SDMX GUIDELINES

GUIDELINES ON THE VERSIONING OF SDMX ARTEFACTS

VERSION 1.0

15/11/2015

Contents

1. Introduction

This document aims at providing guidelines on how to version SDMX artefacts inspired by "*semantic versioning*", i.e. a formal convention for specifying compatibility between the different versions of a "versionable" artefact (a SDMX artefact that has an associated version number). There are slight differences when compared to semantic versioning regarding the numbering and the definition of the criterion triggering the changes in numbering.

A three-component versioning system is proposed, with the third component being optional. The criterion for deciding which component is impacted is the severity of the change.

Versioning is central to SDMX because it guarantees the stability of references to SDMX artefacts. This is of the utmost importance given the sometimes strong dependencies between artefacts, especially in Data Structure Definitions (DSDs).

The document contains three main recommendations:

- numbering system and syntax;
- types of artefact changes and their versioning impact;
- how versioning works for inter-dependent artefacts.

The document's appendix contains examples of several types of changes and their versioning impact.

2. Numbering system and syntax

The proposed versioning system is based on the Semantic Versioning 2.0 specification[1], namely:

`MAJOR.MINOR.PATCH`[2]

However, as the "patch" component will generally not be used extensively in SDMX, it is proposed to limit the coding to `MAJOR.MINOR` as long as no patches are implemented. Concretely, this means that version number 2.1.0 will be abridged to 2.1 as long as no patch is implemented. When a patch is implemented, the version number then becomes 2.1.1. At subsequent `MAJOR` change in the versioning the `PATCH` component will disappear (2.4.7 → 3.0).

The most severe change has always precedence over other types of changes. For example, if the `MAJOR` and `MINOR` parts of the version number are impacted by changes, only the `MAJOR` component will be impacted. This means that version 3.2.1 will become 4.0.

When an artefact is published in production for the first time, the version number of the artefact should be 1.0.

3. Criterion for incrementing the version number

The criterion for deciding which component is impacted is the severity of the change, i.e. the possibility of maintaining backward and forward compatibility between the different versions of an artefact.

a. Description of backward/forward compatibility

Backward compatibility is defined as: An item (e.g. a data message) that was produced and validated with the previous version of an artefact (e.g. a DSD) can still be successfully validated using the newest version of the same artefact. For example, a data message produced and validated with a DSD version 1.1 is still valid against the same DSD (same id and Agency) upgraded to version 1.2.

Forward compatibility is defined as: An item (e.g. a data message) that is produced and validated with the new version of an artefact (e.g. a DSD) can also be validated using the previous version of the same artefact. For example, a data message produced and validated with a DSD version 1.1 is also valid against the same DSD (same id and Agency) having version 1.0 (an earlier version).

Given the syntax specified above, namely `MAJOR.MINOR.PATCH`, implementers should increment the:

- MAJOR version when changes are not backward compatible;
- MINOR version when changes are backward but not forward compatible;
- PATCH version when minor changes (e.g. text clarifications, correction of typos) are both backward and forward compatible.

[1] http://www.semver.org

[2] It should be noted that the SDMX standard specifies no limitation as to the number of components in the versioning system. The option proposed here is thus nothing but a recommended convention.

b. Cost-benefit analysis for a major version change

The cost of imposing a "major" change should be balanced against the benefit of retaining backward compatibility, for example by not deleting codes used in existing data exchanges or by deleting or replacing codes only through a concerted effort of all data exchange partners.

c. Synthesis based on the above syntax and criterion

Change Severity	Version Impact	Description	Example
Major	**+.0**	**Neither** backward **nor** forward compatibility	**1.2 → 2.0**
Minor	**N.+**	Backward **but not** forward compatibility	**1.0 → 1.1**
Patch	**N.M.+**	Backward **and** forward compatibility	**1.2 → 1.2.1**

4. Types of artefact changes and their versioning impact

As a general rule insignificant changes (e.g. textual clarifications or typos) will result in an increment of the patch component of the versioning system (i.e. N.M.+).

