog.
3. Define properties centrally and reuse synonymous properties.
4. Set controls, groups, and display order; ungrouped properties use the default group.
5. Use **Associate Properties** on objects.
6. Use **Associate Objects** for hierarchy and business relationships.
7. Edit the root and associate shared properties and business objects.

Root associations are required. Unassembled definitions can prevent template generation, imports, or complete display.



The screenshot identifies association areas. Use names from your own design.

Review the five design steps

Step 1: Create object types

List actual entity types, such as Agent, Goal, Knowledge, Memory, and Model. Types define detail pages and permission boundaries, not individual records. Enable **Allow Asset Registration** only where needed.

Step 2: Create properties

Define each meaning once. Choose controls, groups, and order, including suitable date, enumeration, and long-text controls.

Step 3: Associate properties

Select existing properties using **Associate Properties**. This references rather than copies them. Check all references before changing a shared property.

Step 4: Define relationships

Use directional, meaningful names for parent-child, reference, dependency, and business relationships. Do not turn a possible association into a hard dependency that misleads searches or impact analysis.

Step 5: Assemble the root

The root is the entry to instances. Associate shared ID, name, logical name, and definition, then object types. Preview the template to check hierarchy and grouping.

Follow design principles

- Start from business objects rather than database tables.
- Keep essential task-related properties first.
- Keep core object/property codes and relationship direction stable.
- Allow future extension without speculative fields.
- Apply consistent naming and record definitions.

Validate before publication

Use a small sample to verify:

1. Template generation and root ID/Name rules.
2. Expected object tree without cycles.
3. Readable property groups and order.
4. Only intended types are available for asset registration.
5. Consistent identifiers across harvesting, import, and lineage.

Verify the design

Check templates, hierarchy, groups, order, unique names, and accidental duplicate properties. Then create instances using [Metamodel import](#).

Object associations are not automatically transformation lineage. Supply separate source-target relationships where needed.

Import and validate custom metamodel data

This page covers DAM Excel import and its differences from SQLink files. Complete [Metamodel design](#), including root assembly, first.

Prepare plugins and a DAM source

1. Install product-provided metamodel source and reverse engineering plugins. The basic template solution uses METAMODELDEMO.
2. Verify enablement. Use an adapted plugin for other sources.
3. Create a custom source in **Metadata > Data Sources**, associate a system, and choose the plugin.
4. Create harvesting normally. This template configuration creates the management structure without reading external business metadata. Its name identifies the schema.

Do not treat inactive connection fields in the demonstration plugin as real external connection settings.

Complete and import Excel

1. Download the model-based template from source details.
2. Preserve the first root sheet's ID and Name; Name corresponds to the schema.
3. Enter children and parent ID/Name references.
4. Complete deeper hierarchy names consistently.
5. Select the source in **Metadata > Datasets** and **Import Metadata**.
6. Inspect failures and correct missing parent IDs and duplicate names.

AI_AGENT.DIG is only a hierarchy format example. Use actual root and child names.

Verify results

Open root instances, shared properties, children, groups, and relationships. For incremental imports, provide changes only, inspect merging, and validate a small sample first.

Understand the SQLink differences

SQLink uses **File** harvesting with a namespace, not the DAM demo plugin. The root ID can be empty; child Name can describe the root. These rules do not apply to DAM root templates.

After execution, filter SQLink's catalog by source and verify the synchronized DAM results.

Add custom lineage

1. Create an Excel lineage task in SQLink and download its template.
2. Enter source, target, namespace, and schema, including many-to-many relationships where needed.
3. Turn off Chinese display in lineage analysis to inspect object paths.
4. Import, parse, and inspect lineage on metadata objects.
5. For empty graphs, verify synchronization to the space's database resources and synchronize manually if needed.

Related: [SQLink administration](#), [Basic lineage](#).

Review model changes and metadata

Compare approved DDM models with database structures harvested into DAM to identify tables, columns, views, and property differences. A typical flow reverse engineers a database into DDM, maintains model metadata, then compares and updates approved DAM properties.

DDM handles reverse engineering, design, and model versions. DAM harvests actual structures, initiates comparisons, and reviews governance associations. Confirm objects, versions, and synchronization direction before updating production metadata.

