FUSION REGISTRY

Structure Mapping

FUSION REGISTRY
VERSION 9

Structure Mapping

This guide describes how Structure Maps can be used to transform the structure and coding schemes of a dataset to conform to a different Data Structure, and how this can be applied to a data transformation.

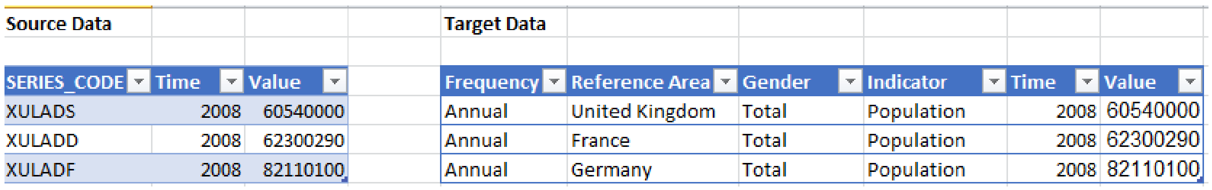
Fusion Registry: 9.4.x
Document Last Updated: April 3rd 2018

version 20180403

1 Contents

2 Overview

A Structure Map is used to define the relationship between two Data Structure Definitions (DSDs) in terms of their structure, and their coding schemes. The simplest example of a Structure Map is one which maps between a DSD with a single Dimension, to a multidimensional DSD, as shown in the following image.



Source Data

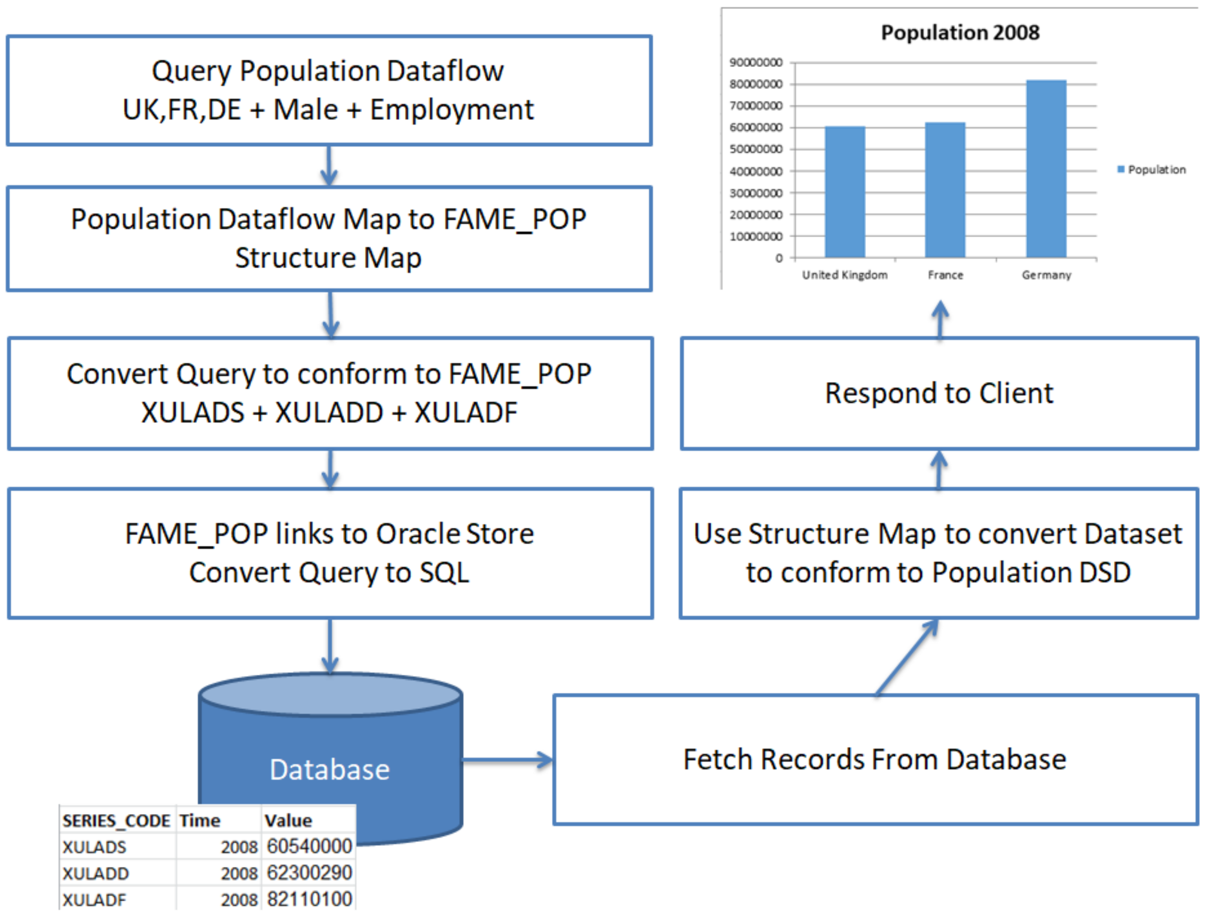
SERIES_CODE	Time	Value
XULADS	2008	60540000
XULADD	2008	62300290
XULADF	2008	82110100

Target Data

Frequency	Reference Area	Gender	Indicator	Time	Value
Annual	United Kingdom	Total	Population	2008	60540000
Annual	France	Total	Population	2008	62300290
Annual	Germany	Total	Population	2008	82110100

Figure 1 showing a relationship between source and target data

By defining a Structure Map to describe this relationship, the Fusion Registry will be able to transform between the source dataset to create the target dataset, and vice-versa. Structure Maps can also be used for data dissemination, enabling the source data to be stored conforming to one DSD, but then exposed via the SDMX Web Services conforming to another DSD, as the following image shows.


