BIRD
Banks' Integrated Reporting Dictionary

Version 1.1

January 2026

BIRD Metadata Manual

Contents

0. Version control

Version	Date	Comments
1.0.0	10/2024	First version of the BIRD metadata manual on how to access, navigate, and integrate BIRD metadata.
1.1.0	01/2026	Enhanced explanation on how to obtain simplified and technical exports (chapter 2.3 Exports).

1. Introduction

This document aims to provide BIRD users with a comprehensive guide to the methods available for accessing BIRD metadata, how to navigate them, and how to maximize their use. It also offers recommendations on how users can integrate BIRD metadata into their systems. Please note that the guidance provided is a simplified overview and presents possible approaches to BIRD implementation; however, banks are ultimately responsible for ensuring the correct implementation and submission of data to the authorities.

To fully benefit from this document, the BIRD team recommends that users familiarize themselves with the following key topics:

- **BIRD Methodology and Components:** These resources, available in the "About BIRD" section of the website, offer deeper insights into the BIRD methodology and its components.
- **Introduction to the Logical Data Model (LDM):** This document explains the structure of the BIRD LDM and how it can be further implemented.
- **LDM Design Principles:** This document outlines the key design principles behind the BIRD Logical Data Model.
- **Conceptual/Logical vs. Physical Modelling:** Users should understand the definitions and differences between conceptual/logical modelling and physical modelling, as BIRD primarily focuses on the former.
- **APIs:** Users should have a basic understanding of APIs and how they work, including typical response formats such as JSON or XML.
- **SMCube documentation:** This document provides extensive information on the SMCube methodology which are necessary to comprehend the technical format of BIRD metadata.
- **Additional Documentation:** While not strictly required, other BIRD documentation may provide further context and enhance understanding.

1.1. Structure of the document

This document is organized as follows:

- **Chapter 2**: Describes the methods available for accessing BIRD metadata, offering guidance on how to retrieve and navigate the metadata referenced in subsequent chapters.
- **Chapter 3**: Explains the various types of BIRD data models, their characteristics, and how to manage them effectively.
- **Chapter 4**: Details the different types of BIRD transformation rules and provides instructions on managing them.
- **Chapter 5**: Details BIRD validation rules and their structure.
- **Chapter 6**: Covers BIRD's historization mechanism, which tracks historical changes in BIRD metadata, including those discussed in Chapters 3 and 4.
- **Chapter 7**: Focuses on mappings within the BIRD framework and their application.
- **Chapter 8**: Offers recommendations for users on how to implement a BIRD solution based on the information provided in previous chapters.
- **Chapter 9**: Provides insights into future developments and enhancements in BIRD.

1.2. List of Abbreviations

Number	Abbreviation	Description
1	BIRD	Banks' Integrated Reporting Dictionary
2	IL	Input Layer
3	EIL	Enriched Input Layer
4	OL	Output Layer (currently referring to the combination of ROL and NROL)
5	ROL	Reference Output Layer
6	NROL	Non-reference Output Layer
7	ERM	Entity Relationship Model
8	LDM	Logical Data Model
9	ELDM	Enriched Logical Data Model
10	API	Application Programming Interface
11	WS P	Workstream on Prototyping
12	WS DM	Workstream on Data Modelling
13	DP	Data Point

Table 1: List of Abbreviations.

Description for the main SMCube and BIRD simplified metadata terms can be found in the BIRD Navigator tutorial page.

2. Methods to access metadata – an overview

2.1. Introduction

BIRD (Banks' Integrated Reporting Dictionary) provides several methods to access and interact with metadata. This chapter provides an overview of the available methods for accessing BIRD metadata, including APIs, technical exports, simplified exports, and the BIRD Navigator. Additional details on how to access metadata for the various BIRD components are provided in the other chapters of this manual.

Tool	Purpose
BIRD API	Enables automated processes, such as integrations using Python.
BIRD Technical exports	Offers exports for technical users or specialized use cases.
BIRD Simplified exports	Provides simplified data exports tailored for business users.
BIRD Navigator	Facilitates browsing and exploration of metadata.

Table 2: List of BIRD tools.

2.2. Application Programming Interfaces (APIs)

BIRD APIs allow applications and procedures to access BIRD metadata easily and efficiently. The BIRD team recommends this method due to its user-friendliness and the structured format of the results. Documentation for the BIRD API can be found here. The following are the main BIRD API endpoints mentioned in this document:

- GET /**dataModels**: Retrieves the data structures, relationships, and attribute restrictions for the BIRD Input Data Models, including the Logical Data Model (LDM), Enriched Logical Data Model (ELDM), Input Layer (IL), Enriched Input Layer (EIL), and Entity Relationship Model (ERM)-based output requirements.
- GET /**entities**: Provides the specific data structures, attributes, and restrictions for ERM or flat model-based entities (input and output).
- GET /**renderingTables**: Provides the template layout for template-based reporting requirements and related data points (valueCombinations) definition.

- GET /**incomingLogicalLineages**: Identifies the source entities, their attributes, allowed values, and algorithms used to generate an output entity or derive attributes of an enriched entity. It can be used to generate BIRD output entities or to derive EIL attributes.
- GET /**renderingIncomingLogicalLineages**: Helps to identify the source entities, attributes, allowed values, and algorithms that are necessary to generate a destination Rendering Table (i.e., template), data point by data point for template-based reporting requirements.

2.3. Exports

BIRD offers two types of exports for accessing metadata – simplified exports and technical exports – which are available in the Metadata & Export section either in excel or csv format.

2.3.1. Simplified exports

Simplified exports are ideal for those who require a user-friendly format for investigating or using the metadata without the complexities associated with detailed technical exports.

The following are the types of simplified exports.

Export Type	Description	Content
***Framework models overview* Export**	The *Framework models overview* export allows users to easily navigate the data models of a framework, by showing all the entities, attributes, restrictions, and allowed values of each data model. This export can be used with any type of framework, even those with large data models.	Includes metadata from both input and output layers.
***Framework logical lineage* Export**[1]	The *Framework logical lineage* export provides information on **derivation and generation transformation rules** and related lineage from the BIRD Input Models to certain destination frameworks, including the destination entities and rendering tables linked to the transformation rules.	The ***Logical transformation rules*** sheet contains a list of WUDEN, derivation, and generation rules. The ***Entity logical lineages*** sheet contains information on the incoming lineage for a certain destination (output) entity, its attributes, and allowed values. The ***Rendering logical lineages*** sheet contains information on the incoming lineage for a certain destination rendering table based on its value combinations (i.e., data points), attributes, and allowed values.
***Framework matrix* Export**	The *Framework matrix* export helps users exploring entities, attributes, restrictions, and allowed values of the selected data models while allowing to easily compare the structure of its entities.	Entities, attributes, restrictions, and allowed values of selected data models.
***Individual entity (Cube)* Export**	The *Individual entity (Cube)* export helps business users to navigate metadata of individual data models entities in a user-friendly way, allowing users to explore all the attributes, restrictions, and allowed values of the selected entities. This export is suitable for those Frameworks with a limited number of entities.	Specific entities structures and possible values.

Table 4: BIRD simplified export types.

[1] When exporting the *Framework logical lineage*, please ensure to select the BIRD Input Models and the destination (output) framework of interest.

2.3.2. Technical exports

For users who require a more detailed and comprehensive view of the BIRD metadata, technical exports provide a highly detailed metadata model that covers all possible use cases covered by the SMCube Methodology. These exports are more complex and require knowledge of the SMCube methodology.

Export Type	Description	Content
***All metadata* technical (SMCube) Export**	Download of all metadata or single tables in their technical, implementation format (SMCube).	All metadata per object type.
***Framework(s)* technical (SMCube) Export**[2]	Download of framework-specific metadata in their technical, implementation format (SMCube).	Framework-specific metadata, including the respective **mapping**, **rendering**, and **transformation** packages.

Table 3: BIRD technical exports.

Please refer to the SMCube documentation and the SDD Framework for a detailed definition of the SMCube metadata model.

2.4. Navigating BIRD metadata

A fourth method to access BIRD metadata is via the BIRD Navigator. The BIRD Navigator allows users to easily navigate the BIRD content. It provides an extensive set of features for BIRD users and contributors to investigate and analyse BIRD data models, glossary, and Transformation Rules.

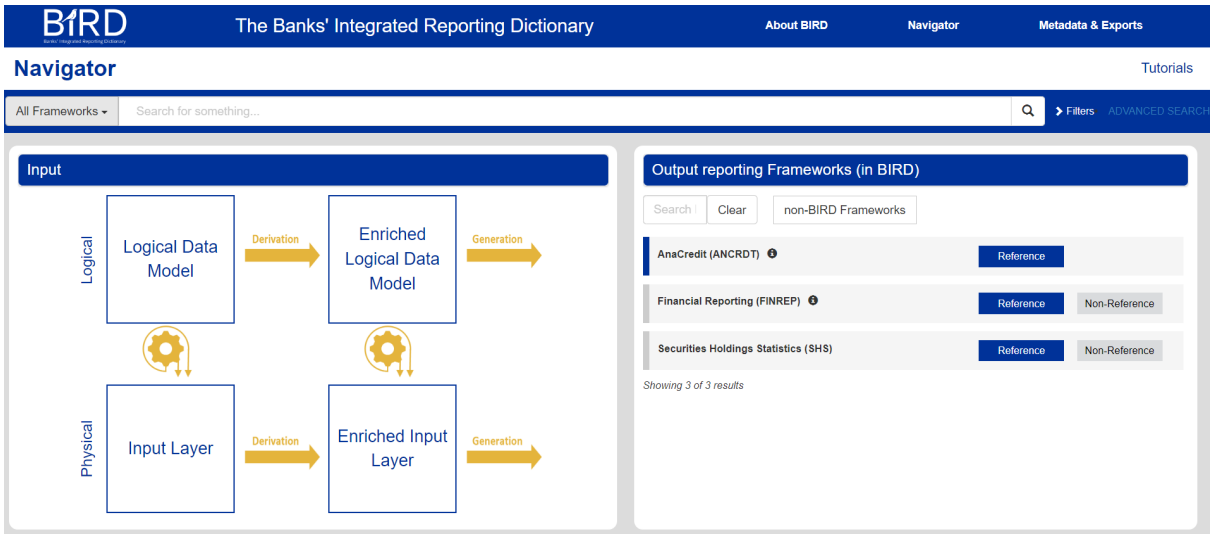


Figure 1: BIRD Navigator page.

Please refer to the following user guide for more information.

[2] To export the logical lineage between two frameworks, ensure all frameworks of interest are selected. For example, to export the transformation package that includes the logical lineage from BIRD Input Models to FINREP, select BIRD and FINREP Reference Frameworks and check the "Including content of the transformation package" option.

3. Data Models

This chapter aims to describe the different types of data models included in the BIRD and their main characteristics.

The first section refers to data models expressed in the form of Entity Relationship Models or flat structures (e.g., AnaCredit, IReF).

The second section refers to data models expressed as templates (e.g., FINREP).

3.1. Entity Relationship Models (ERMs) / Flat dataset

A common type of data model is the Entity Relationship Model (ERM) which is composed of a set of entities and relationships connecting those entities. These entities can also be grouped/classified in accordance with a certain business classification. It should be noted that ERMs can converge to a flat dataset (i.e., single entity) whenever they are de-normalised (i.e., flattened) or extremely simple.

BIRD includes several ERMs of different kinds and in different phases of the BIRD process.