Establish comparable baselines

1. Harvest the target in DAM and verify database, schema, table, and column scope.
2. In DDM Client, [reverse engineer the database \(Chinese guide\)](#) for the same scope.
3. Save to the agreed repository folder and create an identifiable version; record environment and reverse engineering time.
4. Maintain Chinese names, descriptions, standards, column definitions, and properties. Review and publish the comparison version.
5. Align the DAM harvesting time and DDM version before comparing.

DDM can also compare models with reverse engineered databases through [model comparison \(Chinese guide\)](#). Use DAM tasks when applying model results to DAM governance.

Configure comparison in DAM

1. Find the source in **Metadata > Datasets** and select **Model Comparison and Synchronization**.
2. Select the approved DDM model/version; verify repository, name, database type, and schema.
3. Enable comparison and select relevant objects and properties.
4. Review additions, deletions, and changes to tables, columns, views, types, length/precision, defaults, keys, and indexes.
5. Retain the report with DAM objects, DDM version, difference types, decisions, and owners.

Have business owners resolve same-name/different-meaning columns. A difference does not prove either baseline is correct.

Review and update metadata

1. Model and database owners classify causes: database ahead of model, model not implemented, stale DAM harvest, or mismatched scope.

2. Enable synchronization for approved business properties only, such as standards, reference codes, Chinese names, descriptions, or custom properties.
3. Apply the available null/conflict policies. Retain the report and test direction with a few objects.
4. Implement physical changes through database change management, then harvest again. Property synchronization does not replace DDL changes.
5. Compare again and retain reasons and owners for unresolved differences.

Check related impacts

After deletions, type changes, or constraint changes, review standards, quality rules, catalogs, metrics, and lineage. Assign owners and record repairs to broken references. Successful harvesting or comparison does not confirm business impact.

Review after execution

Reopen affected objects and inspect structure, names, descriptions, standards, tags, and relationships. Run limited [Harvesting](#), then compare DAM, DDM, and the database against the approved outcome. Handle design, version, and DDL generation issues in DDM records.

Comparison checklist

- Same environment and schema.
- Reviewed model version and suitable DAM harvesting time.
- Report includes all changes and reasons for deferral.
- Confirmed direction and properties; nulls did not overwrite meaningful values.
- Physical changes followed release procedures and DAM was reharvested.
- Standards, quality, assets, metrics, and lineage impacts reviewed.

Manage report and file metadata

Report metadata describes reports, information items, and table associations. File metadata describes objects harvested from storage servers. Their collection and query paths differ.

Area	Includes	Does not include merely because input is a file
Reports	Supported BI reports and custom DAM/SQLink imports	Dictionaries, DDL, ordinary server files
Files	File server files and folders	Dictionaries, DDL, API metadata, metamodel templates

Create a report

1. Open Reports and select a folder.
2. Create a report with basic properties and folder information.
3. Add information items and table associations.
4. Save and verify properties, folder, and related objects.

Use [Report harvesting](#) for supported BI tools rather than manually duplicating their results. Supported and custom reports share this area.

Import reports

1. Download the report template.
2. Complete properties and relationships.
3. Select update or overwrite behavior and upload.
4. Verify results, counts, and table associations against the input.

Check matching names and scope before overwrite.

Maintain and export reports

View details, edit properties, or select rows for bulk deletion after reviewing scope. Folder export covers the current folder only. View [Basic lineage](#) where parsed relationships exist; text descriptions do not generate lineage.

View and download files

Only File Storage Server harvesting produces these entries. Dictionaries, DDL, APIs, and metamodel files create structured objects in Datasets.

1. Search the Files list.
2. Open details and maintain editable properties as needed.
3. Select files and download.
4. Verify contents and follow organizational storage and use rules.

For missing files, check the subtype, task execution, paths, scope, and permissions. Query Datasets for parsed structured inputs.

Configure source account permissions

A harvesting account must connect to the source and read the catalogs, definitions, and schemas required by the reverse engineering plugin. With insufficient permissions, connection tests can succeed while harvesting fails, returns partial objects, or omits views, procedures, keys, and indexes.

