Structural Metadata Management
This guide is for Agency or Admin users who are administering Structural Metadata in Fusion Registry 10
## Contents

1. **Overview** ................................................................................................................. 4

2. **Structure Maintenance** .......................................................................................... 5
   2.1 **Structure Creation** ............................................................................................... 5
   2.2 **Structure Modification** ......................................................................................... 6
   2.3 **Structure Deletion** ............................................................................................... 7
   2.4 **Modification Details** ............................................................................................ 7
       2.4.1 **Viewing Modification History** ........................................................................ 7
       2.4.2 **Roll Back Modification or Deletion** ................................................................. 9
       2.4.3 **Global Roll Back Modification or Deletion** ....................................................... 9
       2.4.4 **Download Logs for Modification** .................................................................. 10
       2.4.5 **Comparing Prior Revisions of a Structure** ..................................................... 10
   2.5 **Structure Submission from File or URL** ............................................................. 10

3. **Organisations** ........................................................................................................... 12
   3.1 **Overview** ........................................................................................................... 12
   3.2 **Organisation Wizard** .......................................................................................... 12

4. **Category Scheme** .................................................................................................... 13
   4.1 **Overview** ........................................................................................................... 13
   4.2 **Category Scheme Wizard** ................................................................................... 13
   4.3 **Categorising Structures** ...................................................................................... 15

5. **Concept Schemes** .................................................................................................... 17
   5.1 **Overview** ........................................................................................................... 17
   5.2 **Concept Scheme Wizard** ..................................................................................... 18

6. **Codelists** .................................................................................................................. 21
   6.1 **Overview** ........................................................................................................... 21
   6.2 **Codelist Wizard** ................................................................................................ 21
   6.3 **Codelist Inheritance** .......................................................................................... 23
       6.3.1 **Code Ordering** .............................................................................................. 23
       6.3.2 **Code Precedence** ........................................................................................ 23
       6.3.3 **General Behaviour** ...................................................................................... 23
       6.3.4 **Creating a Codelist with Inherited Codes** ................................................... 24

7. **Hierarchical Codelists** ............................................................................................. 25
   7.1 **Overview** ........................................................................................................... 25
   7.2 **Hierarchical Codelist Wizard** ............................................................................. 26

8. **Data Structure Definition (DSD)** ........................................................................... 28
8.1 Overview .......................................................................................................................... 28
8.2 Data Structure Wizard ..................................................................................................... 29
9 Dataflow ................................................................................................................................. 31
  9.1 Overview .......................................................................................................................... 31
  9.2 Dataflow Wizard ............................................................................................................. 31
10 Provision Agreements ........................................................................................................ 32
  10.1 Overview ....................................................................................................................... 32
  10.2 Provision Agreement Wizard ........................................................................................ 32
11 Reporting Constraints ......................................................................................................... 34
  11.1 Overview ....................................................................................................................... 34
  11.2 Reporting Constraint Wizard ....................................................................................... 35
12 Pre-Defined Queries ........................................................................................................... 37
13 Metadata Structure Definition ............................................................................................ 38
  13.1 Overview ....................................................................................................................... 38
  13.2 Metadata Structure Definition Wizard ......................................................................... 39
14 Data Structure and Dataflow Maps ..................................................................................... 41
  14.1 Overview ....................................................................................................................... 41
  14.2 Data Structure Map Wizard ......................................................................................... 41
15 Codelist and Concept Scheme Maps .................................................................................. 44
  15.1 Overview ....................................................................................................................... 44
  15.2 Codelist Map Wizard ................................................................................................... 44
16 Annotations .......................................................................................................................... 46
  16.1 Overview ....................................................................................................................... 46
  16.2 Annotation Wizard ....................................................................................................... 46
  16.3 Viewing Annotations .................................................................................................... 47
17 Structural Usage .................................................................................................................. 48
  17.1 Overview ....................................................................................................................... 48
  17.2 All Structures Cross-Reference Report ......................................................................... 49
  17.3 Structures with no Cross-References Report ............................................................... 49
### Version History

<table>
<thead>
<tr>
<th>Version #</th>
<th>Implemented By</th>
<th>Revision Date</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>20191128</td>
<td>Phil Lazarou</td>
<td>28 Nov 2019</td>
<td>Added information for Merge action when uploading structures</td>
</tr>
<tr>
<td>20200103</td>
<td>Glenn Tice</td>
<td>03 Jan 2020</td>
<td>Added Codelist Inheritance</td>
</tr>
</tbody>
</table>
1 Overview

The Fusion Registry is a web application for Data and Metadata management conforming to the Statistical Data and Metadata Exchange (SDMX) Specification - ISO 17369:2013.

This guide provides information on how to create, modify, and delete Fusion Registry structural metadata using the HTML User Interface (UI).

Structural Metadata includes the following structural artefacts, and have been grouped below in the same sequence as the HTML UI:

- **Organisations**
  - Agencies
  - Data Providers
  - Data Consumers

- **Items**
  - Category Schemes
  - Concept Schemes
  - Codelists
  - Hierarchical Codelists

- **Data**
  - Data Structure Definitions (DSDs)
  - Dataflows
  - Provision Agreements
  - Reporting Constraints
  - Pre-Defined Queries

- **Metadata**
  - Metadata Structure Definitions (MSDs)

- **Structure Maps**
  - Data Structure Maps
  - Dataflow Maps
  - Codelist Maps
  - Concept Scheme Maps
2 Structure Maintenance

To create structures in the Fusion Registry, first log into the HTML-UI as a user with either Admin or Agency permissions.

The Login button is located in the top right of the application, in the page header.

![Figure 1 showing the branding bar with login and language selector](image)

Once logged in all the pages in the Fusion Registry related to Structural Metadata will include a maintenance button as shown in the image below.

![Figure 2 showing the maintenance button for authenticated users](image)

The maintenance button will include, as a minimum, the ability to create, edit, or delete a structure. A structure must first be selected before Edit and Delete operations are enabled.

![Figure 3 showing the action menu for the maintenance button](image)

2.1 Structure Creation

The action of creating a structure will open up a wizard which provides the steps for defining the required information for the structure. Each wizard in the Fusion Registry is different depending on the structure type being created, however the first step of each wizard following a common theme, as described below.
Figure 4 showing step 1 of a structure creation Wizard

Each structure in the Fusion Registry must have an Id and a Name. All structures must also be owned by an Agency. It is the Agency who is responsible for maintaining and updating the structure. Only a user whose account is linked to the Agency is allowed to make modifications to the structure (admin users can modify structures for all Agencies). Most structures include a Version which will default the value to 1.0 if no version is defined.