3.1.1. Logical Data Model (LDM) and Enriched Logical Data Model (ELDM)

The BIRD Logical Data Model (LDM) is a detailed logical Entity Relationship Model. It provides a description of the necessary business requirements and their relationships, helping users understand them for reporting purposes. This model encompasses several structural properties of the business requirements which would otherwise need to be defined as validation rules (e.g., the Legal Entity Identifier is only applicable to organisations).

The BIRD Enriched Logical Data Model (ELDM) is an extension of the BIRD LDM and includes all the attributes (i.e., variables) derived via Derivation Transformation Rules.

In addition to the general ERMs characteristics which can correspond also to physical ERMs. LDMs include information about how entities can be organised in a hierarchical structure where, for example, a Legal person entity can be defined as a subtype of the Party entity (Party entity is considered a supertype of a Legal person entity).

BIRD describes these data models, including these types of relationships, as metadata.

3.1.2. Input Layer (IL) and Enriched Input Layer (EIL)

The BIRD Input Layer (IL) can serve as a possible blueprint for a physical interface model. The IL is derived (i.e., forward engineered) from the BIRD LDM. Even though it includes the same business components as the LDM, it is more compact and easier to browse but less informative and explicit than the LDM. Based on the individual IT landscape of an institution, the institution might decide to create their own individual physical models based on the BIRD LDM.

The BIRD Enriched Input Layer (EIL) is an extension of the BIRD IL and includes all the attributes (i.e., variables) derived via Derivation Transformation Rules.

BIRD describes these data models as metadata. It also provides metadata to describe the Transformation Rules (please refer to the Transformation Rules chapter) depicting data lineage from (E) LDM to (E)IL.

The BIRD IL and EIL models are also the basis for logical Transformation Rules in BIRD.

3.1.3. Output Layer (OL) ERMs

The Output Layer (OL) describes the original regulatory reporting requirements, and it reflects the way data are organised for reporting purposes.

The Output Layer can be divided into the Reference Output Layer (ROL) and the Non-Reference Output Layer (NROL). The former describes reporting requirements using the standardised BIRD "reference" codes and descriptions. The latter uses the codification systems of the related regulation (e.g., for FINREP, from the EBA's DPM database).

Output reporting requirements can be based on ERMs (e.g., AnaCredit, future IReF) and in BIRD are described as such, in a similar way as the (E)IL. These reporting requirements should be then delivered to the authorities, on an entity-by-entity basis.

BIRD describes Generation Transformation Rules necessary to generate these reports from the EIL (please refer to the Generation Transformation Rules) chapter.

3.1.4. How to find or get ERMs metadata?

BIRD input and output BIRD ERMs can be browsed easily and in a user-friendly way from the BIRD Navigator main page (see, for example, the BIRD LDM model).

The BIRD ERMs can also be fetched using BIRD metadata access methods.

3.1.4.1. BIRD API

The BIRD API GET /**dataModels**: Retrieves data structures, relationships, and attribute restrictions for BIRD Input Data Models, including the Logical Data Model (LDM), Enriched Logical Data Model (ELDM), Input Layer, Enriched Input Layer, and ERM-based Output Model.

Similarly, GET /**entities** provides specific data structures, attributes, and restrictions for data models entities, but omitting relationships between them. This can also be used for flat structures.

3.1.4.2. BIRD simplified exports

The BIRD ***Framework models overview*** simplified export retrieves data structures, relationships, and attribute restrictions for BIRD Input Data Models, including the Logical Data Model (LDM), Enriched Logical Data Model (ELDM), Input Layer (IL), Enriched Input Layer (EIL), and ERM/flat dataset-based Output Model.

BIRD also provides other types of simplified exports (e.g., Individual entity export) for more specific needs.

3.1.4.3. BIRD technical exports

BIRD ERMs can also be downloaded via the BIRD technical exports, which are modelled with the SMCube format, in the Metadata & Export section. Please refer to the SMCube documentation for additional information.

3.1.5. What are the main characteristics?

The main characteristics of the ERM are their entities, which are organised in business groups/clusters (e.g., Instrument related, Party related entities, etc., i.e., the colour of the ERM entities in the BIRD Navigator) and the relationships between them which can be either Association or logical Generalisation (i.e., subtyping). To be noted that Association relationships can be used for multiple purposes:

- To define referential integrity between entities (e.g., an Instrument's Debtor Party must exist in the Party table)
- To define automatic joins between entities involved in Transformation rules. When a Generation or Derivation Transformation Rule has its source attributes belonging to several source entities, they can be combined/joined to generate/derive the resulting entity/attribute. To join/combine them, a path should be identified based on the available Association relationships defined by the Entity Relationship Model. This path should also define the necessary associations between attributes that specify the "on" condition of the join. Furthermore, relationships mandatoriness also identify which type of joins should be applied[3]. It should be noted that there could be multiple possible paths to join/combine source entities. In such cases, the user is responsible for identifying and implementing the necessary data joins.

Each entity object is then composed of attributes, some of which are defined as keys (i.e., Dimensions). Every attribute is defined with a certain restriction; if enumerated, it will be associated with a list of allowed values, if non enumerated, it will be associated with a set of constraints (e.g., max value, max length etc.)[4].

3.2. Templates (Rendering Tables)

Reporting requirements of the OL can also be template-based (e.g., FINREP). These reporting requirements are described as rendering tables (i.e., annotated templates), similarly to the Data Point Model (DPM), and are composed of cells associated to data points to be reported. Such templates often represent a pivot table of data aggregated from banks internal system. BIRD describes Generation Transformation Rules necessary to generate these reports from the EIL (see next chapters).

[3] A "Full join" should be defined when neither of the two edges of the relationship is mandatory, "inner join" when they are both mandatory and "left/right" join when only one of the two is mandatory.
[4] Please note that in the SMCube we provide more detailed information for these types of information. For example, an attribute defined as a generic concept is called Variable. An attribute defined in a specific Cube context is called Cube Structure Item. We also distinguish two types of restrictions: a Variable Set is a restriction having Variables as allowed values (e.g., for the DPM metric dimension), a Subdomain is a restriction having Members as allowed values.

3.2.1. How to find or get Rendering Tables metadata?

The BIRD output annotated templates (i.e., Rendering Tables) can be browsed easily and in a user-friendly way from the BIRD Navigator output Frameworks (e.g., FINREP Framework).

The BIRD ERMs can also be fetched using BIRD metadata access methods.

3.2.1.1. BIRD API

The BIRD API GET /**renderingTables**: Retrieves annotated templates (i.e., Rendering Tables), rendering structures, and related data points definition and location.

GET /**entities**: Provides specific data structures, attributes, and restrictions for Rendering Tables underlying aggregated data structures (entities).

3.2.1.2. BIRD simplified exports

The BIRD ***Framework models overview*** simplified export retrieves templates underlying data structures (i.e., entities).

3.2.1.3. BIRD technical exports

The BIRD Output rendering tables can also be downloaded via the BIRD technical exports modelled in SMCube format in the Metadata & Export section. Please refer to the SMCube documentation for additional information.

3.2.2. What are the main characteristics?

The main characteristics of the Rendering Tables are the columns, rows, and sheet headers each of which is defined based on a code (e.g., “020”) and a name (e.g., “Credit card debt” label). Rendering Tables are organised in cells which are associated to certain coordinates based on the combination of the related header codes.

Each cell is then associated to a data point (i.e., value combination) which enables to identify the key characteristics of the data points. These characteristics are instrumental in defining the generation rules that produce the reported data point (please refer to the Generation Transformation Rules chapter).

Related to each template, an entity (i.e., Cube) is defined to represent templates’ underlying “combined” structure. These are aggregated data structures which follow a similar definition as the entities metadata described in the previous chapters.

4. Transformation Rules

This chapter is designed to elucidate the structure and format of the Transformation Rules needed to derive additional BIRD content from “raw” data and to generate the output reporting Frameworks.

Within the BIRD project, we differentiate between two types of logical/semantic transformation rules: derivation and generation rules. Combined, they are used to explain how data is transformed from the Input Layer to produce the Output Layer (i.e., specific regulatory reports).

Derivation rules describe how input data found in the BIRD Data Models can be used to calculate derived information, such as the calculation of the current loan-to-value ratio. Generation rules describe how to filter the BIRD Data Models' entities and attributes and how to identify the type of aggregation that is needed to generate a specific regulatory report. The derivation phase (i.e., enrichment of concepts using primary concepts available in the input) starts from the Logical Data Model (LDM) or the Input Layer (IL). The generation phase (i.e., creation of output reports by filtering, aggregating, and selecting existing concepts) takes place from the Enriched IL (EIL) (Figure 1).

Additionally, in BIRD, Transformation Rules are described in two levels: Semantic and Logical Transformation Rules.

Please note that Validation rules can also be considered a type of Transformation rule, but they will be explained in the next chapter due to their specific characteristics.

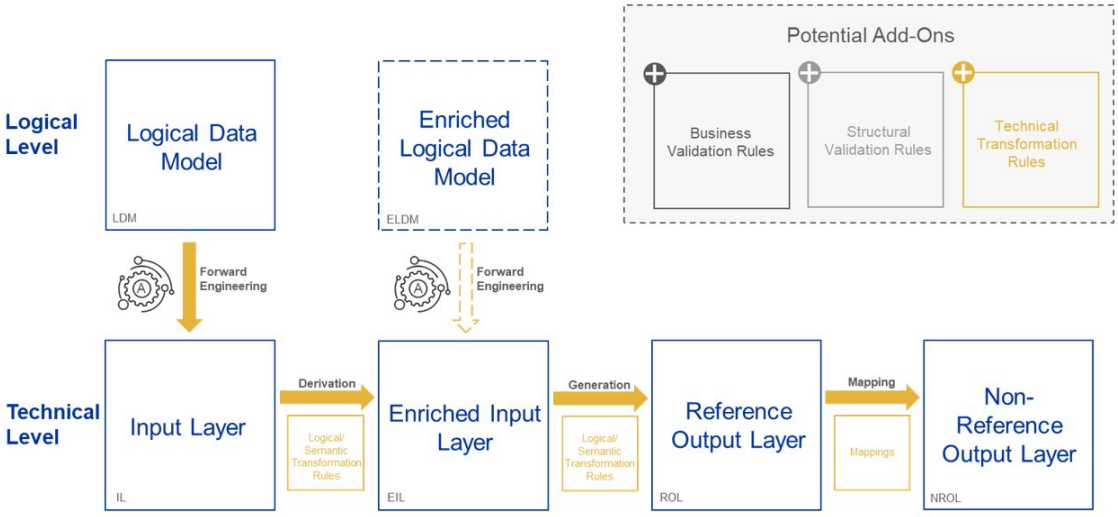


Figure 2. The BIRD process describes how the components of the BIRD are interconnected. The derivation and generation phases are documented using both logical and semantic transformation rules.

4.1. Levels of the Transformation Rules

In BIRD, we describe Transformation Rules on two levels, Semantic and Logical Transformation Rules.

Semantic (a.k.a. Conceptual) Transformation Rules describe the rules to business users on a "conceptual"/descriptive level; they are connected to generic data attribute concepts (e.g., the Gross Carrying Amount Derivation Rule) or to the generated entity/Cube (e.g., the Generation Rule of the FINREP F 05.01 annotated template). This type of transformation is not entirely machine readable but rather a business representation of the rule.

Logical Transformation Rules describe the rules on a "Logical" level; they are machine readable and connected to specific data models entity, attributes, and allowed values (e.g., the Gross Carrying

amount Derivation Rule for the EIL instrument entity) or to the generated entity/Cube (e.g., the Generation Rule of the FINREP F 05.01 entity/Cube and its related data points). They are designed to be translated into a physical implementation by the user.

4.2. Derivation Transformation Rules

Derivation Transformation Rules specify the process for deriving attributes within the BIRD Data Models from the "raw" data or information collected as input in the Data Model.

