Guidelines for INSPIRE application schemas and feature chaining for Geology

Background, Theory and Praxis

GBA Guidance

Version: 0.0

Date: 13.10.20

Author: Kathi Schleidt

Version History

0.0	13.10.20	Basic doc with all classes
0.1	19.10.20	Provision Workflow
0.2	28.10.20	Basic App Schema Configuration
0.3	3.11.20	Resolvable Identifiers and Rewriting framework
0.4	10.11.20	App Schema – extended feature chaining, updated identifiers to GBA scheme
0.5	13.11.20	Extended Class diagrams and descriptions
0.6	18.11.20	Customized Stored Queries to GBA
0.7	23.11.20	Finalized App Schema tutorial, workflow
0.8	27.11.20	Added Vocabularies and semantics, finalized Identifiers and Versioning
0.9	2.12.20	Final Draft for GBA Review
1.0	24.8.21	Final Updated Version

Contents

1. Introduction

Just as INSPIRE covers a wide range of different types of spatial data and services, many different approaches to the provision process have emerged, utilizing different technologies as well as provision paradigms. A part of this spectrum is due to the different forms of data foreseen under INSPIRE; different service types, architectures and server solutions are required for the provision of vector-based datasets than for gridded data or highly dynamic sensor data. Additionally, the existing data landscape must be taken into account, as different management and storage systems may predispose towards certain solutions.

Based on preliminary analysis of the data held by GBA, it was determined that most data relevant to provision by INSPIRE pertains to vector-based datasets, whereby these datasets are relatively static in nature and only rarely updated. The vector-based datasets to be made available via INSPIRE services are stored in a PostgreSQL database with the PostGIS spatial extension, whereby the principles described in this document apply to all spatial databases. However, as some thematic areas are only covered by gridded models modelled as coverages, these data models are also contained, together with information on how to transform these as foreseen under the INSPIRE Good Practice for Coverage data.

In the context of this document, we focus on the utilization of the Open Source GeoServer software for the provision of INSPIRE Download Services, specifically Web Feature Service (WFS) and OGC API – Features (OAPIF); if required, GeoServer can be configured to provide INSPIRE View Services in the form of Web Map Services (WMS) on these data sources. While direct configuration of GeoServer is seen as overly complex and technical by some, this is also what makes it well suited for the complex data models that apply to the GBA datasets. A further benefit of utilizing GeoServer for provision of data in accordance with the INSPIRE data specifications and XML Schemata made available for this purpose by the Joint Research Center (JRC) of the European Commission (EC) ensues from the new SmartDataLoader and Templating extensions, enabling rapid prototyping and simple creation of additional service endpoints serving community specific profiles of available data.

As INSPIRE steadily shifts the spatial data paradigms from the closed-world philosophy reflected in many organizational data systems to an open-world philosophy more closely related to linked-data approaches, various concepts that proved consistent within the local context must be rethought to function within such a wider harmonized data ecosystem. Identifiers of objects must retain uniqueness beyond the direct context of the dataset they are stored in if these objects are to be externally referenced. While INSPIRE has foreseen specific identifier types, how these can be utilized to enable data linking are still under debate, with various options being implemented by different organizations, many largely precluding direct links from external sources. As direct access to specific objects is essential for an SDI, provisions should be put in place to enable this via the download services.

A further aspect that becomes visible in the open-world philosophy pertains to semantics; the actual meaning of the various classifiers utilized within the local data stores. Such local classifiers are often “home-grown”, not based on external definitive sources, and thus hard for external users

of the data to interpret, especially in the context of an international infrastructure such as INSPIRE. For this reason, various types of concept repositories have arisen, ranging from simple codelists over richer thesauri to complex ontologies.

In this document, we provide information on how to best structure the entire data provision workflow from data identification to standardized provision of semantically interoperable data. For readability, the more technical information on how to implement the recommendations from this document is provided in the annexes to this document.

In addition to basic provision of data, we will also detail further considerations that must be taken into account in order to assure that the data is not only available, but truly (re-)usable across contexts and domains. Finally, all recommendations underpin the FAIR principles, assuring that available data is Findable, Accessible, Interoperable and Usable.

2. Infrastructure and Provision Options

2.1. Provision Workflow

2.1.1. Identify dataset(s)

The first step in the INSPIRE data provision workflow pertains to determining what of the data being administrated by the data holder both pertains to INSPIRE and actually belongs to this organization (in contrast to data being held by the organization for operational purposes, but actually belonging to a different organization; in such a case, the organization with actual ownership of the data must fulfil the INSPIRE requirements thereon).

On a general level, most data with a geospatial context fall under INSPIRE requirements, but at a more detailed level, the determination of what parts of the data being investigated pertain to which INSPIRE Themes is not always easy. While INSPIRE aims to enable access to a wide range of spatial data types, the 34 Themes provided in the Annexes to the INSPIRE Directive remain an eclectic mix of fundamental concepts (e.g. Coordinate Reference Systems or Grids), thematic domains (whereby these are not always clearly delineated, see Hydrogeology vs. Hydrology vs. Water Transport Networks) and cross-cutting concepts (e.g. Environmental Monitoring Facilities, that pertain to all environmental media being measured). How these different themes interact has been poorly elaborated to date.

A general concern repeatedly encountered as geospatial data becomes more openly available via services pertains to the concept of a dataset, common in geospatial data management, but increasingly outdated with the shift towards more IT-centric solutions. Within geospatial concepts, a dataset is often aligned with a Layer or ShapeFile, while within a database such delimitations become lost as the data becomes increasingly structured and interrelated. With the advent of linked data such differentiation becomes ever more difficult as ever more data becomes interlinked, both within individual organizations as well as across organizations; it becomes increasingly clear that the dataset concept is very subjective, often being circumscribed by the data requirements of a specific use case.

Pertaining to INSPIRE, while general good practice would advocate a separation of concerns between data identification, selection, encoding and provision, due to some of the INSPIRE requirements to Network Services, a more satisfactory result for both data providers and users may be attained by paying attention to all aspects of the data provision process from the onset. Specifically, the delimitation of a dataset impacts the following aspects:

- Metadata: exactly one metadata record must be provided to describe each dataset. Thus, the content of each dataset should be homogeneous in regard to the information being provided within this record.
- Service endpoints: in the INSPIRE Technical Guidance for INSPIRE Download Services 3.1, Requirement 52 states:

A separate WFS endpoint shall be provided for each INSPIRE dataset thus providing one dataset per GetCapabilities response.

The following table shows the relevant datasets presently identified within GBA as relevant towards INSPIRE:

ID	Title
4241	**Tektonische Linien 1:1 Mio.** Shear Displacement Structures 1:1 Mio
4242	**Tektonische Linien 1:50 000** Shear Displacement Structures 1:50k
4244	**Aerogeophysik Messkampagnen Österreich** airborne geophysics campaigns Austria
4245	**Gravitative Massenbewegungen - Observed Events (Media)** Gravitational Mass Movements - Observed Events (Media)
4253	**Bodengeophysik Profillinien Österreich** ground geophysics profil lines Autsria
4254	**Hydrogeologische Objekte - Natürliche und künstliche Objekte** Hydrogeological Objects - Natural and ManMade Objects
4255	**Hydrogeologische Einheiten - Aquifertypen 1:500 000** Hydrogeological Units - Aquifer Media Types 1:500 000
4256	**Mineralvorkommen und Rohstoffe in Österreich** Mineral Occurrences and Commodities in Austria
7900	**Geologische Einheiten 1:50 000 Österreich (Oberflächengeologie)** Geologic Units 1:50k Austria (Surface Geology)
7905	**Geologische Einheiten 1:200 000 Österreich (Oberflächengeologie)** Geologic Units 1:200k Austria (Surface Geology)
7910	**Geologische Einheiten 1:500 000 Österreich (Oberflächengeologie)** Geologic Units 1:500k Austria (Surface Geology)
7920	**Geologische Einheiten 1:1 Mio. Österreich (Oberflächengeologie)** Geologic Units 1:1 million Austria (Surface Geology)

Table 1: INSPIRE relevant datasets identified in GBA

2.1.2. Identify correct INSPIRE Theme

Once the relevant datasets have been identified, these must be aligned with the Themes and Application Schemas laid down by INSPIRE. To this purpose, the concepts described by the individual tables of the database encompassing the dataset must be analyzed and compared with the classes that make up the individual INSPIRE Application Schemas, in order to determine which Application Schema is best suited for conveying the available data.

The information available from the European Commission's INSPIRE website[1] are a good support for this purpose. The base page on Data Specifications[2] is a good entry point for accessing

[1] https://inspire.ec.europa.eu/
[2] https://inspire.ec.europa.eu/data-specifications/

information on the individual INSPIRE Themes that may be of relevance. For each Theme, one finds all relevant information including Technical Guidelines and the underlying UML data models. A useful tool that can be accessed from here is the INSPIRE Find-Your-Scope[3] utility, that is foreseen to be useful especially in situations when datasets fall under two or more INSPIRE data themes / application schemas content. The application also serves as a catalogue of all objects defined by INSPIRE.

The datasets currently identified as relevant to INSPIRE within GBA have been aligned with INSPIRE Themes and Applications Schemas as shown in the following table.

ID	Theme	Application Schema	Title
4256	MR	MineralResources	**INSPIRE Mineralvorkommen und Rohstoffe in Österreich** INSPIRE Mineral Occurrences and Commodities in Austria
4254	GE	Hydrogeology	**INSPIRE Natürliches hydrogeologisches Objekt** INSPIRE HydrogeologicalObjectNatural Austria
4255	GE	Hydrogeology	**INSPIRE Hydrogeologische Einheiten - Aquifer 1:500 000 Österreich** INSPIRE Hydrogeological Units - Aquifer 1:500k Austria
4254	GE	Hydrogeology	**INSPIRE Künstliches hydrogeologisches Objekt** INSPIRE HydrogeologicalObjectManMade Austria
4241	GE	Geology	**INSPIRE Tektonische Linien (GE.ShearDisplacmentStructure) 1:1 Mio. Österreich** INSPIRE ShearDisplacement Structures 1:1 Mio. Austria
4242	GE	Geology	**INSPIRE Tektonische Linien (GE.ShearDisplacmentStructure) 1:50000 Österreich** INSPIRE ShearDisplacement Structures 1:50k Austria
4245	NZ	NaturalRiskZones	**INSPIRE Gravitative Massenbewegungen - Observed Events (Media) Österreich** INSPIRE Gravitational Mass Movements - Observed Events (Media) Austria
7900	GE	Geology	**INSPIRE Geologische Einheiten 1:50 000 Österreich (Oberflächengeologie)** INSPIRE Geologic Units 1:50k Austria (Surface Geology)
7905	GE	Geology	**INSPIRE Geologische Einheiten 1:200 000 Österreich (Oberflächengeologie)** INSPIRE Geologic Units 1:200k Austria (Surface Geology)
7910	GE	Geology	**INSPIRE Geologische Einheiten 1:500 000 Österreich (Oberflächengeologie)** INSPIRE Geologic Units 1:500k Austria (Surface Geology)
7920	GE	Geology	**INSPIRE Geologische Einheiten 1:1 Mio. Österreich (Oberflächengeologie)** INSPIRE Geologic Units 1:1 million Austria (Surface Geology)
4244	GE	Geophysics	**INSPIRE Aerogeophysikalische Befliegungsgebiete (Kampagne) in Österreich** INSPIRE Flight areas of Airborne Geophysics (Campaign)

[3] https://inspire-regadmin.jrc.ec.europa.eu/dataspecification/FindYourScope.action

4253	GE	Geophysics	**INSPIRE Profillinien bodengeophysikalischer Messungen** INSPIRE profile lines of ground geophysical surveys

Table 2: INSPIRE Themes for GBA datasets

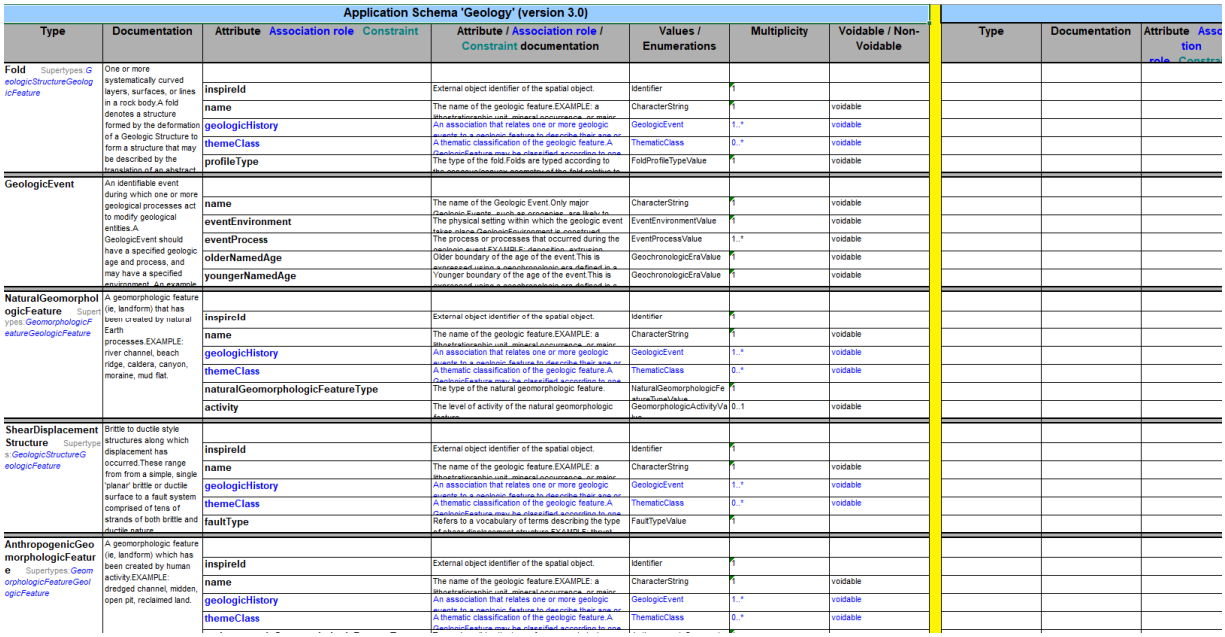
2.1.3.Map data

In order to map between the internal structures of the database in which the data is currently stored and the classes that make up the INSPIRE Application Schemas, one must first identify for which of these classes data is available. Preliminary information on this correspondence has already been collected in the previous step on identifying the correct INSPIRE Theme and Application Schema, and can be utilized for this purpose.