Some structures may be created as a draft. Draft structures are only visible to admin users and users of the same Agency as the structure. Draft structure cannot be seen in the user interface, and web service responses by unauthenticated users. Draft structures may be published at a later date from the user interface by a user with Admin or Agency permissions (they must belong to the Agency who owns the structure).

Name and Description fields are multilingual. The first step of the wizard can be used to add new languages to the structure, which can be used in both the first step and subsequent steps when defining names and descriptions. Description fields are always optional.

Some structures include a validity period (Valid From/Valid To). This information is optional.

Some structures also include a Status, which can be set to either Final or Non-Final. Final structures indicate to the user that the structure will not undergo any major modifications. The Fusion Registry enforces this rule by restricting what can be modified. Certain information in a final structure can be modified, including Names; Descriptions; Valid From/To. It is possible to remove the Final status from a structure to allow for further modification, however it should be noted that if a structure is marked as Final, it is documenting to the user that they should not expect any further changes to this structure.

2.2 Structure Modification

Structures can be modified from the Fusion Registry using the edit menu option under the maintenance button. This action opens the same Wizard used for structure creation. When modifying a structure the wizard steps are pre-populated with the details from the structure that is being modified.
2.3 Structure Deletion

Structures can be deleted from the Fusion Registry using the delete menu option under the maintenance button. To delete a structure, navigate to the appropriate page in the Fusion Registry for the structure type, select the structure in the table, and click on the maintenance button to select the Delete option.

When a structure is deleted, it is important for the Fusion Registry to ensure that no structures cross-reference the deleted structure. The Fusion Registry will check to determine if the structure marked for deletion is referenced, and if it is, the user must choose to either delete all the cross-referencing structures, or cancel the operation. If the structures to be deleted include structures owned by other Agencies, then the user must have the authority to modify these structures (i.e. they must either be an Admin user, or a user with an account linked to multiple Agencies).

2.4 Modification Details

2.4.1 Viewing Modification History

The Fusion Registry records all modifications to structures and makes this information available to the Agency who owns the structure.

The structure modifications can be viewed in the details section which is displayed below the table of structures.

Figure 5 showing the Dataflow Details for the selected Dataflow. This details section appears below the table containing all of the Dataflows in the Fusion Registry.

An alternative approach to view structure modifications is to use the left hand menu to navigate to the Activity-> Structure Submissions page. This page shows the modifications made to each structure, and includes structures which have been subsequently deleted from the system. As the Structure Submissions page contains deleted structures, it is possible to perform an undelete operation from this page.
Figure 6 showing the structure modification history for all the structures in the Fusion Registry

The Fusion Registry provides the means to view the details about each modification. These details include information about the HTTP Request, the authenticated User, and the Fusion Registry application that was running at the time the modification was made.
2.4.2 Roll Back Modification or Deletion
When viewing the modification details for a structure, the user is offered the ability to rolling back to the revision as it was at the given point in time. Rolling back will cause the Fusion Registry to resubmit the structure as it was in the given revision. This will take the effect of creating a new revision in the audit trail for the newly submitted structure.

2.4.3 Global Roll Back Modification or Deletion
It is possible to roll-back the entire content of the Registry to a particular date. In order to do this, navigate to Activity -> SDMX Transactions.

On this page, a table will be displayed containing a list of all the transactions in the Registry and a time selection control.

Selecting one of the transactions will provide the choice to “Roll Back to selected Transaction”. Activating this control will roll back the entire content of the Registry to that selected transaction. This can be thought of as a ‘global roll back’ option.
2.4.4 Download Logs for Modification

Downloading the logs will export a CSV file containing the server logs associated with the modification. The granularity of the logged information will depend on the log setting at the time the modification was made. Log settings are set under the Settings -> Debug menu. See the Setup guide for more information on log levels.

2.4.5 Comparing Prior Revisions of a Structure

The prior revisions of a structure can be compared by activating the “Compare Revisions” control which will bring up a dialog showing the differences between the revisions. The controls at the top of this dialog allow quick comparison between different revisions.

![Figure 8 showing part of the comparison dialog](image)

2.5 Structure Submission from File or URL

If you have Structures defined in a file these can be uploaded into the Fusion Registry via a control from the Registry Home Page.

![Figure 9 showing the "Load Structures" control](image)

Activating this control will open a modal which allows an SDMX structure file to be loaded.

![Figure 10 showing the "Upload Registry Content" window](image)

Upload provides a choice of 4 different upload actions: Append, Replace (the default), “Full Replace” and Merge.

- Append - will only add new structures to those existing in the Registry. No modifications to existing structures will be permitted.
- Replace – will add new structures and can modify any existing structures.
- Full Replace - will remove all existing content from the Registry before replacing it with the contents of the uploaded file.
- Merge – will merge Concept Schemes and Codelists into existing Concept Schemes and Codelists. If the same structure exists in both the file being loaded and the Registry, then
the incoming structure will not remove items from the Registry. For example, the Codelist CL_REF_AREA exists in the Registry with codes BE and DE. Merging in another CL_REF_AREA Codelist which only has the code “CA”, will result in CL_REF_AREA containing all 3 codes. If this was attempted with any other upload option, then REF_AREA would result in only containing 1 code: “CA” as it would be replaced by the contents of the submitted Codelist.

It is also possible to load content from an SDMX compatible Web Service. By selecting “URL” and supplying a valid address of an SDMX web service or structure file. For example the following URL would import all of the codelists from an SDMX Web Service:

http://AnotherRegistryWebService/ws/public/sdmxapi/rest/codelist/all/all/all

Once the upload has succeeded a dialog will be displayed listing the structural changes to the system.
3 Organisations

3.1 Overview

Organisations are used to define ownership of structures or data, allow users to report data, and in the case of a private Fusion Registry Organisations provide users with the ability to access Registry content to view.

Organisations include:

**Agencies:** An Agency is responsible for owning structures in the Fusion Registry. Each structure in the Fusion Registry must be owned by an Agency.

**Data Providers:** A Data Provider is responsible for loading data or registering the URL of where data can be obtained.

**Data Consumers:** A Data Consumer can log into the Fusion Registry, however they have no special access beyond being able to use the user interface. When a Fusion Registry is running in public mode, a Data Consumer has no extra privileges. When a Fusion Registry is running in private mode, access is restricted to authenticated users only. In this condition, a Data Consumer will have access to browsing the content of the Fusion Registry via the Web Service only.

3.2 Organisation Wizard

The Organisation Wizard contains 2 steps. Step 1 is the general details step. Step 2 allows for the provision of information about Contacts.

