**DATA POINT MODELLING (DPM) METHODOLOGY**

(editor: Victor Morilla)

**1 INTRODUCTION**

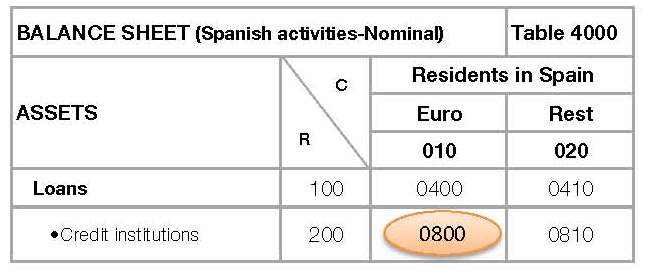
This document describes a methodology for the definition and identification of the data exchanged in financial reporting frameworks, which is supported by XBRL standards. In this document, the term data point will be used to refer to any datum represented in such reporting frameworks, i.e. concepts for which information is required.

# 1.1 Approaches for identifying the financial information

A key component in design of processes for the exchange of information is the scheme used to identify data. Under certain approaches, each datum is considered a cell in a template (a graphical representation of a set of data); the datum has no meaning out of the context of its template. Other approaches consider data independently of its graphical representation; templates can be considered sets of data points, partial views of the whole set of data represented.

These two different views of the information correspond to two different set of approaches for identifying data: form centric approaches and data centric approaches.

1. *Form centric approaches*: Under these approaches, concepts are identified by its position in the template they are represented in, i.e. the row and column (or the number of cell) of the template designed for requiring the data. The data has no meaning out of the context of the template. In the following example, the data point (cell) 0800 can be identified as Table 4000 row 200 and column 010 or as Table 4000 cell 0800.



1. *Data centric approaches*: Under these approaches, concepts are identified by a set of characteristics that are intrinsic to their meaning: the values of a set of dimensions/attributes. The data is meaningful on its own; no positional context (template, row and column) is needed for its identification. In the previous example, the data point (cell) 0800 is identified by all attributes that identify the concept from a business point of view:



In summary, a “form” centric approach is orientated exclusively to the visualization of the data with the form of the tables in which they are included.

By the contrary under a “data” centric approach methodology, any data point is expressed in terms of the dimensions necessary for their identification, like sector of the counterparty, currency of the instrument, country of residence, accounting portfolio, etc. This enables that the same data point is defined only once in a reporting framework, regardless of whether it is included in more than one table and to know which data is reported in any data point. In any case, there are visualization tools that allow that the data collected under data centric approaches can be visualized in the form of the tables in which they are required in the relevant reporting framework as in the case of “form” centric approaches.

# 1.2 Data Point Model (DPM) methodology

The *Data Point Model (DPM) methodology* is a data centric approach that has been designed for developing data models, which enables the identification (in the most detailed way) of the data points of any financial, prudential and statistical reporting framework, in exchange data processes. The DPM is based on the XBRL language, although it can be used in any IT solution.

In the Data Point Model (DPM) methodology, a data point is identified by a primary item and the members of the dimensions and possibly, by a temporal reference (information referred to the beginning of a period vs. information referred to end of the period). Hence, with the DPM methodology, a data point will be identified by its business characteristics or attributes.

The goal of a data model designed with DPM is to identify data points under a data centric approach.

Some characteristics of the DPM methodology are:

1. There is always a specific dimension, called primary item,which determines simultaneously the basic characteristics of the data.
2. There is no fixed number of dimensions in the model. This implies that it is possible to add any new dimension to the reporting framework when needed without restrictions.
3. The data are identified only by the strictly necessary dimensions. When these are not used in their identification, the model assumes that they take the default member of the dimension (e.g. Not applicable/All).

# 1.3 Dictionary and Frameworks

One of the advantages of the DPM methodology is that the elements of the DPM can be used by the different frameworks of financial reporting (e.g. FINREP, COREP, the ECB requirements, national requirements, etc.). This enables the definition of integrated reporting schemes in a more efficient way, because the elements necessary to identify the data points (primary items, dimensions and members) that are common to several reporting frameworks can be reused. E.g., when designing COREP, FINREP should be taken in consideration as well.

For that reason, the DPM methodology allows the use of a common dictionary of elements for any financial reporting framework. Any concept included to define the requirements of a framework can be reused in the future by other frameworks.



The use of a common dictionary for any framework has a lot of advantages, among them the reduction of the reporting burden to the regulators and institutions, the increase of the consistency and quality of the data.

The following template shows how the different elements of the dictionary are use by the different frameworks



1. **Description of the elements of the dictionary**
   1. **Primary items**

The XBRL language requires the inclusion in a specific dimension, called primary item, of some basic characteristic of the data exchanged:

* Its basic data type. This characteristic specifies the kind of data to be reported: a number, a date, a text, a monetary amount (a number plus a currency). This information is also used by IT applications to determine the way data is represented in electronic files.
* The period type to which the data refers: does it refer to a specific point in time (“instant”) or to an interval of time (“duration”).
* In addition to these two characteristics, the primary element in the DPM methodology provides additional detail of the type of amount or data adding a business definition (e.g. carrying amount, nominal, fair value, number of events, etc.).

Due to the information provided, the primary item is also referred to as “Data type”.