One of the many useful resources available from the European Commission's INSPIRE website for each Theme are mapping tables for each Application Schema. These tables provide a good starting point for aligning the requirements from INSPIRE with the available data. For each class or type specified, all attributes and associations are listed together with their types and descriptions at the left of the spreadsheet. At the right of the sheet there are columns for the provision of information on the corresponding tables and columns of the local data source.

A few caveats to the INSPIRE mapping tables:

- In order to allow for widespread use, the mapping tables are encoded in an open XML format. However, this makes it unclear how to access/open these files, Windows does not automatically open them with Excel. The trick lies in downloading the XML locally, then manually opening the file with Excel and storing in a current Excel format.
- The mapping tables have been automatically generated, and provide only the direct attributes of a type. In use, this causes issues, as some of the types are in practice nested within other types; in such cases, these nested blocks must first be copied to their container classes before performing the alignment with the local data source.
- The columns foreseen for description of the local data source are often not suited for the required information, thus it is often more efficient to replace the right side of the sheet with columns such as the following:
    - Table
    - Column
    - Constant (for provision of a constant value not taken from the DB)
    - Join information, for data not stemming from the main table



Application Schema 'Geology' (version 3.0)									
Type	Documentation	Attribute Association role Constraint	Attribute / Association role / Constraint documentation	Values / Enumerations	Multiplicity	Voidable / Non-Voidable	Type	Documentation	Attribute Association role Constra
Fold Supertypes:GeologicStructureGeologicFeature	One or more systematically curved layers, surfaces, or lines in a rock body.A fold denotes a structure formed by the deformation of a Geologic Structure to form a structure that may be described by the translation of an abstract								
		inspireId	External object identifier of the spatial object.	Identifier	1				
		name	The name of the geologic feature.EXAMPLE: a lithostratigraphic unit, mineral occurrence, or major	CharacterString	1	voidable			
		geologicHistory	An association that relates one or more geologic events to a geologic feature to describe their age or	GeologicEvent	1..*	voidable			
		themeClass	A thematic classification of the geologic feature.A GeologicFeature may be classified according to one	ThematicClass	0..*	voidable			
		profileType	The type of the fold.Folds are typed according to the preservation geometry of the fold relative to a	FoldProfileTypeValue	1	voidable			
GeologicEvent	An identifiable event during which one or more geological processes act to modify geological entities.A GeologicEvent should have a specified geologic age and process, and may have a specified environment. An example								
		name	The name of the Geologic Event.Only major GeologicEvents such as orogenies are likely to	CharacterString	1	voidable			
		eventEnvironment	The physical setting within which the geologic event took place.GeologicEnvironment is construed	EventEnvironmentValue	1	voidable			
		eventProcess	The process or processes that occurred during the geologic event.EXAMPLE: deposition, extrusive	EventProcessValue	1..*	voidable			
		olderNamedAge	Older boundary of the age of the event.This is expressed using a geochronologic era defined in a	GeochronologicEraValue	1	voidable			
		youngerNamedAge	Younger boundary of the age of the event.This is expressed using a geochronologic era defined in a	GeochronologicEraValue	1	voidable			
NaturalGeomorphologicFeature Supertypes:GeomorphologicFeatureGeologicFeature	A geomorphologic feature (ie, landform) that has been created by natural Earth processes.EXAMPLE: river channel, beach ridge, caldera, canyon, moraine, mud flat.								
		inspireId	External object identifier of the spatial object.	Identifier	1				
		name	The name of the geologic feature.EXAMPLE: a lithostratigraphic unit, mineral occurrence, or major	CharacterString	1	voidable			
		geologicHistory	An association that relates one or more geologic events to a geologic feature to describe their age or	GeologicEvent	1..*	voidable			
		themeClass	A thematic classification of the geologic feature.A GeologicFeature may be classified according to one	ThematicClass	0..*	voidable			
		naturalGeomorphologicFeatureType	The type of the natural geomorphologic feature.	NaturalGeomorphologicFeatureTypeValue	1				
		activity	The level of activity of the natural geomorphologic feature	GeomorphologicActivityValue	0..1	voidable			
ShearDisplacementStructure Supertypes:GeologicStructureGeologicFeature	Brittle to ductile style structures along which displacement has occurred.These range from from a simple, single 'planar' brittle or ductile surface to a fault system comprised of tens of strands of both brittle and ductile nature.								
		inspireId	External object identifier of the spatial object.	Identifier	1				
		name	The name of the geologic feature.EXAMPLE: a lithostratigraphic unit, mineral occurrence, or major	CharacterString	1	voidable			
		geologicHistory	An association that relates one or more geologic events to a geologic feature to describe their age or	GeologicEvent	1..*	voidable			
		themeClass	A thematic classification of the geologic feature.A GeologicFeature may be classified according to one	ThematicClass	0..*	voidable			
		faultType	Refers to a vocabulary of terms describing the type of shear displacement structure.EXAMPLE: thrust	FaultTypeValue	1				
AnthropogenicGeomorphologicFeature Supertypes:GeomorphologicFeatureGeologicFeature	A geomorphologic feature (ie, landform) which has been created by human activity.EXAMPLE: dredged channel, midden, open pit, reclaimed land.								
		inspireId	External object identifier of the spatial object.	Identifier	1				
		name	The name of the geologic feature.EXAMPLE: a lithostratigraphic unit, mineral occurrence, or major	CharacterString	1	voidable			
		geologicHistory	An association that relates one or more geologic events to a geologic feature to describe their age or	GeologicEvent	1..*	voidable			
		themeClass	A thematic classification of the geologic feature.A GeologicFeature may be classified according to one	ThematicClass	0..*	voidable			

Figure 1: INSPIRE Mapping Table for Geology

Please note that the separation of mapping vs. transformation is not as clear as described in this document. In reality, as the mapping process ensues analyzing and understanding the structure of the underlying data source while aligning this data to the structures foreseen in the relevant INSPIRE Application Schemas, a well-done alignment and mapping paves the way for the transformation process. How the existing database tables are to be joined must be indicated in the mapping as relevant context, this information can then be directly transferred to the data transformation process described in the next section.

2.1.4.Transform data

Once the data has been mapped to the applicable INSPIRE Application Specification, the data must be transformed as specified in the mapping. Various approaches and tools exist for this transformation process, e.g. in some cases, it may be useful to utilize ETL (Extract, Transform and Load) tools, that provide visual interactive interfaces to help streamline the transformation process. However, experience has often shown that those with the necessary database understanding required to correctly configure the transformation are also at ease with more technical approaches such as transformation directly in SQL, while those without such in-depth knowledge of SQL will require expert support in creating such transformations. Thus, in this document, we will describe the well-tested workflow of creating database views aligned with the structure of the classes of the applicable INSPIRE Application Specification. Once these are available, the final step of configuring the service endpoints for provision becomes very straightforward.

The type of view to be utilized depends both on the volatility of the underlying data as well as the size and complexity of the data source. With smaller sources or rapidly changing data, normal

database views are preferred. However, when dealing with large data sources and complex joins underlying the views, it may be advantageous to utilize materialized views; in such cases, care must be taken to update these materialized views when changes have been made to the underlying data sources.

When transforming the data, great care must be taken pertaining to the cardinalities of nested parts of the relevant INSPIRE Application Schema in comparison to the existing data. In many cases, multiple entries for a concept are foreseen by the Application Schema, but locally, there is no data available to support this multiplicity. In such cases, the nested part can be seen as a part of the encompassing type and integrated into the same database view that provides the data for the main type, while when higher multiplicities are warranted by the underlying data, these concepts must be provided in a separate view, joined to the main view by a foreign key relation.

We illustrate this principle based on the INSPIRE Base Type DocumentCitation datatype, that allows for multiple URLs to be provided via the “link” attribute as shown in Figure 2: INSPIRE DocumentCitation Type below.

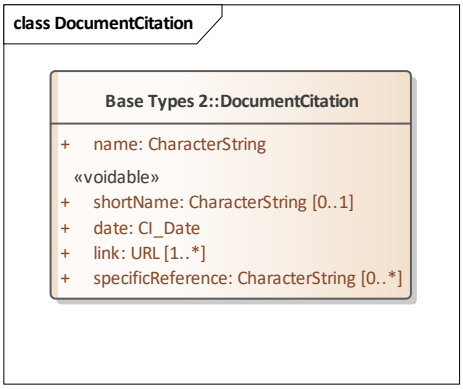


Figure 2: INSPIRE DocumentCitation Type

In the case that the data source only has one URL to be provided via the “link” attribute, the corresponding data can be integrated directly into the table or view providing the base attributes for the DocumentCitation datatype. However, if multiple URLs are to be provided here, an additional table or view must be created and linked to the main table via foreign key relation, to allow for provision of all URLs relevant to the specific DocumentCitation.

2.1.5.Configure Provision Service

Once the preceding steps have been completed, the actual configuration of the provision service becomes relatively simple; the core of this task consists of transferring the information previously collected to the structure of the GeoServer APP-Schema configuration files. Before performing this step, it is advised to create a simple example of the expected output XML. Dummy values can be provided as data in this example, but care should be taken that all XML Elements and Attributes are contained.

The GeoServer App-Schema configuration files consist of the following main sections:

- **Namespaces**: in this section, the namespace prefix and URI for all required namespaces must be provided. Inspection of the example output XML can be useful, as one often misses namespaces that have been included within used types. More details on the configuration of namespaces are provided in the Annex section “7.2.2 Namespaces”.
- **Data Stores**: in this section, database configuration information is provided to allow GeoServer to access the necessary database tables or views. More details on the configuration of data stores are provided in the Annex section “7.2.3 Database Configuration”.
- **Feature Mapping**: this section is the core of the App-Schema configuration, containing a dedicated section for each featureType to be provided. In the header block of this section, information on the featureType as well as the database table or view from which the data is to be taken is provided. In the following attribute mapping subsections, the database column containing the data is configured for each XML Element or Attribute. The individual Elements and Attributes are addressed by their XPATH in the configuration file; this can be extracted from the example XML file created before mapping. More details on the configuration of data stores are provided in further sections of “Annex C - GeoServer App Schema Configuration”.
  As App-Schema allows for on-the-fly data manipulation, e.g., concatenation of DB contents with static values, as often required to prepend URIs to codes stored in the DB, a decision must be reached as to where these transformations are to be performed:
  - During data transformation: Advantage is that all transformations are performed in one place, disadvantage is that many long strings must be stored.
  - During feature mapping: while saving on DB storage, this option can cause issues with subsequent queries.

A detailed description of the GeoServer App-Schema configuration process including examples is described in “Annex C - GeoServer App Schema Configuration”.

2.2. Profiling Alternatives

While the process described above is well suited for provision of data in accordance with strict schema requirements, a related problem often encountered by data providers is the provision of diverse alternative profiles, often required by specific communities or projects.

GeoServer has recently introduced an interesting approach for the provision of such alternative profiles in its Features-Templating Extension[4], utilizing a simple “What You See Is What You Get” approach. You start with a static example (the template) of how you envision the output data to be formatted; for all dynamic portions (those that change per feature), you provide the location of

[4] https://docs.geoserver.org/latest/en/user/community/features-templating/index.html

this piece of information in an existing endpoint (utilizing the XPATH in the original structure) within the template, GeoServer takes care of the rest.

This templating functionality is now available for both WFS and OGC API – Features, and supports the following output formats:

- GeoJSON
- GML
- JSON-LD
- HTML

2.3. Required Tools & Access Requirements

While the technical issues detailed above pose significant challenges, often the greatest hurdle to be faced in the data provision process pertains to more administrative issues, foremost access to the relevant data sources as well as the availability of required tools.

Pertaining to data access, in many cases it proves useful to create a duplicate database where one can perform all steps required for data transformation without requiring access rights to the production system or impacting day-to-day work. Once all steps have been performed, it is relatively simple to transfer the database view created on the local copy to the production system, to prepare for the configuration of provision services. Alternatively, read-only access can be granted for the data, but this often precludes the creation of views, and thus hindering the data provision process.

In order to configure GeoServer, admin rights for this server must be available. While most functionalities are available from the GeoServer GUI, direct access to the workspace directories is very valuable when troubleshooting issues; thus, options for accessing these directories should be possible.

In addition to access to the data source and GeoServer, several types of tools have proven to be most useful if not essential for the configuration of GeoServer configurations. The SW recommendations provided are based on experience, but should not preclude usage of alternative SW solutions. The toolset recommended for GeoServer configuration consists of the following:

- Database Tool: some form of database access and administration functionality is essential for this work as one must be able to read the DB, and also in many cases create views. While all this can be done via SQL, and can also be performed via command line tools such as PSQL, DB Tools offering a full GUI are most useful. Postgres provides PGAdmin

for this purpose, but this SW has various issues. At present, we find DBeaver[5] to be the most useful open-source tool available.

