Semantic Versioning: Unlocking the Power of SDMX 3.X

OCTOBER 1, 2025

Allen Boddie

Versioning in SDMX

Every **Maintainable** SDMX Artefact has a URN which is a combination of Agency, ID, version and artefact type.

$$\underbrace{\text{urn:sdmx:org.sdmx.infomodel}}_{\text{Namespace}}.\underbrace{\text{codelist}}_{\text{Package}}.\underbrace{\text{Codelist}}_{\text{Class}}=\underbrace{\text{SDMX}}_{\text{Agency}}:\underbrace{\text{CL\_FREQ}}_{\text{ID}}(\underbrace{\text{2.1}}_{\text{Ver}})$$

In SDMX 3.X there are two types of versioning:

- 2-digit version
- 3-digit version numbers which must be semantically versioned

Semantic versioning

5.2.7-draft

Major: Backwards incompatible

Minor: Backwards compatible

Patch: Forward and backwards compatible

Extension: Optional, if present the artefact is mutable

SDMX 3.X’s best feature...

Wildcard referencing - No longer need to reference a specific version of an artefact!

Levels:

- Major: Wildcard at the major level allowing all major, minor and patch updates 1+.0.0
- Minor: Wildcard at minor level allowing all minor and patch updates 2.3+.0
- Patch: Wildcard at patch level allows all patch updates 3.1.8+

Enables:

- Codelist inheritance
- Evolving data structures
- Etc.

What's missing from the standard?

Artefact specific guidance on impacts on versioning of changes

- Is a given change a minor or major change?
- What flexibility do I have to decide the version number?