Query Population Dataflow
UK,FR,DE + Male + Employment
Population Dataflow Map to FAME_POP Structure Map
Convert Query to conform to FAME_POP
XULADS + XULADD + XULADF
FAME_POP links to Oracle Store
Convert Query to SQL
Database
Fetch Records From Database
Use Structure Map to convert Dataset to conform to Population DSD
Respond to Client
Population 2008
90000000
80000000
70000000
60000000
50000000
40000000
30000000
20000000
10000000
0
United Kingdom
France
Germany
Population

SERIES_CODE	Time	Value
XULADS	2008	60540000
XULADD	2008	62300290
XULADF	2008	82110100

Figure 2 showing a workflow from data query to data display using mapping to transform both the incoming query and outgoing dataset

Note: Mapping data stores is beyond the scope of this document. Detailed information on using a Structure Map to map data stores is provided in the Data Collection Management Document.

More complex mapping scenarios can be described for example: when both DSDs are multidimensional; where both series and observation attributes need to be mapped; even the observation value can be mapped.

The remainder of this document describes how to create a Structure Map in the Fusion Registry, and how, once a Structure Map is defined, the Fusion Registry can assist in verifying the map definition is correct.

3 Anatomy of a Structure Map

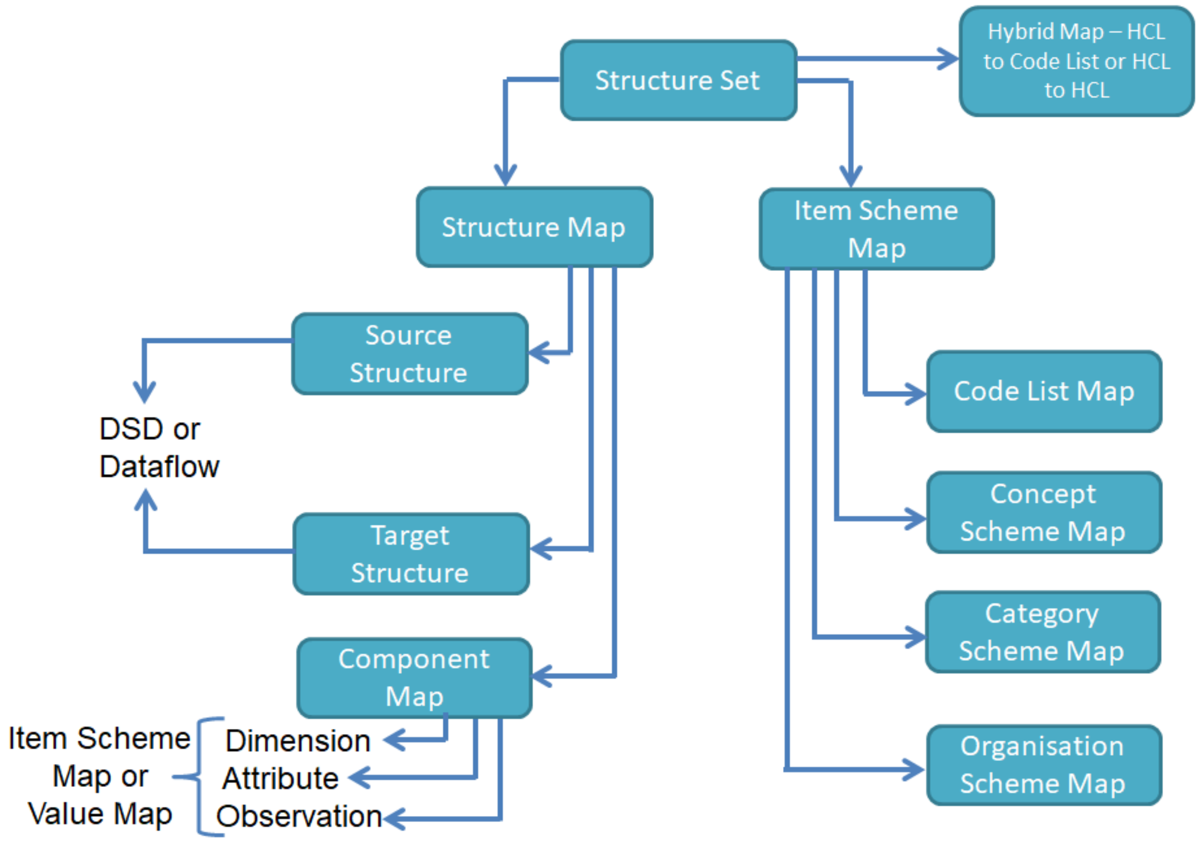


Figure 3 showing a Structure Map and related structures

A **Structure Map** 'resides' inside a bigger container called a **Structure Set**. A Structure Set can contain any number of Structure Maps, and may additionally include other types of maps including Codelist Maps, Concept Scheme Maps, etc. For the purpose of this user guide, only Structure Maps and Codelist Maps are used.