- XML Editor: As GeoServer App-Schema configuration relies on XPATHs to indicate where in the resulting output data the data from a specific DB column is to be provided, having an XML example of the desired target data structure is most valuable. Good XML editors simplify this task, as they analyze the desired XML Schema and can make suggestions for required content. In addition, the validation functionality assures that the example of the target format is syntactically correct; the same validation should also be applied to examples of the output data provide by GeoServer services. Commonly used XML Editors are XMLSpy[6] and Oxygen[7].
- Robust text editor: While a dedicated XML editor is essential for creating valid GeoServer services, an additional robust text editor is often quite useful, as this allows for faster viewing and editing of XML files. The disadvantages posed by such an editor not doing regular XML validation are made up for by having immediate access to the file regardless of syntactic correctness. Notepad++[8] is a useful open-source tool for such applications, also providing support for very large files.
- GIS Tool: while syntactic validation as performed by XML Editors as well as validation by the INSPIRE validation tools are essential, the final proof of correct mapping usually comes by viewing the resulting output data via a GIS tool. This step allows for simple heuristic checks on the data content, while assuring that the geolocation is provided correctly. QGIS[9] is a commonly used open-source tool, that also allows for interactive use of complex features provided by a WFS2 endpoint.

2.4. Architecture Considerations

In this section, we provide some architecture considerations that have proven useful. However, these are provided without a full analysis of the current data landscape at GBA, and should thus be seen as general suggestions, that must be correctly applied within the explicit context of the GBA servers.

One simple approach for assuring that valuable data cannot be corrupted by external actors when making it available for data provision is to only serve data from a mirror instance of the database. In addition, read-only access should be configured for the database user utilized by GeoServer. By positioning both this mirror database and the GeoServer instance in the DMZ, and configuring updates to be performed by push from the internal DB, the essential infrastructure is kept safe,

[5] https://dbeaver.io/
[6] https://www.altova.com/xmlspy-xml-editor
[7] https://www.oxygenxml.com/
[8] https://notepad-plus-plus.org/
[9] https://www.qgis.org/en/site/?

while the lightweight configuration positioned in the DMZ can be rapidly reloaded in case of cyberattack.

Such a split between internal operational systems and externalized hosting can also impact update and versioning. As many database systems do not include fine-grained versioning of individual objects, one can advantageously utilize this split of data storage for a pragmatic versioning approach based on the update cycles of the externalized provision DB: Each time one does an update of the provision DB, ALL objects in the update are assigned a new version. While this will create dummy version updates on many objects, it is exceedingly simple and foolproof. If these versioned objects are also stored beyond their lifespan, they can easily be utilized for the creation of temporal snapshots.

3. Vocabularies and Semantics

While INSPIRE has led to a great deal of progress pertaining to the provision of harmonized spatial data across Europe, true data harmonization requires more than standardized data models. In order to integrate data from disparate sources, it is essential that they reference the same terms for the same concepts. As long as all data being utilized comes from the same organization, consistent concepts are usually utilized; however, especially in domains such as geology, different institutions have their own classification traditions, leading to diverse and often overlapping concepts being utilized dependent on data source. This issue becomes increasingly pressing pertaining to cross-border data sharing as linguistic issues arise in addition to local concepts being utilized to described individual data objects. While simple concepts can be aligned with the use of modern translation technology, letting us set “water” as equivalent to “wasser” and “woda”, such approaches usually fail when dealing with specific domain vocabularies.

Standardized referenceable vocabularies can provide help to all aspects of this dilemma. If different institutions utilize common vocabularies within the description of their data, merging data across sources becomes far easier, as one needn’t first align basic concepts. As most widely used vocabularies have been translated, data providers across Europe can utilize concepts in their native language, with the assurance that data users will have access to all translations pertaining to this concept.

While flat code-lists providing sets of concepts are a good start, the true power of vocabularies comes to play when semantic constructs are utilized. At the simplest level we see thesauri, that integrate simple relations such as “broader”, “narrower”, or “related” between concepts to allow for the creation of taxonomies. Based on such simple hierarchical structures, additional relations can be added as required, whereby there is a continuum between thesauri and ontologies, with no clear line as to where richly structured thesauri end and ontologies begin. These relations can then be leveraged during data discovery and use, e.g., during search, where one can specify a higher-level term including all children in order to identify all data dealing with a broad topic area.

Recent work organized under the auspices of CODATA[10] and the DDI Alliance[11] has analyzed how the FAIR Principles can be applied to vocabularies[12]. In this work, they have defined Ten Simple Rules for making a vocabulary FAIR as follows:

- Rule 1. Verify that the legacy-vocabulary license allows repurposing
- Rule 2. Determine the governance arrangements and custodian responsible for the legacy vocabulary
- Rule 3. Check minimal term definition completeness
- Rule 4. Select a domain and service for the web identifiers
- Rule 5. Design an identifier scheme and pattern
- Rule 6. Create a semantic-standards based vocabulary - Interoperability

[10] https://codata.org
[11] https://ddialliance.org/
[12] https://arxiv.org/abs/2012.02325

- Rule 7. Add rich metadata - Reusability
- Rule 8. Register the vocabulary - Findability
- Rule 9. Make the IRIs resolve - Accessibility
- Rule 10. Implement a process for maintaining the FAIR vocabulary

These rules should be taken into account when exposing one’s own concepts as vocabularies, in order to ensure a durable robust system that will assure that data remains interpretable over time. For details, please see the cited publication.

4. Identifiers and Versioning

4.1. Identifier Management

Clear stable identifiers are essential for the consistent reuse of data, especially when repeatedly accessing the same data source over time. While all data systems utilize some form of identification, in many cases this is just a local identifier that while unique within a specific dataset or database, cannot be directly utilized for identification within a wider scope. Additionally, these local identifiers are often automatically assigned by the database itself using a serial identifier scheme; when such a database is backed up and reloaded, all automatically generated identifiers are reassigned, and traceability to older versions of this data source are lost. Several mechanisms can be utilized to ameliorate this issue.

A first essential step is to reduce dependence on automatically assigned identifiers. While such automatically assigned identifiers columns are commonly used in databases, these values should NOT be utilized for external identification. For the case where these automatically assigned identifiers have become the de-facto identifier for the objects being managed, it is advised to create an additional column for the persistent identifier and copy the values from the automatically assigned identifiers to this column. Once this has been done, the persistent identifier column will remain aligned, even when the data is reloaded, thus modifying the automatically assigned identifiers.

Additional issues ensue from duplicate identifiers across datasets. These often occur when different departments administrate their data separately; while the identifiers within one dataset may be consistent, these identifiers are no longer unique when the datasets are merged, leading to all kinds of issues and inconsistencies. A simple pragmatic solution to this dilemma consists of assigning unique internal namespaces at the necessary level of granularity, e.g., per dataset or department. This internal namespace can then be concatenated with the internal identifier of the object, utilizing the namespace as a prefix, and thus assuring unique identifiers across all digital objects within all datasets. Care should be taken to assure that this prefix as well as the internal identifiers already utilized are formatted in line with identifier requirements ensuing from GML. While the gml:id is slowly losing significance as other access modalities progress, at present, the following requirements must be met:

> gml:id cannot start with a number. It must be a letter or underscore “_”, after this characters may be letters, numbers or one of “_”, “-“, “.”

4.2. Resolvable URLs

Resolvable URLs are an essential aspect of any distributed SDI, as individual features must be clearly referenceable and retrievable. This pertains to both direct access to a feature as well as to providing relevant interlinkages between features, supplying essential contextual information. While it is conceivable to utilize the service URLs made available via the Web Feature Service

(WFS), these are complex, fragile and prone to change over time. URL rewriting is useful in those cases where one wishes to provide a compact and usable URL in place of a more complex service URL.

Example:

In order to specify access to an explicit feature utilizing the service URL, one would need to provide a URL as follows:

http://service.datacove.eu/geoserver/tn-a/ows?service=WFS&version=2.0&request=GetFeature&typeNames=tn-a:RunwayArea&outputFormat=gml32&FeatureId=1-2017-11-10

Apart from being long and messy, this type of URL is prone to change over time as different technologies and versions are utilized for provision. While utilization of OGC API – Features (OAPIF) will create a slightly less messy URL, direct utilization of the OAPIF URL as an identifier as follow is still suboptimal:

https://service.datacove.eu/geoserver/ogc/features/collections/tn-a:RunwayArea/items/1-2017-11-10

Far nicer would be a URL in the following form:

http://guid.datacove.eu/tn.RunwayArea/1-2017-11-10

In this section, in addition to explaining the theory of URL rewriting and its utilization in the provision of complex data sources, we will also provide all information required to implement these concepts, both pertaining to the necessary rewriting to be performed via the web server providing basic HTTP functionality as well as the details of GeoServer Feature Chaining, to be configured via the GeoServer App Schema extension. We will also show how rewriting can help to support efficient versioning approaches.

4.3. Relevance to INSPIRE

Within INSPIRE, the associations between featureTypes are provided via the xlink:href attribute. Utilizing a clean simple URL for this purpose not only provides cleaner more readable XML, it also gives duration to the reference, as technology can change over time, while this URL can continue to be resolved to the currently utilized technology. Thus, this URL can also be utilized as a persistent Globally Unique Identifier (GUID), not only providing a unique name for an object, but also the mechanisms for direct access through HTTP resolution.

This approach has already been adopted as one of two options within the Austrian INSPIRE metadata recommendations [INSPIRE-AT_Metadatenerfassungsleitfaden_v2-2.pdf] for the identification of a metadata record, whereby the following syntax has been recommended:

Basis-URL/Geodatenstelle/Ressourcenbezeichner

Examples:

> **https://data.inspire.gv.at/0002/37d564f9-5d63-4760-aae6-29d3f98ee1b4**
> *Version mit UUID als Ressourcenbezeichner*
>
> https://data.inspire.gv.at/0019/HAZARD_AREA_HQ30
> *Version mit sprechendem Ressourcenbezeichner*

This pattern is continued in the data recommendations, where the HTTP option is specified as follows:

> **Inspire Id:**
>
> > **Namespace:** Metadata URL/[Theme].[FeatureType]
> >
> > **LocalId:** Local ID of the feature, must be unique within the FeatureType
>
> **Resolvable URI:**
>
> > Namespace/LocalId
>
> **Example:**
>
> > https://data.inspire.gv.at/0019/HAZARD_AREA_HQ30/nz.HazardArea/K2494213
>
> **Theme:** 2 character abbreviation of INSPIRE Theme
>
> **FeatureType:** class name of the feature being provided

4.4. Clean Base URL

In order to assure that the rewritten URLs are persistent, it is essential to assure that a clean base URL is utilized. A common mistake is taking one's agency's URL in some form, only to find one rebranded the next day, URL gone. As costs for a domain are quite low, it would be sensible to acquire a dedicated URL for the long-term provision of rewritten URLs.

What URLs are finally chosen for data provision are often guided more by administrative concerns than technical feasibility. However, rewriting gives us a great degree of flexibility between what URLs are utilized for data provision vs. those serving for data identification. Multilevel redirects are easily possible, allowing for continued usage of the national identifier scheme as an initial access point. Undue burden on the central coordinating organization can be reduced by defining a cascade of redirections, with the first national level just distributing requests to the relevant data

provider, relying on a second level of redirects within the data providers infrastructure to address individual data objects.

4.5. Versioning

4.5.1. Versioning Considerations

An eternal issue in data provision pertains to versioning. As we move away from the provision of monolithic datasets towards linked-data approaches this challenge becomes increasingly complex, throwing up diverse issues such as

1. What degree of change constitutes a change of version?
2. Which version do linked objects reference after updates?
3. How to enable access to specific versions, e.g., temporal snapshots?

Pertaining to the first point, while one is often tempted to do minor changes without versioning, even fixing small spelling mistakes can impact users in unexpected ways if they are not aware of these changes. Thus, we recommend changing the version whenever any of the data content of the feature in question have been modified in any manner. In order to enable this, we must first resolve the further questions.

On the question of versions and links, it is usually sufficient to assure that all links resolve to the most current version of the feature, if not specified differently. This will assure that general data access to the current version always returns current data, also when providing linked features. However, this leaves open which version an older version of a feature should reference; for this we first need to answer the question of how to access temporal snapshots of the data.

Most general use cases require access to the most current dataset; when accessing such data, the user will assume that all features returned will be the most current version available. In other cases, the version that was available at a specific point in time will be required; for such temporal snapshots, care must be taken that all features returned are of the same consistent version.

INSPIRE provides support for versioning both through the inclusion of the optional version attribute within the InspireId as well as via the beginLifeSpan and endLifeSpan attributes common to most INSPIRE featureTypes. However, there is no clear guidance on how to effectively utilize these facets to enable explicit access to versions via the Web Services or APIs used for provision. The following recommendations have been gleaned from diverse deployments across various topics throughout Europe.

4.5.2. Inspire ID

As mentioned, the datatype defined for the inspireId provided for the identification of featureTypes in INSPIRE provides a version attribute, but places no requirements as to if and how this attribute

is utilized. While an intuitive approach for this attribute is a simple sequential number scheme, an interesting alternative is to utilize a timestamp. If this timestamp corresponds to the endLifeSpan concept of the feature, it becomes simple to identify the current version, e.g., the one without a version.

4.5.3. Database Considerations

In order to enable versioning in data provision, the underlying database must support some sort of versioning. The simplest solution is to utilize the beginLifeSpan and endLifeSpan concepts from the INSPIRE data models, initializing the beginLifeSpan column when a new version is created and adding a value to endLifeSpan when a new version has been created. As this may cause issues with legacy applications, SQL functions can be created and triggers set that both automate setting these LifeSpan values as well as automate access to the most current version by filtering out all entries with a value for endLifeSpan. For read-only access, such as required by the dedicated endpoint for the current version described in the section below, a (materialized) view can be utilized.

4.5.4. Endpoints

As most users wish to access to the most current version, a dedicated endpoint that only serves the latest versions of all features should be provided. When querying this endpoint, the user is not even aware of the fact that previous versions exist; they receive what for them is the definitive version of the features they have requested.

For those users requiring access to specific versions and temporal snapshots, an additional endpoint should be configured, providing access to all versions. The reason for providing this as a separate endpoint is that general users would not want to receive all versions of an object when performing a spatial query, and may not be aware of more complex querying mechanisms available. Stored queries should be foreseen for this endpoint, allowing users to provide a date to be applied a filter pertaining to the beginLifeSpan and endLifeSpan attributes, enabling access to the data available at this point in time.