Have a DBA grant access. Prefer a dedicated read-only account with least privilege for the target database and schema. Do not default to application accounts or administrator roles. Replace example accounts, databases, schemas, and passwords. Database, authentication, and plugin versions may require additional catalog access; obtain DBA review first.

Separate metadata access from data access

Purpose	Required capability	Reads business records?
Connection test	Log in to the database or service	No
Reverse engineering	Read catalogs, definitions, schemas, tables, columns, views, routines, constraints, indexes	Normally no
Sampling or profiling	Limited queries on approved tables/views	Yes
Quality validation and other tasks	Read columns/records required by task SQL	Yes

Structure-only projects should not receive unrestricted table reads because sampling is available. Restrict approved sampling by database, schema, table, row count, and account purpose.

Prepare permissions by database

Type	Reverse engineering focus	Additional checks
MySQL	Definitions and references in target databases	Views, routines, sampling
Oracle	Sessions and data dictionary	Cross-schema scope and table reads
DM Database	Dictionary, views, sequences, target objects	Restriction to target schema
SQL Server	Definitions and dependencies	DBA-defined read-only permissions instead of defaulting to db_owner
Hive/Impala	Target databases and objects	Metastore, Kerberos, platform permissions
PostgreSQL/Greenplum	Connect, schema usage, catalogs	Table SELECT for sampling and defaults for new tables
openGauss	Required pg_catalog objects	Catalog scope for the installed plugin
MongoDB	Collections and structure in target database	Sampling authorization and authentication database
Vertica/ClickHouse/Doris	Target schema/database structures	Current plugin requirements and project checklist

This table describes goals, not a substitute for official authorization scripts. A DBA must review production statements and restrict their scope.

Example grants

Commands contain no real credentials. `user_name`, `database_name`, `schema_name`, and `password` are placeholders.

MySQL

Read structure and data across databases:

```
mysql
```

```
GRANT SELECT ON *.* TO 'user_name'@'%';
```

Limit access to one database:

```
mysql
```

```
GRANT SELECT ON database_name.* TO 'user_name'@'%';  
GRANT REFERENCES ON database_name.* TO 'user_name'@'%';
```

For structure only, have the DBA grant access to catalogs actually read by the plugin. Add target table SELECT only for approved sampling.

Oracle

Create a session and read the dictionary:

□sql

```
CREATE USER user_name IDENTIFIED BY "password";  
GRANT CREATE SESSION TO user_name;  
GRANT SELECT ANY DICTIONARY TO user_name;
```

Prefer approved object-level grants for limited schemas. Check additional access for cross-schema views, routines, and dependencies.

DM Database

Example metadata and object reads:

□sql

```
CREATE USER user_name IDENTIFIED BY "password";  
GRANT SELECT ANY DICTIONARY,  
      SELECT ANY MATERIALIZED VIEW,  
      SELECT ANY SEQUENCE,  
      SELECT ANY VIEW,  
      SELECT ANY TABLE TO user_name;
```

Narrow production grants to target schemas. Have the DBA confirm statements for the installed DM version.

SQL Server

The account needs definitions and dependencies. An adapted database-level example is:

□sql

```
USE database_name;  
ALTER ROLE [db_datareader] ADD MEMBER [user_name];  
GRANT VIEW DEFINITION TO [user_name];
```

Some older environments use `db_owner` for quick verification. Its broad privileges are unsuitable as a production default. Replace it with DBA-approved database, schema, and object permissions after testing.

Hive and Impala

Example database-level read access:

□sql

```
GRANT SELECT ON DATABASE database_name TO USER user_name;
```

Also verify Metastore, authentication, Kerberos, and platform requirements. Grant mechanisms differ by distribution.