A Structure Map defines both a **Source** and **Target** structure. Source and target structures must refer to either a Data Structure Definition (DSD), or Dataflow. As a Dataflow references a DSD, mapping between two Dataflows is still ultimately mapping the underlying structure of the DSD, the only difference is the mapping is applied at the level of the Dataflow[1]. Mapping the Dataflow has the advantage of allowing the mapping to be used against a data store to support data collection and dissemination use cases.

Each Structure Map contains a **Component Map**. A Component Map describes how the Components of the source and target DSD are mapped[2]. Each Component Map defines the source component, e.g. SERIES_CODE, and target component, e.g. REF_AREA.

Each **Component Map** can also describe how the **values** of the source Component map to the values of the target Component. For example the series key *"XULADS"* maps to the REF_AREA code *"UK"*.

[1] Consider the use case where one DSD, for example World Development Indicators (WDI), is used by multiple Dataflows, for example: Poverty; Debt; and Education. A mapping at the Dataflow level allows individual Datasets to be mapped to another Dataflow. All WDI Dataflows could map to the same target Dataflow or different target Dataflows.

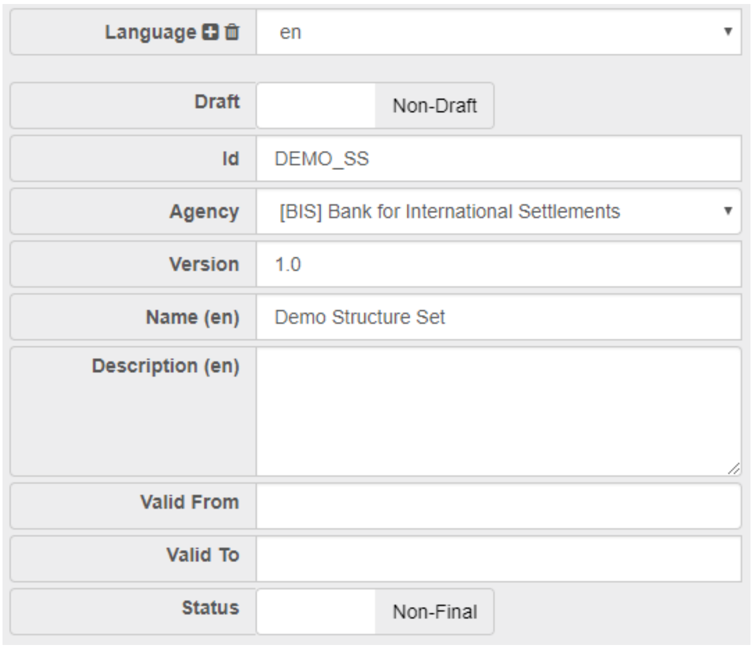
[2] A Component is the collective terms for Dimension, Attribute, and Observation

4 Creating a Structure Map

4.1 Defining a Structure Set

To create a Structure Map, a Structure Set container must first be created. To create a Structure Set, log into the Fusion Registry as either an Admin or Agency user, and navigate to **Structure Maps -> Structure Sets**. Click the cogs icon on the top left of the page, and select 'Create New Structure Set'.

This will launch a single page wizard requiring an Id, Agency Id, and Version of the Structure Set to be created.



Language	en
Draft	Non-Draft
Id	DEMO_SS
Agency	[BIS] Bank for International Settlements
Version	1.0
Name (en)	Demo Structure Set
Description (en)	
Valid From	
Valid To	
Status	Non-Final

Figure 4 showing the minimum details for a Structure Set to be created

4.2 Structure Map Wizard

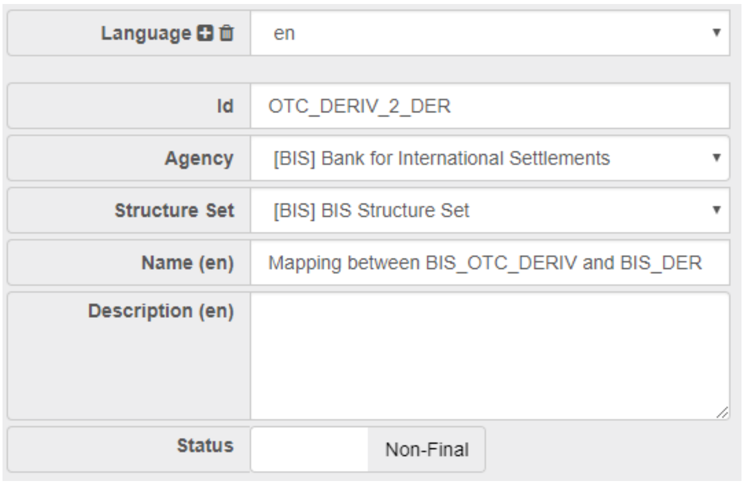
Once the Structure Set is created, a **Data Structure Map** or **Dataflow Map** can be created. The Data Structure Map maps between two DSDs and a Dataflow Map maps between two Dataflows. Both types of map use the same Wizard with the only exception being step 2 of the Wizard where the choice is either: a source and target DSD; or source and target Dataflow, depending on the chosen map type.