Derived attributes provide with additional business value to better align raw input information to the reporting requirement representation of data.

4.2.1. What are types of derivation rules?

Derivation rules can be defined in various ways to derive both non-enumerated attributes (e.g., Gross carrying amount) and enumerated attributes (e.g., Enterprise size). These rules can either be independent, based solely on raw source data, or dependent on other derived attributes, using them as source information. When derivation rules depend on other derived attributes, users must identify and manage the chain of dependencies between these rules.

Furthermore, derivation rules can be defined using a formal derivation algorithm or described semantically in words.

4.2.2. How to identify derived attributes?

In the BIRD Data Models, derived attributes are included under derived entities, which are identified in the Enriched Logical Data Model (ELDM) with the word "derived" (Figure 2).

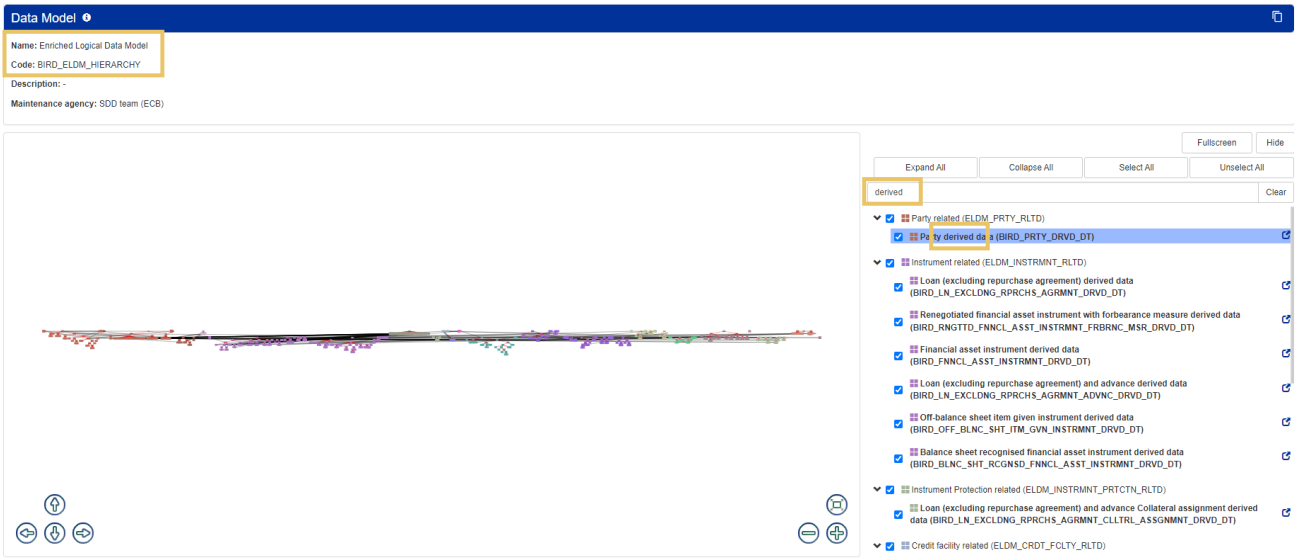


Figure 3. How to identify derived attributes in the ELDM via the BIRD website.

4.2.3. How to find the Derivation Rules?

Semantic Derivation Transformation Rules can be found in the BIRD website under the respective Variable's page. Via the search bar in the BIRD Navigator[5], it is possible to get to a specific Variable page by typing in either the name or code of said Variable. For example, by typing "current loan-to-value ratio" in the search bar, the first option that pops-up is the respective Variable. Once at the Variable's page, there is a link "How to derive this Variable: Semantic transformation rule" that opens a pop-up with the description of that Derivation Rule (Figure 3).

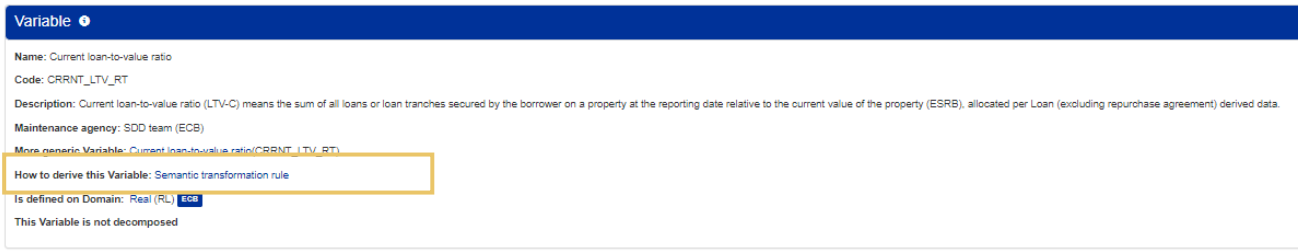


Figure 4. How to find the Semantic Derivation Rules in the BIRD website.

Additionally, BIRD provides multiple formats for accessing Logical Derivation Rules, which can be readily converted into a practical or technical implementation.

BIRD APIs

The BIRD API GET /**incomingLogicalLineage** can be used to fetch Logical Derivation Rules for a certain entity which includes the related derived attributes. Such entity can be selected by code or businessId parameters.

Simplified & Technical Exports

The BIRD Derivation Rules can be downloaded via the BIRD simplified and technical exports, which describe the derivation process to obtain derived attributes included in the entities of a certain Framework.

Please select the BIRD Framework[6] to download BIRD Logical Derivation Rules.

4.2.4. How are the Derivation Rules structured?

Semantic Derivation Rules

[5] For more information on how to navigate the BIRD website and explore its different functionalities, please consult the BIRD Navigator User guide.
[6] Derived attributes are included in the ELDM and EIL data models which are part of the BIRD (input models) Framework.

The description of the Semantic Derivation Rules is divided in seven sections – Derived Attribute, Input Attributes, Type of transformation, Verbal description, Legal and other background documentation, Formal description, Technical (pseudo-SQL) description and, for some cases, also Examples.

In the first two sections, it is possible to find information about the Derived Attribute and the respective Input Attributes needed for the calculations. Under the "Transformation description", there is information about the type of derivation, and a verbal, formal, and pseudo-technical description of the Derivation Rules (Figure 4).

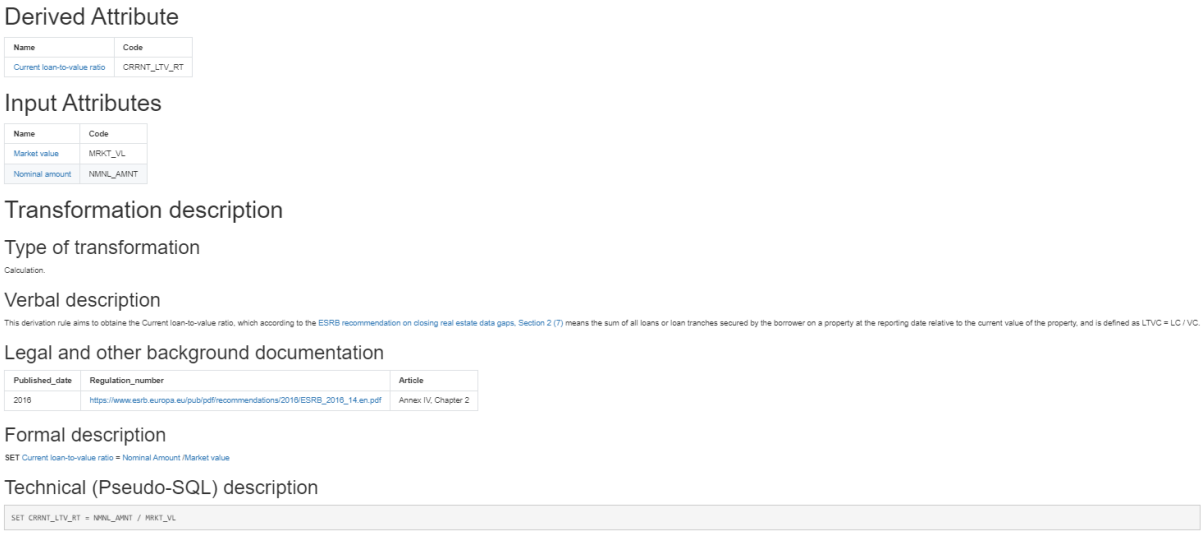


Figure 5. Derivation Rules for Current loan-to-value ratio.

Please note that Attributes and Allowed Values are always described by a Name and a Code. For Allowed values, their codes are represented either by numbers or a combination of letters and numbers. In the Formal Description of the derivation rule the Allowed Values' codes are showcased under brackets (e.g., "Performing (11)". In the Technical Description, the Allowed Values are represented by their IDs, which are a combination of the Domain where these values belong and the respective member code. For example, the allowed value "Performing" is represented by the code 11 under the Domain Credit Quality (CRDT_QLTY) and, thus, its ID is "CRDT_QLTY_11".

Logical Derivation Rules

On the other hand, Logical Derivation Rules focus on describing the rule formally in terms of their algorithm and the connection between source (i.e., input of the formula) and destination (i.e., derived information) entities and attributes of the derivation. One Logical Transformation Rule per derivation rule is defined with type "DER" and linked to the correspondent Semantic Derivation Rule.

For every Derivation rule, a destination entity and attribute are defined, the rule then can provide the derivation algorithm to be implemented and the source entity and attributes that are required for its derivation.

4.3. Generation Transformation Rules

Generation Transformation Rules describe how from the BIRD Data Models one can obtain the aggregated figures to populate the regulatory reports. These transformations indicate the filter criteria that should be applied to determine the necessary data sets, what are the observations to be generated, and the type of aggregation that should be used (e.g., using the sum function or more complex calculations).

4.3.1. How to find the Generation Rules?

Semantic Generation Transformation Rules can be found in the BIRD website under the page of the template one wants to generate using the BIRD Data Models. Via the BIRD Navigator[7], it is possible to get to a specific template page by going first to the Framework page of interest and then searching for a specific template. For example, by clicking on FINREP's reference Framework, and searching for a specific template, e.g., FINREP template F_05.01, there is a link "How to generate this Cube: Semantic transformation rule" that opens a pop-up with the description of the generation rule (Figure 5).

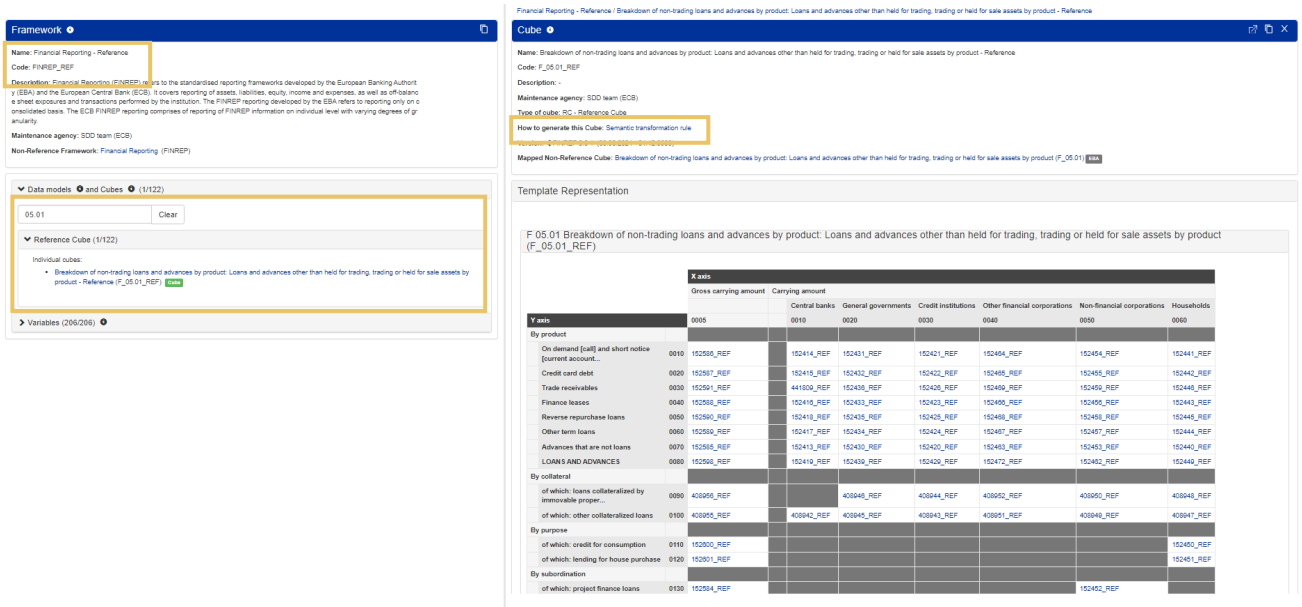


Figure 6. How to find the Generation Rules in the BIRD website.