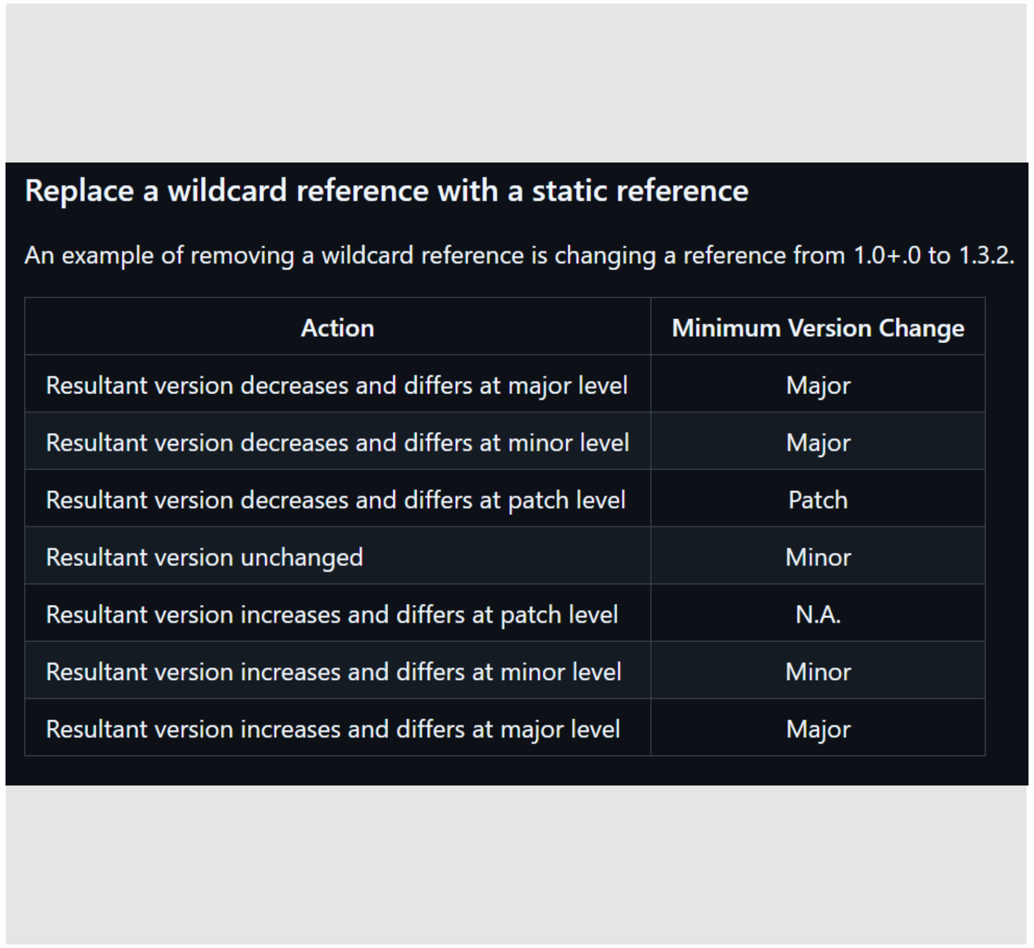
Guidance on wildcard references

- What are the implications of having wildcard references in different scenarios?
- Are their any concerns around having a Major or Minor wildcard reference?

Guidance on migration