To create a Structure Map navigate to **Structure Maps -> Data Structure Map** or **Structure Maps -> Dataflow Map**. Click on the cogs icon on the top left of the page and select the 'Create new…' option to launch the Wizard.

4.2.1 Step 1 – Structure Map Details

Step one of the Wizard requires the Id of the Structure Map to be filled in, along with the Name. In addition the owning Agency must be selected. Once the owning Agency is selected, a list of Structure Sets that the Agency maintains will be displayed in the Structure Set drop-down. The Structure Map will be saved into the chosen Structure Set container.

Note: Later in this document Codelist Maps are introduced as a way to describe the relationship between two codelists. Codelist Maps can be used by Structure Maps, but they must be contained in the same Structure Set.



Language	en
Id	OTC_DERIV_2_DER
Agency	[BIS] Bank for International Settlements
Structure Set	[BIS] BIS Structure Set
Name (en)	Mapping between BIS_OTC_DERIV and BIS_DER
Description (en)	
Status	Non-Final

Figure 5 showing step one of the Structure Map Wizard

4.3 Step 2 – Source and Target Structure

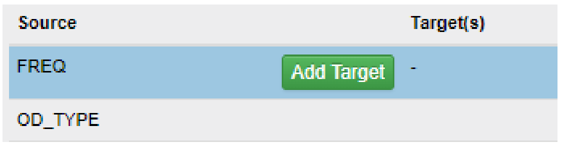
The second step of the Structure Map Wizard defines the source and target Dataflow/DSD. A well-defined Structure Map will allow the mapping to be bi-directional, so data can be converted from source to target, or target to source, therefore there should be no technical restrictions when choosing which structure is the source and which is the target.

4.3.1 Step 3 – Map Components

The third step of the Structure Map Wizard enables the Dimensions and Attributes of the source structure to be mapped to those of the target structure. It is possible to map Dimensions to Dimensions, or Dimensions to Attributes. It is possible to create one-to-one, on-to-many or many-to-many relationships between source and targets.

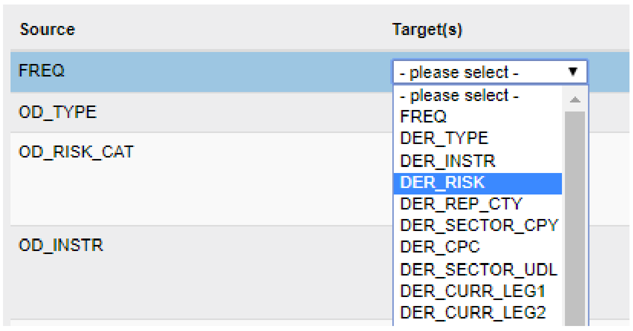
If a Component has no mapping to a target Component, it can be provided with a fixed value. For unmapped Dimensions providing a fixed value allows the mapping to be bi-directional.

The Mapped Components table lists all the source Components in the order defined by the source DSD, along with the mapped target Components (if any). To add a new mapped target, click the relevant row in the table, which will highlight the row, and an 'Add Target' button, which when clicked will create a drop-down list of available selections, as shown below:



Source		Target(s)
FREQ	Add Target	-
OD_TYPE		

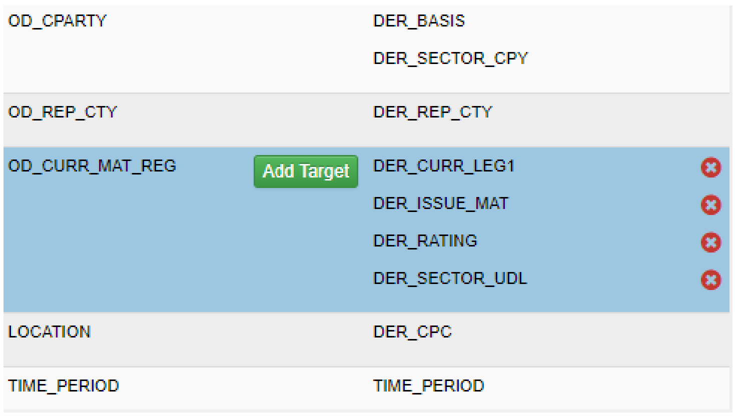
Figure 6 showing the Add Target button for the selected row



Source	Target(s)
FREQ	- please select -
OD_TYPE	
OD_RISK_CAT	
OD_INSTR	

Figure 7 showing the available targets for the selected source

To remove a target, select the row in the Mapped Components table, and all the current targets will be shown with a red 'x' icon. Clicking on this icon will remove the Component mapping.