[7] For more information on how to navigate the BIRD website and explore its different functionalities, please consult the BIRD Navigator User guide.

Additionally, BIRD offers Logical Generation Rules which are based in two different approaches:

- For template-based reporting Frameworks (e.g., FINREP), they eventually specify generation rules for every data point of the templates (i.e., rendering tables) to be reported. It is also possible, via BIRD, to generate the annotated template's underlying entity/Cube.
- For ERM/flat table-based reporting Frameworks they specify the rules for the generation of a certain entity.

For more information on the different aspects of the implementation, please refer to Chapter 5.
Logical Generation rules are available in the below formats.

BIRD APIs

The BIRD API GET /**incomingLogicalLineage** and /**incomingRenderingLogicalLineage** can be used to fetch Generation Rules for a certain reporting entity (Cube) or rendering Table (template), respectively. User can select such entity/reportingTable by code or businessId parameters.

Simplified & Technical exports

The BIRD Generation Rules can be downloaded via the BIRD simplified and technical exports for both template-based and ERM/flat dataset-based reporting Frameworks.

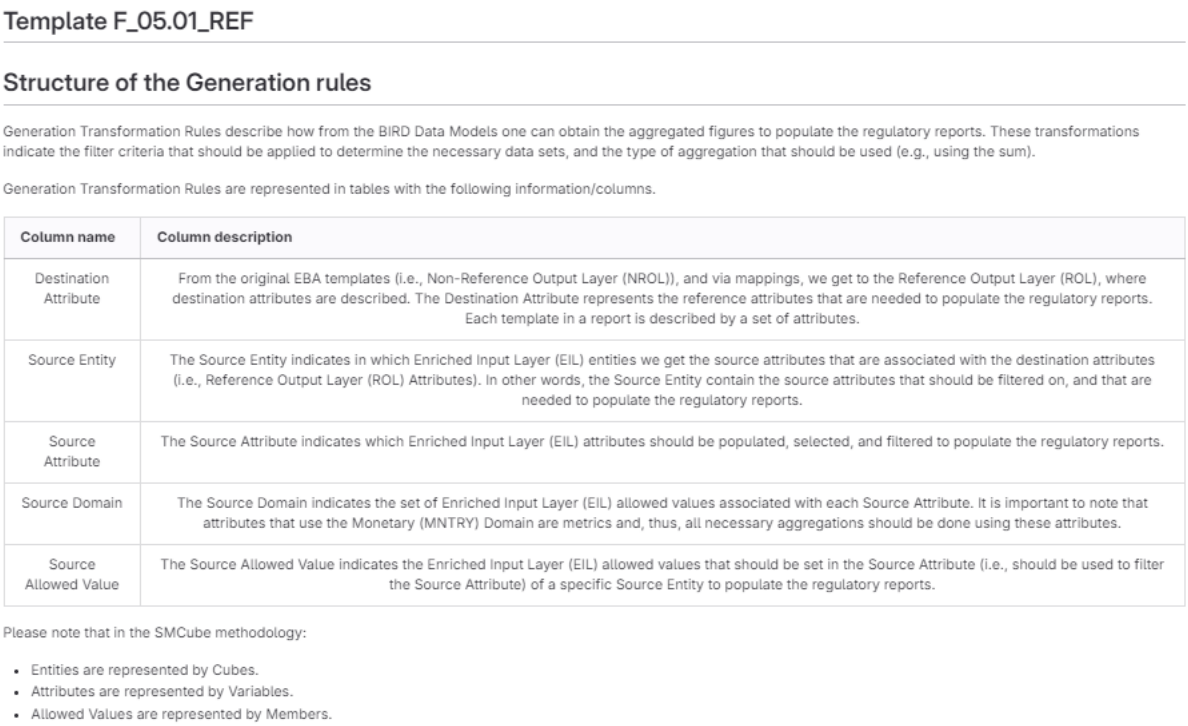
Please select the BIRD Input Models and the destination (output) Framework of interest.

4.3.2. How are the Generation Rules structured?

Semantic Generation Rules

The description of the Semantic Generation Rules is divided in two main sections: structure of the Generation Rules and the Generation Rules themselves.

Under "Structure of the Generation Rules" there are descriptions of the columns that constitute the transformation. In summary, the Destination Attribute represents the reference attributes that are needed to populate the regulatory reports. On the other hand, the Source Entity, Source Attribute, and Source Allowed Value reflect the needed input information that can be found in the BIRD Enriched Input Layer (EIL) (Figure 6). The Source Entity contains the source attributes that are needed to generate the template. The Source Attributes indicates which EIL attributes are needed to generate the content of the template, being used as filter criteria or to calculate the metrics that need to be reported. The Source Allowed Value indicates the EIL allowed values that should be used for a respective attribute (i.e., should be used to filter the Source Attribute) of a specific Source Entity to populate the regulatory reports.



Template F_05.01_REF

Structure of the Generation rules

Generation Transformation Rules describe how from the BIRD Data Models one can obtain the aggregated figures to populate the regulatory reports. These transformations indicate the filter criteria that should be applied to determine the necessary data sets, and the type of aggregation that should be used (e.g., using the sum).

Generation Transformation Rules are represented in tables with the following information/columns.

Column name	Column description
Destination Attribute	From the original EBA templates (i.e., Non-Reference Output Layer (NROL)), and via mappings, we get to the Reference Output Layer (ROL), where destination attributes are described. The Destination Attribute represents the reference attributes that are needed to populate the regulatory reports. Each template in a report is described by a set of attributes.
Source Entity	The Source Entity indicates in which Enriched Input Layer (EIL) entities we get the source attributes that are associated with the destination attributes (i.e., Reference Output Layer (ROL) Attributes). In other words, the Source Entity contain the source attributes that should be filtered on, and that are needed to populate the regulatory reports.
Source Attribute	The Source Attribute indicates which Enriched Input Layer (EIL) attributes should be populated, selected, and filtered to populate the regulatory reports.
Source Domain	The Source Domain indicates the set of Enriched Input Layer (EIL) allowed values associated with each Source Attribute. It is important to note that attributes that use the Monetary (MNTRY) Domain are metrics and, thus, all necessary aggregations should be done using these attributes.
Source Allowed Value	The Source Allowed Value indicates the Enriched Input Layer (EIL) allowed values that should be set in the Source Attribute (i.e., should be used to filter the Source Attribute) of a specific Source Entity to populate the regulatory reports.

Please note that in the SMCube methodology:

- Entities are represented by Cubes.
- Attributes are represented by Variables.
- Allowed Values are represented by Members.

Figure 7. Structure applied to all the Generation Rules.

Under “Generation rules” the information needed to reconstruct a template is broken down into three sub-sections: 1) filters applied to the full template, 2) filters applied to rows, and 3) filters applied to columns (Figure 7). All three sections have information not only about the Source Attributes that should be used, but also about the Source Entities from where these attributes should be taken from, and the Source Allowed Values that should be used. Please note that the “marginal” approach on filtering is a simplification to represent Semantic rules in a business-friendly way. Please refer to the Logical Generation Rules for a solution which is easier to implement.

Semantic Generation Rules outlines the process for populating each cell in a specific reporting template. It begins by applying the BIRD Input Layer model to organize the foundational data set. From this structured data set, specific subsets of data are then isolated through filtering criteria. These subsets are subsequently aggregated to produce the final values required for each cell in the report.

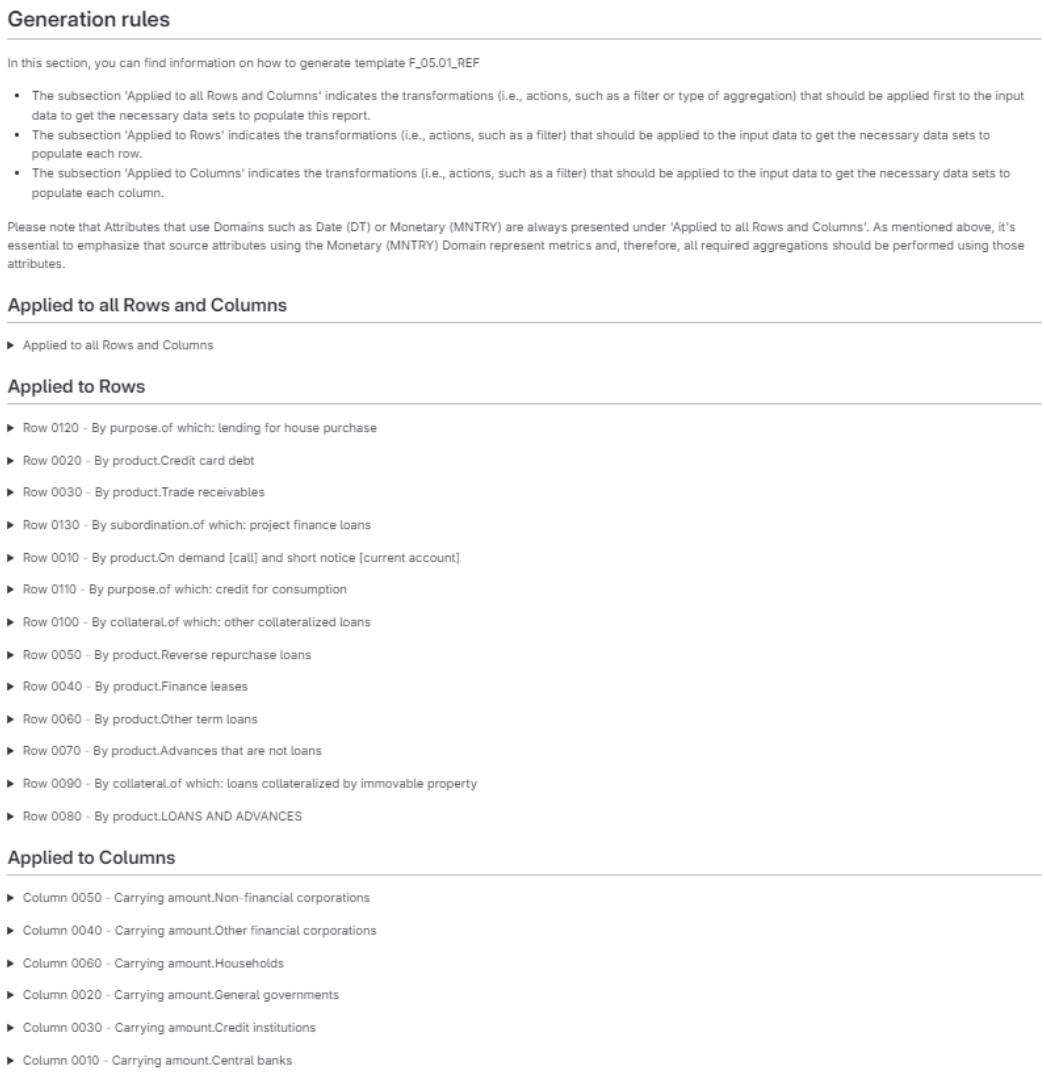


Figure 8. Excerpt of the Generation Transformation Rules documentation for F_05.01.