* 1. **Dimensions**

A *dimension* is each one of the additional “characteristics/breakdowns/disaggregations/attributes” that identify in detail the information described by data points (e.g. main category, currency of the instruments, sector of the counterparty, residence of the counterparty, location of the activity,...). Every “dimension” must have two or more possible members (values) and it is not possible to use the same “dimension” more than once to identify a data point.

In some occasions, the number of dimensions is a matter of opinion, because it is possible to state a dimension for any different attribute or to combine more than one attribute in a single dimension (e.g., type of product and collateral). For that reason, when deciding if it is necessary to add a new dimension to the dictionary, it is important to consider if it could be useful to use the same dimension in other potential reporting frameworks and if it is easier for the reporting institutions to split the dimensions. In this context, it is important to remember that data points are identified only by the strictly necessary dimensions. When these are not used in their identification, the model assumes that they take the default member of the dimension (e.g. Not applicable/All).

To identify properly a data point, at least the two following dimensions, in addition to the primary item (or data type), are necessary:

1. *Base item*: This dimension shows the basic conceptual [financial/prudential/statistical] meaning (nature) of the data (e.g. assets, liabilities, exposures, own fund, memorandum item, etc.). This dimension is key because it provides an additional basic characteristic of the data that affects to the convention of signs: it determines whether the data is a “debit” or a “credit”. The default member of this dimension is “Memorandum item” with the treatment of a “debit”. For that reason, if a data point is a memorandum item, it is not necessary to include explicitly this dimension to identify it.
2. *Main category:* This dimension specifies the item that the data point refers to (e.g. loans, debt securities, etc.). For that reason it is necessary to use this dimension to define the data point.

The rest of dimensions are only necessary for the identification of a data point when they have a value different than the default member.

Basic rules of dimensions:

1. A dimension can only contextualize the members of one domain.
2. For one data point, the same dimension cannot be used more than once.
3. The default member of a dimension is the default member of the domain it is associated to.
   1. **Members**

A *membe*r is anyone of the possible values that the dimensions of a data point can be assigned to (e.g. “Cash”, “Loans”, “Shares and other equity” are “members” of the dimension “Main category”).

Each member must have an accurate definition, so two concepts that share some characteristics, but are not exactly the same should be identified by two different members. For example, “accounting” trading book is not exactly equal to “prudential” trading book, even if they are almost the same. Each of these concepts must be represented in the model using a member; they cannot share the same definition.

Basic rules of members:

1. One member can only belong to one domain.
2. One member can be assigned to more than one dimension.
3. Members are contextualized by dimensions to identify data points.
4. One member can be used more than once to identify a data point (if used in the context of different dimensions).
   1. **Domains**

A *domain* is a set of “members” that share a common semantic nature.

Some dimensions can share the same “members” when they have exactly the same meaning, although they are used in a different context. For that reason, domains are necessary to avoid redundancies when the same member is used by more than a dimension (e.g. the domain “Geographical area” is used to include all members related to countries and regions, regardless of the dimension in which they are used: location of the activity, residence of the issuer, residence of the counterparty, …).

Every domain has a “default member” (e.g. “Not applicable/All” or “Not applicable”). When a data point is not identified in the model by a specific dimension, it is assumed that the member of that dimension is the default member. For that reason, it is very important to designate properly the “default member” of any domain, taking in consideration not only the necessities of the current reporting frameworks, but possible needs in the future.

There are two types of domains:

* 1. *Explicit domains*: Domains that have an exhaustive list of members.
  2. *Typed domains*: Domains whose members must be typed (e.g. dates, identification codes of securitisations, percentages, ..)

In some occasions, the inclusion of the members of some dimensions in a domain is a matter of opinion. From an IT point of view, they can be included in the same domain provided that they share the semantic nature and have the same default member, but, at least in some cases, it is more useful to split them into more than one domain according to its conceptual nature.

Domains should not be confused with dimensions, despite the fact that a domain can be associated to a single dimension (in fact, this situation is quite often). A domain is the set of members, while a dimension is the context of the usage of a member in the definition of a data point. Only when the members of one domain can be used in different contexts, a domain will contain more than one dimension.

**2.5 Hierarchies**

*Hierarchies of members* provide additional information regarding the relationship between the members included in a domain. The members of a domain are organized in nodes with their children, indicating the relation among them: equal than, higher than, … (e.g. the node “All” of Currencies is equal than “Euro” and “Currencies other than euro”).

The relation between the different nodes and their children in a hierarchy of members of a domain reflects the validation rules that can be applied to the templates they appear in. For that reason, hierarchies of members simplify the definition of the validation rules between the different concepts of a reporting framework.

An example of hierarchy could be found in the scheme below. In this scheme, the first column indicates the arithmetical relationship (“equal to” or “higher than”) between the member and the members located one level below in the hierarchy; the second column indicates whether the members should be added or subtracted to the other members in the same level to verify the relationship (with the member of a higher level) expressed in the first column; the last column conveys the level of each member in the hierarchy.

|  |  |  |  |
| --- | --- | --- | --- |
| Approach for market risk | | | |
| **=** |  | 1 | Total/NA |
| **=** | **+** | 2 | Standardised approaches |
| **=** | **+** | 3 | Normal treatment |
|  | **+** | 4 | Maturity-based approach |
|  | **+** | 4 | Duration-based approach |
| **=** | **+** | 4 | Other |
|  | **+** | 5 | Debt securities under the first category |
|  | **+** | 5 | Debt securities under the second category |
|  | **+** | 5 | Debt securities under the third category |
|  | **+** | 5 | Debt securities under the fourth category |
|  | **+** | 5 | Securitisation exposures subject to 1250% risk weighting or deduction and unrated liquidity facilities |
|  | **+** | 3 | Particular approach for CIUs |
|  | **+** | 3 | Additional requirements for options |

**2.6 Identification codes**

To simplify the identification of the data points, it is necessary to use a codification system to name the different primary items (data type), dimensions, members and domains always with the same code.