![Contact form](image)

*Figure 11 showing the creation of a Contact within an Organisation*
4 Category Scheme

4.1 Overview

Category Schemes are the container for one or more Categories. Categories can be built into a hierarchy. A Category is used to provide the means to categorise any other structure type, this creates a link between the Category and the target structure.

A Categorisation is an independently maintained structure, which can be used to satisfy any use case where a structure requires a high level grouping. A typical use case for data dissemination is to link a Dataflow to one or more Categories using Categorisations. This use case allows a user to find Dataflows by Category.

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Description</th>
<th>Linked Structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Demographic and social statistics</td>
<td>Covers work in population and migration, labour, etc.</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Population and migration</td>
<td>Population and migration covers work in population and migration</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Labour</td>
<td>Covers statistics on labour force, labour market, etc.</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Education</td>
<td>Includes educational participation, literacy, educational, etc.</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Health</td>
<td>Health covers the health and mortality related statistics</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Income and consumption</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>Social protection</td>
<td>Social protection – deals with statistics on measure, etc.</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Human settlements and housing</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>Justice and crime</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Culture</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>Political and other community activities</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

*Figure 12 showing the Categories in a Category Scheme*

4.2 Category Scheme Wizard

The steps in a Category Scheme Wizard includes step 1 which provides the generic, high level details about the Category Scheme.

The second step allows the user to import Categories from CSV. CSV text can be copied and pasted into the text field provided, and on clicking ‘Next’, the CSV is checked for correctness. If valid, Categories are created and added to the Category Scheme, shown in step 3.
When Importing Categories via CSV it is important to note that the following:

1. Categories will be added to the Category Scheme, and therefore this step can be used to add additional Categories to a Category Scheme which already has Categories.
2. The import language for the name and description fields is defined in the drop down list above the text area.
3. If a Category with the given Id already exists, it will be modified based on the information supplied in the CSV. Modifications include the addition of names or descriptions in a new language into an existing Category.
4. Sub-Categories can be created by using the dot ‘.’ notation to include the ids of the parent Category(ies). This is shown in the image above where 1.1 is a sub-Category of Category 1.
5. The same number of delimiters are required for each line, even if there is no information for the field, shown above where Frequency includes an additional comma followed by no text (there is no Parent Concept Id)
6. If a field contains the delimiter (for example if a name includes a comma) then the text can be put in double quotes

The third step is to manually add, edit, delete, Categories. Additionally this step can be used to organise Category position in the Category hierarchy, this is achieved using the drag and drop support on the Category in the tree.
4.3 Categorising Structures

A Categorisation is used to create a link between a Category and any other structure in the Registry.

To create a Categorisation, first select a Category Scheme from the Category Schemes page, and then click View Category Scheme.

![Category Scheme View](image1)

**Figure 14 showing the view Category Scheme button**

From the Category Scheme view, select the Category to link the structure to, and click **Create Categorisation**, as highlighted in the image below.

![Create Categorisation](image2)

**Figure 15 showing the view of a Category Scheme with the ability to create a Categorisation**

The final step is to choose which structures to Categorise. First select which Agency will own the Categorisation, and then select the target structure type followed by the target structure(s). This is shown in the image below.
Figure 16 showing 4 Dataflows selected to be linked to a Category – this will result in 4 Categorisation structures being created which will all be owned by the Eurostat Agency.

The Category Scheme view includes the linked structures. The link can be deleted from this view for each individual structure, as shown below, or in bulk by clicking Delete Categorisations, which will open up a bulk delete window.

Figure 17 showing the Categorisations for a Category with Delete controls.
5 Concept Schemes

5.1 Overview

Concept Schemes are a container for Concepts. Concepts are used by Dimensions, Attributes, Measures, and Metadata Attributes to provide them with a semantic meaning. Concepts can also be used to define a default representation, for example it is possible to link the Concept of Frequency to a Codelist containing all the Frequency Codes, any structure using the Concept will inherit the default representation.

![Diagram of Concept Scheme](image)

Figure 18 showing Concepts in a Concept Scheme

The above shows the Concepts in a Concept Scheme, the image below shows how Concepts are used to give various Components of a Data Structure a name. Concepts can be used by any number of structures, and a structure (such as a DSD) can reference Concepts from any number of Concept Schemes.
Figure 19 showing the Concepts used by a Data Structure Definition providing the names and descriptions for each Dimension and Attribute of the Data Structure

5.2 Concept Scheme Wizard

The Concept Scheme Wizard includes the first generic step for information about the Concept Scheme.

The second step allows the user to import Concepts from CSV. CSV text can be copied and pasted into the text field provided. On clicking ‘Next’ Concepts will be created based on this CSV and added to the Concept Scheme.

When importing Concepts it is important to note that the following:

1. Concepts will be added to the Concept Scheme, and therefore this step can be used to add additional Concepts to an existing scheme.
2. The import language for the name and description fields is defined in the drop down list above the text area.
3. If a Concept already exists it will be modified based on the information supplied in the CSV. Modifications include addition of a name or description in a new language into an existing Concept.
4. Sub Concepts can be created by using parent Concept Id field. This is shown in the image above where SUB_CONCEPT is a sub-concept of REF_AREA.
5. The same number of delimiters are required for each line, even if there is no information for the field, shown above where Frequency includes an additional comma followed by no text (there is no Parent Concept Id).

6. If a field contains the delimiter (for example if a name includes a comma) then the text can be put in double quotes.

The third step is to manually add, edit, and delete Concepts.

![Concept Id, Name, Description, Representation table]

**Figure 21 showing wizard step 3, the manual editing of Concepts**

Each Concept can be assigned a default representation. This representation is inherited by Components of a Data Structure Definition (DSD) as a default. Default representation can be overridden by each Component of the DSD if required. To apply a Representation to a Concept, select the Concept in the table, and click the Change Representation button.

Representation can be non enumerated or enumerated. Non enumerated formats refer to data formats which are restricted by data type as oppose to enumerated representation which provide a list of allowed values. Non enumerated format includes types such as String, Integer, and Decimal, amongst others. Each non enumerated format can optionally provide additional details relevant to the selected format, such as min/max length for String. Enumerated formats allow a Concept to be linked to a Codelist. The Codelist is used to define the enumerated list of all the allowable values for the Concept.

**Data Type and Restrictions**

<table>
<thead>
<tr>
<th>Data Format</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal Places</td>
<td></td>
</tr>
<tr>
<td>Has Sequence</td>
<td>False</td>
</tr>
<tr>
<td>Min Value</td>
<td></td>
</tr>
<tr>
<td>Max Value</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 22 showing an example non enumerated representation form*
### Figure 23 showing the list of Codelists that could be assigned to a Concept

The final step of the Wizard allows the user to organise the Concepts into a Hierarchy using drag and drop.