4.5.5. Rewriting and versioning

The approach described above on rewriting can also be extended to support versioning. When the URI is formed with only the featureType and localId as follows, the current version is returned:

http://guid.datacove.eu/tn.RunwayArea/1-2017-11-10

Older versions can be explicitly addressed by appending the version to the end of the feature URI with a slash (/):

http://guid.datacove.eu/tn.RunwayArea/1-2017-11-10/2020-05-21

4.6. Backlink on Metadata

In order to allow a data user who has accessed a feature via the clean rewritten URL to also interrogate the service itself, access to the service URL should be provided. This allows the user to access other related features provided by this service, but not directly linked to the initial feature accessed; it also allows for filtering on these additional features.

A simple way of fulfilling this requirement is to provide a link to the metadata record of this feature within the gml:metadata element of the individual features. As the metadata record contains information on the service URL, additional features can be accessed in a fully automated and standardized manner.

5. Annex A – Resolvable Identifier Configuration

5.1. How to Configure URL Rewriting

For this type of rewriting, the web server must be configured with the proper rewriting rules. In the example, we have utilized Apache, but the same principles apply to all web servers. In this text, we utilize examples from AustroControl.

Under Apache, the rewrite rule has the following syntax:

RewriteRule *Pattern Substitution [flags]*

Whereby the arguments of the rule are defined as follows:

1. Pattern: which incoming URLs should be affected by the rule, described as a regular expression;
2. Substitution: where should the matching requests be sent;
3. [flags]: options affecting the rewritten request.

The Apache rewrite rule for the example above as applied to the domain guid.datacove.eu is as follows:

RewriteRule ^/tn-a.RunwayArea/(.*)$ http://service.datacove.eu:8080/geoserver/tn-a/ows?service=WFS&version=2.0&request=GetFeature&typeNames=tn-a:RunwayArea&featureID=$1 [L]

The text “RewriteRule” tells the Apache Server that it is a rewrite rule. This is followed by the Pattern, a regular expression containing the pattern for the incoming URL, whereby the groups in parenthesis () specify individual variables to be extracted from the incoming URL, and the symbol ^ represents the base URL of the service. Thus, the pattern ^/tn-a.RunwayArea/(.*)$ corresponds to all incoming URLs of the form guid.datacove.eu/tn-a.RunwayArea/XXX, whereby everything after the “tn-a.RunwayArea/”, in this example “XXX” is assigned to the first variable.

The Substitution provides the final target URL, whereby the variables extracted from the Pattern can be dynamically inserted. In our example, $1 provides the identifier of the GeographicalName feature provided after the “tn-a.RunwayArea/” in the clean URI. Thus, the simple URL shown above resolves to the service URL described.

5.2. GBA Identifiers and Links

In order to create a sustainable identifier scheme for the featureTypes to be served under the geological themes, we must first analyze the relevant identifiers. In addition, we must define which identifiers are utilized for the provision of a persistent URL-based GUID for referencing purposes.

5.2.1. Identifiers in INSPIRE

The following identifiers pertain to INSPIRE features:

- **gml:id:** cannot start with a number. It must be a letter or underscore "_", after this characters may be letters, numbers or one of "_", "-", "."
- **gml:identifier** (note: this is currently not utilized, but provided for completeness and future compatibility)
    - codespace: URI
    - id: string
- **net:inspireId**
    - namespace: URI, ideally related with the metadata URL
    - localId: string
    - version: string, ideally a timestamp

5.2.2. Identifiers in GBA

Due to the wide range of feature types hosted by GBA, originally maintained in various thematic departments, there is not one consistent identifier scheme that can be applied to all datasets falling under GBA auspices. For the scope of this document, we will illustrate the principles based on the Mineral Occurrence (MO) dataset; the same principles apply to all datasets. At this general level, we recommend joint administration of all dataset level concerns, e.g., creation and maintenance of a commonly accessible resource documenting:

- All metadata concepts pertaining to the dataset required for INSPIRE metadata;
- All information pertaining to the mapping of identifiers toward INSPIRE identifiers;
- Location of detailed thematic mapping tables;
- All necessary administrative and contact information.

Only by keeping all information pertaining to a dataset tightly coupled can robust data management and provision be assured.

Pertaining to the MO dataset, we can document the progression of the identifiers from the original thematic dataset over internally used views to the final tables utilized for the provision of INSPIRE data.

The following information is available locally to identify INSPIRE features:

- **<GBA_ID>**
  - **rst.IRIS.ID** (local GBA ID)
  - **IRIS_ID** in the provided views
  - **irisID** in the data provision tables
- **<featureType>**: <featureNS>.<featureType>
  - **<featureNS>:** in our example, this is 'mr-core'
  - **<featureType>:** the class name of the feature being served, in our example this is 'MineralOccurrence'
- **<GBA_metadata_URL>:** <MD_Base_URI>/<MD_UUID>
  - **<MD_Base_URI>:** Base URI for MD URL
    - Currently utilizing the national MD URLs https://data.inspire.gv.at/
    - Could alternatively utilize a dedicated URL also serving data, e.g., https://data.geologie.at/
  - **<MD_UUID>:** UUID of the Metadata Record d69f276f-24b4-4c16-aed7-349135921fa1
  - Leads to a metadata URL such as: https://data.inspire.gv.at/d69f276f-24b4-4c16-aed7-349135921fa1
- **<GBA_Version>:** e.g. timestamp as of which this version is valid

5.2.3.Identifier Mapping GBA -> INSPIRE?

InspireId

- **<namespace>** = <GBA_metadata_URL>/<featureNS>.<featureType>
- **<localId>** = <GBA_ID>
- **<version>** = < GBA_Version>

gml:id = AT.<MD_UUID>.<featureNS>.<featureType>.<GBA_id>

gml:identifier

- **<codespace>** = http://inspire.ec.europa.eu/ids
- **<identifier>** = <Resolvable URI>

<Resolvable URI> = <namespace>/<localId>

Example

Base Information:

- **<GBA_ID>**: MB00001
- **<featureType>:** nz-core:ObservedEvent
    - **<featureNS>:** nz-core
    - **<featureType>:** ObservedEvent
- **< GBA_metadata_URL>:** https://data.inspire.gv.at/d69f276f-24b4-4c16-aed7-349135921fa1
    - **<MD_Base_URI>:** https://data.inspire.gv.at
    - **<MD_UUID>:** d69f276f-24b4-4c16-aed7-349135921fa1

InspireId:

- **Namespace** = https://data.inspire.gv.at/d69f276f-24b4-4c16-aed7-349135921fa1/nz.ObservedEvent
    - **<MD_Base_URI>**: https://data.inspire.gv.at/
    - **<MD_UUID>**: d69f276f-24b4-4c16-aed7-349135921fa1
    - **<featureNS>**: nz
    - **<featureType>**: ObservedEvent
- **LocalId** = MB00001
    - **<GBA_ID>**: MB00001

gml:id:

- AT.d69f276f-24b4-4c16-aed7-349135921fa1.nz.ObservedEvent.MB00001

gml:identifier:

- **<codespace>** = http://inspire.ec.europa.eu/ids
- **<identifier>** = https://data.inspire.gv.at/d69f276f-24b4-4c16-aed7-349135921fa1/nz.ObservedEvent/MB00001

Resolvable URI:

https://data.inspire.gv.at/d69f276f-24b4-4c16-aed7-349135921fa1/nz.ObservedEvent/MB00001

Resolvable URI with version:

https://data.inspire.gv.at/d69f276f-24b4-4c16-aed7-349135921fa1/nz.ObservedEvent/MB00001/2018-11-21

5.2.4. Feature Metadata Link

In addition, the gml:metaDataProperty should be provided, whereby the xlink:href attribute should contain the Metadata URL.

5.3. Rewriting Rules

Due to the solidity of the metadata scheme described in the sections above, individual features can be accessed either via their inspireId or their gml:id, as shown in the following examples, both accessing the tn-a RunwayArea feature with id 1:

> https://sdigeot-free.austrocontrol.at/geoserver/tn-a/wfs?service=WFS&version=2.0.0&request=GetFeature&**STOREDQUERY_ID**=GetRunwayAreaInspireID2&localid=tn-a.RunwayArea/1&ns=https://sdigeo-free.austrocontrol.at/0012/6bed1778-d6bf-11e8-9f8b-f2801f1b9fd1/

> https://sdigeot-free.austrocontrol.at/geoserver/tn-a/wfs?service=WFS&version=2.0.0&request=GetFeature&typeNames=tn-a:RunwayArea&**featureID**=AT.0012.6bed1778-d6bf-11e8-9f8b-f2801f1b9fd1.tn-a.RunwayArea.1.2017-11-10

GBA Guidance

The URI for this feature should be as follows (base URI for austrocontrol to be adjusted):

> https://sdigeo-free.austrocontrol.at/0012/6bed1778-d6bf-11e8-9f8b-f2801f1b9fd1/tn-a.RunwayArea/1

Regex for rewriting:

> ^/0012/6bed1778-d6bf-11e8-9f8b-f2801f1b9fd1/tn-a.(.+)/(.+)$

Output via featureID:

> https://sdigeot-free.austrocontrol.at/geoserver/tn-a/wfs?service=WFS&version=2.0&request=GetFeature&typeNames=tn-a:AirRoute&**featureID**=AT.0012.6bed1778-d6bf-11e8-9f8b-f2801f1b9fd1.tn-a.**$1**.**$2**

Output via inspireId:

> https://sdigeot-free.austrocontrol.at/geoserver/tn-a/wfs?service=WFS&version=2.0.0&request=GetFeature&**STOREDQUERY_ID**=Get**$1**InspireID2&localid=tn-a.**$1**/**$2**&ns=https://sdigeo-free.austrocontrol.at/0012/6bed1778-d6bf-11e8-9f8b-f2801f1b9fd1/

5.3.1.Some examples taken from the Austro Control Project

Metadata for a feature:

```
https://inspire.austrocontrol.at/AT.0012.6bed1778-d6bf-11e8-9f8b-f2801f1b9fd1
```

Translates via rewriting into:

```
https://sdimdt-free.austrocontrol.at/geonetwork/srv/eng/csw-inspire?SERVICE=CSW&VERSION=2.0.2&REQUEST=GetRecordById&id=6bed1778-d6bf-11e8-9f8b-f2801f1b9fd1&resulttype=results&outputSchema=http://www.isotc211.org/2005/gmd&typenames=gmd:MD_Metadata&constraint=%3CFilter%20xmlns=%22http://www.opengis.net/ogc%22%20xmlns:gml=%22http://www.opengis.net/gml%22/%3E&constraintlanguage=FILTER&constraint_language_version=1.1.0
```

Requesting **all** features of one particular feature type:

```
https://inspire.austrocontrol.at/AT.0012.6bed1778-d6bf-11e8-9f8b-f2801f1b9fd1/tn-a.AerodromeNode
```

Translates via rewriting into:

```
https://sdigeo-free.austrocontrol.at/geoserver/tn-a/wfs?service=WFS&version=2.0.0&request=GetFeature&typeNames=tn-a:AerodromeNode&outputFormat=gml32&srsName=http://www.opengis.net/def/crs/EPSG/0/4258
```

Requesting **one** features of one particular feature type:

```
https://inspire.austrocontrol.at/AT.0012.6bed1778-d6bf-11e8-9f8b-f2801f1b9fd1/tn-a.AerodromeNode.42
```

Translates via rewriting into:

https://sdigeo-free.austrocontrol.at/geoserver/tn-a/wfs?service=WFS&version=2.0.0&request=GetFeature&typeNames=tn-a:AerodromeNode&featureId=AT.0012.6bed1778-d6bf-11e8-9f8b-f2801f1b9fd1.tn-a.AerodromeNode.42&outputFormat=gml32&srsName=http://www.opengis.net/def/crs/EPSG/0/4258

5.4. Stored Queries for Access via INSPIRE ID

My favorite Stored Query Resource: https://www.weichand.de/2012/04/22/wfs-2-0-stored-queries-beispiele/

5.4.1.Stored Query

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:CreateStoredQuery xmlns="http://www.opengis.net/wfs/2.0"
xmlns:wfs="http://www.opengis.net/wfs/2.0" xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:fes="http://www.opengis.net/fes/2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:nz-
core="http://inspire.ec.europa.eu/schemas/nz-core/4.0" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:base="http://inspire.ec.europa.eu/schemas/base/3.3"
xmlns:ns1="http://www.opengis.net/ows/1.1" service="WFS" version="2.0.0"
xsi:schemaLocation="http://www.opengis.net/wfs/2.0 http://schemas.opengis.net/wfs/2.0/wfs.xsd
http://www.opengis.net/gml/3.2 http://schemas.opengis.net/gml/3.2.1/gml.xsd">
 <wfs:StoredQueryDefinition id="GetObservedEventInspireID">
  <!-- Definition Template-Parameter -->
  <wfs:Parameter name="localid" type="xsd:string"/>
  <wfs:Parameter name="ns" type="xsd:string"/>
  <wfs:QueryExpressionText returnFeatureTypes="nz-core:ObservedEvent"
language="urn:ogc:def:queryLanguage:OGC-WFS::WFS_QueryExpression">
   <wfs:Query typeNames="nz-core:ObservedEvent">
    <fes:Filter>
     <fes:And>
      <fes:PropertyIsEqualTo>
       <fes:ValueReference>nz-core:ObservedEvent/nz-
core:inspireId/base:Identifier/base:localId</fes:ValueReference>
       <fes:Literal>${localid}</fes:Literal>
      </fes:PropertyIsEqualTo>
      <fes:PropertyIsEqualTo>
       <fes:ValueReference>nz-core:ObservedEvent/nz-
core:inspireId/base:Identifier/base:namespace</fes:ValueReference>
       <fes:Literal>${ns}</fes:Literal>
```