- Semantic versioning is great; how can I actually implement it in my environment?

General guidance



Replace a wildcard reference with a static reference

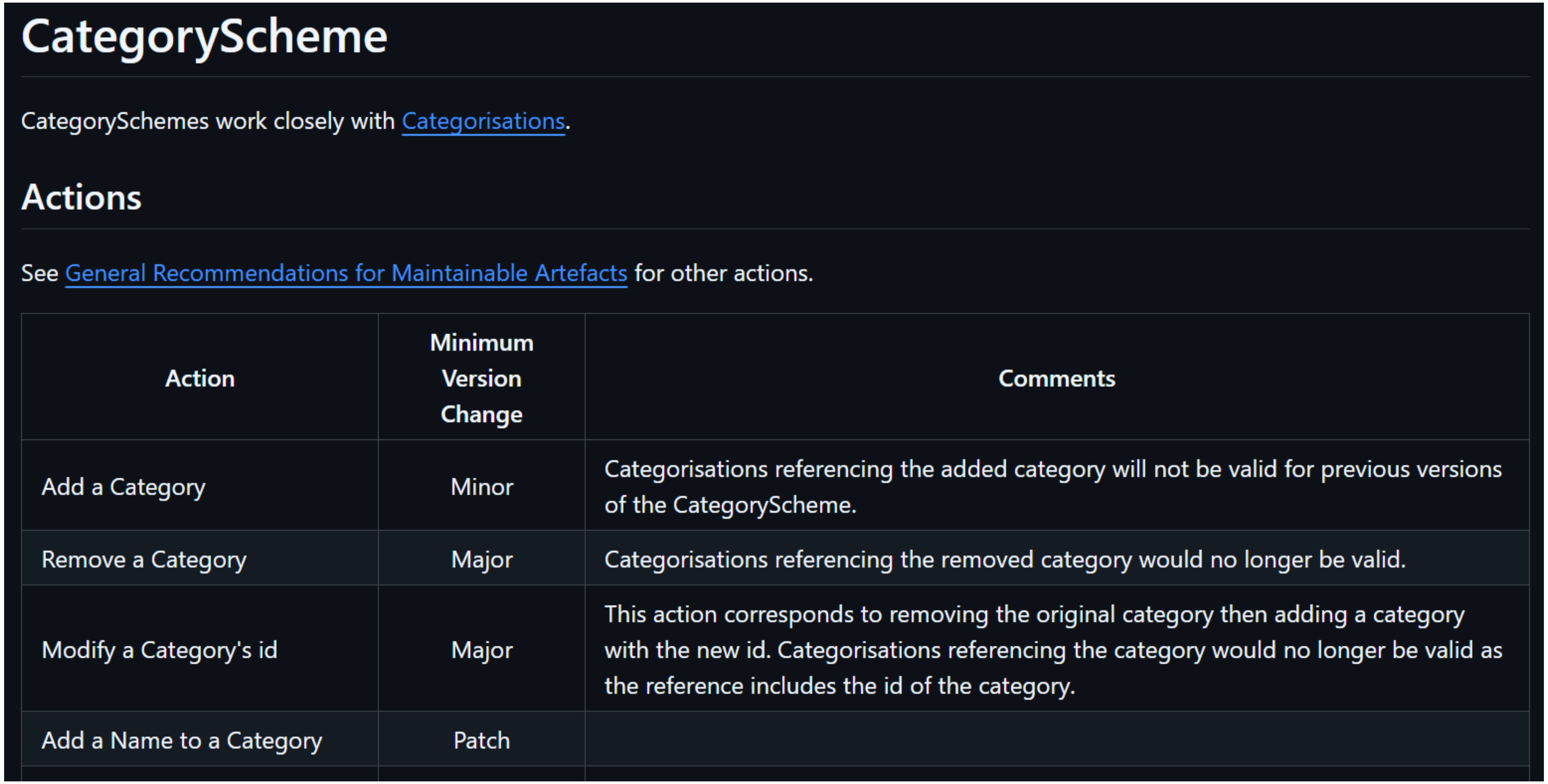
An example of removing a wildcard reference is changing a reference from 1.0+.0 to 1.3.2.