<table>
<thead>
<tr>
<th>Agency</th>
<th>Id</th>
<th>Name</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESTAT</td>
<td>CL_ACCOUNT_ENTRY</td>
<td>Accounting entry code list</td>
<td>1.0</td>
</tr>
<tr>
<td>ESTAT</td>
<td>CL_ACTIVITY</td>
<td>Activity codes</td>
<td>1.0</td>
</tr>
<tr>
<td>ESTAT</td>
<td>CL_ADJUSTMENT</td>
<td>Adjustment indicator code list</td>
<td>1.0</td>
</tr>
<tr>
<td>ESTAT</td>
<td>CL_AREA</td>
<td>Area code list</td>
<td>1.0</td>
</tr>
<tr>
<td>ESTAT</td>
<td>CL_COICOP</td>
<td>COICOP</td>
<td>1.0</td>
</tr>
<tr>
<td>ESTAT</td>
<td>CL_CONF_STATUS</td>
<td>Confidentiality status code list</td>
<td>1.0</td>
</tr>
<tr>
<td>ESTAT</td>
<td>CL_DECIMALS</td>
<td>Decimals code list</td>
<td>1.0</td>
</tr>
<tr>
<td>ESTAT</td>
<td>CL_FREQ</td>
<td>Frequency code list</td>
<td>1.0</td>
</tr>
<tr>
<td>ESTAT</td>
<td>CL_INSTR_ASSET</td>
<td>Financial instruments and assets classification list</td>
<td>1.0</td>
</tr>
<tr>
<td>ESTAT</td>
<td>CL_NA_PRICES</td>
<td>Prices</td>
<td>1.0</td>
</tr>
<tr>
<td>ESTAT</td>
<td>CL_NA_STD</td>
<td>Stocks, transactions, other flows</td>
<td>1.0</td>
</tr>
<tr>
<td>ESTAT</td>
<td>CL_NA_TABLEIO</td>
<td>NA Table IOs</td>
<td>1.0</td>
</tr>
</tbody>
</table>
6 Codelists

6.1 Overview
Codelists provide an enumerated list of allowable values. Codelists can be used by Dimensions, Concepts, and Metadata Attributes to define the allowable content for both data and metadata reporting. Each Codelist contains zero to many Codes, each Code must belong to a Codelist.

![Codelist Diagram]

Each Codelist contains zero to many Codes, each Code must belong to a Codelist.

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABW</td>
<td>Aruba</td>
</tr>
<tr>
<td>ADO</td>
<td>Andorra</td>
</tr>
<tr>
<td>AFG</td>
<td>Afghanistan</td>
</tr>
<tr>
<td>AJO</td>
<td>Angola</td>
</tr>
<tr>
<td>ALB</td>
<td>Albania</td>
</tr>
<tr>
<td>ARE</td>
<td>Arab World</td>
</tr>
<tr>
<td>ARE</td>
<td>United Arab Emirates</td>
</tr>
</tbody>
</table>

*Figure 24 showing part of a Reference Area Codelist*

6.2 Codelist Wizard
The Codelist Wizard includes the first generic step for information about the Codelist.

The second step allows the user to import Codes from CSV. CSV text can be copied and pasted into the text field. On clicking ‘Next’, the CSV is checked for correctness. If valid, Codes are created and added to the Codelist.

*Figure 25 showing an example of codes being imported using the CSV import*
When importing Codes it is important to note that the following:

1. Codes will be added to the Codelist, and therefore this step can be used to add additional Codes to an existing scheme.
2. The import language for the name and description fields is defined in the drop down list above the text area.
3. If a Code already exists it will be modified based on the information supplied on this page, this includes the addition of new or description in a new language into an existing Codelist.
4. Sub Codes can be created by using parent Code Id field. This is shown in the image above where UK, FR and DE are sub-codes of EUR.
5. The same number of delimiters are required for each line, even if there is no information for the field, shown above where ‘Europe’ includes an additional comma followed by no text (since it has no Parent Code Id).
6. If a field contains the delimiter (for example if a name includes a comma) then the text can be put in double quotes.

The third step is to manually add, edit, and delete Codes in a very similar manner to Category Schemes and Concept Schemes.

The fourth step permits the specification of a natural order to the codes and the ability to build a hierarchy.

There is an input field on the left-hand side of the page. This is pre-populated with the list of items in the Codelist. If the item has a parent then the item will be followed with a comma and the parent ID. In the example above we can see that DE, UK and FR are all children of EUR. This is shown in the hierarchy view on the right-hand side.

To specify the order or give items parent codes, edit the input field and then click the ‘Update View’ button so that the Hierarchy View updates. There are controls on this page to sort the items by name or by ID.
When creating a hierarchy with many levels, each code only requires its immediate parent. To create a hierarchy with code A having a child of code B, with a child C, with a child D, etc. you would enter the following in the input field:

![Diagram of hierarchy]

**Figure 27 showing an example hierarchy definition**

### 6.3 Codelist Inheritance

In addition to a literal list of codes, Codelists can also inherit codes from one or more other Codelists. This is helpful where Codelists are best created and maintained by ‘extending’ other Codelists.

**Example:**

Consider the need for a Frequency Codelist where a special ‘Bi-Weekly’ frequency is required. Codelist Inheritance could be used to create a new Codelist by extending the standard SDMX:CL_FREQ Codelist with a single additional ‘[I] Bi-Weekly’ value.

The new Codelist is considered to have a single ‘parent’ SDMX:CL_FREQ extended with the addition of the [I] code.

<table>
<thead>
<tr>
<th>Inheritance</th>
<th>Position</th>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from SDMX:CL_FREQ</td>
<td>1</td>
<td>A</td>
<td>Annual</td>
</tr>
<tr>
<td>Inherited from SDMX:CL_FREQ</td>
<td>2</td>
<td>S</td>
<td>Half-yearly</td>
</tr>
<tr>
<td>Inherited from SDMX:CL_FREQ</td>
<td>3</td>
<td>Q</td>
<td>Quarterly</td>
</tr>
<tr>
<td>Inherited from SDMX:CL_FREQ</td>
<td>4</td>
<td>M</td>
<td>Monthly</td>
</tr>
<tr>
<td>Inherited from SDMX:CL_FREQ</td>
<td>5</td>
<td>W</td>
<td>Weekly</td>
</tr>
<tr>
<td>Inherited from SDMX:CL_FREQ</td>
<td>6</td>
<td>D</td>
<td>Daily</td>
</tr>
<tr>
<td>Inherited from SDMX:CL_FREQ</td>
<td>7</td>
<td>H</td>
<td>Hourly</td>
</tr>
<tr>
<td>Inherited from SDMX:CL_FREQ</td>
<td>8</td>
<td>B</td>
<td>Daily – business week</td>
</tr>
<tr>
<td>Inherited from SDMX:CL_FREQ</td>
<td>9</td>
<td>N</td>
<td>Minutely</td>
</tr>
<tr>
<td>Extended Value</td>
<td>10</td>
<td>I</td>
<td>Bi-Weekly</td>
</tr>
</tbody>
</table>

#### 6.3.1 Code Ordering

Codes in an extended Codelist are ordered in the sequence defined for the parents, followed by any Extended Values.