Logical Generation Rules

Logical Generation Rules are more detailed than their semantic counterpart, as logical rules also describe the source and destination generated entities/rendering tables, attributes, and allowed values that should be used in the generation phase. Furthermore, for observation values, the rules also specify the aggregation function.

Logical Generation rules have a different nature depending on the type of reporting requirement.

ERM/flat structure-based Generation rules:

For every destination entity and attribute to be reported, the Generation rule provides the related source entities and attributes to fetch EIL information from. Additionally, the rule provides the source allowed

value to be filtered from the EIL and how they are reported on the output (destination allowed value). It also provides additional filtering conditions (e.g. for non-enumerated attributes) to be combined (in AND condition) with the allowed values-based filters described before. Finally, the rule provides aggregation function for every destination attribute subject to aggregation.

Template-based Generation rules:

For every reporting data point, a series of destination attributes are defined either as filters (e.g. Institutional sector, identified as “keys”) or measures (e.g. Gross carrying amount). For the former, an allowed value is defined representing the (aggregated) concept associated to respective data point attribute. Such concept is related to a set of source allowed values which represent the values to be filtered from the source attributes of the source entities. To be noted that, for this type of reporting requirement, destination data points attributes and allowed values used as filters are just “logical” metadata used for the generation process and don’t have to be reported as fields.

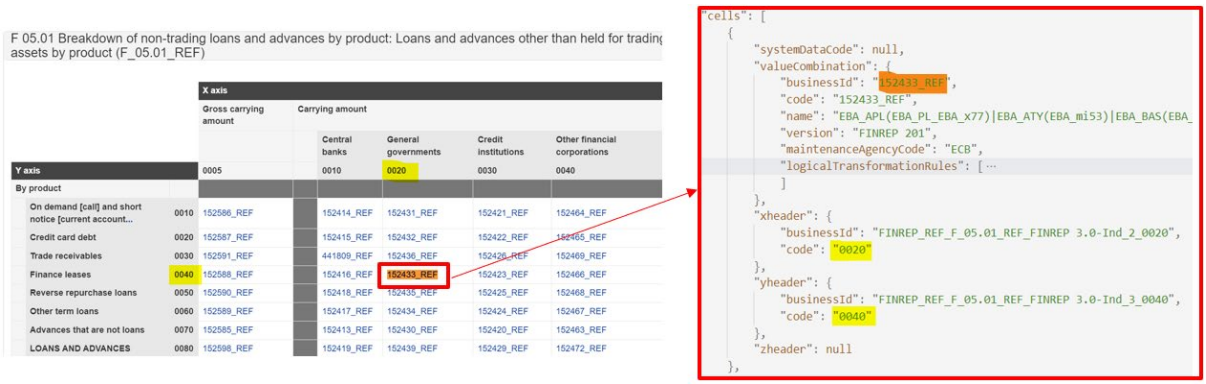


Figure 9: Reporting data points generation (BIRD API)

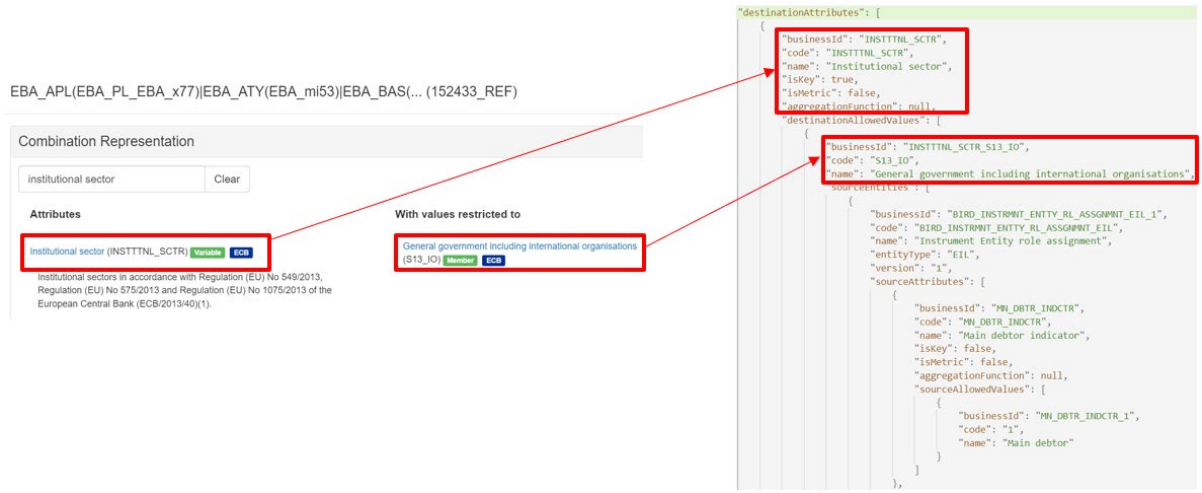


Figure 10: Definition of destination data points attributes and allowed values in BIRD generation rules (BIRD API).

Figure 11: Definition of source entities, attributes and allowed values in BIRD generation rules (BIRD API)

4.4. Handling of *Not Applicable, Not Available*, and *Total*

Some IL or EIL entities (Cubes) can include attributes which are not mandatory and can be not applicable for some records. For example, during the forward engineering procedure, some (E)LDM entities are sometimes “wrapped up” into a more generic entity of the subtype tree to the (E)IL. When this happens, enumerated attributes included in the more specific (E)LDM entities will also be wrapped up into a more generic one with an assigned default value.

For enumerated attributes the default allowed value is “Not applicable (0)”.

For non-enumerated attributes the WS DM and WS P will investigate the possibility to use Null Explanatory Values (NEVs) to distinguish between “Not available” and “Not applicable” (default) Null values.

When the allowed value “Total (-1)” is defined for a certain Transformation Rule attribute on the destination side, all corresponding source attribute values should be considered for the rule, and no filter is needed for that attribute.

4.5. Handling of de-normalisation and aggregation

For every Logical Transformation Rule, a destination entity and a set of source entities are defined. When applying the Transformation Rule, the source entities combined data should be firstly de-normalised and then aggregated in accordance with the dimensional definition of the destination entity.

Consider, as an example, a generation rule involving the following simplified tables:

Instrument

Instrument ID	Instrument type	Carrying amount	Debtor ID
INSTRMNT1	Credit Card Debt	150	PARTY1

INSTRMNT2	Other Loan	250	PARTY2
INSTRMNT3	Other Loan	200	PARTY1

Table 3: Instrument table.

Party

Party ID	Institutional Sector
PARTY1	Financial Institution
PARTY2	Non-Financial Institution

Table 4: Party table.

Instrument – Protection item

Instrument ID	Protection ID	Protection allocated value
INSTRMNT1	ITEM1	50
INSTRMNT2	ITEM1	100
INSTRMNT2	ITEM2	80
INSTRMNT3	ITEM2	70
INSTRMNT1	ITEM3	40
INSTRMNT3	ITEM3	60

Table 5: Instrument - Protection item table

Protection Item

Protection ID	Protection type
ITEM1	Real Estate Protection
ITEM2	Security Protection
ITEM3	Security Protection

Table 6: Protection Item table.

Let us consider now the following output template (i.e., rendering table).

T_X (rendering table)

	Financial Institution	Non-financial Institution
Credit Card Debt	DP1	DP2
Other Loan	DP3	DP4
Loans	DP5	DP6
Real Estate – protected	DP7	DP8
Security – protected	DP9	DP10

Table 7: Template (T_X) rendering table.

Where for example DP1, DP6, DP7 data points are defined as follows:

	DP1	DP6	DP7
Type of instrument	Credit Card Debt	Loans	Total (-1)
Institutional sector	Financial Institution	Non-financial Institution	Financial Institution
Protection type	Total (-1)	Total (-1)	Real Estate
Metric	Carrying Amount	Carrying Amount	Protection allocated Value

Table 8: Definition of DP1, DP6, DP7 data points

Where the defined Dimensions (key attributes) are **Type of instrument, Institutional sector,** and **Protection type** for the rendering table's underlying entity (Cube) while for example DP7 data point we can consider only **Institutional sector** and **Protection type** as relevant.

Furthermore, thanks to the Logical Transformation Rules, we know that "Loans" is a generic (allowed) value which can be either "Credit Card Debt" or "Other Loan", to be included in the source filters on Type of instrument for the generation of DP6. In addition, "Total (-1)" refers to all possible source values for the related attributes.

Thanks to the relationships of the source model entities, we also know how the four source entities can be joined together to obtain the following de-normalised ("flattened") table:

De-normalised table:

Instrument ID	Instrument type	Carrying amount	Debtor ID	Institutional Sector	Protection ID	Protection type	Protection Allocated value
INSTRMNT1	Credit Card Debt	150	PARTY1	Financial Institution	ITEM1	Real Estate Protection	50
INSTRMNT2	Other Loan	250	PARTY2	Non-Financial Institution	ITEM1	Real Estate Protection	100
INSTRMNT2	Other Loan	250	PARTY2	Non-Financial Institution	ITEM2	Security Protection	80
INSTRMNT3	Other Loan	200	PARTY1	Financial Institution	ITEM2	Security Protection	70
INSTRMNT1	Credit Card Debt	150	PARTY1	Financial Institution	ITEM3	Security Protection	40
INSTRMNT3	Other Loan	200	PARTY1	Financial Institution	ITEM3	Security Protection	60

Table 9: De-normalised table. Light blue columns are those part of the primary key.

To be noted that the dimensionality of this De-normalised entity is different than the one of destination entities or data points. This means that an aggregation rule should be applied to generate the results.

We can now execute the transformation by applying the defined filters and the specified aggregation for each destination measure. We then obtain the following values:

- DP1: 150
- DP6: 250
- DP7: 150

It is important to observe that, with a full de-normalisation approach, a distinct should be applied on the source key and metric attributes of the entity where the metric is included to avoid double counting. For example, to generate DP1 we need to use the Instrument's Carrying amount as metric. That means that a distinct should be applied to "Instrument ID", "Carrying Amount" of the Instrument entity after filtering and before applying the aggregation function "SUM".

Please note that the user can decide to apply different levels of de-normalisation for the generation of T_X by, for example, implementing two intermediary tables:

- **Instrument** join **Party (Debtor)** to generate the first section of the template: DP1, DP6 etc.

- **Instrument** join **Party (Debtor)** join **Instrument – Protection** join **Protection Item** to generate the section at the bottom of the template for data points: DP7, DP8 etc.

5. Validation Rules

5.1. Introduction to Validation Rules

A data validation rule is a set of criteria or constraints used to ensure that BIRD input data is correct, complete, and coherent. There are two types of validation rules that can be defined:

- **Structural Validation Rules:** These rules ensure that the data fed into the BIRD physical model is "structurally sound," meaning it is defined in accordance with the BIRD Logical Data Model (LDM). In simpler terms, structural validation rules are used to ensure that the BIRD LDM and BIRD Input Layer (IL) are semantically equivalent. These rules can be derived from the structure of the BIRD LDM. For example, if the LDM specifies that an entity such as a "Legal person" can be subtyped into "Natural person" and "Organisation," structural validation rules ensure that attributes like "First name" apply only to "Natural persons" and not to "Organisations."
- **Business Validation Rules:** These rules ensure that the data adheres to additional business constraints. For example, a business rule might state that the "Maturity date should be later than the Inception date." These validations are considered on top of the structural validation rules and cannot be derived from the BIRD LDM.