Action	Minimum Version Change
Resultant version decreases and differs at major level	Major
Resultant version decreases and differs at minor level	Major
Resultant version decreases and differs at patch level	Patch
Resultant version unchanged	Minor
Resultant version increases and differs at patch level	N.A.
Resultant version increases and differs at minor level	Minor
Resultant version increases and differs at major level	Major

- Actions relevant to all maintainable artefacts.
  - Changing description
  - Changing validFrom
  - Etc.

- Detailed guidance on changing references

Artefact specific guidance



CategoryScheme

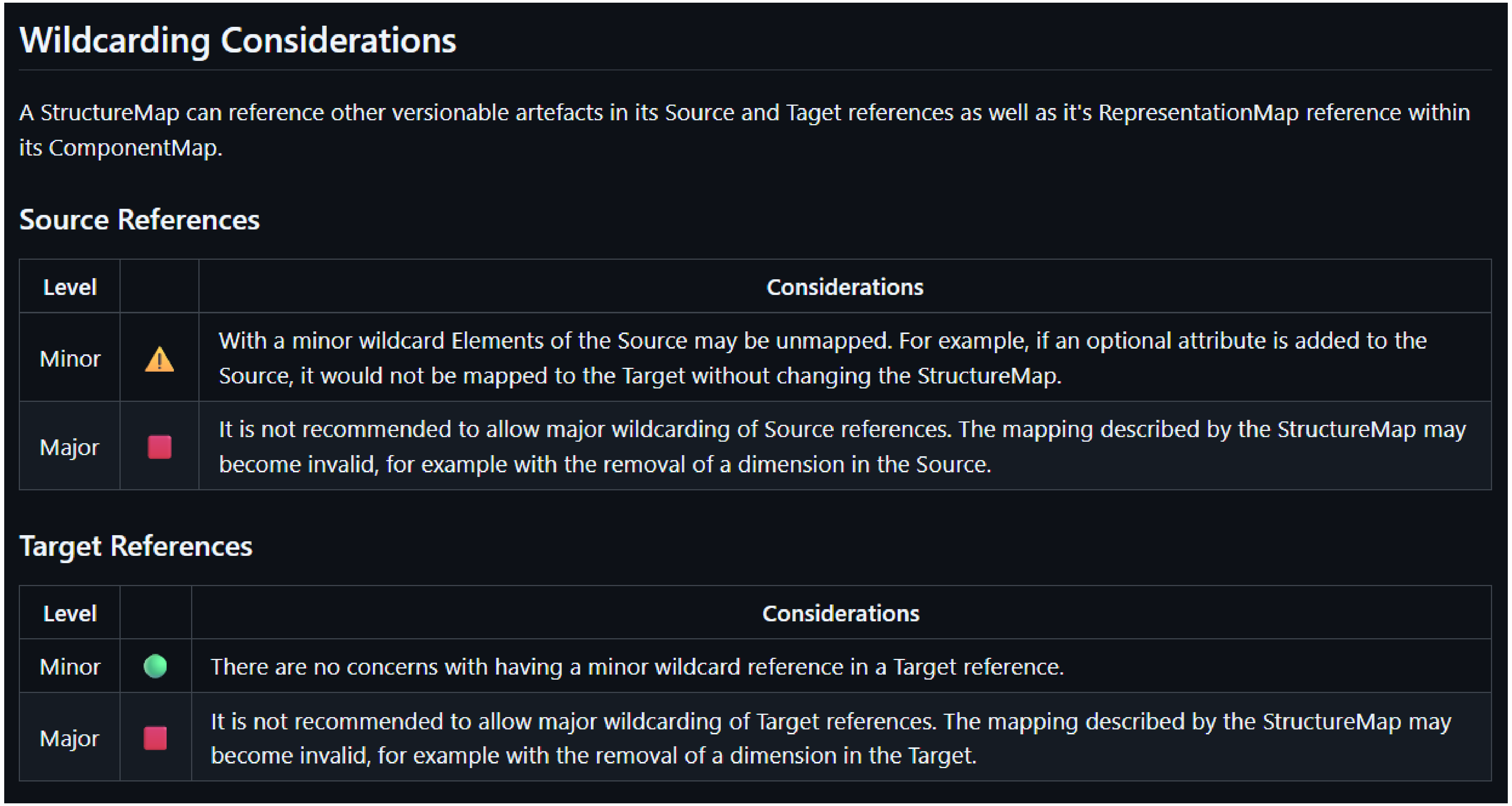
CategorySchemes work closely with Categorisations.