Ideally the codification system should be an international codification to allow its use in any reporting framework, but at least the different European regulators should use the same codes.

|  |  |
| --- | --- |
| CODIFICATION | |
| PRIMARYITEM | They are identified by two lower case letters followed by a number.  The first letter indicates if the data is monetary, percentage, etc. (for example, “m” would be monetary, “p” percentage, …)  The second letter indicates if it is a stock data (i: instant) or a flow data of the period (d: duration).  The number indicates the detailed data reported (e.g. 1, for fair value).    Example: “mi1” identifies that the data is the fair value of a stock monetary data. |
| DIMENSIONS | They are identified by two or more capital letters (e.g. “CU” for currency) |
| MEMBERS | They are identified by “x(number)” (e.g. “x1” could be the code for “cash”), unless there is a standard organization codification that is widely recognized, as for ISO countries and currencies.  There is no rule for the assignation of the number to the members, except that the number cannot be reused for other members even if in the future the previous member is not necessary. |
| DOMAINS | They are identified by two or more capital letters (e.g. “GA” for Geographical area). |
| DATA POINTS | They are identified by the combination of the codes of the primary item (data type) and the pairs dimension/member. |

**3. OPEN STRUCTURED TEMPLATES**

# 3.1 Introduction

This part of the document is intended to shed some light on the topic of modelling and election of dimensions in tables where there are a variable number of rows or columns (from now on, “open tables”). Because of its graphical representation, this kind of tables can be a little confusing compared to tables where all columns and rows are predefined (from now on “fixed tables”). To illustrate this issue, the following example table will be used:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Entity code** | **Entity name** | **Profit (loss) of Investee** | **Accumulated equity interest (%)** | **Jurisdiction of Incorporation** | **Carrying amount** | **Security code** | **Equity interest (%)** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

In this table, a credit institution reports certain information of its subsidiaries and of its holdings of securities issued by the subsidiaries.

# 3.2 Steps to model open structure tables

In order to undertake the analysis of such a table, the following steps are suggested:

## Identification of subject concepts

The first step consists in identifying what is the concept or concepts we are representing information about. In other words, what the *subject* of the data to be reported is. In the example, we have information describing two concepts: subsidiaries and securities held by the reporting institution.

## Identification of key fields

For each concept described in the table, we should identify one column that identifies univocally the concept we are describing (this is referred to as “key dimension” in IT terms). This key is usually a code; names are not a good choice, as names can be reported with slight differences (upper case / lower case, different number of spaces,…) making difficult the analysis of data by information systems. In our example, the concept “subsidiary” is clearly identified by the “entity code” (two different subsidiaries cannot share the same entity code) and the concept security is clearly identified with by the “security code”.

## Normalization of the table

If there is more than one concept described in a table, it must be split in sub-tables, each sub-table representing the information of a single concept: this process is called “*normalization*”. Each sub-table must contain the “key dimension” that identifies its concept and the columns describing it. The table of our example must be split in two sub-tables. The first one corresponds to information of the subsidiaries of the reporting institution, whereas the second one corresponds to information about its holdings of securities issued by the subsidiaries.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity code** | **Entity name** | **Profit (loss) of Investee** | **Accumulated equity interest (%)** | **Jurisdiction of Incorporation** |
|  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Security code** | **Entity code** | **Equity interest** (%) | **Carrying amount** |
|  |  |  |  |

It is very likely that the key dimension of one concept (or sub-table) could be a property describing another one. In this case, that dimension will appear in two or more sub-tables. But a given sub-table must not contain columns describing other sub-table’s concept other than the key dimension of the other sub-table’s concept.

In our example, the “entity code” is present in both tables. In the first table, the “entity code” is used to *identify* the subsidiary we are requesting information about (it is the “key dimension” of the table). However, in the second table it represents a piece of information about each security owned by the reporting institution. As we will see later, this different role has a reflection in the model. Somehow, the entity code in the second table can be seen as a link to the first table, where the information about a subsidiary is reported. The advantage of this approach is that information is not duplicated in normalized tables as it would happen in the following example:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Entity code** | **Entity name** | **Profit (loss) of Investee** | **Accumulated equity interest (%)** | **Jurisdiction of Incorporation** | **Carrying amount** | **Security code** | **Equity interest (%)** |
| UK0123 | Acme Inc. | 150.000 € | 60% | UK | 100.000 € | UK1234567890 | 25% |
| UK0123 | Acme Inc. | 150.000 € | 60% | UK | 200.000 € | UK7777555552 | 15% |
| UK0123 | Acme Inc. | 150.000 € | 60% | UK | 200.000 € | UK8888222203 | 20% |
| US7777 | Big Comp. | 600.000 € | 65% | US | 200.000 € | US2222000033 | 20% |
| US7777 | Big Comp. | 600.000 € | 65% | US | 300.000 € | US3333555533 | 45% |