```
          </fes:PropertyIsEqualTo>
        </fes:And>
      </fes:Filter>
    </wfs:Query>
  </wfs:QueryExpressionText>
 </wfs:StoredQueryDefinition>
</wfs:CreateStoredQuery>
```

5.4.2.Request

http://localhost:8080/geoserver/ows?service=WFS&version=2.0.0&request=GetFeature&STOREDQUERY_ID=GetObservedEventInspireID&localid=MB00001&ns=https://data.inspire.gv.at/d69f276f-24b4-4c16-aed7-349135921fa1/nz.ObservedEvent

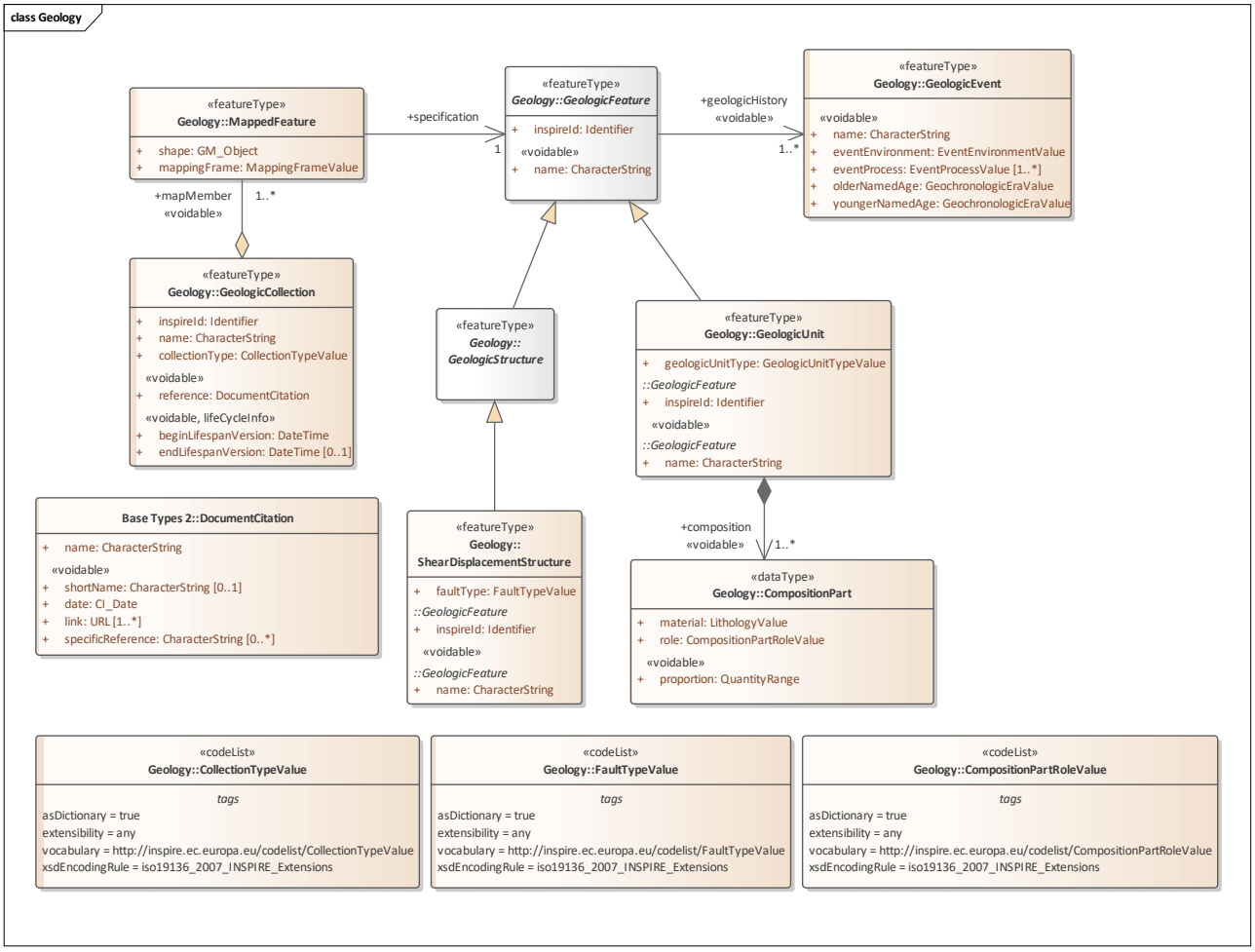
6. Annex B – INSPIRE Classes with Context

6.1. Overview

One of the difficulties in interpreting the geology relevant models stems from the complex derivation hierarchy spanning multiple models. Many of the base classes from HydroGeology and GeoPhysics have been derived from classes defined in the Geology Model, while others rely on classes provided by the O&M data model. In other cases, while all classes are contained within the same model, it is still difficult to traverse the entire derivation hierarchy. Thus, the following section provides a simplified view of the target classes with all derived attributes displayed.

6.2. Geology

6.2.1.Overview



6.2.2.GeologicFeature

Definition: A conceptual geological feature that is hypothesized to exist coherently in the world.

Description: This corresponds with a "legend item" from a traditional geologic map. While the bounding coordinates of a Geologic Feature may be described, its shape is not.

The implemented Geologic Feature instance acts as the "description package"

6.2.3.GeologicStructure

Definition: A configuration of matter in the Earth based on describable inhomogeneity, pattern, or fracture in an earth material.

Description: The identity of a GeologicStructure is independent of the material that is the substrate for the structure.

6.2.4.Mapped Feature

Definition: A spatial representation of a GeologicFeature.

Description: A MappedFeature is part of a geological interpretation.

It provides a link between a notional feature (description package) and one spatial representation of it, or part of it (exposures, surface traces and intercepts, etc) which forms the specific bounded occurrence, such as an outcrop or map polygon.

6.2.5.GeologicCollection

Definition: A collection of geological or geophysical objects.

Description: Geologic objects are commonly grouped into collections such as geological maps, thematic maps, or the required input to a geological model.

6.2.6.GeologicEvent

Definition: An identifiable event during which one or more geological processes act to modify geological entities.

Description: A GeologicEvent should have a specified geologic age and process, and may have a specified environment. An example might be a cratonic uplift event during which erosion, sedimentation, and volcanism all take place. A GeologicEvent age can represent an instant in time or an interval of time.

6.2.7.ShearDisplacementStructure

Definition: Brittle to ductile style structures along which displacement has occurred.

Description: These range from from a simple, single 'planar' brittle or ductile surface to a fault system comprised of tens of strands of both brittle and ductile nature.

6.2.8.GeologicUnit

Definition: A volume of rock with distinct characteristics.

Description: Includes both formal units (i.e. formally adopted and named in an official lexicon) and informal units (i.e. named but not promoted to the lexicon) and unnamed units (i.e. recognisable and described and delineable in the field but not otherwise formalised).

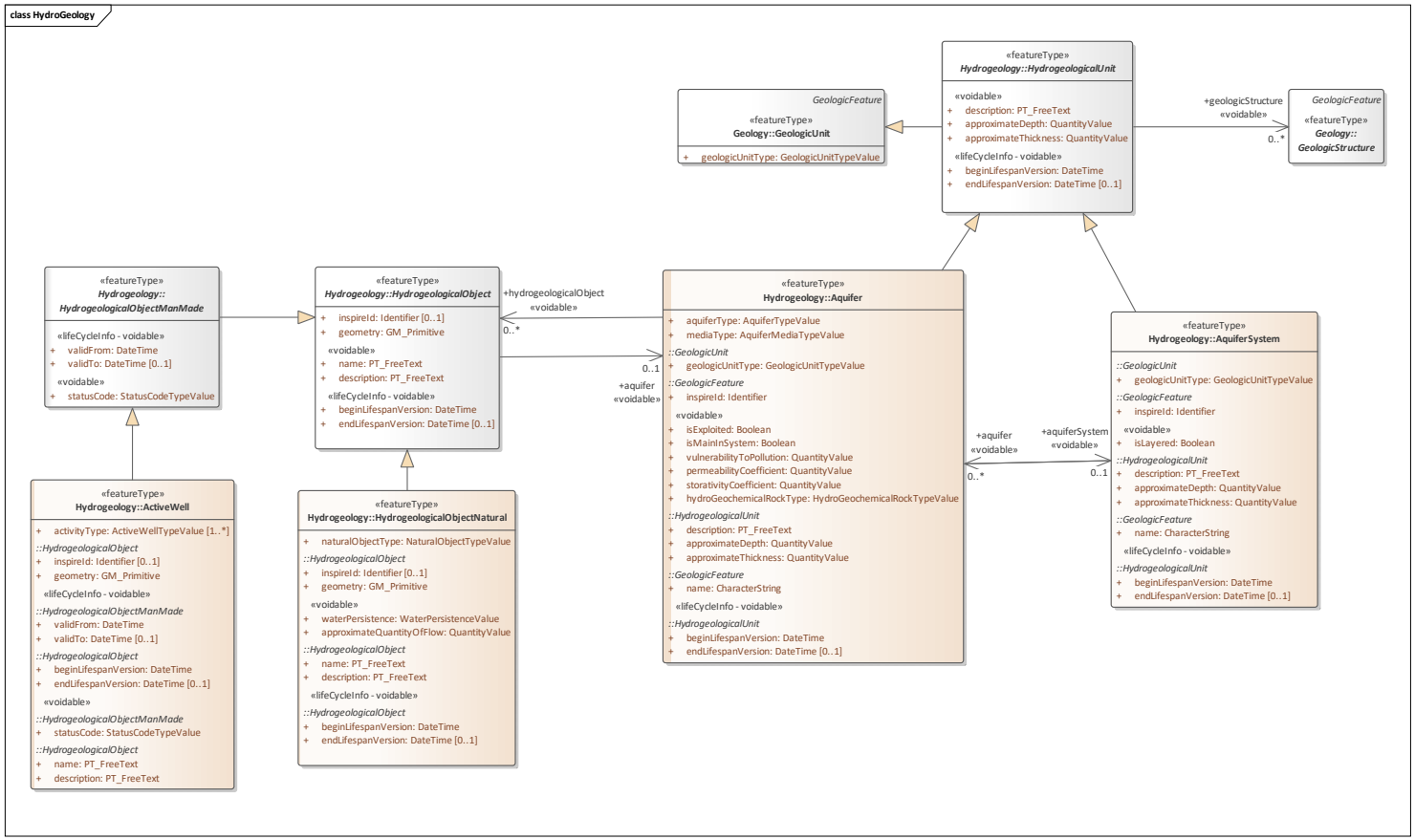
Spatial properties are only available through association with a MappedFeature.

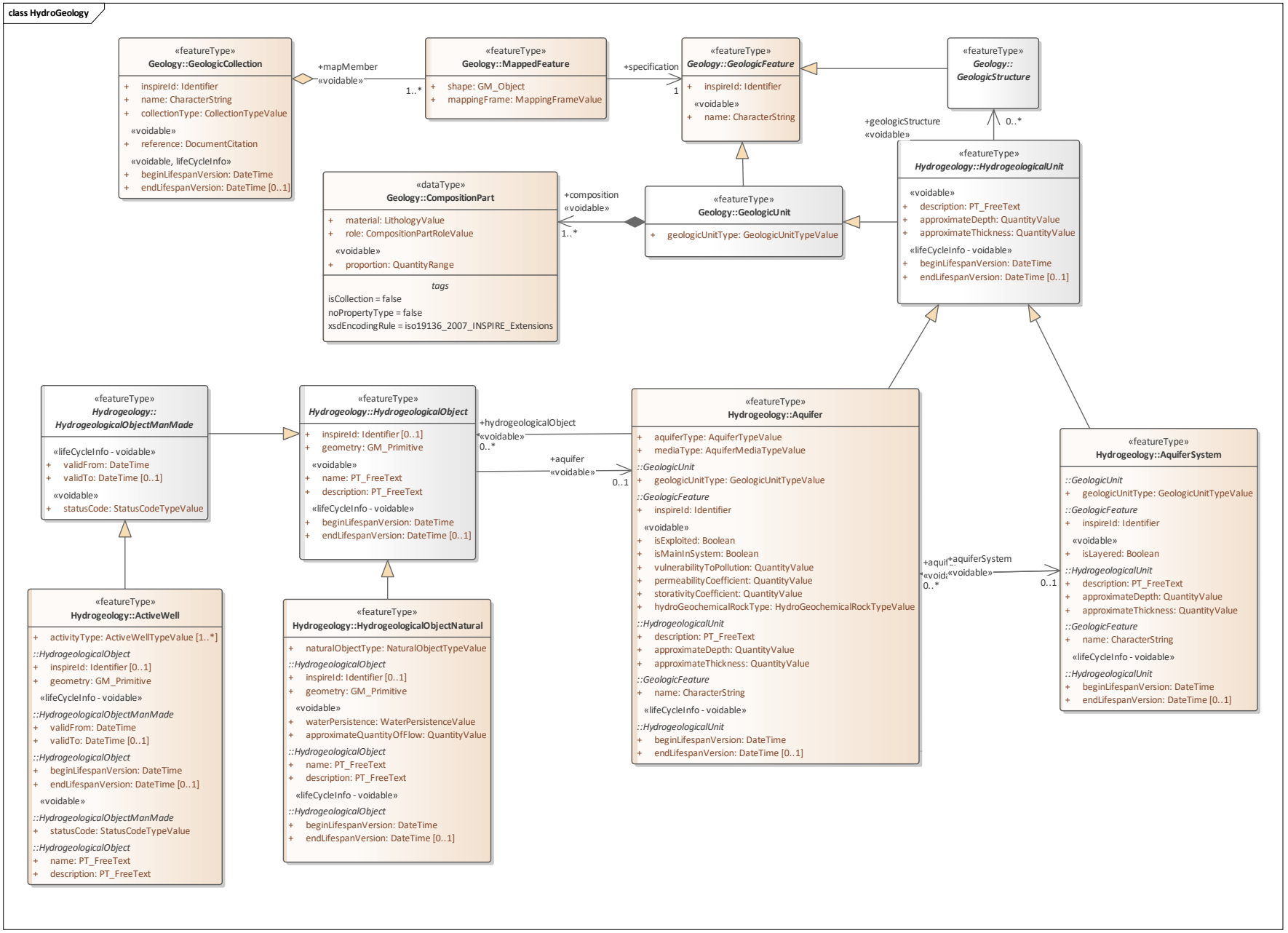
6.2.9.CompositionPart

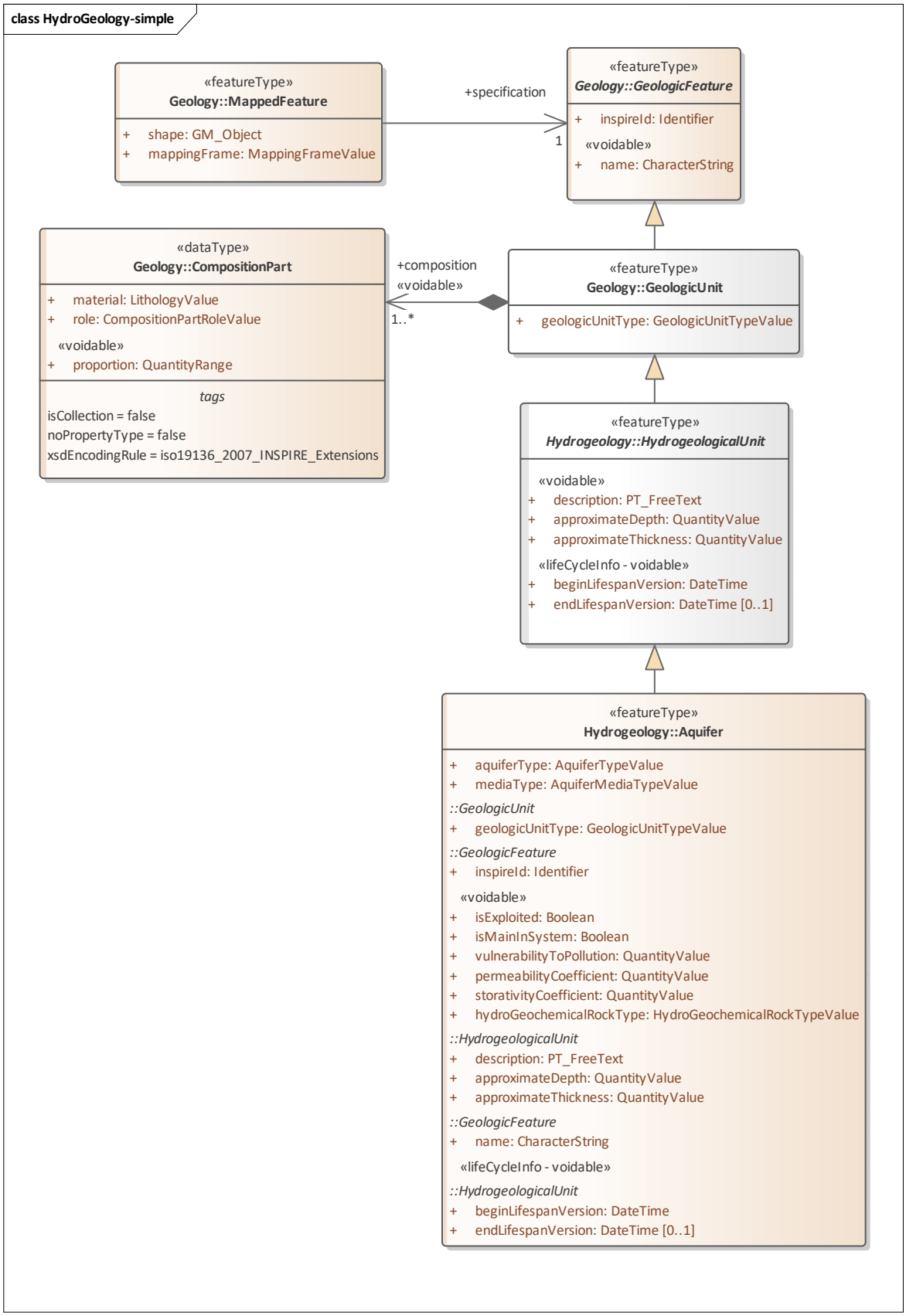
Definition: The composition of a geologic unit in terms of lithological constituents.

6.3. HydroGeology

6.3.1.Overview




class HydroGeology
«featureType»
Geology::GeologicCollection
+ inspireId: Identifier
+ name: CharacterString
+ collectionType: CollectionTypeValue
«voidable»
+ reference: DocumentCitation
«voidable, lifeCycleInfo»
+ beginLifespanVersion: DateTime
+ endLifespanVersion: DateTime [0..1]
+mapMember
«voidable»
1..*
«featureType»
Geology::MappedFeature
+ shape: GM_Object
+ mappingFrame: MappingFrameValue
+specification
1
«featureType»
Geology::GeologicFeature
+ inspireId: Identifier
«voidable»
+ name: CharacterString
«featureType»
Geology::
GeologicStructure
+geologicStructure
«voidable»
0..*
«featureType»
Hydrogeology::HydrogeologicalUnit
«voidable»
+ description: PT_FreeText
+ approximateDepth: QuantityValue
+ approximateThickness: QuantityValue
«lifeCycleInfo - voidable»
+ beginLifespanVersion: DateTime
+ endLifespanVersion: DateTime [0..1]
«dataType»
Geology::CompositionPart
+ material: LithologyValue
+ role: CompositionPartRoleValue
«voidable»
+ proportion: QuantityRange
tags
isCollection = false
noPropertyType = false
xsdEncodingRule = iso19136_2007_INSPIRE_Extensions
+composition
«voidable»
1..*
«featureType»
Geology::GeologicUnit
+ geologicUnitType: GeologicUnitTypeValue
«featureType»
Hydrogeology::
HydrogeologicalObjectManMade
«lifeCycleInfo - voidable»
+ validFrom: DateTime
+ validTo: DateTime [0..1]
«voidable»
+ statusCode: StatusCodeTypeValue
«featureType»
Hydrogeology::HydrogeologicalObject
+ inspireId: Identifier [0..1]
+ geometry: GM_Primitive
«voidable»
+ name: PT_FreeText
+ description: PT_FreeText
«lifeCycleInfo - voidable»
+ beginLifespanVersion: DateTime
+ endLifespanVersion: DateTime [0..1]
+hydrogeologicalObject
«voidable»
0..*
+aquifer
«voidable»
0..1
«featureType»
Hydrogeology::Aquifer
+ aquiferType: AquiferTypeValue
+ mediaType: AquiferMediaTypeValue
::GeologicUnit
+ geologicUnitType: GeologicUnitTypeValue
::GeologicFeature