OD_CPARTY		DER_BASIS DER_SECTOR_CPY
OD_REP_CTY		DER_REP_CTY
OD_CURR_MAT_REG	Add Target	DER_CURR_LEG1 DER_ISSUE_MAT DER_RATING DER_SECTOR_UDL
LOCATION		DER_CPC
TIME_PERIOD		TIME_PERIOD

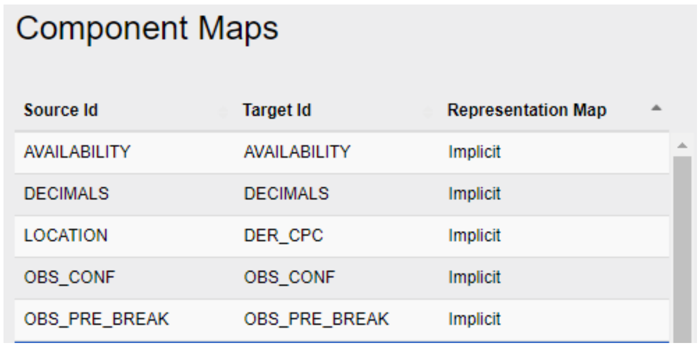
Figure 8 showing the remove target buttons for the selected row

Once all the source and target Components have been selected, the final step of the Wizard requires information on how the values for each Component map, this is known as a Mapping Representation.

4.3.2 Step 4 – Mapping Representation

Mapping representation falls into two main categories **Implicit** and **Explicit** mapping.

Implicit Mapping can be used when the target value is the same as the source value, for example if the underlying Codelist is the same. In this instance no mapping information is required, simply knowing that the source Component and target Component map is enough. The Fusion Registry will copy the data verbatim from source to target.



Component Maps

Source Id	Target Id	Representation Map
AVAILABILITY	AVAILABILITY	Implicit
DECIMALS	DECIMALS	Implicit
LOCATION	DER_CPC	Implicit
OBS_CONF	OBS_CONF	Implicit
OBS_PRE_BREAK	OBS_PRE_BREAK	Implicit

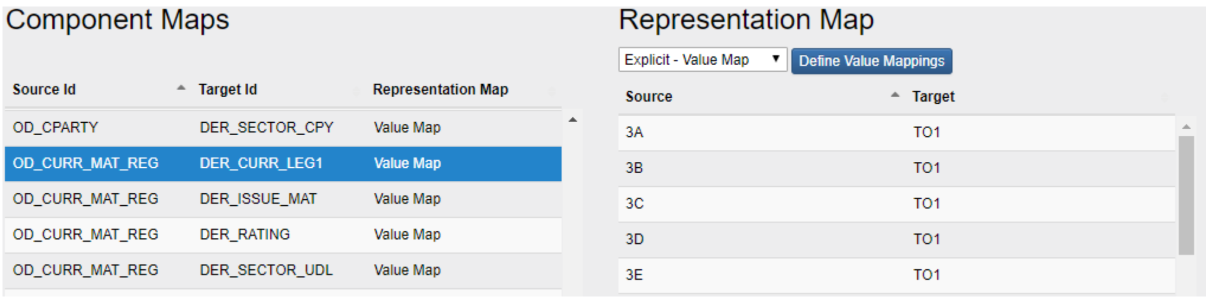
Figure 9 showing Component Map that use implicit mapping. In this example the source and target Id are also the same, although there is no requirement that they must be the same

Explicit Mapping provides information about how the source and target values map. There are two ways an explicit map can be defined: one is by referencing a **Codelist Map**[3]; the other is using a **Value Map**.

A **Codelist Map** has an id, Name, source and target Codelists, and defines how source codes map to target codes. The benefit of a Codelist Map is that it can be defined once, it can be maintained independently of the Structure Map, and it can be reused by multiple Structure Maps. A Codelist Map, however, can-not map between source and target Components if the source or target Component does not reference a Codelist. In this instance a Value Map should be used instead.

[3] Only Codelist Maps which exist in the same Structure Map can be referenced

A **Value Map** is simply a mapping of source value to target value, where the value is any valid representation of source and target. For instance if the source Component defines its allowable representation as Integer and the target Component is a Boolean, then the value map should map Integers to Booleans, e.g. *1 = true*.



Component Maps

Source Id	Target Id	Representation Map
OD_CPARTY	DER_SECTOR_CPY	Value Map
OD_CURR_MAT_REG	DER_CURR_LEG1	Value Map
OD_CURR_MAT_REG	DER_ISSUE_MAT	Value Map
OD_CURR_MAT_REG	DER_RATING	Value Map
OD_CURR_MAT_REG	DER_SECTOR_UDL	Value Map

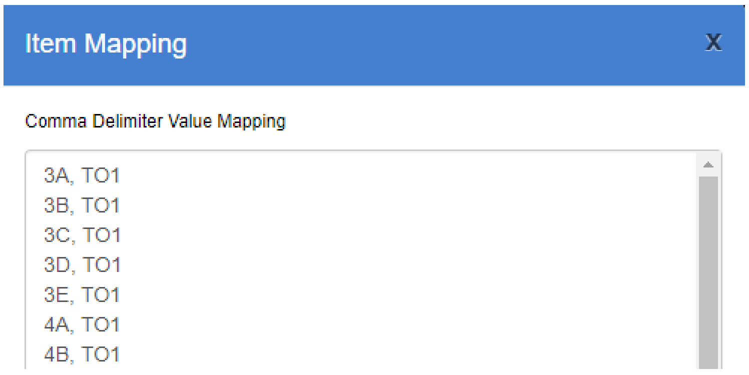
Representation Map

Explicit - Value Map

Source	Target
3A	TO1
3B	TO1
3C	TO1
3D	TO1
3E	TO1

Figure 10 showing a Value Map for the selected source/target Component. In this instance multiple source values map to the same target value, TO1.