The example shows how a lot of information is repeated. The consequence is that more data is requested to credit institutions and there is a higher likelihood of mistakes in the reporting process. However, the normalized version does not have this problem:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity code** | **Entity name** | **Profit (loss) of Investee** | **Accumulated equity interest (%)** | **Jurisdiction of Incorporation** |
| UK0123 | Acme Inc. | 150.000 € | 60% | UK |
| US7777 | Big Comp. | 600.000 € | 65% | US |

|  |  |  |  |
| --- | --- | --- | --- |
| **Security code** | **Entity code** | **Equity interest (%)** | **Carrying amount** |
| UK1234567890 | UK0123 | 25% | 100.000 € |
| UK7777555552 | UK0123 | 15% | 200.000 € |
| UK8888222203 | UK0123 | 20% | 200.000 € |
| US2222000033 | US7777 | 20% | 200.000 € |
| US3333555533 | US7777 | 45% | 300.000 € |

As this example shows, normalized tables help to reduce the amount of data required and prevent inconsistency issues.

## Standard DPM analysis

The next step consists in applying the DPM analysis individually on each sub-table. As “open tables” can be a bit confusing, we can try a little trick that consists on converting the “key dimension” into a fixed number of rows that represent specific elements. This way, we obtain a “fixed” version of the table so that we can continue with a regular DPM analysis.

In the first sub-table of our example, we will represent the information of two specific companies. Beware this change is just an exercise and is not meant to be represented in a final version of the model:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Code (Key dimension)** | **Entity name** | **Profit (loss) of Investee** | **Accumulated equity interest (%)** | **Jurisdiction of Incorporation** |
| **Entity UK0123** | Acme Inc. | 150.000 € | 60% | UK |
| **Entity US7777** | Big Comp. | 600.000 € | 65% | US |

A possible analysis for the columns and rows and common characteristics in this table follows:

Columns (x axis)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Column name*** | ***Data type*** | | | ***Main category*** | **Related parties/Relationship** |
| Entity name | String | Instant | Name | Entity name | Subsidiaries |
| Profit (loss) of Investee | Monetary | Duration | Current period | Profit or loss | Subsidiaries |
| Accumulated equity interest(%) | Percent | Instant | Percentage of the reporting entity | Accumulated equity interest | Subsidiaries |
| Jurisdiction of Incorporation | Geographical area | Instant | Country | Jurisdiction of Incorporation | Subsidiaries |

Rows (y axis)

|  |  |
| --- | --- |
| ***Row name*** | ***Entity code (Key dimension)*** |
| Entity UK0123 | UK0123 |
| Entity US7777 | US7777 |

Regarding the second sub-table, we could obtain the following model:

|  |  |  |  |
| --- | --- | --- | --- |
| **Security code (key dimension)** | **Entity code** | **Equity interest (%)** | **Carrying amount** |
| **Security UK1234567890** | UK0123 | 25% | 100.000 € |
| **Security US2222000033** | US7777 | 20% | 200.000 € |

Columns (x axis):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Column name*** | ***Data type*** | | | ***Main category*** | ***Base item*** |
| Entity code | Entity code | Instant | Entity code-- | Equity investments | Default member |
| Equity interest(%) | Percent | Instant | Percentage of equity interest | Equity investments | Default member |
| Carrying amount | Monetary | Instant | Carrying amount | Equity investments | Assets |

Rows (y axis):

|  |  |
| --- | --- |
| ***Row name*** | ***Security code (Key dimension)*** |
| Security UK1234567890 | UK1234567890 |
| Security US2222000033 | US2222000033 |

An important point to remark in this analysis is that the entity code is represented in two different ways. This in fact reflects the different roles played by this column in each sub-table:

* In the first sub-table, the entity code is used to *identify* the data that is being requested. When a reporting institution files information about the financial concept “Profit (loss) of investee”, it is necessary to clarify which subsidiary this amount refers to. Consequently, the entity code has been modelled as the “key dimension” .
* In the second sub-table, the entity code is ***not*** necessary to identify a security (in fact, the only field necessary to identify a security is its code). The entity code is information we are requesting about each security, it is data required, very much like “Equity interest”. But this data is not a monetary amount, it is a code; thus, it has a different data type. In fact, the data type of this field is the domain “entity code” in the first table.
* Similarly, the “Jurisdiction of Incorporation” is information requested about each subsidiary: a country. So, the data type of this data point is “Country”. It is possible that in other tables, the “Country” concept is used to identify the data we are requesting, for instance, a breakdown of assets by country. In such case, this field would be modelled as a dimension (e.g. “Residence of counterparty).

## Variable axes and open dimensions

In the two sub-tables of our examples we included a few rows to represent very specific subsidiaries and securities so that they could be represented as fixed tables. However, in a real reporting process this approach is not useful, as it is not feasible to know in advance what the set of possible values for subsidiaries and securities are. This information is not known by the supervisor; it is reported by the filer.

This is precisely the purpose of open dimensions: open dimensions are just like any other regular dimension. However, it’s possible values are not enumerated in advance; the actual values are created by the supervised institution during the filing process. In order to model an open dimension, it is necessary to define the data type of its values (the data type of its domain); for this reason, this kind of dimensions are known as typed dimension in XBRL technical terminology.

So, in the final step, the “key” column is represented as an open dimension. To represent formally the graphical representation of this kind of tables, open dimensions are included in the definition of a variable axis, as the following picture shows. The variable axis, as opposed to fixed axes, has not a determined number of positions (rows in this example):



So, open tables can be considered just a special case of fixed ones where the number of entries is not determined a priori.