#### 6.3.2 Code Precedence

Where the same code occurs, Extended Values take top precedence, followed by the parent Codelists in the sequence defined.

#### 6.3.3 General Behaviour

- Codes added to parent Codelists are automatically inherited.
• Codes removed from parent Codelists are removed from Codelists that inherit them, only if they are unused. If a parent removes a code in use in an extended Codelist, the code will be converted to a literal.

• Codelists with inherited Codes act just as normal ‘literal’ Codelists when used. When exported, the list of codes is materialised making it possible to share such Codelists with other SDMX aware systems as static ‘literal’ Codelists without requiring any special precautions. Inheritance information is preserved as annotations allowing systems that understand inheritance to manage the structure appropriately.

```xml
<st:st:st:Codelist urn="urn:sdmx:org.sdmx.infomodel.codelist.Codelist=METATECH:
   urn:sdmx.org.sdmx.infomodel.codelist.Codelist=SDMX:CL_FREQ(2.0)"
</com:Annotation>
</com:Annotations>
```

Figure 28 Example annotation for a Codelist with inheritance in SDMX-ML 2.1

6.3.4 Creating a Codelist with Inherited Codes

Codelists with inherited Codes are created using the Codelist Wizard as described in section 6.2.

To extend existing Codelists, use the ‘Add Inheritance’ button to choose one or more parent Codelists. The sequence of the parent Codelists is shown in the Codelist Inheritance table and can be modified using the ‘Move Up’ and ‘Move Down’ buttons.

Figure 29 showing the Codelist Wizard with Codelist Inheritance

Codes values which extend those inherited from the parent(s) are added and maintained using the normal procedure.
7 Hierarchical Codelists

7.1 Overview

A Hierarchical Codelist provides a hierarchical view over one or more ‘standard’ Codelists. A Hierarchical Codelist can be thought of in the same way as a database view: it does not define any new Codes, it is used to group any number of existing Codes from any number of existing Codelists.

A Hierarchical Codelist consists of one or more Hierarchies, each Hierarchy organises codes from one or more Codelists into a hierarchical structure. The same Code may appear multiple times in a hierarchy.

A Hierarchical Codelist does not have a defined purpose. However they are typically used for data dissemination to provide another view on a standard Codelist.
The Hierarchical Codelist Wizard includes the first generic step for information about the Hierarchical Codelist.

The second step requires information on the Codelists that will be required when building the Hierarchies.

### Referenced Codelists

<table>
<thead>
<tr>
<th>Agency Id</th>
<th>Id</th>
<th>Name</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>WB</td>
<td>CL_SERIES_WDI</td>
<td>Series code list</td>
<td>1.0</td>
</tr>
<tr>
<td>WB</td>
<td>TOPICS</td>
<td>Series Topics</td>
<td>1.0</td>
</tr>
</tbody>
</table>

In addition this step includes the definition of each Hierarchy contained in the Hierarchical Codelist.
Figure 32 showing step 2 of the Hierarchical Codelist Wizard and the hierarchies contained in the Hierarchical Codelist

Step 3 of the wizard provides Codes management in each Hierarchy. Each Code added to the Hierarchy is clickable and can be dragged and dropped into a new position as required.

Step 4 of the wizard allows the user to provide a formal definition for each level of the hierarchy. For example if the root codes in a Hierarchy are all related to Industry Sector with child codes are all related to Industry Type, then the Hierarchy could be given formal levels to define this.
8 Data Structure Definition (DSD)

8.1 Overview

A Data Structure Definition (DSD) defines a dataset in terms of its Dimensionality and allowable content. All reported datasets must conform to the definition defined by the DSD.

A DSD Consists of Dimensions, Attributes, and Measures, collectively these are termed Components. Each Component of a DSD references a Concept to provide a semantic meaning, and optionally a Codelist to provide an enumerated list of allowable content for reported data.

A DSD can include a special type of Dimension known as a Measure Dimension. The Measure Dimension supports the use case of multiple measures. Unlike a 'normal’ Dimension, the Measure Dimension cannot reference a Codelist, however it must reference a Concept Scheme which is used to list the allowable measures. Each Concept in the Concept Scheme may provide its own representation which can be enumerated (Codelist) or non-enumerated. When data is reported for the Measure Dimension, the allowable values will depend on the Concept that is being reported in the linked Concept Scheme.

When Viewing a DSD the Components of the DSD are displayed by showing the Name of the Concept as the Component label, and the allowable content, which may be the referenced Codelist, as shown below.

Figure 33 showing a DSD for World Development Indicators
8.2 Data Structure Wizard

The Data Structure Wizard includes the first generic step for information about the DSD. In addition, the first step asks two further questions: is the DSD describing Time Series data, if so a Time Dimension is automatically created and added and if there is a TIME_PERIOD Concept in the Fusion Registry this will be used to provide the semantic for this Dimension; and secondly what is the Concept used for the Primary Measure, if there is an OBS_STATUS Concept in the Fusion Registry this will be used by default. To change the default Concepts chosen by the wizard for either the Time Dimension or the Primary Measure, click on the text field to open up a list of all available Concepts.

The second step of the Wizard allows the user to define all the other Concepts which will be used by the DSD. Each Concept can be assigned a role of Dimension, Measure Dimension, or Attribute.

![Figure 34 showing the second step of the DSD Wizard](image)

The third step of the wizard allows the user to define the allowable content for each Component. If the Concept has a default Representation this will be selected by default, however this Representation can be changed if required.

![Figure 35 showing the third step of the DSD Wizard](image)

The allowable content can be enumerated (which is a reference to a Codelist) or non-enumerated (for example Text, Integer, Double, Boolean etc). The allowable content defines what data a user can report when they are supplying a dataset.

The final step of the Wizard is to define the assignment status (required or optional) and attachment level (dataset, series, dimension group, observation, group) of Attributes as shown in the screenshot below.
The purpose of each attachment levels is described below.