CODE LIST (CL)		
Type of Change	**Impact**	**Comments**
Addition into an existing CL of one or more new codes not having the CodeList:Code:ParentCode attribute	**Minor**: **N.+**(3)	Data exchanged/disseminated using the old CL can still be exchanged/disseminated using the new CL
Addition of one or more new hierarchies represented using the CodeList:Code:ParentCode attribute (not using the Hierarchical Code List artefact)	**Minor**: **N.+**(3)	Data exchanged/disseminated using the old CL can still be exchanged/disseminated using the new CL as already existing hierarchies still represent the same aggregations
Addition of one or more new codes into existing hierarchies represented using the CodeList:Code:ParentCode attribute (not using the Hierarchical Code List artefact)	**Major**: **+.0**	After the change, the parent code for the changed hierarchy does not represent the same aggregation any more, thus resulting in a break in backward compatibility
Aggregation, disaggregation, reorganisation or removal of one or more codes	**Major**: **+.0**	Data exchanged/disseminated using an old version of the CL can no longer be exchanged/disseminated using the new version of the CL

[3] The overall impact on compatibility should be assessed when there are several "minor" version impact changes. For example, it may be that the effect of adding several new Code List or HCL codes results in an implicit change in the meaning of existing Code List or HCL codes which may not be completely backward compatible, therefore (depending on the analysis) the overall version impact may be "Major +.0".

HIERARCHICAL CODE LIST (HCL)		
Type of Change	**Impact**	**Comments**
Addition of new hierarchies in the HCL. Existing hierarchies are unaffected	**Minor**: **N.+**[3]	Data represented using the old HCL can still be represented using the new HCL
Addition of codes into existing hierarchies in the HCL. Existing hierarchies are thus affected	**Major**: **+.0**	The HCL resulting from this change does not represent the same aggregation any more, thus breaking backward compatibility
Removal of one or more codes in the HCL or removal of one or more codes in the referenced code lists	**Major**: **+.0**	Data represented using the old HCL can no longer be represented using the new HCL, thus resulting in a break in backward compatibility
Addition, modification or removal of one or more hierarchical levels	**Major: +.0**	The reorganisation of codes within hierarchies has a significant impact on the code aggregations

CONCEPT SCHEME (CS)		
Type of change	**Impact**	**Comments**
Addition of one or more new concepts in an existing CS	**Minor**: **N.+**	Data exchanged/disseminated using the old version of the CS can still be exchanged/disseminated using the new CS
Removal of one or more existing concepts	**Major: +.0**	Data exchanged/disseminated using the old version of the CS can no longer be exchanged/disseminated using the new version with less concepts

DATA STRUCTURE DEFINITION (DSD)		
Type of change	**Impact**	**Comments**
Addition of a dimension	**Major**: **+.0**	Adding a new dimension has a strong impact because a dimension represents the identifier of a dataset, thus requiring a remodelling of the data as existing structural validation will fail
Addition of a mandatory attribute	**Major**: **+.0**	If the attribute is mandatory, the situation is the same as under point “Addition of a dimension”
Addition of a conditional attribute	**Minor**: **N.+**	If the attribute is conditional backward compatibility is maintained
Removal of a dimension or attribute	**Major**: **+.0**	Whatever the type of component, the change does not guarantee backward compatibility

For concrete examples, see the Appendix.

5. How versioning works for inter-dependent artefacts

This section describes how version changes to inter-dependent or parent/child artefacts affect each other. For example, how a Concept Scheme is affected when one of the Code Lists that it references changes version.

Some artefacts have references to other artefacts. For example:

- each of a Concept Scheme's Concepts may reference a Code List;
- a DSD can reference one or more Concept Schemes;
- each of a DSD's Concepts may reference a Code List. (Note that if a Concept-Code List reference exists both in a DSD and a Concept Scheme, the Concept-Code List reference in the DSD overrides the reference in the Concept Scheme);
- a Hierarchical Code List references one or more Code Lists whose codes are arranged in the hierarchical structure.

In the text below, the following concepts will be used:

- **Parent artefact**: an artefact that contains a reference to another artefact. For example, a Concept Scheme is a parent to a Code List that it references, and the Code List is the child;
- **Child artefact**: an artefact that is referenced by another artefact. For example, a Code List is a child of a Concept Scheme that contains a reference to it, and the Concept Scheme is the parent.

It is important to note that a new version of a child artefact does not automatically trigger a version update of the parent artefact. A version change to the parent artefact is made only if the new version of the child artefact is adopted by the parent artefact.

a. Impact on parent artefact when child artefact version changes

The replacement of a reference with a different reference has the same impact for every artefact.