Going back to our example, the final model for each sub-table is obtained just by converting the y axis into a variable one. The rest of the model remains the same:

**Sub-table 1**, rows (variable axis): key dimension “Entity code” (domain “entity code”)

**Sub-table 2**, rows (variable axis): key dimension “Security code” (domain “ISIN code”)

**4. Sign convention**

Strictly speaking, the “sign convention” indicates if a data point should be reported by the entities to the supervisors with positive or negative figures. However, sometimes other issues are generally referred under the label “sign convention”. These issues are: i) the restrictions and arithmetical relationship that convey the validation rules “implicit” in the templates (i.e. the indications provided using bracket or signs in the label of the related row/column in the template) and ii) the “rendering convention” that indicates how data are visualized in the IT tools.

As explained above, the sign convention “stricto senso” for the data points with “monetary” value refers to the rule for reporting increases/decreases (e.g. "*Any amount that increases the own funds or the capital requirements*…"). The rule for increases/decreases is conveyed in the DPM by assigning the "credit/debit attribute" in the base items (e.g. “own funds” and “liabilities” have the “credit” attribute, “exposures” and “assets” have the “debit” attribute). For instance, “Accumulated other comprehensive income” has as base item the member (value) “own funds” that has a “credit” attribute; consequently when this data point increases own funds should be positive and when it decreases them negative.

The “restrictions” are conventions that indicate that the data point could be reported with a positive or negative figure, or that the data point should be necessarily reported as negative. An example of this group of conventions is when in the label of the template is included “(+/-)“ to indicate that the data item could be reported either positive or negative or to include “(-)“ in the label to indicate that the data item should be necessarily reported as negative [e.g. “(-) Own shares”].

The restriction that makes a data point negative is expressed through a validation rule. For instance, the data point for the “amount” (table column label) of “(-) Direct holdings of CET1 instruments” (table row label) is a decrease of own funds and, thus, negative; to make sure that the “credit/debit attribute” is interpreted correctly, a validation rule establishing that this data point is always negative should be introduced.

Additionally, the conventions can indicate the “arithmetical relationships” that the data points must meet. These conventions indicate if the data point should be added or subtracted to other data points (with the same level in the hierarchy) to obtain a total (i.e. another data point with higher level in the hierarchy). An example of this group of conventions is to use brackets in the row/column labels of the templates to indicate that an item should be subtracted:

Sign convention: Income 100, (Expenses) 80, Profit or loss 20.

“Implicit” validation rule: 100 – 80 = 20.

Consequently, the names of the members of a domain shall not include “(-)“ (only the table rows/columns labels could use this convention) because:

1. A given data point is characterized by the primary item, base item and main category plus, if needed, other dimensions applicable. Thus, the name of the applicable members shall not include “(-)” to avoid confusion among implementers as well as to avoid the inconsistencies derived from the combination of members with different signs. For instance, if members names include “(-)”, implementers would have to analyze the names of the 3, 5 or 10 members that characterize a given data point to grasp the sign. With the “credit/debit attribute”, they only need to know whether the base items is “credit” or “debit” (easy as the number of base items is very reduced).
2. The name of the members should be non-positional. Domains are a list of members that does not contain information on how members related to each other. To include “(-)” in the name of a member could lead to inconsistencies as this member could need to be negative in relation with another members but positive in other context. For instance, the scope of CR SEC SA is securitization positions but the scope of CR SA is all exposures classes minus securization positions.

Regarding hierarchies, in principle, they could contain negative signs. However, it is a “good practice” to avoid the use of negative signs. For instance, instead of saying “Exposure net of value adjustments and provisions” = “Original Exposure” – “Value adjustment and provisions”, it is better to say that “Original exposure” = “Exposure net of value adjustments and provisions” + “Value adjustment and provisions”. Anyway, if a negative sign is required in a hierarchy, it could be used.

Finally, the “rendering convention” is an issue to be solved by national supervisors and software developers following the “sign convention” and the validation rules provided in the taxonomies. Thus, the templates, reporting instructions, illustrative examples and DPM does not need to provide a “rendering convention” rather than they should express clearly the “sign convention” and “validation rules” applied to make possible the development of appropriate “rendering conventions”.

In any case, the conventions should be clearly defined in the reporting instructions, consistently applied in the templates and clearly illustrated with examples in order to convey the “implicit” or “automatic” validation rules in the templates

# 5 .VALIDATION RULES

# 5.1 Introduction

Validation rules state mathematical or logic relationships that data points have to comply with. These validation rules should be included as part of XBRL taxonomies.

The creation of these validations with current market tools requires a deep knowledge of XBRL technical standards. Though we foresee that this situation will change soon, as long as a proper tool enabling business users to express these validations in XBRL language by themselves is not available, it is advisable to agree on a certain notation so that business experts can communicate validation rules to technical experts in an accurate way.

# 5.2 Identification of data

The identification of the set of data affected by a validation rule could be expressed using two different approaches:

* In terms of the graphical representation of the information (tables, rows and columns). We will refer to this approach as RC.
* In terms of the concepts defined in the model (base items, dimensions and members). We will refer to the approach as DPM.