**Dataset Attachment**
An Attribute attaching to a dataset will mean that when data is reported for the dataset, there will be a single value which is provided for the dataset. For example, the Unit of Measure Attribute could be attached to the dataset if it is expected that all of the observations will always be measured using the same unit.

**Series Attachment**
If the Attribute attaches to a Series, then the Attribute will attach itself to every Dimension in the Data Structure Definition, except the Time Dimension. This attachment is used to define that the value for the Attribute can vary for each Series in the dataset. For example, the Series Title Attribute could attach to a Series, as each series will have a different title.

It is possible to modify this attachment to specify a subset of the Dimensions, if the value for the attribute only relates to a subset.

**Observation Attachment**
If a value for an Attribute can change from Observation to Observation, then the attachment level should be set to Observation. An example of an Observation attachment is Observation Confidentiality.

**Group Attachment**
The final attachment is to a pre-defined Group of Dimensions. This is similar to Series Attachment, in that the Attribute will attach to a subset of Dimensions, however the subset is defined by the referenced Group.

This option is only available in the list if a Group has been defined. To define, modify, or delete a Group, click on the ‘Manage Groups’ button, which will open a window, as shown below.
9 Dataflow

9.1 Overview

A Dataflow is a structure on which data is collected and disseminated. A Dataflow references a Data Structure Definition (DSD) which is used as the underlying template to which the data must conform.

When viewing a Dataflow, it is the underlying DSD that is shown, however it is shown in the context of the Dataflow and therefore any additional restrictions which have been applied to the Dataflow (see section 11 Reporting Constraints) will be reflected in this view.

9.2 Dataflow Wizard

The Dataflow Wizard includes the first generic step for information about the Dataflow.

The second step requires the selection of the Data Structure Definition (DSD) the Dataflow will use. To select or modify the DSD selection click in the input field, this will open a list of available DSDs to select from. The details of the selected DSD are shown on this step.

The third step is optional and are only required if setting up a data collection environment. The information in these steps can be managed from the Provision Agreements page however for convenience the management is also included in the Dataflow Wizard.

The third step allows the user to define which Data Providers have permission to submit data for this Dataflow. The output of this step is to create a Provision Agreement for each Data Provider linked to this Dataflow.

![Dataflow Wizard](image)

*Figure 37 showing step 3 of the Dataflow Wizard – choose Data Providers for Dataflow*

For more information about data stores, please refer to the Data Collection Management guide.
10 Provision Agreements

10.1 Overview

A Provision Agreement is the union of a Dataflow with a Data Provider. A Provision Agreement is a definition that the Data Provider is allowed to provide data for the Dataflow. Data is always reported by a Data Provider against the Provision Agreement.

When viewing a Provision Agreement, it is the underlying DSD that is shown, however it is shown in the context of the Provision Agreement and therefore any additional restrictions which have been applied to the Provision Agreement (see section 11 Reporting Constraints) will be reflected in this view.

10.2 Provision Agreement Wizard

The Provision Agreement wizard includes the first generic step for information about the Provision Agreement.

The second step requires the selection of both the Dataflow and Data Provider the Provision Agreement is combining. The Dataflow provides the definition of the allowed data. The Data Provider defines which Organisation can report data for the Dataflow.

It is possible to define the data reporting mechanism for that is allowed for this Provision Agreement on the second step of the wizard. The default reporting mechanism is for the Data Provider to supply a URL to an external data file, or SDMX web service.

**Note:** When viewing a Provision Agreement it is possible to modify the linked data store as shown in the image below.
Linked Data store for Provision Agreement
In this example data is reported as a link to an external URL which resolves to an SDMX REST Web Service

Figure 38 showing the Provision Agreements view, where the details section allows modification to the linked data store
11 Reporting Constraints

11.1 Overview

Reporting Constraints are used to further restrict the allowable content of a Codelist in the context of a DSD, Dataflow, Provision Agreement, or Data Provider. Collectively these are termed as Constrainable Structures. The restrictions imposed by Reporting Constraints are taken into account when validating a dataset reported by a Data Provider. A Reporting Constraint defines restrictions against Dimensions and Attributes of a DSD which take allowable content from an enumerated list (e.g. Codelist).

The Codelists which can be constrained are determined from the Constrainable structures that the Reporting Constraint is referencing.

A Reporting Constraint can impose a rule on any of the available Codelists to remove any number of allowable Codes, therefore reducing the universe of the enumerated list. A Codelist is constrained by either defining which Codes in the Codelist are included (remain valid) or which codes are excluded (are not valid). An included or excluded Code may also be marked as cascade, which means any Codes which are children of that Code also inherit the rule of inclusion or exclusion.

When viewing a Reporting Constraint, the Included Cube shows which Codes remain valid for any of the Constrained Components. The excluded Cube shows which Codes are no longer valid.

![Figure 39 showing the view of the included Cube for a Reporting Constraint](image)

A Constrainable Structure, such as a Provision Agreement will inherit any Reporting Constraints which are attached to lower level structures that it references. For example the Provision Agreement will inherit any Reporting Constraints against the Dataflow it uses, and the DSD that the Dataflow uses. It will also inherit any Reporting Constraints against the Data Provider it references.
The rules defined by Reporting Constraints will be enforced when performing data validation. The rules are also reflected when viewing a DSD, Dataflow, or Provision Agreement, as shown below.

11.2 Reporting Constraint Wizard

The Reporting Constraint Wizard includes the first generic step for information.

The second step is used to define what Structures are being constrained and of the constrained structures, which Components will be included in the Constraints. Only Components which reference a Codelist are included in the available list. If the Constrained structure is a Data Provider, the list of Components will be derived from all the Dataflows the Data Provider has a Provision Agreement for.

The third and 4 steps are to define the included and excluded values respectively. A Component can only be defined to contain included codes or excluded Codes not both.
### Included Values

<table>
<thead>
<tr>
<th>Component ID</th>
<th>Value</th>
<th>Include Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREQ</td>
<td>A</td>
<td>false</td>
</tr>
<tr>
<td>REF_AREA</td>
<td>ASM</td>
<td>false</td>
</tr>
<tr>
<td>REF_AREA</td>
<td>AUS</td>
<td>false</td>
</tr>
<tr>
<td>SERIES</td>
<td>BO_GER_NF8Y_GQ_Z8</td>
<td>false</td>
</tr>
<tr>
<td>SERIES</td>
<td>BM_GER_CMCP_Z8</td>
<td>false</td>
</tr>
<tr>
<td>SERIES</td>
<td>BM_GER_FCTY_CD</td>
<td>false</td>
</tr>
<tr>
<td>SERIES</td>
<td>BM_GER_GNFS_CO</td>
<td>false</td>
</tr>
<tr>
<td>SERIES</td>
<td>BM_GER_NOF_Z8</td>
<td>false</td>
</tr>
<tr>
<td>SERIES</td>
<td>BM_GER_URCN_CD</td>
<td>false</td>
</tr>
<tr>
<td>SERIES</td>
<td>BM_QAR_NF8Y_CD</td>
<td>false</td>
</tr>
<tr>
<td>SERIES</td>
<td>BM_QAR_ROYL_CO</td>
<td>false</td>
</tr>
<tr>
<td>SERIES</td>
<td>BM_QAR_TOTI_CO</td>
<td>false</td>
</tr>
<tr>
<td>SERIES</td>
<td>BM_QAR_TRAN_Z9</td>
<td>false</td>
</tr>
</tbody>
</table>