Currently, BIRD offers only structural validation rules in a prototype format, which is not available as part of the SMCube or simplified metadata but can be downloaded from the About BIRD section of the BIRD website. All types of BIRD validation rules will be available at a later stage as part of the BIRD metadata.

5.2. Generation of Validation Rules

The basis for generating structural validation rules is the Logical Data Model (LDM). The LDM's structure specifies which attributes apply to which data objects and how different data objects relate to each other.

When the Input Layer (IL) is forward-engineered from the LDM/ELDM, much of this information is lost because all subtypes are merged into a single table. To preserve this LDM information in the IL, we run an additional script in SQL Developer to recreate validation rules. This ensures that the information present in the LDM/ELDM is also available in the IL/EIL, maintaining semantic equivalence between the LDM and the IL.

When the ELDM is updated, these validation rules will also be updated to ensure their applicability to the resulting EIL.

5.3. Structure and Format

We provide validation rules in two JSON files: one containing constraints on relationships or foreign keys, and the other containing constraints on columns of the EIL.

Relationship Constraints JSON File

This file has the following JSON keys:

- **Type:** Always "RELATIONSHIP," indicating the rule applies to a relationship or foreign key.
- **Table:** The table in the EIL involved in these relationships.
- **Entity:** The entity in the ELDM from which the table is generated during forward engineering.
- **Relationship:** A list of lists specifying the subtype entities to which the relationships are constrained.

Example:

```json
{
  "type": "RELATIONSHIP",
  "table": "ASST_PL_DBT_SCRTY_PSTN_ASSGNMNT",
  "entity": "Asset pool Debt security position assignment",
  "relationship": [
    ["ASST_PL", "ASST_PL_DBT_SCRTY_PSTN_ASSGNMNT"],
    ["LNG_DBT_SCRTY_PSTN", "ASST_PL_DBT_SCRTY_PSTN_ASSGNMNT"]
  ]
}
```

This rule indicates that the table `ASST_PL_DBT_SCRTY_PSTN_ASSGNMNT` is connected to a hierarchical entity `ASST_PL` on one side and to a subtype entity `LNG_DBT_SCRTY_PSTN` on the other, restricting the original relationship with the table `SCRTY_PSTN` to one of its subtypes.

Column Constraints JSON File

This file has the following JSON keys:

- **Table:** The table in the EIL containing the restricted column.
- **Type:** Always "IF," indicating the rule applies to a column.
- **Attr:** The discriminator column on which the rule is based.
- **Comparator:** "=" (equal) or "!=" (not equal), indicating whether `Attr` should be equal to or not equal to the following entity.

- **OriginalEntityName:** The member of the discriminator column's domain to which this rule applies.
- **Entities:** A dictionary of codes and names of entities in the ELDM corresponding to `OriginalEntityName`.
- **Value:** The column constrained by the rule.
- **OriginalValueName:** The full name of the column in `Value`.
- **AssertComparator:** "equal" or "not equal to," connecting `Value` and `AssertValue`.
- **AssertValue:** "NULL" or "Not applicable," limiting the `Value` column in the rule.

Example:

```json
{
  "table": "PRTY",
  "type": "IF",
  "attr": "TYP_PRTY",
  "comparator": "!=",
  "entities": {
    "19": "Self-employed natural person",
    "18": "Non-self-employed natural person"
  },
  "value": "LST_NM",
  "originalValueName": "Last name",
  "assertComparator": "=",
  "assertValue": ["NULL"]
}
```

This rule indicates that in the table `PRTY`, if the column `TYP_PRTY` is not equal to `SLF_EMPLYD_NTRL_PRSN` or `NN_SLF_EMPLYD_NTRL_PRSN`, then the column `LST_NM` must be `NULL`.

5.4. Usage

The validation rules are provided in JSON format, allowing for translation into any programming language or format for implementation in the database. They can be used as additional checks on database content or as constraints at the input level. The current format serves as a prototype, and we welcome suggestions for improving the content and/or format.

6. Historization

BIRD offers an historisation system to represent metadata definitions which are valid at different points in time, for the same underlying logical definition. For example, the BIRD LDM Party entity can be updated in different points in time (e.g., by adding a new attribute) and the change should normally be depicted by introducing a new version of such entity. Every version is defined for a certain time range (valid from – valid to). Some BIRD metadata objects (e.g., restrictions/Subdomain) are not versioned but only "historized"; this means that they are not defined based on a series of versions, but their components (items) are defined in accordance with a certain validity range.

In this chapter we will explain how to handle versioning and historization in BIRD, for the various metadata access methods.

Please note that, as of July 2024, the BIRD does not make use of the versioning system explicitly but keeps every input model metadata object with its first version. The BIRD team will however soon introduce historisation for its metadata.

6.1. BIRD API simplified endpoints versioning system

BIRD API simplified endpoints normally include three parameters to handle historization validities:

- **validOn**: It allows to retrieve only metadata valid at a certain point in time, including the related information (when includeRelated=True). Normally this corresponds to a single version of the data model.
- **validAfter**: It allows to retrieve all metadata valid after a certain date, including the related information. This can retrieve multiple versions for the same underlying logical information.
- **validBefore**: It allows to retrieve all metadata valid before a certain date, including the related information. This can retrieve multiple versions for the same underlying logical information.

Additionally, for versioned objects (such as the entity), it is also possible to filter by the version field using the "filter" parameter. Please note that this will only affect objects of the level 1 of the result object.

6.2. BIRD simplified exports versioning system

In the BIRD Metadata & Export section, it is possible to extract metadata which are valid for a certain date by selecting the validity date in the export form.

6.3. BIRD technical exports versioning system

The BIRD technical exports are modelled in accordance with the SMCube implementation model. The SMCube offers the possibility to historize some of its object types.

For example:

- A Subdomain is **historised**: each of its SUBDOMAIN_ENUMERATION is associated to a VALID_FROM, VALID_TO dates which define validities of its single components.

For more details on the SMCube methodology and the versioning of its metadata objects, please refer to the SMCube documentation and the SDD Framework.

7. Mappings

Mappings in BIRD represent a specialised type of rules that are needed to translate dictionaries that are not semantically integrated into the "reference" dictionary of BIRD. The "non-reference" dictionaries of the Non-Reference Output Layer (NROL) use different definitions, concepts, and codes, which need to be mapped to the "reference" dictionary of BIRD.

Currently, the main purpose of mappings is to relate BIRD definitions for supervisory frameworks to the original EBA ITS definitions provided in the DPM database. This allows users to compare the different sets of definitions and provide feedback on their consistency.

For every supervisory template/entity (Cube), we provide an entity mapping that describes the mappings between each attribute (for measures and dimensions, e.g., Institutional Sector in BIRD with Counterparty Sector in DPM) and the related allowed value (e.g., S12 in BIRD with x8 in DPM for Financial corporations).

8. BIRD implementation guide

This chapter specifies possible approaches to implement a BIRD solution. Please note that the BIRD is a non-mandatory product which aims at describing business requirements in a logical way while leaving freedom on the implementation solution. The implementation is determined by each users individual IT landscape and needs. Steps documented in this chapter are only a suggested approach and can be amended or complemented with additional steps/procedures depending on the user's individual need.

8.1. Implementation of the input model

8.1.1. Implementation of the Enriched Input Layer

A common approach is to implement EIL model directly without implementing first the IL. Derivation rules will be then implemented from/to the same implementation model or simply ignored (i.e., derived attributes will be then directly populated by the bank, when necessary).

Implementing the EIL involves transforming the raw data into a format that can be easily utilized for reporting.

Implementation Steps:

1	**Implementation of the EIL**	Create tables, columns, keys, and technical data types based on the EIL definitions[8].
2	**Extraction of relevant data from Source Systems**	Identify the relevant data sources of the user system.
		Extract raw data that includes all necessary attributes and values.
3	**Source data mapping and feeding the EIL**	Map raw data from source systems into the EIL implementation system.

[8] Please note that EIL is very close to the proposed physical implementation of the BIRD database but formally is still a logical model. In fact, it does provide only logical data types. Technical data types should be generated by users in accordance with the logical types.

4	**Validation rules**	Implement and run validation rules on the content of the EIL[9]

Table 10: Steps to implement the EIL.

8.1.2. Alternative approach: Implementation of a custom Input Layer model

In certain instances, such as for technical reasons, organizations might choose to implement a custom IL. This approach offers greater flexibility but necessitates a thorough understanding of the LDM, enabling the user to apply a custom forward engineering procedure to the custom IL.

Implementation Steps:

1	**Define a custom EIL by forward engineering the LDM.**	Decide how to de-normalise/wrap the LDM entities.
		Generate the custom IL implementation model.
2	**Define a new set of structural validation rules based on the custom EIL implementation.**	Generate a custom set of validation rules which depend on the de-normalisation and wrapping choices made as part of the forward engineering procedure.
3	**Data extraction, mapping/feeding and validation**	Reproduce step 2, 3 and 4 of the previous section with the newly defined structural validation rules.

Table 11: Steps for the implementation of a custom Input Layer model.

8.2. Derivation of attributes

Derived attributes are derived from raw data and provide enriched data that is necessary for reporting. This chapter details the (currently) most used methods among users to identify and derive attributes and explores an alternative approach to manage the IL and EIL.

Implementation Steps:

1	**Identification of derived attributes**	Identify derived attributes in the EIL for which Derivation Transformation Rules are defined.
		Alternatively, the user can identify the attributes present in the EIL but not in the IL or whether the attribute belongs to a derived entity in the ELDM.
2	**Apply Derivation Rules**	Apply algorithms of the derivation rules from source to destination attributes. Please note that this operation should be done iteratively, identifying the chain of dependencies occurring between different derivation rules (please refer to chapter 3). Please note that this could require aggregation of different records in case the keys (dimensions) of the destination entity are different than the combined keys of the source entities.
		Store the results in the EIL destination attributes.

Table 12: Population of derived attributes.

8.2.1. Alternative approach: separation of the Input Layer from the Enriched Input Layer

An alternative approach to managing derived attributes involves maintaining a clear separation between the IL and the EIL. This approach can offer several benefits, including improved data management, flexibility, and clarity in the data transformation process.

Implementation Steps:

1	**Definition of the Input Layer**	Implement the IL based on the metadata definitions using (e.g.) BIRD **dataModel** API endpoint.
2	**Definition of the Enriched Input Layer**	Implement the EIL based on the metadata definitions using (e.g.) BIRD **dataModel** API enpoint.
3	**Implementation of derivation rules**	Implement derivation rules and store results in the EIL. Please note that in some cases Transformation Rules are defined iteratively. Therefore, derivation rules source

[9] Please note that, as of Q2 2024, BIRD does not include validation rules which will be introduced at a later stage.

		attributes can either refer to raw data (i.e., included in the IL) or attributes which have been already previously derived (i.e., attributes which are already defined in the EIL).

Table 13: Implementation of the derivation rules by separating IL from EIL.

8.3. Generating reporting Frameworks

This chapter covers the processes involved in generating Entity Relationship Model (ERM) / flat structure-based reporting requirements and template-based reporting requirements.