Actions

See General Recommendations for Maintainable Artefacts for other actions.

Action	Minimum Version Change	Comments
Add a Category	Minor	Categorisations referencing the added category will not be valid for previous versions of the CategoryScheme.
Remove a Category	Major	Categorisations referencing the removed category would no longer be valid.
Modify a Category's id	Major	This action corresponds to removing the original category then adding a category with the new id. Categorisations referencing the category would no longer be valid as the reference includes the id of the category.
Add a Name to a Category	Patch	

Guidance on wildcard references



Wildcarding Considerations

A StructureMap can reference other versionable artefacts in its Source and Taget references as well as it's RepresentationMap reference within its ComponentMap.

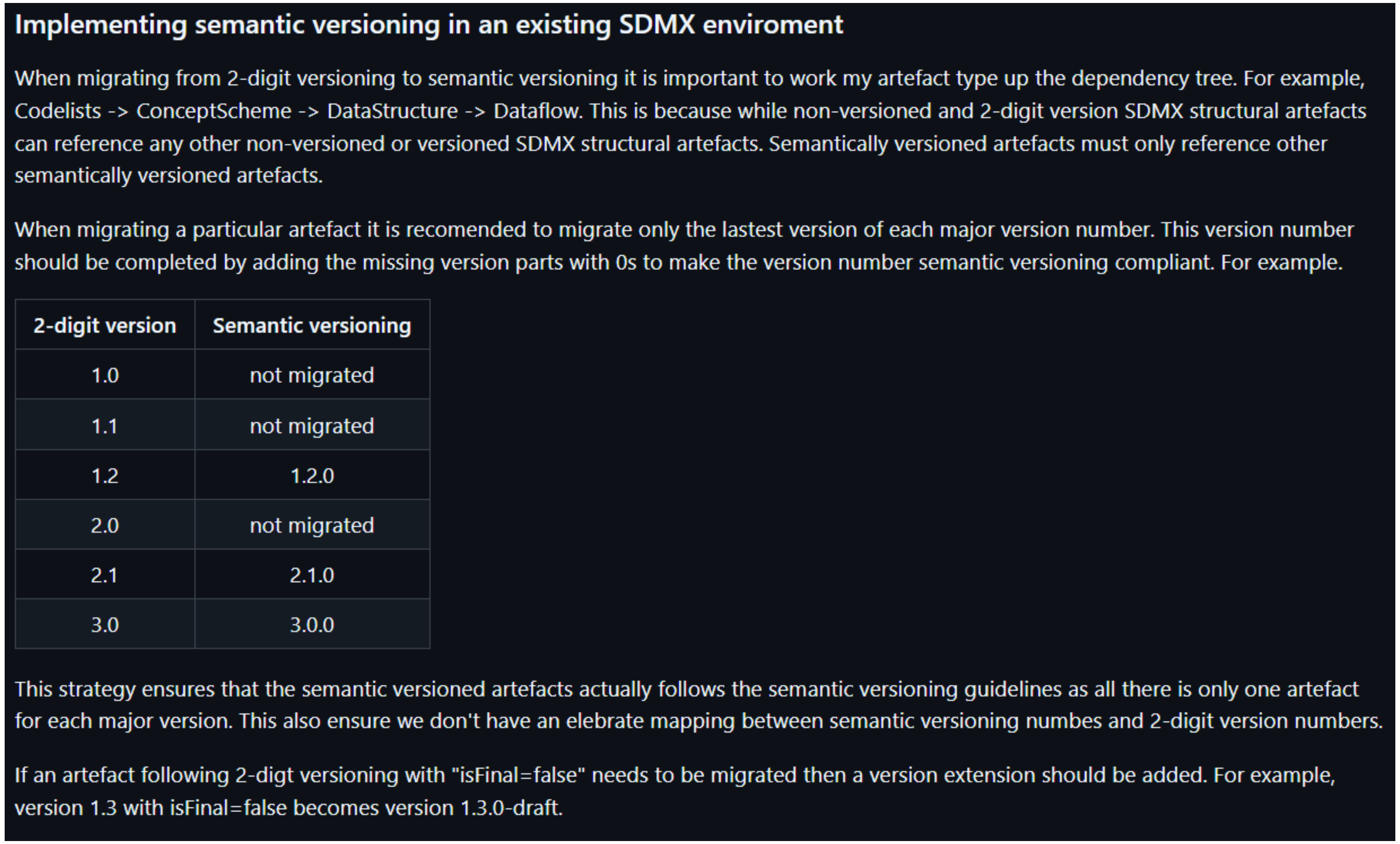
Source References

Level		Considerations
Minor	⚠️	With a minor wildcard Elements of the Source may be unmapped. For example, if an optional attribute is added to the Source, it would not be mapped to the Target without changing the StructureMap.
Major	🟥	It is not recommended to allow major wildcarding of Source references. The mapping described by the StructureMap may become invalid, for example with the removal of a dimension in the Source.

Target References

Level		Considerations
Minor	🟢	There are no concerns with having a minor wildcard reference in a Target reference.
Major	🟥	It is not recommended to allow major wildcarding of Target references. The mapping described by the StructureMap may become invalid, for example with the removal of a dimension in the Target.

Guidance on migration



Implementing semantic versioning in an existing SDMX enviroment

When migrating from 2-digit versioning to semantic versioning it is important to work my artefact type up the dependency tree. For example, Codelists -> ConceptScheme -> DataStructure -> Dataflow. This is because while non-versioned and 2-digit version SDMX structural artefacts can reference any other non-versioned or versioned SDMX structural artefacts. Semantically versioned artefacts must only reference other semantically versioned artefacts.

When migrating a particular artefact it is recomended to migrate only the lastest version of each major version number. This version number should be completed by adding the missing version parts with 0s to make the version number semantic versioning compliant. For example.

2-digit version	Semantic versioning
1.0	not migrated
1.1	not migrated
1.2	1.2.0
2.0	not migrated
2.1	2.1.0
3.0	3.0.0

This strategy ensures that the semantic versioned artefacts actually follows the semantic versioning guidelines as all there is only one artefact for each major version. This also ensure we don't have an elebrate mapping between semantic versioning numbes and 2-digit version numbers.

If an artefact following 2-digt versioning with "isFinal=false" needs to be migrated then a version extension should be added. For example, version 1.3 with isFinal=false becomes version 1.3.0-draft.

Next steps

Feedback on Guidance

Continue to update with new versions of SDMX

- SDMX 3.0
- SDMX 3.1

Integration into standard and tools

Q&A

https://github.com/sdmx-twg/semver