*Figure 42 showing step 3 of the Reporting Constraint Wizard*
12 Pre-Defined Queries

A Pre-Defined query is stored in the Fusion Registry as an SDMX Content Constraint. In SDMX the Content Constraint can be used to define allowable content (see section 11 Reporting Constraints) or to define data present. The latter type of Content Constraint has been termed Pre-Defined Query in the Fusion Registry. The creation of a Pre-Defined query follows the same steps as a Reporting Constraint, with the exception that the constrained structure must be a Dataflow.

The view of the Pre-Defined Query is the same as the Reporting Constraint, with the additional feature of being able to run the query.

![Pre-Defined Queries](image)

**Figure 43 showing the view of a Pre-Defined Query, with the Run Query button in the details header**

Running the query loads the browse data page with the Codes pre-selected based on the Pre-Defined Query definition.

**Education**
- **Reference Area**: United States
- **Frequency**: Annual

![Data Table](image)

**Figure 44 showing the resultant data table after running the Pre-Defined Query above**
13 Metadata Structure Definition

13.1 Overview

The Metadata Structure Definition (MSD) is analogous to the Data Structure Definition (DSD), it defines the structure and allowable content of a Metadata Report (a DSD defines the structure and allowable content of a Dataset). The MSD also defines what types of artefact in the Fusion Registry the Metadata Report can be reported for.

![UML diagram for Metadata Structure Definition](image-url)

Figure 45 showing (simplified) UML for a Metadata Structure Definition

The Metadata Structure Definition (MSD) provides a template for Metadata Reports, and is used to assist in authoring and validation of a Metadata Report. When a user authors a Metadata Report, the report must conform to the MSD this is analogous to a user reporting data which must conform to the Data Structure Definition (DSD). There are a few main differences between a DSD and a MSD. Once such difference is that a MSD contains the definition of allowable Targets, these define what the Metadata Report can be authored against, a DSD does not have this concept. The MSD contains a hierarchy of Reported Attributes, a Reported Attribute is almost identical to a DSD Component (Dimension or Attribute) however a DSD Component is not hierarchical.

![MSD view](image-url)

Figure 46 showing the view of a MSD

**Note:** Details on authoring Reference Metadata can be found in the Reference Metadata Management guide.
13.2 Metadata Structure Definition Wizard
The Metadata Structure Definition Wizard includes the first generic step for information.

The second step requires information on the Report and allowable Targets. Targets can include Data related targets, this includes:

- Observation
- Series
- Code in the context of a Dataflow
- Concept in the context of a Dimension

Structure targets can include any Identifiable Structure, or specific structures can be selected.

![Figure 47 showing the Report Attachments step of the Metadata Structure Definition Wizard](image)

The third step is used to build the report template definition, by building the Report Attributes hierarchy. Each Reported Attribute must reference a Concept (which provides the name and description). This step can be used to build a hierarchy by using drag and drop.

The final step in the Metadata Structure Wizard is to specify the allowable content for each Metadata Attribute. Content type includes formats such as HTML, String, Numerical and enumerated formats, amongst others. For textual content (HTML, String) the MSD can also define if multilingual text is supported. If multilingual support is set to true in the definition then Metadata can be reported for the Attribute in multiple languages.

If the Metadata Attribute is used simply as a presentational node then it can be set to allow no content. A presentational node can be used if an Attribute has child Attributes, as shown in the following example:
As shown in the above contact report, this step also allows the user to specify the minimum number of times the Attribute can appear in the Report. A Min Occurs of 1 would indicate that the Attributes appearance is mandatory. If Max Occurs is left empty then this will be interpreted as unbounded (it can be reported multiple times with no imposed limit). In the above example there is no upper limit on the number of Contacts, but there must be at least one. In the above example each Contact must have 1 Address but no more than 2. Each Address must have a house number or name, a Street, Country and post/zip code. The list of countries will be provided by the Reference Area Codelist that the COUNTRY Attribute has been linked to.
14 Data Structure and Dataflow Maps

14.1 Overview

Dataflow Maps are almost identical to a Data Structure Map, except that the mapping is performed between two Dataflows. The remainder of this section describes Data Structure Maps, however all statements hold true for a Dataflow map.

Data Structure Maps are used to map Components (Dimensions and Attributes) from one Data Structure to another. This relationship between two Data Structures can be used to map data queries, or datasets from the source Data Structure to the target Data Structure.

Figure 49 showing the view of a Data Structure Map

14.2 Data Structure Map Wizard

The Data Structure Map Wizard includes the first generic step for information.

The second step of the wizard is to pick a source and target Data Structure; this defines which two Data Structures will be mapped. To add or modify a source or target, click in the input field to open a list of available structures.
The third step is used to define how the Components of the source Data Structure map to the Components of the target Data Structure. The **Source** table on the left of the page shows the available Components of the source Data Structure. Selecting a source will update the **Target** table in the middle of the page, which shows all currently mapped targets. The **Available** table on the right of the page shows all the Components of the target Data Structure which are not yet mapped to the selected Source. Components can be moved from Available to Target and vice versa by selecting the component in the target or available table and clicking the left or right arrow.

The last step of the wizard provides a mechanism to define, for each mapped Component, how their representation is mapped. For example if two Reference Area Components are being mapped, with the source Reference Area using an ISO 2 character country Codelist and the target using the ISO 3 character Country Codelist, this step of the wizard provides the mechanism to define the mapping used between each Code in the source and target Reference Area Codelists.

The above image shows the Component Maps on the left of the page, as defined in step 3 of the wizard. Each mapped Component can have a Representation Map defined. A Representation Map falls into one of the following three categories:
1. **Implicit** – this type of mapping states that the source and targets values will be the same and therefore there is no need explicitly define the mapping. For example if the content is textual, or if the components use the same Codelist, or the source and target Codelists share the same Code Ids then implicit mapping can be used.