8.3.1. Generating ERM/flat-based reporting requirements

ERM/flat structure-based reporting frameworks (e.g., AnaCredit and future IReF) are structured to represent data in a relational format. The generation process for these frameworks involves several key steps:

Implementation Steps:

1	**Identification of reporting requirements**	Determine the specific reporting requirements that need to be fulfilled. Identify the list of businessIds/codes for the entities (Cubes) to be reported. You can (e.g.,) use the BIRD API **entities** endpoint filtering for (e.g., in the future) IReF Framework to identify the necessary dissemination entities to be reported.
2	**Identification and extraction of required source information**	For each entity to be reported (destination), identify the required source entities, attributes and allowed values to be used as input for the generation. You can use (e.g.,) the BIRD API **incomingLogicalLineages** endpoint, filtering for the business IDs/codes obtained in the previous step.
3	**Generation of output entities**	For each entity to be reported, generate the required (destination) attributes from the related source attributes, properly filtered by the defined filters. Filters should be the combination of filters defined by the source allowed values (for every source attribute, in an AND condition) and the defined "additional filters" (to be applied in AND condition to the source valued-based filters). To be noted that, if the keys of the destination entities are different from the combined dimensions of the source entities, an aggregation should be applied to every observation, to obtain each destination attribute, based on the specified aggregation formula.

Table 14: Implementation steps to generate ERM/flat structure-based reporting requirements.

8.3.2. Generating Template-based reporting requirements

Template-based reporting frameworks are those defined mainly as a set of "pivot" tables of aggregated data. The generation process for such frameworks involves generating every data point of a set of reporting templates. This can be achieved by generating firstly intermediary tables or by generating the required data points directly.

8.3.2.1. Phases for implementing the generation of supervisory reports

A possible fully-fledged approach to generate supervisory reporting requirements can be represented as follows:

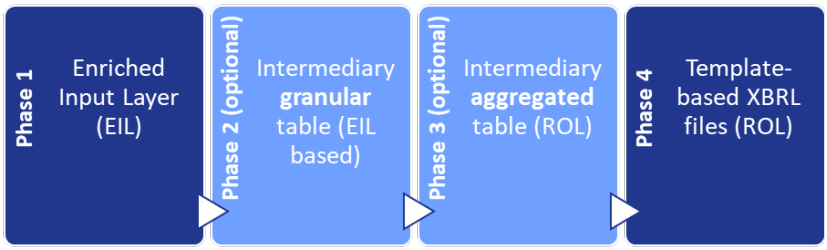


Figure 12: Phases for implementing the generation of supervisory reports

Starting from the EIL, the defined process flow comprises three additional implementation phases: the implementation of an **intermediary granular table** which is closer to the EIL structure and volume, the implementation of an **intermediary aggregated table** based on a data point-by-data points generation process and, ultimately, the creation of the **template-based XBRL files**.

To be noted that the intermediary granular and aggregated tables are optional steps which can be implemented for reasons of efficiency.

8.3.2.2. Implementation of intermediary tables

For each annotated supervisory template, BIRD provides an underlying entity (Cube) containing metadata relevant to the template. This entity offers an overview of all applicable attributes and their associated allowed values, both aggregated and granular (e.g., “credit card debt” and “loans and advances”). Users can leverage this entity to create intermediary tables, streamlining the generation of output reporting requirements, rather than generating each data point directly from the EIL. The entity's level of granularity matches that of the annotated templates, but combined with BIRD Generation rules, it can be used to create either a granular intermediary table or an aggregated one with the necessary data points.

8.3.2.2.1. Implementation of a granular intermediary table

A common approach is to implement, for each reporting template, an intermediary dataset including all the EIL joined (flattened) information necessary to generate such annotated template (e.g., F 05.01). The granularity of this dataset corresponds to the granularity of the combined EIL entities involved in the template generation process.

Implementation Steps:

1	**Identification of reporting requirements**	Determine the specific reporting requirements that need to be fulfilled. Identify the list of businessIds/codes for the reference[10] entities (Cubes) to be reported. You can (e.g.,) use the BIRD API **entities** endpoint (includeRelated=False) filtering for the FINREP_REF Framework (frameworkCodes=FINREP_REF).
2	**Identification and extraction of required EIL information**	For each entity to be reported (destination), identify the required source entities, attributes and distinct allowed values to be used as input for the generation. You can use (e.g.) the BIRD API **incomingLogicalLineages** endpoint, filtering for the business IDs/codes obtained in the previous step.
3	**Implementation of granular intermediary datasets**	For each entity of the previous step, create an intermediary dataset including all the necessary source information from the EIL including all the key attributes (dimensions) of the source entities Necessary join conditions should be applied by the user based on the relevant relationships between entities depicted by the EIL model.

Table 15: Implementation steps of a granular intermediary table.

8.3.2.2.2. Creation of data point-based intermediary table

The next step is to define an intermediary dataset that includes data for all the relevant data points of the template. This dataset's granularity matches that of the output reporting template. Consequently, generating this intermediary dataset requires aggregating the data obtained in phase 2.

[10] The reference layer is the only one needed for the generation phase since it provides more easily the filter conditions and measures included in the EIL.

Implementation Steps:

1	**Identification of required data points**	Identify the list of businessIds/codes for the reference[11] templates (Rendering Tables) to be reported. You can (e.g.,) use the BIRD API **renderingTables** endpoint (includeRelated=False) filtering for the FINREP_REF Framework (frameworkCodes=FINREP_REF).
2	**Generation of intermediary aggregated tables**	For each template to be reported, generate each required data point (or cell) from the related source attributes, properly filtered by the defined filters, while aggregating the defined metric and append the resulting data to the intermediary aggregated table. Users can include in this intermediary table the codes and ids of the generated data point, to be used for output validation purposes or additional optimisations. For the generation process, you can use (e.g.) the BIRD API **incomingRenderingLogicalLineages** endpoint. Please note however that, if implemented, the source information should be gathered from the intermediary tables created in phase 2, instead of the EIL.

Table 16: Implementation steps to create a data point-based intermediary table.

8.3.2.3. Creation of XBRL transmission files

The last step is to create the XBRL files including the generated data points and to deliver them to authorities.

Implementation:

1	**Generation of transmission files and deliverance**	Based on the intermediary aggregated table obtained in phase 3, create dissemination XBRL files and deliver them to the competent authority. The creation of the XBRL file should be done by the user. It should be noted that users can also use the BIRD API **renderingTables** endpoint (includeRelated=True) to obtain, for each single template, the logical structure (i.e. table cells, coordinates and related data points) of the XBRL files.

Table 17: Implementation of XBRL transmission files.

8.3.2.4. Alternative approach: generating dissemination files directly

As an alternative approach, users can generate dissemination files directly from the EIL, without implementing the intermediary tables. The implementation steps for this approach are the following:

Implementation Steps:

1	**Identification of reporting requirements**	Determine the specific reporting requirements that need to be fulfilled. Identify the list of businessIds/codes for the reference[12] annotated templates (renderingTables) to be reported. You can (e.g.) use the BIRD API **renderingTables** endpoint (includeRelated=False) filtering for the FINREP_REF Framework (frameworkCodes=FINREP_REF).
2	**Identification and combination of required source information**	For each template to be reported (destination), identify the required EIL attributes and allowed values to be used as source for the generation and the related source entities to be combined (joined). You can use (e.g.) the BIRD API **incomingRenderingLogicalLineages** endpoint, filtering for the business IDs/codes obtained in the previous step.
3	**Generation of output rendering tables**	For each data point (valueCombination) to be reported, generate the respective value based on the measure, filters and aggregation type defined by the generation rule.
4	**Creation of transmission files and deliverance**	Organise the data points generated in the previous step in a XBRL transmission file and deliver it to authorities.

Table 18: Implementation steps to generated dissemination files directly.

[11] The reference layer is the only one needed for the generation phase since it provides more easily the filter conditions and measures included in the EIL.
[12] (Generation) transformation rules refer to the reference output layer.

9. Future developments

This section outlines the forthcoming main features of BIRD, which are either part of a project plan or expected to be implemented in the coming years.

- **SMCube 2.0 Metadata Model Update.** As part of the BIRD Operational Tasks (OT) project, the SMCube metadata model will be updated to a new version. This update is scheduled for release before the running phase of the BIRD OT project. The changes to the SMCube are expected to have a medium to low impact on the BIRD technical exports and a minor impact on the BIRD simplified exports.
- **Metadata Historisation.** The BIRD team will introduce historisation for its metadata. This feature will enable banks to track business changes in the BIRD data models and transformation rules over time.
- **BIRD Releases.** BIRD will implement a release system to archive outdated BIRD releases instead of replacing them. A retrieval mechanism will be provided to fetch these archived releases.
- **BIRD Technical APIs.** The BIRD website will offer additional API endpoints for accessing BIRD metadata in its technical format (SMCube). For example, it will be possible to fetch a generic Cube object according to the SMCube format, including fetching previous releases.
- **Validation Rules via API and Exports.** BIRD will provide a more advanced format for both structural and business validation rules. These will be accessible via API and exports.
- **Other Improvements in the BIRD Navigator.** Several enhancements will be made to the BIRD Navigator, including:
    - Advanced logical lineage graphical representation.
    - Improved Logical Data Model visualisation.
    - Improved search bar functionality.
    - Enhanced layout for mobile devices.

Entity mapping objects can be obtained from the BIRD API endpoint `GET/**incomingLogicalLineages**` for a specific reference template entity. Similarly, this information is also available in the BIRD technical export (i.e., Cube Mappings and other related mapping objects). Please refer to the SMCube documentation for further details.

However, please note that for template-based reporting requirements, mappings are not relevant for the generation rules, as supervisory templates are delivered cell by cell rather than attribute by attribute. For such reporting frameworks, reference attributes and allowed values are sufficient to identify the filtering conditions and measures necessary to generate reporting data points.

10. Annex

10.1. How to combine (join) ERM entities in BIRD

The BIRD Transformation Rules do not explicitly specify how the source entities should be combined or joined. However, the BIRD IL and EIL ERMs illustrate all potential associations between entities, which can be used to establish join conditions.

Consider the following simplified and reduced example that demonstrates the associations between relevant entities to obtain Loans and advances for FINREP F 05.01, excluding non-negotiable bonds.

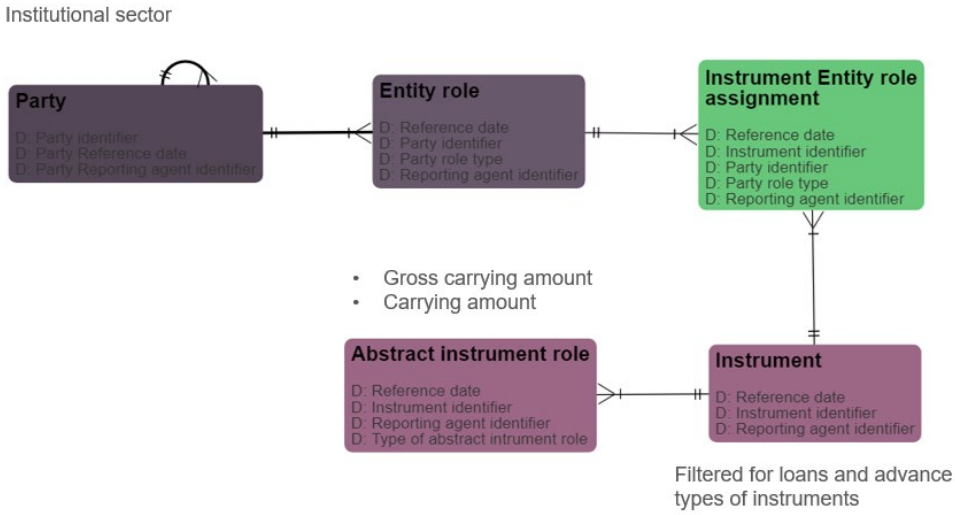


*Figure 13: Associative path from **Party** to **Abstract instrument role** to obtain F 05.01 Loans and advances relevant data records and attributes (this path does not consider non-negotiable bonds)*

This path suggests that the **Party** entity can be joined with **Entity role**, **Instrument Entity Role**, **Instrument** and **Abstract Instrument role** based on the associated attributes which are described in the relationship. Furthermore, it suggests which type of join can be applied between entities. For example, an inner join can be applied between **Party** and **Entity Role**, since both edges of the associative relationship are mandatory.