+ inspireId: Identifier
«voidable»
+ isExploited: Boolean
+ isMainInSystem: Boolean
+ vulnerabilityToPollution: QuantityValue
+ permeabilityCoefficient: QuantityValue
+ storativityCoefficient: QuantityValue
+ hydroGeochemicalRockType: HydroGeochemicalRockTypeValue
::HydrogeologicalUnit
+ description: PT_FreeText
+ approximateDepth: QuantityValue
+ approximateThickness: QuantityValue
::GeologicFeature
+ name: CharacterString
«lifeCycleInfo - voidable»
::HydrogeologicalUnit
+ beginLifespanVersion: DateTime
+ endLifespanVersion: DateTime [0..1]
+aquifer
«voidable»
0..*
+aquiferSystem
«voidable»
0..1
«featureType»
Hydrogeology::AquiferSystem
::GeologicUnit
+ geologicUnitType: GeologicUnitTypeValue
::GeologicFeature
+ inspireId: Identifier
«voidable»
+ isLayered: Boolean
::HydrogeologicalUnit
+ description: PT_FreeText
+ approximateDepth: QuantityValue
+ approximateThickness: QuantityValue
::GeologicFeature
+ name: CharacterString
«lifeCycleInfo - voidable»
::HydrogeologicalUnit
+ beginLifespanVersion: DateTime
+ endLifespanVersion: DateTime [0..1]
«featureType»
Hydrogeology::ActiveWell
+ activityType: ActiveWellTypeValue [1..*]
::HydrogeologicalObject
+ inspireId: Identifier [0..1]
+ geometry: GM_Primitive
«lifeCycleInfo - voidable»
::HydrogeologicalObjectManMade
+ validFrom: DateTime
+ validTo: DateTime [0..1]
::HydrogeologicalObject
+ beginLifespanVersion: DateTime
+ endLifespanVersion: DateTime [0..1]
«voidable»
::HydrogeologicalObjectManMade
+ statusCode: StatusCodeTypeValue
::HydrogeologicalObject
+ name: PT_FreeText
+ description: PT_FreeText
«featureType»
Hydrogeology::HydrogeologicalObjectNatural
+ naturalObjectType: NaturalObjectTypeValue
::HydrogeologicalObject
+ inspireId: Identifier [0..1]
+ geometry: GM_Primitive
«voidable»
+ waterPersistence: WaterPersistenceValue
+ approximateQuantityOfFlow: QuantityValue
::HydrogeologicalObject
+ name: PT_FreeText
+ description: PT_FreeText
«lifeCycleInfo - voidable»
::HydrogeologicalObject
+ beginLifespanVersion: DateTime
+ endLifespanVersion: DateTime [0..1]


class HydroGeology-simple
«featureType»
Geology::MappedFeature
+ shape: GM_Object
+ mappingFrame: MappingFrameValue
+specification
1
«featureType»
Geology::GeologicFeature
+ inspireId: Identifier
«voidable»
+ name: CharacterString
«dataType»
Geology::CompositionPart
+ material: LithologyValue
+ role: CompositionPartRoleValue
«voidable»
+ proportion: QuantityRange
tags
isCollection = false
noPropertyType = false
xsdEncodingRule = iso19136_2007_INSPIRE_Extensions
+composition
«voidable»
1..*
«featureType»
Geology::GeologicUnit
+ geologicUnitType: GeologicUnitTypeValue
«featureType»
Hydrogeology::HydrogeologicalUnit
«voidable»
+ description: PT_FreeText
+ approximateDepth: QuantityValue
+ approximateThickness: QuantityValue
«lifeCycleInfo - voidable»
+ beginLifespanVersion: DateTime
+ endLifespanVersion: DateTime [0..1]
«featureType»
Hydrogeology::Aquifer
+ aquiferType: AquiferTypeValue
+ mediaType: AquiferMediaTypeValue
::GeologicUnit
+ geologicUnitType: GeologicUnitTypeValue
::GeologicFeature
+ inspireId: Identifier
«voidable»
+ isExploited: Boolean
+ isMainInSystem: Boolean
+ vulnerabilityToPollution: QuantityValue
+ permeabilityCoefficient: QuantityValue
+ storativityCoefficient: QuantityValue
+ hydroGeochemicalRockType: HydroGeochemicalRockTypeValue
::HydrogeologicalUnit
+ description: PT_FreeText
+ approximateDepth: QuantityValue
+ approximateThickness: QuantityValue
::GeologicFeature
+ name: CharacterString
«lifeCycleInfo - voidable»
::HydrogeologicalUnit
+ beginLifespanVersion: DateTime
+ endLifespanVersion: DateTime [0..1]

6.3.2.GeologicUnit

Definition: A volume of rock with distinct characteristics.

Description: Includes both formal units (i.e. formally adopted and named in an official lexicon) and informal units (i.e. named but not promoted to the lexicon) and unnamed units (i.e. recognisable and described and delineable in the field but not otherwise formalised).

Spatial properties are only available through association with a MappedFeature.

6.3.3.HydrogeologicalUnit

Definition: A part of the lithosphere with distinctive parameters for water storage and conduction.

6.3.4.Aquifer

Definition: A wet underground layer of water-bearing permeable rock or unconsolidated materials (gravel, sand, silt, or clay) from which groundwater can be usefully extracted using a water well.

Description: An underground geological formation able to store and yield water.

6.3.5.AquiferSystem

Definition: A collection of aquifers and aquitards, which together constitute the environment of groundwater - "communicating vessels", that are filled or can be filled with water.

Description: Attributes of Aquifer System and its components determine the feasibility of water collection, its movement, as well as the impact on its chemical state.

NOTE: The Aquifer System components and their attributes (including geometry) are relatively stable over time except in special cases.

6.3.6.HydrogeologicalObject

Definition: An abstract class for man-made facilities or natural features that have an interaction with the hydrogeological system.

Description: Hydrogeological objects may be natural (eg. spring) or the manmade (eg. wells). The vast majority of hydrogeological objects are manmade.

6.3.7.HydrogeologicalObjectNatural

Definition: HydrogeologicalObject which was created by natural processes.

Description: Examples of natural hydrogeological objects are: a source, vanishing point and geyser.

6.3.8.HydrogeologicalObjectManMade

Definition: A man-made hydrogeological object.

Description: Examples of manmade hydrogeological objects are: well, groundwater intake, groundwater monitoring station or monitoring well.

6.3.9.ActiveWell