ALL ARTEFACTS		
Type of change	**Impact**	**Comments**
Replacement of a child artefact having a different version, but same id and Agency	**The child artefact version change is replicated in the parent artefact**	If a child artefact (e.g. a Code List) has a minor version change, then the parent artefact (e.g. a Concept Scheme) should also have a minor version change. If there are several child artefact version changes, the most severe impact is replicated in the parent artefact. For example, if two Code Lists have minor changes, and one Code List has a major change at the same time, the parent Concept Scheme has a major version change
Replacement of a referenced child artefact having a different id or Agency	**The parent artefact version impact depends on the backward/ forward compatibility as shown in the tables above**	Technically, the child artefact is not considered to be related to the previous child artefact. It needs to be checked whether exchange contracts can still be guaranteed (backward/forward compatibility principle)

b. Addition or removal of referenced artefacts

CONCEPT SCHEME (CS)		
Type of change	**Impact**	**Comments**
Addition or removal of a child Code List	**Minor: N.+**	The child Code Lists in a Data Structure Definition have priority over those referenced in a Concept Scheme. Child Code Lists added to or removed from a Concept Scheme do not have a direct impact on the data exchange. Backward/forward compatibility depends on the way Code Lists are referenced in Data Structure Definitions referencing the concept scheme. This needs to be taken into account when creating a new version of a DSD accordingly

DATA STRUCTURE DEFINITION (DSD)		
Type of change	**Impact**	**Comments**
Addition or removal of a child Code List	**If same id and Agency, then the child artefact version change is replicated in the parent artefact.** **If different id or Agency, impact wil depend on the backward/forward compatibility as shown in the tables above**	If a child Code List has a minor version change, then the DSD should also have a minor version change. If there are several Code List version changes, the most severe impact is replicated in the DSD. For example, if two Code Lists have minor changes, and one Code List has a major change at the same time, the parent DSD has a major version change

6. Appendix - Examples

Example 1 – Change to a Code List name, for clarification purposes. Patch Impact: N.M.+

Id	Old Name	New Name
CL_ADJUSTMENT	Adjustment codes	Adjustment code list

Example 2 – Change to a Concept name, for clarification purposes. Patch impact: N.M.+

Id	Old name	New name
PRODUCT_TO	Product classification	Product classification (input-output product*product)

Example 3 – Change in the substance of codes. Major impact: +.0

Id	Old name	New name
CP01115	Other products	Pizza and quiche

Example 4 - Aggregation, disaggregation or reorganisation of codes. Major impact: +.0

AGGREGATION OF EXISTING CODES	
Old version	**New version**
2011 Heifers (female bovine that never calved), live **2012** Cows, live	**2010** Heifers and cows, live
Codes **2011** and **2012** are fully[4] **removed** and replaced with one **brand new** code. In this case there is a many to 1 correspondence between the codes.	

DISAGGREGATION OF EXISTING CODES	
Old version	**New version**
1010 Live horses	**1011** Pure bred breeding horses, live **1012** Other horses, live
Code **1010** is fully **removed** and replaced with two **brand new** codes. In this case there is a 1 to m correspondence between the codes.	

[4] i.e. without integration into or combination with another existing code.

REORGANISATION OF EXISTING CODES	
Old version	**New version**
3010 Fowls, weighing ≤ 185 g **3020** Ducks, , weighing ≤ 185 g **3030** Other poultry, weighing ≤ 185 g **3040** Fowls, weighing > 185 g **3050** Ducks, , weighing > 185 g **3060** Other poultry, weighing > 185 g	**3025** Poultry, weighing ≤ 175 g **3045** Poultry, weighing > 175 g
Codes **3010**, **3020**, **3030**, **3040**, **3050** and **3060** are fully removed and replaced with two brand new codes; furthermore the criterion for the classification used in the old version has been changed in the new version (185 g criterion versus 175 g criterion), so that it is not possible to exactly aggregate the codes from the old version to the codes of the new version (e.g. a part of **3010** goes to **3025**, another part to **3045**). In this case there is a m to n correspondence between the two sets of codes	

Example 5 – Changes to hierarchies in a Code List. Major impact: +.0

ADDING A NEW CODE IN AN EXISTING HIERARCHY – CODE LIST	
Old version	**New version**
• 0213 - Beer o 02131 - Lager beer o 02132 - Other alcoholic beer	• 0213 - Beer o 02131 - Lager beer o 02132 - Other alcoholic beer o **02133 - Low and non-alcoholic beer**
Code 02133 has been added to hierarchy 0213	

Example 6 – Changes to hierarchies in a Hierarchical Code List. Major impact: +.0

ADDING A NEW CODE IN AN EXISTING HIERARCHY – HIERARCHICAL CODE LIST	
Old version	**New version**
• A1 - World (codelist ref. ECB@CL_AREAS@1.0) o E1 - Europe (ECB@CL_COUNTRIES@1.0) ▪ ES - Spain ▪ FR - France ▪ GR - Greece ▪ IT - Italy o E4 - Africa ▪ etc.	• A1=World (codelist ref. ECB@CL_AREAS@1.0) o E1 = Europe (ECB@CL_COUNTRIES@1.0) ▪ ES = Spain ▪ FR = France ▪ GR = Greece ▪ IT = Italy ▪ **DE= Germany** o E4 = Africa ▪ etc.
The id of the hierarchical codes are assumed to be equal to those of the code lists referenced. The code DE has been added to hierarchy E1	