The former has the advantage of being the one most users are familiar with, whereas the latter is expressed in terms of data points, and thus, independent of changes in tables and aligned with the Data Point Model in the ITS on supervisory reporting. Validations in XBRL are expressed in terms of the model, and so, if validations were decided to be officially published in terms of rows and columns (RC approach), they would also have to be necessarily translated to the DPM approach. It is also important to clarify that there is not a one to one correspondence between the two approaches: in some cases, two or more validation rules expressed in row/column terms can be expressed with a single validation rule in DPM terms.

In this document we will include examples in both RC and DPM terms; however, due to lack of time, only one of them shall be decided as the official representation in the incoming ITS on supervisory reporting. It is expected that, by the end of this year, the documentation of validations rules could be obtained automatically from taxonomy files using the proper tool and thus, published using both approaches to satisfy every user.

# 5.3 Format of references (or data identifiers)

In both approaches, the reference used to identify a certain data point or a set of data points is composed of different parts. In RC approach, these parts are the table, the row, and the column and pairs dimension/member for the Z axis in the case of tables with multiple layers (e.g.: table CRSADetail in COREP) or for the variable parts in tables with an open number of rows or columns (table GS in COREP). In the DPM approach, these parts are the base item, pairs dimension/member and in some cases, a reference to previous period of time (for instance, previous year balance). In both cases, we propose the following notation for references:

*{part1, part2, …, partN}*

Where each part will be replaced by the notation specific of the approach used (RC or DPM), as indicated in the text below.

In the case of DPM approach, the following notation is proposed:

* The base item is identified by the code of the base item itself and before any other part of the identifier. For example: mi1 refers to member 1 of the “base”, monetary, instant.
* A temporal reference to a previous period is described by a capital T followed by a negative integer number that represents the number of periods the information refers to in relation to the declaring period. For instance, T-1 refers to information of the previous period.
* A pair dimension / member is represented by the code of the dimension followed by the code of the member[[1]](#footnote-1). If a dimension is assigned to the default member of its domain, it is not necessary to include it explicitly. In the example below, CP5 refers to the pair Counterparty = Government

For instance:

*{mi1, T-1, CP5, MC1, AT20}*

An alternative representation for this notation could consist of the labels of the concepts involved. For instance:

*{Base = Assets, Previous period, Counterparty = Government, Main category =Loans and advances, Amount type = Fair value}*

Though clearer, this notation is too verbose and of little practical use for this purpose. However, it could be automatically produced in the future by a tool as a complement to other types of notation.

In the case of RC approach, the following notation is proposed:

* A table is represented by a small capital “t” followed by the code of the table.
* A row is represented by a small capital “r” followed by the number assigned to the row. The number will include only significant digits (no zero padding). This enables a short notation and avoids limitations caused by the choice of identifiers of arbitrary fixed length.
* A column is represented by a small capital “c” followed by the number assigned to the row. The number will show only significant digits (no zero padding).
* Pairs dimension / member in Z axes or variable columns or rows are represented as described in the DPM approach.

For instance:

*{tCRSA, r230, c10}*

*{tCRSADetails, r10, c20, SA1)*

It is not strictly required that parts are written in a given order. However, it is suggested to follow a common criterion in order to facilitate its reading. In the case of RC notation, it is proposed to follow a more to less general approach: first the table, then dimensions in z axes if any, then row and finally columns. In the case of DP notation, it is suggested to use first the base item, then the temporal reference if any, and then dimensions in alphabetical order.

# 5.4 Multiple references

In some cases it can be advisable to include a reference not to a single data point but to several ones. This can be expressed with a list of references separated by a comma between brackets:

( *{tCRSA, r230, c10} , {tCRSA, r240, c10}* )

In order to enable shorter references, a reference can include different values for a part:

*{tCRSA, (r230, r240, r250), c10}*

This reference is equivalent to:

*( {tCRSA, r230, c10}, {tCRSA, r240, c10}, {tCRSA, r250, c10} )*

In the case of columns and rows, the minus symbol (-) can be used to express a range. The following example shows how to include a reference to all the rows whose number is between 230 and 500.

*{tCRSA,* ***r230-r500****, c290}*

The asterisk symbol (\*) can be used to include a reference to all possible values of a part of the reference:

*{tCRSA,* ***r\*****, c290}*, where r\* represents all rows (in table tCRSA)

*{mi1, T-1,* ***CP\*****, MN1, AT20}*, where CP\* represents all possible values for dimension CP (default member included).

In the case of DP notation, multiple references can be expressed also in terms of a hierarchy of the domain associated to a dimension:

*{mi1, T-1,* ***CP.H2.31.\*****, MN1, AT20}*, where CP.H2.31.\* represents all the members that are direct children of member 31 in hierarchy H2 of the domain associated to dimension CP (members in yellow background):



*{mi1, T-1,* ***CP.H2.31.\*\*****, MN1, AT20}*, where CP.H2.31.\*\* represents all the members under member 31 (direct and indirect children) in hierarchy H2 of the domain associated to dimension CP.



*{mi1, T-1,* ***CP.H2.\*****, MN1, AT20}*, where CP.H2.\* represents all top members in hierarchy H2 of the domain associated to dimension CP:



*{mi1, T-1,* ***CP.H2.\*\*****, MN1, AT20}*, where CP.H2.\*\* represents all members in hierarchy H2 of the domain associated to the dimension CP:



If a reference contains more than one of these multiple parts, that reference represents the Cartesian product of all those parts:

{tCRSA, **r230-r500**, **c\***} represents all the cells between rows 230 and 500 in any column of table CRSA.