To define, or modify a value mapping, click the '**Define Value Mappings**' button, and either type or paste a CSV definition into the modal text area. The CSV must conform to the format *source,target.* This is shown in the following image.



Item Mapping

Comma Delimiter Value Mapping

```
3A, TO1
3B, TO1
3C, TO1
3D, TO1
3E, TO1
4A, TO1
4B, TO1
```

Figure 11 showing the CSV list of source to target values

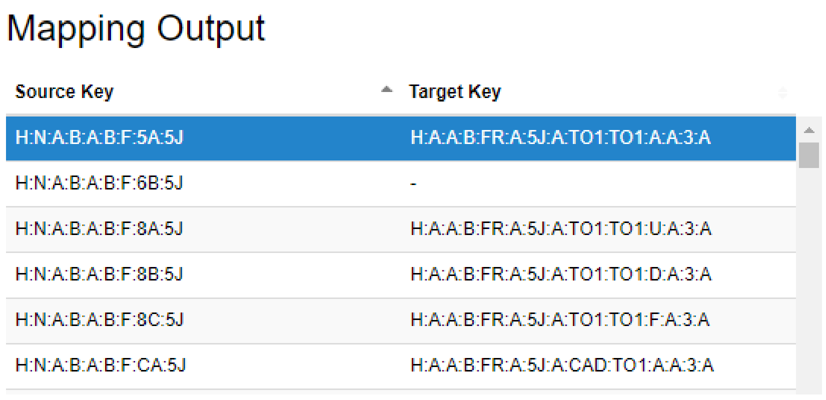
5 Test Mapping

Structure Maps can be complex to build, especially when mapping two structures with a large amount of Dimensions. The Fusion Registry provides the means to test a mapping, and drill into the details of how two series were mapped, or not mapped, in order to understand how to fix any errors.

To test a mapping, navigate to the **Structure Maps -> Data Structure Maps** or **Structure Maps -> Dataflow Maps** page. Select the map to test, and then click the '**Test Mapping**' button.

The next step is to choose a data file to load into the system. The data file must conform to either the source or target structure, and must contain information about which structure it is using. Because of this using a SDMX v2.1 data file is recommended, as this information is contained in the header of the dataset.

On loading a dataset, a page will be displayed showing how the source series-key maps to a target series-key. Any unmapped keys are denoted by a dash in the target, as shown in the image below.



Mapping Output

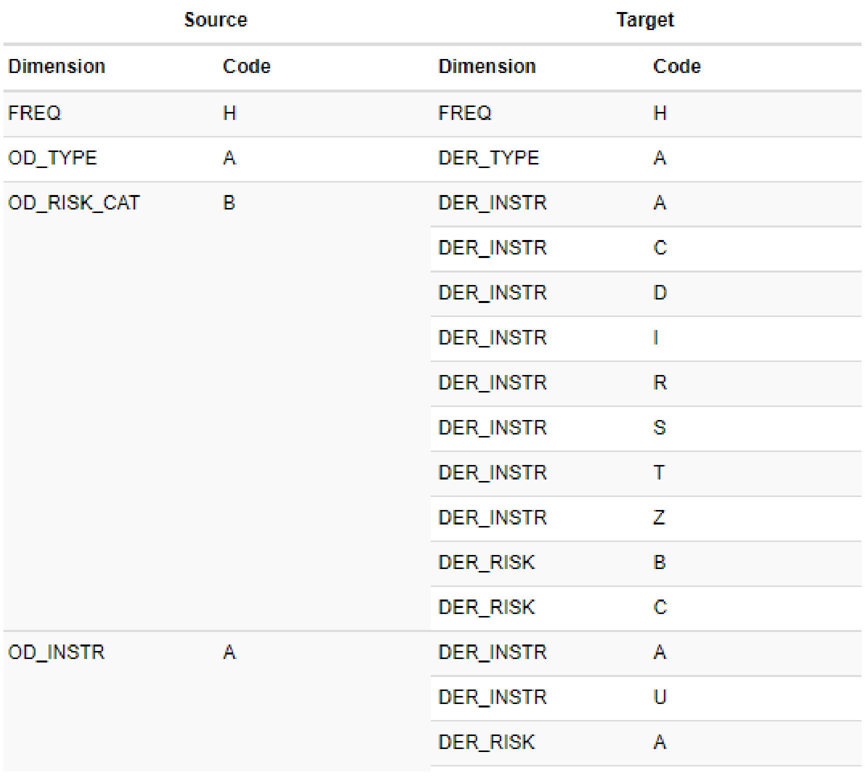
Source Key	Target Key
H:N:A:B:A:B:F:5A:5J	H:A:A:B:FR:A:5J:A:TO1:TO1:A:A:3:A
H:N:A:B:A:B:F:6B:5J	-
H:N:A:B:A:B:F:8A:5J	H:A:A:B:FR:A:5J:A:TO1:TO1:U:A:3:A
H:N:A:B:A:B:F:8B:5J	H:A:A:B:FR:A:5J:A:TO1:TO1:D:A:3:A
H:N:A:B:A:B:F:8C:5J	H:A:A:B:FR:A:5J:A:TO1:TO1:F:A:3:A
H:N:A:B:A:B:F:CA:5J	H:A:A:B:FR:A:5J:A:CAD:TO1:A:A:3:A