2. **Explicit (Value Map)** – this type of mapping is ‘inline’ and stored as part of this Data Structure Map. A value map is imported using CSV, of source to target (as shown below). A Value Map can be used to map any text content and therefore does not have to be a valid ID ($, £, % can all be mapped for example).

   ![Figure 53 Creating a Value Map from CSV](image1)

   ![Figure 54 The resulting Value Map in a Table](image2)

3. **Explicit (Codelist Map)** – this type of mapping references an existing Codelist Map, which is created and maintained separately of this Data Structure Map. This option is only available if there is a Codelist Map defined for the Agency which maps between the source and target Codelist as used by the source and target Components that are being mapped. Codelist Maps are discussed in the next section of this document.

   ![Figure 55 showing the use of an Explicit Mapping to define how the values between two Components map](image3)
15 Codelist and Concept Scheme Maps

15.1 Overview

Concept Scheme Maps are almost identical to a Codelist Map, except that the mapping is performed between two Concept Schemes as oppose to two Codelists. The remainder of this section refers to Codelist Maps, however all statements hold true for a Concept Scheme map.

Codelist Maps are used to map Codes from one Codelist to another. This relationship between two Codelists can be used when mapping between Dataflows or Data Structures (see Data Structure Maps and Dataflow Maps). This information may also be used for reference or processing from external applications.

15.2 Codelist Map Wizard

The Codelist Map wizard includes the first generic step for information.

The second step of the wizard is to pick a source and target Codelist, this defines which two Codelists will be mapped. To pick (or edit) a source or target Codelist, click in the input box and a pop up window will appear showing all the Codelists in the system, allowing one of them to be selected.

The third step allows Code mapping information to be imported from CSV. This step is optional if Manual Mapping is preferred (step 4) then this can be used.

The CSV is expected to have 2 entries per line, one for the source Code Id, and one for the target Code Id. This is shown in the image below.

![Figure 56 showing step 3 of the Codelist Map Wizard](image)

The final step allows codes to be manually mapped to the source code. To manually map a code, select the source code, the current Target and a list of Available targets will be presented, as shown
in the image below. On selecting an available target, it can be moved to be a target by clicking on the left arrow. Current Targets can be moved into the Available target by clicking on the right arrow.

Figure 57 showing step 4 of the Codelist Map Wizard
16 Annotations

16.1 Overview
Annotations are used to provide extra information to structures, almost every structure can be annotated, and can contain zero to many annotations. Annotations provide placeholders for Annotation Id, Annotation Title, Annotation Type, Annotation Text, Annotation URI all of which are optional fields.

16.2 Annotation Wizard
To create annotations, ensure you are logged into the system as an admin user or an Agency user. Navigate to the structure to be annotated (or the parent structure, for example to annotate a Code, navigate to the Codelist that owns the Codes). On structure selection, click the maintenance icon at the top left of the page, to select Manage Annotations to open up an Annotation Wizard.

![Figure 58 showing the Manage Annotations menu option](image)

The first step of the wizard is to add annotations to the parent structure that was selected, this is the top level parent structure, for example the Codelist. The subsequent steps relate to annotating the sub structures, such as the Codes that belong to the Codelist.

The second step allows Annotations to be imported against the sub structures, these are known as Identifiable structures, for example a Code in a Codelist. The step provides the means to import the annotation(s) as CSV.

![Figure 59 showing step 2 of the Annotation Wizard](image)

The only required fields are Identifiable Id (which must reference an item in the Maintainable parent), and Annotation Id.
The third step allows Annotations to be manually created, edited, and deleted against the sub structures (Identifiable), i.e import for Codes in a Codelist.

This step provides a table view of all the current annotations against the sub-structures, with the means to edit the selected structure, or to create new or delete existing annotations.

16.3 Viewing Annotations

If a structure, or its Identifiable sub-structures, contains annotations, then this will be displayed in the table of details section for the selected structure (under the main table).

The View Annotations button will be present, and on clicking will open a pop up which will show all the annotations against the structure, or any of its sub-structures.

Figure 60 showing the Codelists page, the selected Codelist (FREQ) has annotations for it
17 Structural Usage

17.1 Overview

The Registry provides the ability to see how structures relate to each other. On an individual structure pages, selecting an item in the table shows information about that structure beneath it. For example, on the Data Structure Definitions page, selecting an individual Data Structure shows which Structures are Cross Referenced by it and also the number of Dataflows that reference this DSD.

Fusion Registry also provides a dedicated page to explore the relationships between structures. This Structure References page allows the navigation from one structure to any of its dependencies.

Use the two drop-downs on the page to select the type of structure to view and optionally filter by agency. A list of structures matching those selections will be displayed in the list on the left. Selecting an item in the list shows the Structures that Reference it and those Structures that are referenced by it.

![Structure References Page](image)

*Figure 6.1 showing the Structure References page and the references to and from a DSD*

Clicking on a structure displayed in the list on the right, will display that structure in the list on the left. The structures referencing and referenced by this structure will of course be displayed in the list on the right. By clicking on the items on the right hand side, the user can navigate around the structures, observing the relationships between them.

This page also provides the ability to generate reports about all of the structures in the system. There are currently 2 reports that can be generated.
17.2 All Structures Cross-Reference Report
This report has 3 worksheets. The first worksheet (titled “Overview”) lists all of the Structures in the Registry that have references and for each structure lists the count of how many structures it references (the “Reference Count”) and how many structures are referenced by it (the “Referenced By Count”). For a structure such as a DSD you would expect a high number for the “Reference Count” (as it probably references a number of Concepts) and a low number for the “Referenced By Count” (only Constraints and Dataflows may reference a DSD). A Codelist will always have zero for the “Reference Count” as it does not reference any other structures. From this worksheet the popularity of particular structures can be observed.

The second worksheet (titled “Consolidated”) lists all of the Maintainables in the system that reference another Maintainable and explicitly states what that reference is. A single row only shows a single relationship, so if a single Maintainable references a number of items (e.g. a DataStructure references a number of ConceptSchemes) then this Maintainable is repeated with each relationship stated on a different row.

The third worksheet (titled “Granular”) is very similar to the second worksheet, but shows the relationships at the Identifiable level. E.g. a Dimension might reference a Codelist and a Concept. A single row only shows a single relationship, so if a single Identifiable references a number of items, that identifiable is repeated on a number of rows against the different references. From this worksheet the actual relationships can be observed. Since this worksheet is likely to be long, there are filters on the first row to aid in identifying the structures you may be interested in.

17.3 Structures with no Cross-References Report
This is a single worksheet report which lists Maintainables that are not cross-referenced and do not cross-reference any other structure. From this report it is easy to see which structures in the Registry are not used and are candidates for deletion.