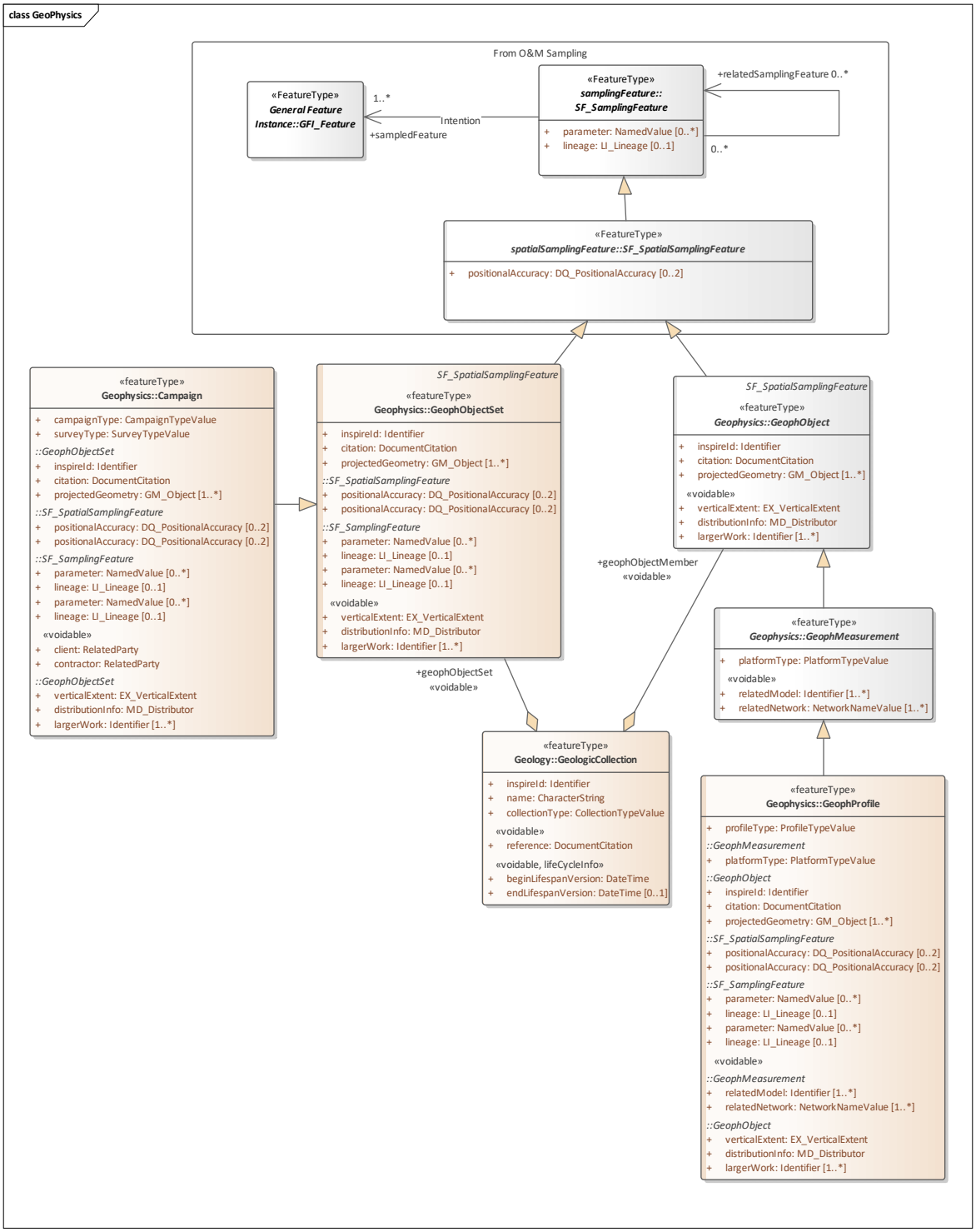
Definition: A well influencing the groundwater resources of the aquifer.

Description: The most common examples of Active Well are: extracting, artificial recharging, or dewatering wells.

NOTE: ActiveWell by extracting, recharging or dewatering influences and changes the state of groundwater resources.

6.4. GeoPhysics

6.4.1.Overview



6.4.2.GFI_Feature

The class GFI_Feature (Figure 2) is an instance of the «metaclass» GF_FeatureType (ISO 19109). It represents the set of all classes which are feature types.

NOTE GFI_Feature is implemented in GML (ISO 19136:2007) by the element gml:AbstractFeature and type gml:AbstractFeatureType.

In an implementation this abstract class shall be substituted by a concrete class representing a feature type from an application schema associated with a domain of discourse (ISO 19109, ISO 19101). Sampling Features (Clause 8) are a class of feature-types whose role is primarily associated with observations.

6.4.3.SF_SamplingFeature

Role of sampling features

Sampling features are artefacts of an observational strategy, and have no significant function outside of their role in the observation process. The physical characteristics of the features themselves are of little interest, except perhaps to the manager of a sampling campaign.

EXAMPLE A “station” is essentially an identifiable locality where a sensor system or procedure may be deployed and an observation made. In the context of the observation model, it connotes the “world in the vicinity of the station”, so the observed properties relate to the physical medium at the station, and not to any physical artefact such as a mooring, buoy, benchmark, monument, well, etc.

NOTE A transient sampling feature, such as a ships-track or flight-line, may be identified and described, but is unlikely to be revisited exactly.

A sampling feature is intended to sample some feature of interest in an application domain. However, in some cases the identity, and even the exact type, of the sampled feature may not be known when observations are made using the sampling features.

Classification of sampling features

A small number of sampling patterns are common across disciplines in observational science. These provide a basis for processing and portrayal tools which are similar across domains, and depend primarily on the geometry of the sample design. Common names for sampling features include specimen, station, profile, transect, path, swath, and scene. These are classified primarily by their topological dimension. Point-located samples are defined in this Clause, extensive samples of various dimensions in Clause 9 and specimens in Clause 10.

6.4.4.SF_SpatialSamplingFeature

When observations are made to estimate properties of a geospatial feature, in particular where the value of a property varies within the scope of the feature, a spatial sampling feature is used. Depending on accessibility and on the nature of the expected property variation, the sampling feature may be extensive in one, two or three spatial dimensions. Processing and visualization methods are often dependent on the topological dimension of the sampling manifold, so this provides a natural classification system for sampling features.

This classification follows common practice in focussing on conventional spatial dimensions. Properties observed on sampling features may be time-dependent, but the temporal axis does not generally contribute to the classification of sampling feature classes. Sampling feature identity is usually less time-dependent than is the property value.

6.4.5.GeophObject

Definition: A generic class for geophysical objects.

Description: GeophObject models single geophysical entities that are used for spatial sampling either by means of data acquisition or data processing.

6.4.6.GeophMeasurement

Definition: Generic spatial object type for geophysical measurements.

Description: Geophysical measurements collect data outside or on the boundary of the observed spatial domain.

6.4.7.GeophProfile

Definition: Geophysical measurement spatially referenced to a curve

Description: Used to collect data along a curve. Examples: 2D seismic line (field measurement), borehole logging, airborne geophysical flight line

NOTE1. Processing results of geophProfiles are often vertical surface coverages

6.4.8.GeophObjectSet

Definition: Generic class for collections of geophysical objects

Description: It is a set of geophysical objects that are grouped by some common property. p.e: created in the same measuring campaign. GeophObjectSets are used for spatial sampling either

by means of data acquisition or data processing. The produced result of a geophObjectSet is always collective, e.g. a map constructed from the results of the individual member objects.

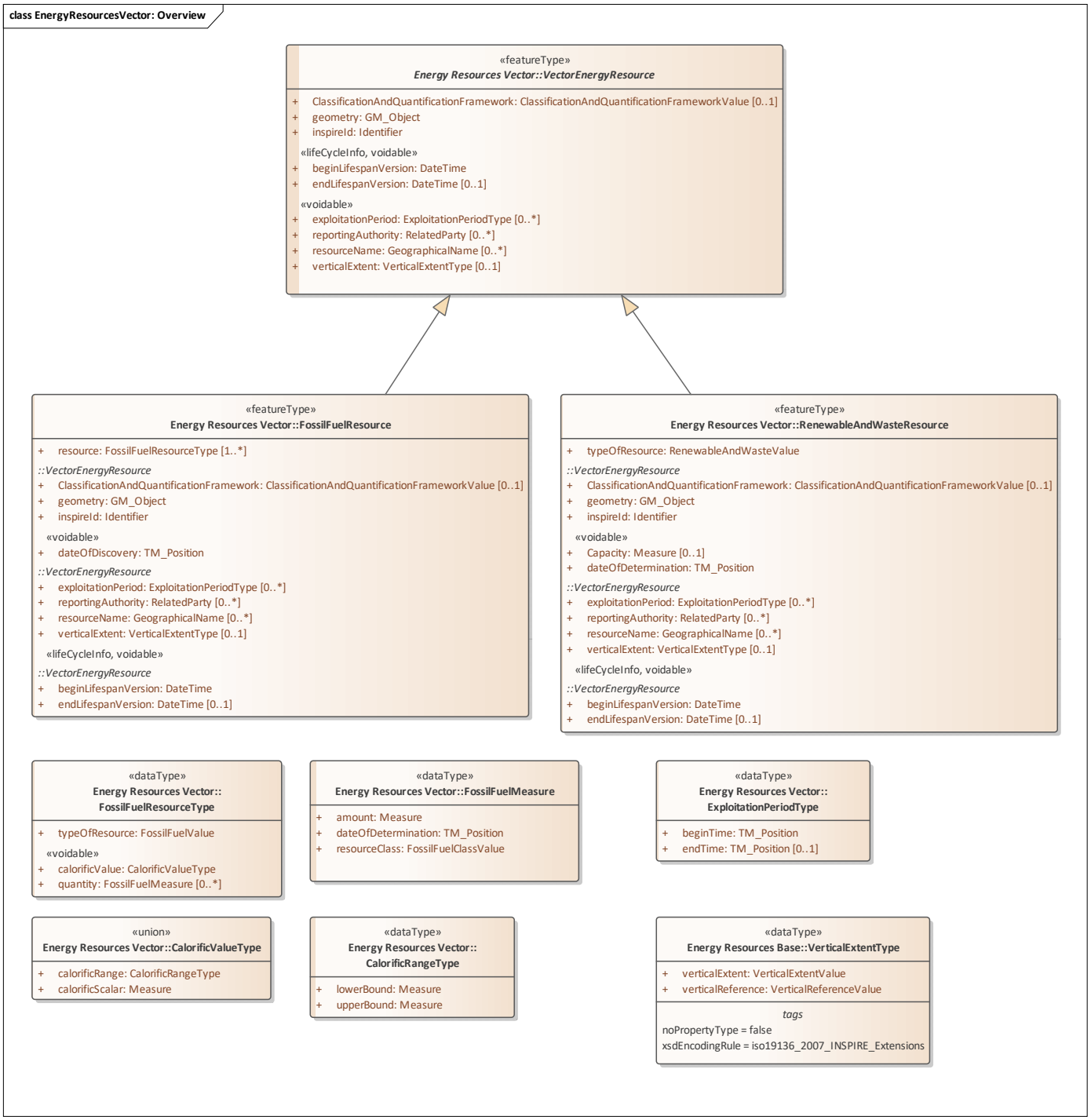
6.4.9.Campaign

Definition: Geophysical activity extending over a limited time range and limited area for producing similar geophysical measurements, processing results or models.

Description: Campaigns can be considered as parents of geophysical measurements or models. Children may refer to parent campaigns through the largerWork identifier.

6.5. EnergyResourcesVector:

6.5.1.Overview



6.5.2.VectorEnergyResource

Definition: A vector spatial object defining an inferred or observable spatial extent of a resource that can be or has been used as a source of energy.

6.5.3.RenewableAndWasteResource

Definition: spatial object defining an inferred or observable spatial extent of a resource that can be, or has been used as a source of renewable energy or waste.