PostgreSQL and Greenplum

Create the account and configure schema access:

□sql

```
CREATE USER user_name;
ALTER USER user_name WITH PASSWORD 'password';
GRANT CONNECT ON DATABASE database_name TO user_name;
GRANT USAGE ON SCHEMA schema_name TO user_name;
GRANT SELECT ON ALL TABLES IN SCHEMA schema_name TO user_name;
ALTER DEFAULT PRIVILEGES IN SCHEMA schema_name
    GRANT SELECT ON TABLES TO user_name;
```

The plugin also needs version-appropriate `pg_catalog` access. Restrict sampling to approved schemas/tables. Validate Greenplum against its cluster version after applying the PostgreSQL approach.

openGauss

Example catalog access:

□sql

```
GRANT SELECT ON pg_catalog.pg_namespace TO user_name;
GRANT SELECT ON pg_catalog.pg_class TO user_name;
GRANT SELECT ON pg_catalog.pg_attribute TO user_name;
GRANT SELECT ON pg_catalog.pg_proc TO user_name;
GRANT SELECT ON pg_catalog.pg_constraint TO user_name;
GRANT SELECT ON pg_catalog.pg_inherits TO user_name;
GRANT SELECT ON pg_catalog.pg_index TO user_name;
GRANT SELECT ON pg_catalog.pg_type TO user_name;
GRANT SELECT ON pg_catalog.pg_roles TO user_name;
GRANT SELECT ON pg_catalog.pg_user TO user_name;
GRANT SELECT ON pg_catalog.pg_extension TO user_name;
```

Add database connection, schema usage, and approved table reads. Test a few objects for any additional plugin-specific catalogs.

MongoDB

Grant a read role for the target database:

 javascript

```
db.grantRolesToUser(
  "user_name",
  [{ role: "read", db: "database_name" }]
);
```

Verify authentication database, target database, and user domain. Restrict collections and sample scope according to policy.

Vertica, ClickHouse, and Doris

Vertica schema example:

 sql

```
GRANT SELECT ON SCHEMA schema_name TO user_name;
```

ClickHouse read example:

 sql

```
GRANT SELECT ON *.* TO user_name;
```

Doris single-database example:

 sql

```
CREATE USER 'user_name'@'%' IDENTIFIED BY 'password';
GRANT SELECT_PRIV ON database_name.* TO 'user_name'@'%';
```

All-database Doris access is only for approved administration scenarios. Verify catalogs, views, and sampling requirements for each installed plugin.

MaxCompute

Configure Endpoint, Access ID, Access Key, and Project Name, with project-side metadata permissions. Provide keys through secure connection configuration or secret management, never manuals, screenshots, or shared documents.

Grant and verify access

1. Confirm the system, database, schema, object types, and sampling needs.
2. Create a dedicated account with appropriate rotation, source-address, and expiration policies.
3. Grant only required login, catalog, and object access.
4. Test the DAM connection. Success proves login, not complete reverse engineering access.
5. Harvest a few known objects and compare counts for tables, columns, views, procedures, functions, keys, and indexes.
6. Add approved sampling reads if needed and test a few records.
7. Expand after verification; record account, scope, plugin version, and verification date.

Troubleshoot incomplete reverse engineering

Symptom	Check first
Connection failure	Network, host, port, authentication, account status, login rights
Connected but no objects	Catalog access, database/schema scope, plugin, default database
Tables/columns only	Definition, routine, and catalog permissions
Some schemas only	Account visibility, filters, allowlists, cross-schema access
Structure works, sampling fails	Table SELECT, row security, query limits
Grants changed but results unchanged	Rerun, account in use, caches/plugin reload

Use task logs to identify the failing object and read operation. Have the DBA grant the minimum missing access and retest a limited scope rather than repeatedly adding global privileges.

Map columns to data standards

Mappings connect definitions, technical requirements, and reference code domains to metadata. Confirm matching meaning rather than relying on similar names.

Prerequisites

Harvest the column and ensure access to it and suitable standards. Standards and reference codes must be usable. Resolve ambiguous business definitions with the owner.

Create a mapping

1. Find the table in Datasets and open the column.
2. Edit Data Standard under **Technical Properties**.
3. Search data or metric standards and verify definitions, types, lengths, and value domains.
4. Click **Select**.
5. If prompted to use the standard's reference codes, verify value semantics first.
6. Return to table details and check the association.

Use the separate reference code edit action where needed.

Verify references

Open the standard's **Reference Information** and confirm column and source. Mapping does not alter the database or validate record values. Use standards validation and quality tasks for conformance.

Change or remove mappings

Verify both objects before correcting semantics. Unbind columns through standard references. Review dependent standards validation, quality, and governance work after unbinding.

Related: [Basic standards](#), [Reference codes](#), [Mapping and validation](#).