Example 7.1 – Dependencies between artefacts: Concept Scheme and Code List. Minor impact: N.+

Id:Artefact Type:Details	Change type	Version Impact	Old version	New version
CL_OBS_STATUS:Code List	Addition of a new code X	**Minor: N.+**	**1.0**	**1.1**
CS_TRADE:Concept Scheme: References CL_OBS_STATUS v1.0 above	Adoption of new code X ***Change type***: Replacement of a child artefact having a different version, but the same id and Agency	**Minor: N.+** The child version impact is replicated in the parent artefact	**2.0**	**2.1**

Example 7.2 – Dependencies between artefacts: Concept Scheme and Code List. Major impact: +.0

Id:Artefact Type:Details	Change type	Version Impact	Old version	New version
CL_OBS_STATUS:Code List	Removal of code U	**Major: +.0**	**1.0**	**2.0**
CS_TRADE:Concept Scheme:References CL_OBS_STATUS v1.0 above	Adoption of new CL_OBS_STATUS without U. ***Change type***: Replacement of a child artefact having a different version, but the same id and Agency	**Major: +.0** The child version impact is replicated in the parent artefact.	**2.0**	**3.0**

Example 7.3 – Dependencies between artefacts: Concept Scheme and Code List. Variable impact (see below)

Id:Artefact Type:Details	Change type	Version Impact	Old version	New version
CL_XYZ: Code List	a) Maintenance agency changes from A to B for governance reasons. Nothing else changes in the code list.	**New artefact**	**CL_XYZ (Agency A)**	**CL_XYZ (Agency B) (new maintenance agency)**
	b) Maintenance agency changes from A to B and at the same time new codes are added			
	c) Maintenance agency changes from A to B. Since B has different coding rules, the code list itself changes as well.			
CS_TRADE: Concept Scheme: References CL_XYZ (Agency A)	Replacement of a child artefact having a different Agency. CL_XYZ (Agency A) changes to CL_XYZ (Agency B).	**Case a): Patch: N.M.+** There is no impact on data exchange	**2.0**	**2.0.1**
		Case b): Minor: N. + The impact is the same as a new minor version of the code list	**2.0**	**2.1**
		Case c) Major: +.0 The impact is the same as a new major version of the code list.	**2.0**	**3.0**

Example 7.4 – Dependencies between artefacts: Concept Scheme and DSD. Variable impact (see below)

Id:Artefact Type:Details	Change type	Version Impact	Old version	New version
CS_TRADE: Concept Scheme containing Concepts C1, C2, C3	Addition of new Concept C4	**Minor: N.+**	**1.4**	**1.5**
TRADE: Data Structure Definition: references Concepts C1 and C2	None Concept C3 is not used	**None**	**1.0**	**1.0**

CS_TRADE: Concept Scheme containing Concepts C1, C2, C3	Change of description in Concept C3 (typo)	**Patch: N.M.+**	**1.4**	**1.4.1**
TRADE: Data Structure Definition: references Concepts C1 and C2	None Concept C3 is not used	**None**	**1.0**	**1.0**
CS_TRADE: Concept Scheme containing Concepts C1, C2, C3	Removal of Concept C3	**Major: +.0**	**1.4**	**2.0**
TRADE: Data Structure Definition: references Concepts C1 and C2	None concept C3 is not used	**None**	**1.0**	**1.0**
Remark: Once a new version of the DSD is needed for some other reasons (e.g. a change in a code list), it is recommended to update all concept references to the newest available concept scheme if possible: i.e. DSD version 1.1 would then update its concept scheme references from 1.4 to 2.0.				
CS_TRADE: Concept Scheme containing Concepts C1, C2, C3	Change of description in Concept C2 (typo)	**Patch: N.M.+**	**1.4**	**1.4.1**
TRADE: Data Structure Definition: references Concepts C1 and C2	Correction should be taken into account, concept C2 is used	**Patch: N.M.+** ***or*** **None**	**1.0**	**1.0.1** ***or*** **1.0**
Remark: Since the change of a typo in a Concept of the Concept Scheme does not have a direct impact on the DSD itself (the link is by reference), there is strictly speaking no need to update the DSD. Both DSDs (1.0 and 1.0.1) will have exactly the same syntax. However, if maintainers want to highlight the correction for users of the DSD or for some other reason the DSD is updated anyway; it should reference the newer Concept Scheme.				