{mi1, T-1, (**CP5, CP6)**, **(MN1, MN5)**, AT20} is equivalent to:

( {mi1, T-1, CP5, MN1, AT20},

{mi1, T-1, CP5, MN5, AT20},

{mi1, T-1, CP6, MN1, AT20},

{mi1, T-1, CP6, MN5, AT20} )

# 5.5 Partial references

As it will more evident in the following chapters, it is advisable in some cases to express references that do not include information about all possible parts of a data point reference. The syntax to express these partial references is the same as complete ones; they just do not include some of the parts. For example:

{tCRSA, r20} is a reference to row 20 in table CRSA but it does not include any information about the columns.

{MN1, AT20} is a reference with two pairs dimension/member that does not include any information about the base item.

Partial references can be used in different sections of the definition of a validation rule. When combined together, they constitute complete references to one or several data points.

# 5.6 Parts of a validation rule

A validation rule is composed of the following parts:

* A validation region, which establishes the part of the ITS where the test is to be applied.
* A Boolean[[2]](#footnote-2) expression (we will refer to it as “test expression”) that defines the check to be applied to data within the boundaries of the validation region.
* Optionally, a validation rule can include a definition of the error margin to be used on account of the deviations introduced by rounding in the reporting process. If not specified, the default approach will be used.

# 5.7 Validation region

The validation region defines the part of the ITS where a validation is to be applied. If no validation region is provided, the validation is assumed to be applicable to the whole ITS.

The validation region is defined using partial references. For instance, {tCRSA} as the region of a validation rule, implies that the expression defined is only applicable to table CRSA.

The separation between the validation region and the text expression is marked by a colon (:).

# 5.8 Test expression

Test expressions are Boolean expressions that perform the desired checks in the data. These expressions are composed of operators and functions that work on variables, constants and other expressions.

Variables represent data points and thus, can be represented in this notation using references or partial references in combination with the validation region. In XBRL, variables are given a name, and it is this name the one which is linked to data points. For instance, the following validation rule checks that the column 10 in table OPR is equal to column 20 plus column30 for each row between 230 and 500:

*{tOPR, r230-r500}:* $a = $b + $c

$a: *{c10}*

$b: *{c20}*

$c: *{c30}*

However, we will use a shorter notation by replacing the names of variables directly by the notation of the reference:

*{tOPR, r230-r500}: {c10}* = *{c20}* + *{c30}*

Anyway, variables can be helpful in some cases where references are complex or in cases where the variable is defined in terms of an expression of other variables.

It is also worth mentioning that, the references in the validation region and the test expression complement each other to univocally identify all parts of the data points involved in a validation rule. For instance, in the previous example, the validation region identifies the table and a range of rows, whereas the test expression includes references to the columns.

# 5.9 Functions and operators

XBRL expressions are based on the W3C standard XPath 2.0. In this language, expressions are defined using a simple notation similar to the one used in arithmetic’s books and most common programming languages. Thus, we propose to use directly this notation in validation rules. There is no benefit in making up a new notation when there is one already available, well documented and supported by software tools.

Some of the most common XPath operators and functions are:

* Arithmetic:
  + + : addition. E.g.: {MC1} **+** {MC2}
  + - : subtraction. E.g.: {r20} **–** {r30}
  + \* : multiplication. E.g.: {r20, c30} **\*** {r21, c32}
  + div : division. E.g.: {tCRSA, c20} **div** 100
* Comparison:
  + = : equal. E.g.: {MC1} **=** {MC2}
  + !=: not equal. E.g.: {r30} **!=** 0
  + > : greater than. E.g.: {MC1} **>** {MC2}
  + >= : greater than or equal. E.g.: {r20} **>=** {r30}
  + < : less than. E.g.: {r20} **<** {r30}
  + <= : less than or equal. E.g.: {r20, c30} **<=** {r21, c32}
* Logical:
  + or: logical disjunction. E.g.: {MC1} = {MC2} **or** {MC1} = {MC3}
  + and: logical conjunction. E.g.: {MC1} > {MC2} **and** {MC2} > {MC3}
  + not: logical negation. E.g.: {MC1} > {MC2} and **not**({MC2} > {MC3})[[3]](#footnote-3)
* Conditional:
  + if *condition* then *expresion1* else *expresion2.*

E.g.: **if** {MC1} > 0 **then** {MC1} = {MC3} **else** {MC1} < {MC3}

* Other:
  + abs: absolute value. E.g.: {MC1} > **abs**({MC2})
  + min: minimum value of multiple values. E.g.: min({r20-r30}) returns the fact in rows 20 to 30 with the minimum value.
  + max: maximum value of multiple values. E.g.: max({r20-r30})
  + sum: addition of several values[[4]](#footnote-4). E.g.: sum({c1-c40})

# 5.10 Custom functions

In those cases where XPath does not provide a simple syntax to perform a certain operation, it is possible to define a custom function. Once defined, those custom functions should be used consistently in all validations that perform that kind of operation. For instance, a validation that checks that “if a certain data point A is zero then another data point B is zero and if B is zero then A must be zero” could be expressed like this:

if {CP1} = 0 then {CP2} = 0 else {CP2} != 0

Or like this:

( {CP1} = 0 and {CP2} = 0 ) or ( {CP1} != 0 and {CP2} != 0 )

However, using a custom function for bi-conditional expressions can be shorter and more consistent:

***if-and-only-if***( {CP1} = 0, {CP2} = 0 )

# 5.11 Constants

As some of the examples show, the operands of an expression can include constants. A constant represents a fixed value. Constant numbers are represented as a set of digits preceded by a minus in the case of negative numbers (positive numbers can be preceded by a plus symbol but it is not required). The decimal separator for rational numbers is the point “.”. Thousands separators are not allowed.