Figure 12 showing how series from the loaded dataset (source key) map to the target structure (target key)

Note: the limitation of the Test Mapping service is that it only shows information about Series Keys, it does not provide any analysis on how Attributes map.

To understand why a mapping resulted in no valid target key, or to further understand how a mapping was derived, the **Explain Plan** can be used. The explain plan is shown on the right of the page. The explain plan is generated for the selected source-target key in the Mapping Output table, so select the desired row in the Mapping Output table to update the explain plan.

Amongst other things, the explain plan shows how each part of the loaded series-key maps to target Component-Code combinations, as shown below:



Source		Target	
Dimension	Code	Dimension	Code
FREQ	H	FREQ	H
OD_TYPE	A	DER_TYPE	A
OD_RISK_CAT	B	DER_INSTR	A
		DER_INSTR	C
		DER_INSTR	D
		DER_INSTR	I
		DER_INSTR	R
		DER_INSTR	S
		DER_INSTR	T
		DER_INSTR	Z
		DER_RISK	B
		DER_RISK	C
OD_INSTR	A	DER_INSTR	A
		DER_INSTR	U
		DER_RISK	A

Figure 13 showing values for the source series key, along with the resulting output values for the target keys

In some instances a source value may result in multiple target values for a particular dimension. In the image above the OD_RISK_CAT value of 'B' maps to multiple target values for DER_INSTR. In order to create a valid output there must be only one possible target value, however it is important to note a target Dimension may be mapped by more than one source Dimension. In this image above the OD_INSTR Dimension also maps to the DER_INSTR. The final output for DER_INSTR can be derived by taking the intersection of all input values for the target Dimension. This is shown in the following image.

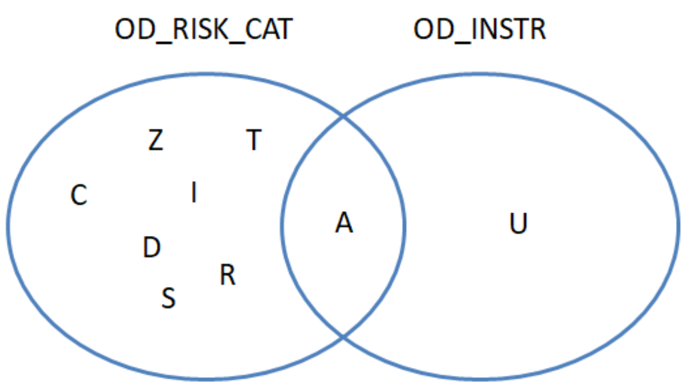
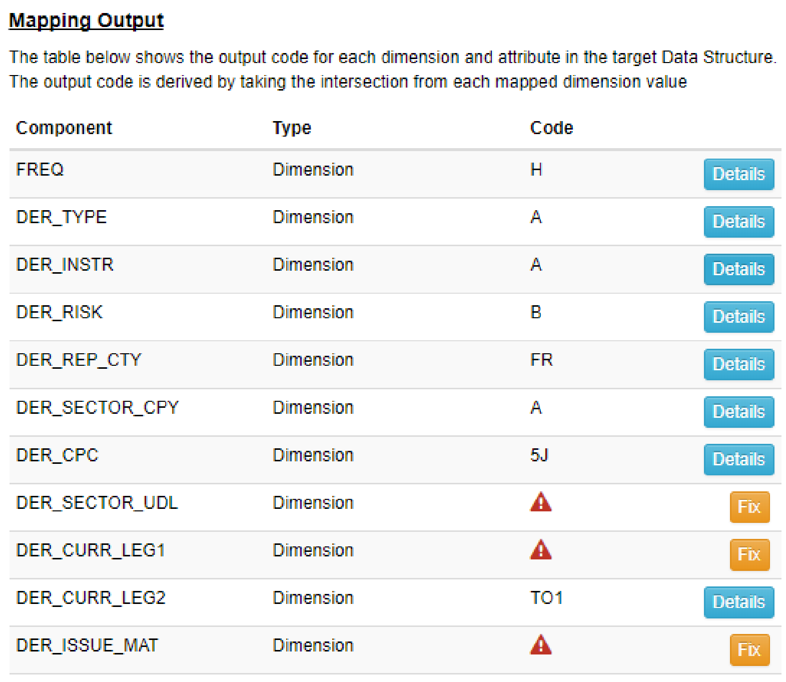


Figure 14 showing a Venn diagram representing all the possible sets relating to the target DER_INSTR Dimension. The intersection contains a single value 'A', which is the output of the mapping for the target Dimension.