10.2. BIRD implementation example with a business case

In this annex we will consider, as a business case, example data from the AnaCredit manual to generate FINREP template F 05.01 output data following the BIRD implementation approach described in chapter 6, including four phases for the generation process.

Implementation of the Enriched Input Layer:

The following tables illustrate how EIL entities are implemented and populated with data from the AnaCredit manual based on the BIRD documentation. Attributes highlighted in yellow constitute the business key of the entity.

RPRTNG_AGNT_ID	DT_RFRNC	INSTRMNT_ID	TYP_INSTRMNT	RPYMNT_RGHTS	LTGTN_STTS	CRRNT_LTV_RT	ACCMLTD	CMMRCL_RL_ESTT_LN_INDCTR	BLNC_SHT_NTTNG_ID	CRDT_FCLTY_ID	CRRNCY	CSH
BLZ10	09/30/2018	123321	Other loans (1022)	Other than on demand or short notice (2)	Not in litigation pre litigation (3)	0	0	Not a commercial real estate loan (2)			Euro (EUR)	
BLZ10	09/30/2018	GUA28569811	Financial guarantee instrument not	Not applicable (0)	In litigation (2)	0	0	Not a commercial real estate loan (2)			Egyptian Pound (EGP)	
BLZ10	09/30/2018	otc_derivative1	Other OTC Derivative instrument (5	Other than on demand or short notice (2)	In litigation (2)	0	0	Not a commercial real estate loan (2)			Euro (EUR)	
BLZ10	09/30/2018	loan_to_central_bank	Other loans (1022)	Other than on demand or short notice (2)	Not in litigation pre litigation (3)	0	0	Not a commercial real estate loan (2)			Euro (EUR)	

Figure 14: Implementation and population of Instrument EIL table

PRTY_RPRTNG_AGNT_ID	PRTY_RFRNC_DT	PRTY_ID	TYP_PRTY	INSTTTNL_SCTR	ECNMC_ACTVTY	ENTRPRS_INDCTR	ENTRPRS_SZ	TRRTRL_UNT
BLZ10	09/30/2018	BLZ10	Non self employed natural person (18)	Other financial corporations excluding financial vehicle corporations (S125_W	Financial service activities except insurance and pe	Enterprise (1)	Large enterprise (1)	Denmark (DK)
BLZ10	09/30/2018	63829150	Non self employed natural person (18)	Other financial corporations excluding financial vehicle corporations (S125_W	Manufacture of rubber and plastic products (22)	Enterprise (1)	Large enterprise (1)	Denmark (DK)
BLZ10	09/30/2018	78451209	Non self employed natural person (18)	Households (S14)	Manufacture of rubber and plastic products (22)	Enterprise (1)	Large enterprise (1)	Denmark (DK)
BLZ10	09/30/2018	central_bar	Non self employed natural person (18)	Other financial corporations excluding financial vehicle corporations (S125_W	Activities of membership organisations (94)	Not applicable (0)	Large enterprise (1)	Not applicable (0)

Figure 15: Implementation and population of Party EIL table

RPRTNG_AGNT_ID	DT_RFRNC	INSTRMNT_ID	TYP_ABSTRCT_INSTRMNT_RL	ACCNTNG_CLSSFCTN	PRPS	CRRYNG_AMNT	CRRNT_LTV_RT	PRFRMNG_STTS	TM_PST_DU_BND	FRBRNC_MSR_CN
BLZ10	09/30/2018	123321	Non balance sheet recognised financial asset instrument (10)	IFRS Financial assets designated at fair value through profit or loss (4)	Construction investment (8)	83491250	0	Non performing (1)	1m 3m (9)	
BLZ10	09/30/2018	123321	Non balance sheet recognised financial asset instrument (10)	IFRS Financial assets designated at fair value through profit or loss (4)	Construction investment (8)	0	0	Non performing (1)	1m 3m (9)	
BLZ10	09/30/2018	123321	Over the counter OTC Derivative role (2)	IFRS Financial assets designated at fair value through profit or loss (4)	Construction investment (8)	83491250	0	Non performing (1)	1m 3m (9)	
BLZ10	09/30/2018	123321	Non balance sheet recognised financial asset instrument (10)	IFRS Financial assets designated at fair value through profit or loss (4)	Construction investment (8)	83491250	0	Non performing (1)	1m 3m (9)	

Figure 16: Implementation and population of Abstract Instrument Role EIL table

Derivation of Gross carrying amount:

The **Gross carrying amount attribute** is present in the **EIL Abstract Instrument Role** entity but not in the corresponding IL entity. This indicates that the attribute is derived, and that the relevant Derivation Transformation rule can be found in the BIRD metadata.

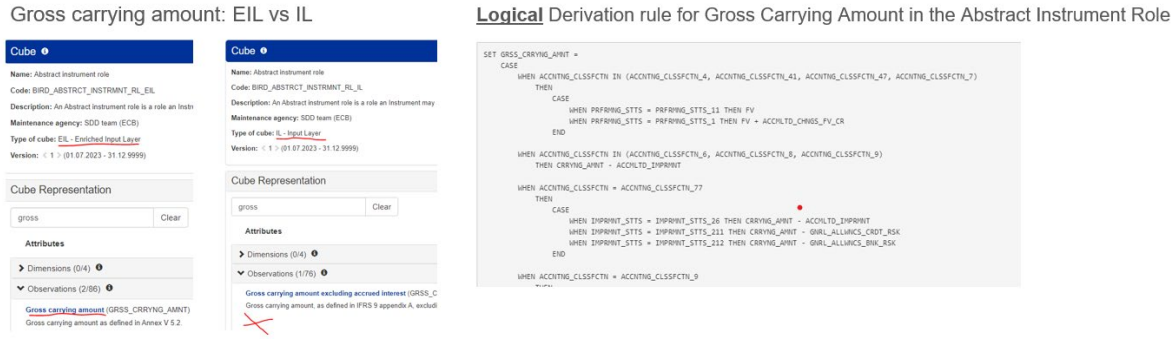


Figure 17: Derivation of the Gross carrying amount

Implementation of the intermediary granular table:

The following is a simplified representation for the intermediary **granular** table for the generation of F 05.01 FINREP template. Attributes (columns) highlighted in yellow, which have been merged from the source entities, constitute the business key for this table. To be noted that this table's granularity matches that of the combined source entities.

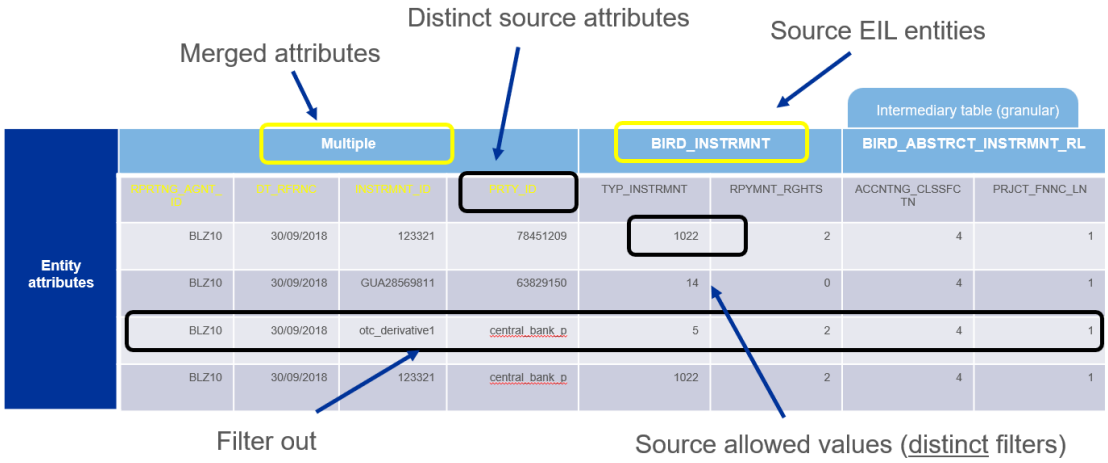


Figure 18: Implementation and population of F 05.01 intermediary granular table (simplified)

Implementation of the intermediary aggregated table and XBRL file creation:

The following is a simplified representation of the generated intermediary **aggregated** table for FINREP F 05.01 and a graphical (logical) representation of the generation of the template based XBRL file. It should be noted that this phase requires aggregating data from the previous phase to obtain aggregated data points. The field highlighted in yellow define the key (i.e. granularity) for this table. To be noted that the table can also be extended to include the rendering table coordinates where data points should be populated in the XBRL file.

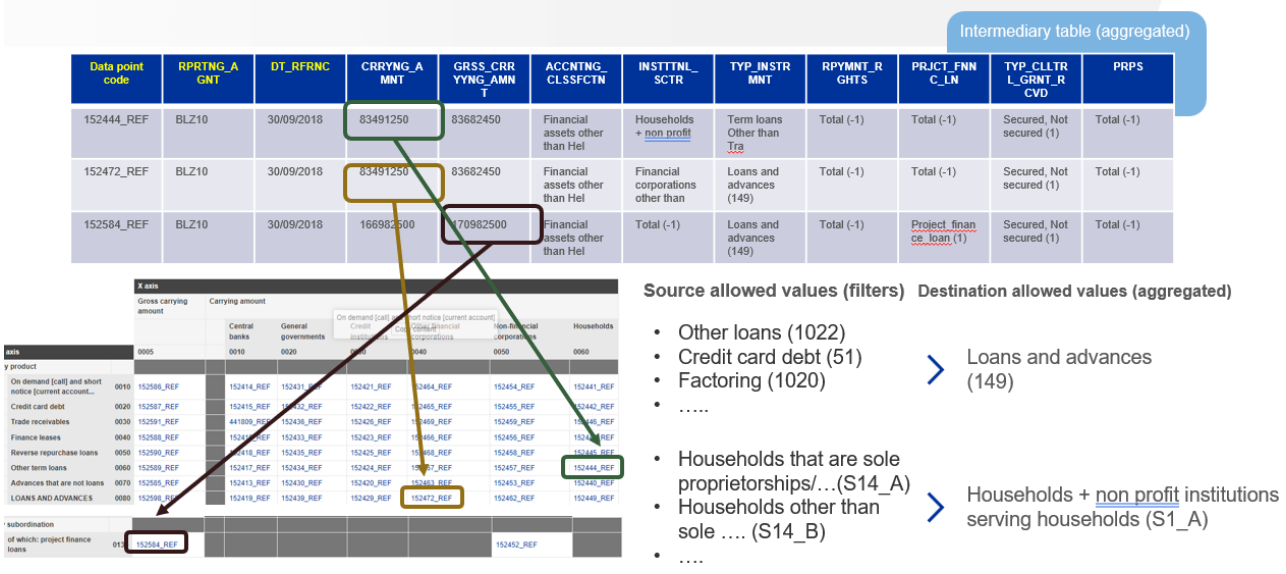


Figure 19: Implementation and population of intermediary aggregated table and template based XBRL for F 05.01. The markers show to which data points the Carrying Amounts refers to.