String constants are represented by a text surrounded by a simple quote: ‘Example string constant’

# 5.12 Error margin

When figures reported are not precise numbers (for instance, when data is reported in thousands of a currency rather than cents), validations must include error margins in order to cope with the imprecision introduced by rounding. By default, validation rules will be assumed to have an error margin that will be automatically computed considering the error margin of input data and the operations performed in the test expression. For instance, in a test expression like  *{CP1} = {CP2} + {CP3},* the error margin considered will be the addition error margin of each one of the three amounts involved.

However, if the validation rule has to consider a different approach, it should be explicitly indicated between square brackets. For instance, the following validation rule will check that columns 10 and 20 have exactly the same value:

{c10} = {c20} [*exact*]

# Examples

Pending: include examples in DPM notation

|  |  |  |  |
| --- | --- | --- | --- |
| **Nº** | **Validation as published** | | **Validation as proposed** |
| CA | | | |
| 1 | {CA1;230} = { GS;010;290} | | {tCA1, r230} = {tGS, r10, c290} |
| 2 | {CA1;370} = {CA4;030}-{CA4;080} | | {tCA1, r370} = {tCA4, r30} - {tCA4, r80} |
| 3 | IF {CA4;230}+{CA4;300}+{CA4;370}-{CA4;680}-{CA4;700}-{CA4;720}>0 Then  {CA1;480} = -Max[({CA4;230}+{CA4;300}+{CA4;370}-{CA4;680}-{CA4;700}-{CA4;720}-{CA4;190}) × ({CA4;230}-{CA4;680}) ÷ ({CA4;230}+{CA4;300}+{CA4;370}-{CA4;680}-{CA4;700}-{CA4;720});0]  Else {CA1;480} = 0 | | if sum({tCA4, (r230, r300, r370, )} -sum {tCA4, (r680, r700, r720)} > 0  then {tCA1, r480} = -max((sum({tCA4, (r230, r300, r370)} – sum({tCA4, (r680, r700, r720, r190)})) \* ({tCA4, r230}-{tCA4, r680}) div ({tCA4, (r230,r300,r370)} - {tCA4, (r680,r700,r720)}), 0) ) else {tCA1, r480} = 0 |
| 4 | {CA1;720} = -{CA1;970} | | tCA1: {r720} = -{r970} |
| 5 | {CA1;930} = Min [{CA4;170}; 1,25% × {CA4;180}] | | {tCA1, r930} = min (({tCA4, r170}, 0.0125 \* {tCA4, r180})) |
| CR SA | | | |
|  | columns | Validation rules applicable for all CR SA templates |  |
| 6 | For columns 010 to 130 and 180, 190 and 320 | {010;\*} = {020;\*} + {030;\*} + {040;\*} + {050;\*} + {060;\*} | (tCRSATotal, t CRSADetails), (c010- c130, c180, c190, c320): {r10} = {r20} + {r30} + {r40} + {r50} + {r60} |
|  | columns | Additional validation rules applicable to CR SA Total only |  |
| 7 | For columns 010 and 020 to 320 | {300;\*} ≥ {310;\*} | tCRSATotal, (c010-c320): {r300} ≥ {r310} |
| CR EQU | | | |
| 8 | {010;080}={020;080}+{050;080}+{100;080} | | tCREQU, c80: {r10} = {r20} + {r50} + {r100} |
| 9 | For columns 020, 060, 080, 090: {020;\*}={040-001;\*}+{040-002;\*}+…+{040-NNN;\*} | | tCREQU, (c20, c60, c80, c90): {r20} = sum({OG\*}) |
| 10 | absolute value of {CR EQU IRB;020;090} + absolute value of {CR EQU IRB;050;090} = absolute value of {CA1;230} | | abs({tCREQUIRB, r20, c90}) + abs({tCREQUIRB, r50, c90}) = abs({tCA1, r230}) |
| CR SETT | | | |
| 11 | All items in CR SETT ≥ 0 | | tCRSETT, r\*, c\*: . >= 0 |
| MKR SA EQU | |  |  |
| 12 | row 010 column 100 | row 010 column 090\*12,5 Link to CA template row 2.3.1.3 | tMKRSAEQU, r10: {c100} = {c90} \* 12.5 |
| {tMKRSAEQU, r10, c100} = {tCA, r590} |

1. Member codes are unique within a domain, and since dimensions can accept only members from a single domain, this notation is short and univocal. [↑](#footnote-ref-1)
2. A Boolean expression is a combination of arithmetic, logic, comparison and other operators whose result can be “true” or “false”. [↑](#footnote-ref-2)
3. Note that functions are defined by a name followed by an argument between brackets. For instance: not({MC1}) or abs({r20}). If a function takes multiple arguments, they are separated by a comma symbol. [↑](#footnote-ref-3)
4. Though the sum function can be considered redundant with the + operator, there are cases where this function is necessary as it is not possible to anticipate the number of cells involved, for instance, in the case of the addition of the values of a column in a table with a variable number of rows (see example 9). Moreover, the sum function is in some cases more compact than its counterparty. [↑](#footnote-ref-4)