The final mapping output can be seen, along with any errors which may have occurred to prevent a mapped series being possible.

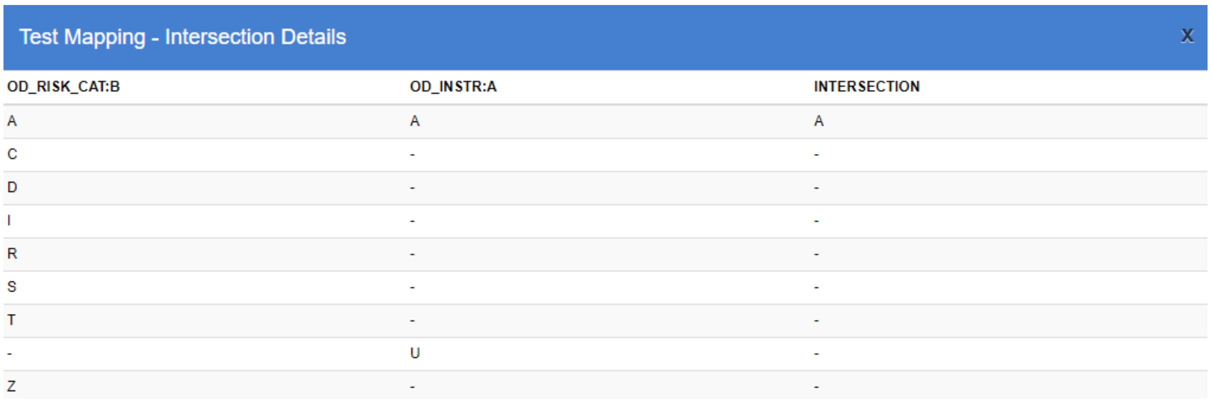

Mapping Output

The table below shows the output code for each dimension and attribute in the target Data Structure.
The output code is derived by taking the intersection from each mapped dimension value

Component	Type	Code	
FREQ	Dimension	H	Details
DER_TYPE	Dimension	A	Details
DER_INSTR	Dimension	A	Details
DER_RISK	Dimension	B	Details
DER_REP_CTY	Dimension	FR	Details
DER_SECTOR_CPY	Dimension	A	Details
DER_CPC	Dimension	5J	Details
DER_SECTOR_UDL	Dimension	⚠	Fix
DER_CURR_LEG1	Dimension	⚠	Fix
DER_CURR_LEG2	Dimension	TO1	Details
DER_ISSUE_MAT	Dimension	⚠	Fix

Figure 15 showing the output of a mapping, in this instance the source key could not be mapped due to the errors shown

Clicking Details on any of the Dimensions shows a tabular representation of a Venn diagram, which identifies the input sets, and the final intersection.


Test Mapping - Intersection Details

X

OD_RISK_CAT:B	OD_INSTR:A	INTERSECTION
A	A	A
C	-	-
D	-	-
I	-	-
R	-	-
S	-	-
T	-	-
-	U	-
Z	-	-

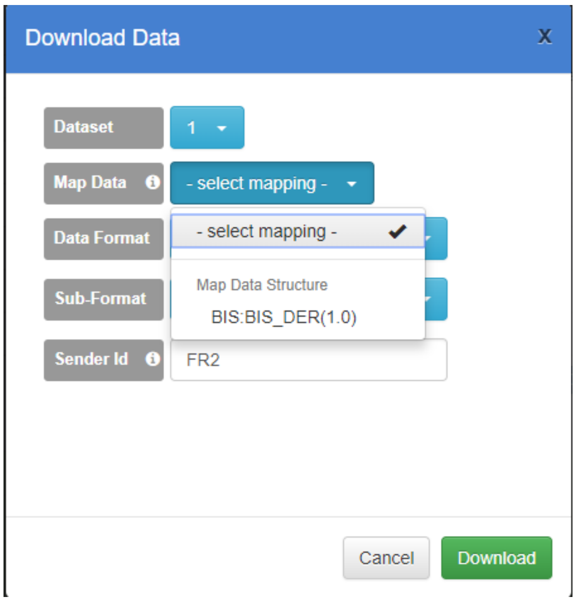
Figure 16 showing the intersection details for the DER_INSTR Dimension.

6 Mapping Data

Once a mapping has been defined it is possible to apply the mapping to transform a Dataset. This can be performed via the web services, and is documented in the Web Service documentation, under the Transformation section. Alternatively, the mapping can be performed via the web browser.

To perform a mapping via the Fusion Registry UI, navigate to the **Data-> Load Data** page, and select a dataset to load.

Once the dataset is loaded, click Convert Data. The resulting modal includes the option to map the data. This option will be populated with all available Structure Maps based on the loaded Datasets linked DSD and Dataflow. Select the Structure Map to use, select the desired output data format, and click Download. The downloaded dataset will conform to the mapped structure, in the requested data format.



Dataset: 1

Map Data: - select mapping -

- select mapping - ✓

Map Data Structure

BIS:BIS_DER(1.0)

Data Format

Sub-Format

Sender Id: FR2

Figure 17 showing the Download Data modal with the option to Map the data