Description: Renewable energy is energy that is naturally occurring and theoretically inexhausatable that is not derived from fossil or nuclear fuel. Waste is a fuel that may consist of many materials coming from combustible industrial, institutional, hospital and household wastes such as rubber, plastics, waste fossil oils and other similar commodities. It is either solid or liquid in form, renewable or non-renewable, biodegradable or non-biodegradable.

6.5.4.ExploitationPeriodType

Definition: The exploitationPeriod defines the start and, if applicable, the end date of the exploitation or application.

6.5.5.VerticalExtentType

Definition: Vertical dimensional property consisting of an absolute measure or range of measures referenced to a well-defined vertical reference level which is commonly taken as origin (ground level, mean sea level, etc.).

6.5.6.FossilFuelResource

Definition: A spatial object defining an inferred or observable spatial extent of a resource that can be, or has been used as a source of fossil fuel energy. The most common fossil fuel types are coal, natural gas and crude oil.

Description: Solid fossil fuels are those non-renewable hydrocarbon energy resources that are naturally found in the solid state i.e. coals and peat. Hydrocarbons cover various types of natural gas and petroleum resources.

6.5.7.FossilFuelResourceType

Definition: Type and amount of resource according to specific categorisation.

6.5.8. FossilFuelMeasure

Definition: Amount of resources according to the specific categorisation.

6.5.9. CalorificValueType

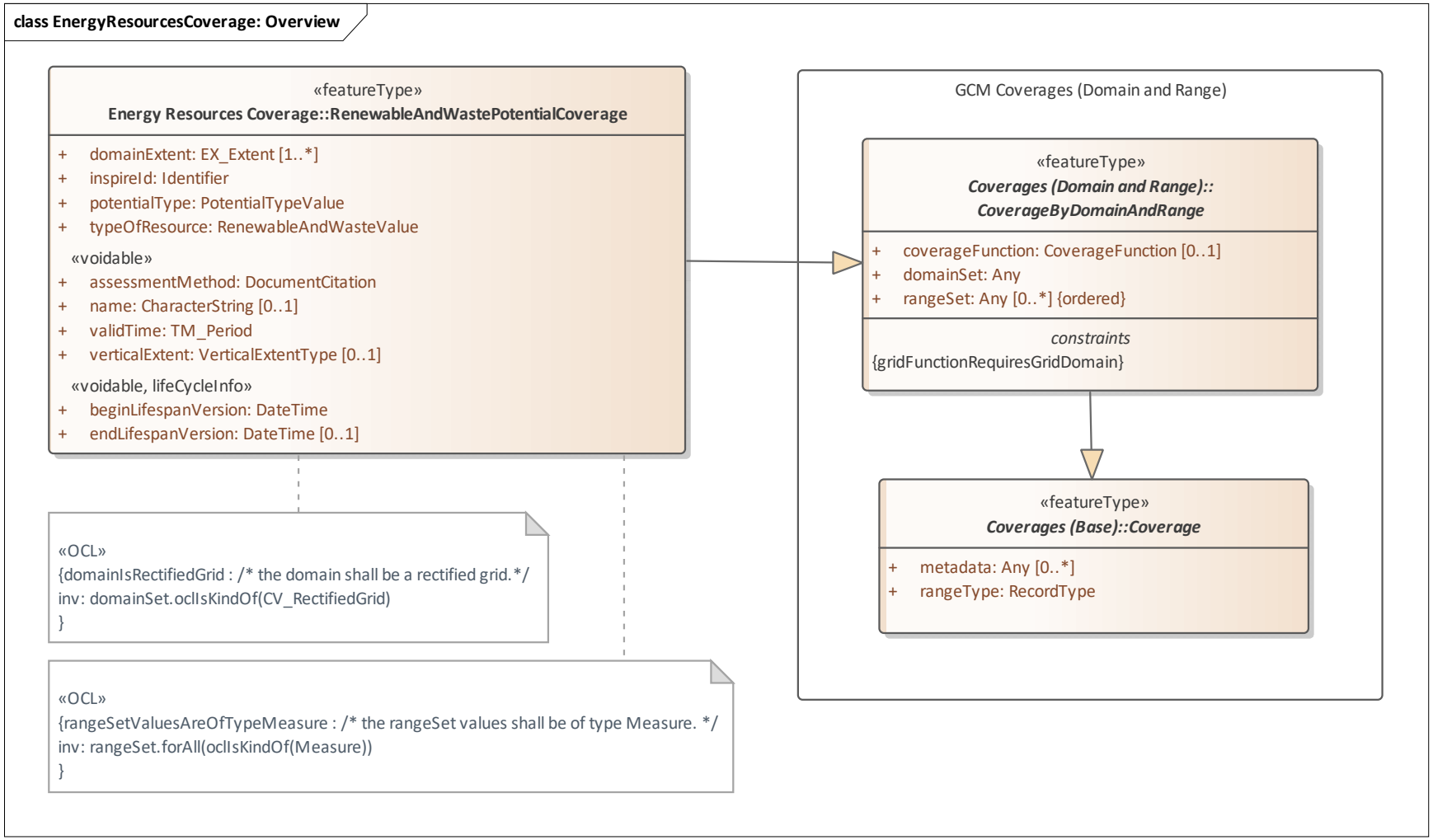
Definition: Value or range of values describing the calorific value of an Energy Resource.

6.5.10. CalorificRangeType

Definition: Value indicating the upper and lower bounds of the calorific range of the energy resource.

6.6. EnergyResourcesCoverage

6.6.1. Overview


class EnergyResourcesCoverage: Overview
«featureType»
Energy Resources Coverage::RenewableAndWastePotentialCoverage
+ domainExtent: EX_Extent [1..*]
+ inspireId: Identifier
+ potentialType: PotentialTypeValue
+ typeOfResource: RenewableAndWasteValue
«voidable»
+ assessmentMethod: DocumentCitation
+ name: CharacterString [0..1]
+ validTime: TM_Period
+ verticalExtent: VerticalExtentType [0..1]
«voidable, lifeCycleInfo»
+ beginLifespanVersion: DateTime
+ endLifespanVersion: DateTime [0..1]
«OCL»
{domainIsRectifiedGrid : /* the domain shall be a rectified grid.*/
inv: domainSet.oclIsKindOf(CV_RectifiedGrid)
}
«OCL»
{rangeSetValuesAreOfTypeMeasure : /* the rangeSet values shall be of type Measure. */
inv: rangeSet.forAll(oclIsKindOf(Measure))
}
GCM Coverages (Domain and Range)
«featureType»
Coverages (Domain and Range)::
CoverageByDomainAndRange
+ coverageFunction: CoverageFunction [0..1]
+ domainSet: Any
+ rangeSet: Any [0..*] {ordered}
constraints
{gridFunctionRequiresGridDomain}
«featureType»
Coverages (Base)::Coverage
+ metadata: Any [0..*]
+ rangeType: RecordType

6.6.2.Coverage

Definition: Spatial object that acts as a function to return values from its range for any direct position within its spatial, temporal or spatiotemporal domain.

Description: EXAMPLE Examples include a raster image, polygon overlay or digital elevation matrix.

NOTE In other words, a coverage is a feature that has multiple values for each attribute type, where each direct position within the geometric representation of the feature has a single value for each attribute type.

6.6.3.CoverageByDomainAndRange

Definition: Coverage which provide the domain and range as separate properties.

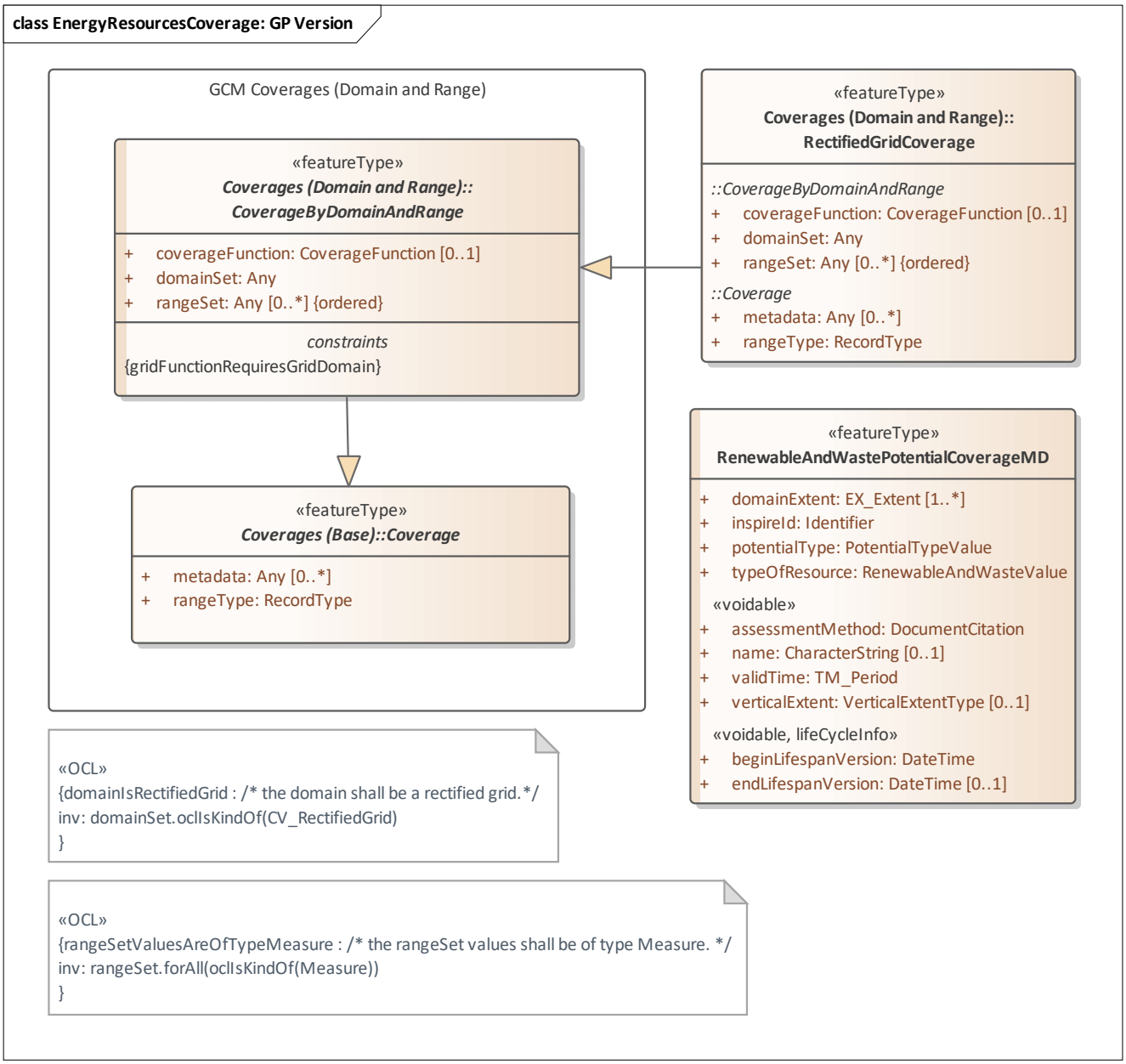
6.6.4.RenewableAndWastePotentialCoverage

Definition: Function that returns an energy potential value from its range for any direct position within its spatial, temporal or spatio-temporal domain.

Description: SOURCE Adapted from "Coverage" [ISO 19123:2005].

6.7. EnergyResourcesCoverage: GP Version

6.7.1.Overview



6.7.2.RectifiedGridCoverage

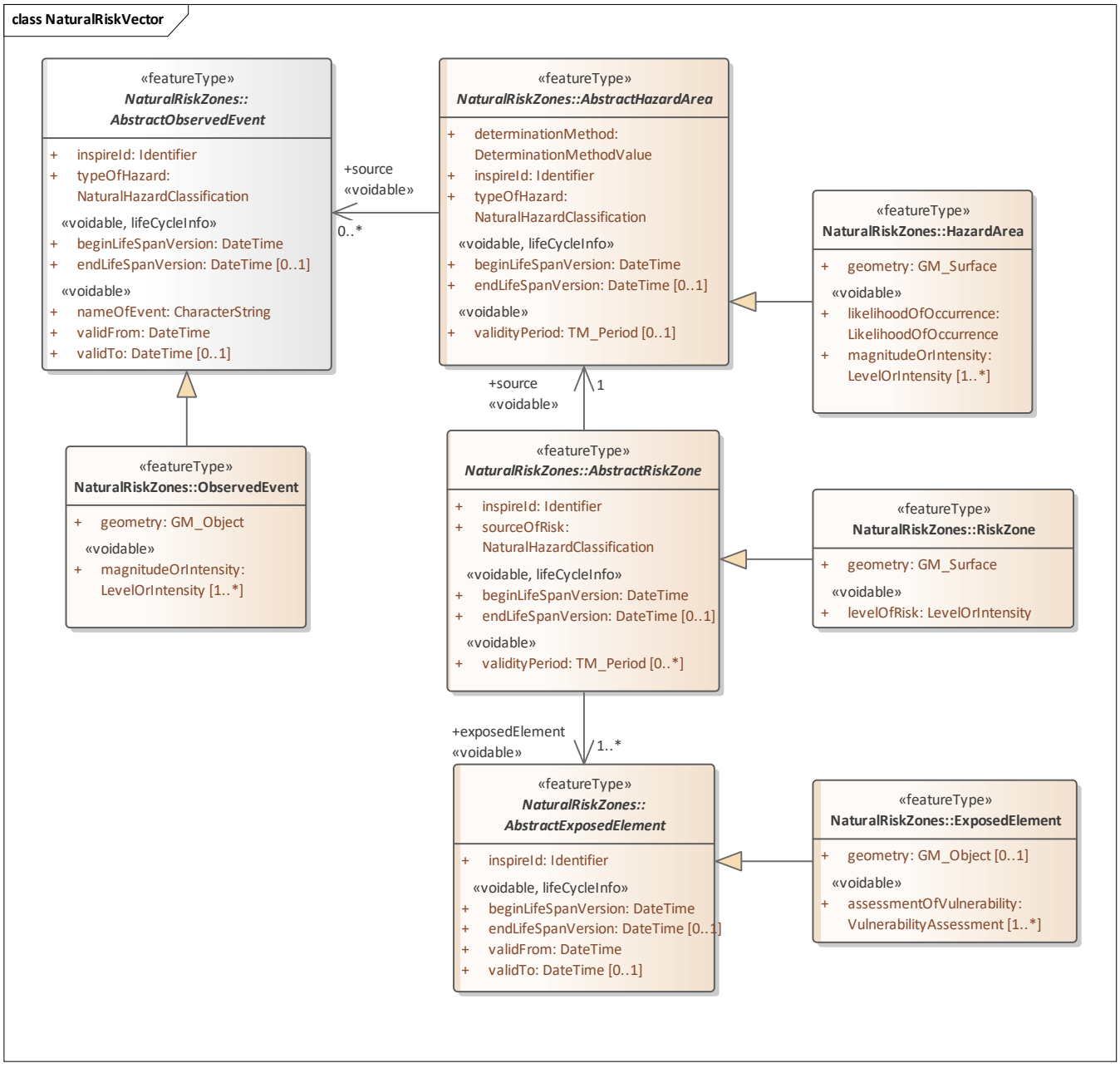
Definition: Coverage whose domain consists of a rectified grid

Description: A rectified grid is a grid for which there is an affine transformation between the grid coordinates and the coordinates of a coordinate reference system.

NOTE This type can be used for both discrete and continuous coverages.

6.8. NaturalRiskVector

6.8.1.Overview



6.8.2.AbstractObservedEvent

Definition: A natural phenomenon relevant to the study of natural hazards which occurred and which has been observed.

6.8.3.ObservedEvent

Definition: Discrete spatial objects representing natural phenomenon relevant to the study of natural hazards which occurred, or is currently occuring, and which has been observed.

6.8.4.AbstractHazardArea

Definition: An area affected by a natural hazard.

Description: A natural hazard is a natural process or phenomenon that may cause loss of life, injury or other impacts, property damage, loss livelihoods and services, social and economic disruption, or environmental damage. [Council of The European Union - Commission Staff Working Paper - Risk Assessment and Mapping Guidelines for Disaster Management].

6.8.5.HazardArea

Definition: Discrete spatial objects representing a natural hazard.

6.8.6.AbstractRiskZone

Definition: A risk zone is the spatial extent of a combination of the consequences of an event (hazard) and the associated probability/likelihood of its occurrence.

6.8.7.RiskZone

Definition: Discrete spatial objects representing the spatial extent of a combination of the consequences of an event (hazard) and the associated probability/likelihood of its occurrence.

6.8.8.AbstractExposedElement

Definition: People, property, systems, or other elements present in hazard zones that are thereby subject to potential losses.

SOURCE : [UNISDR, 2009]

7. Annex C - GeoServer App Schema Configuration

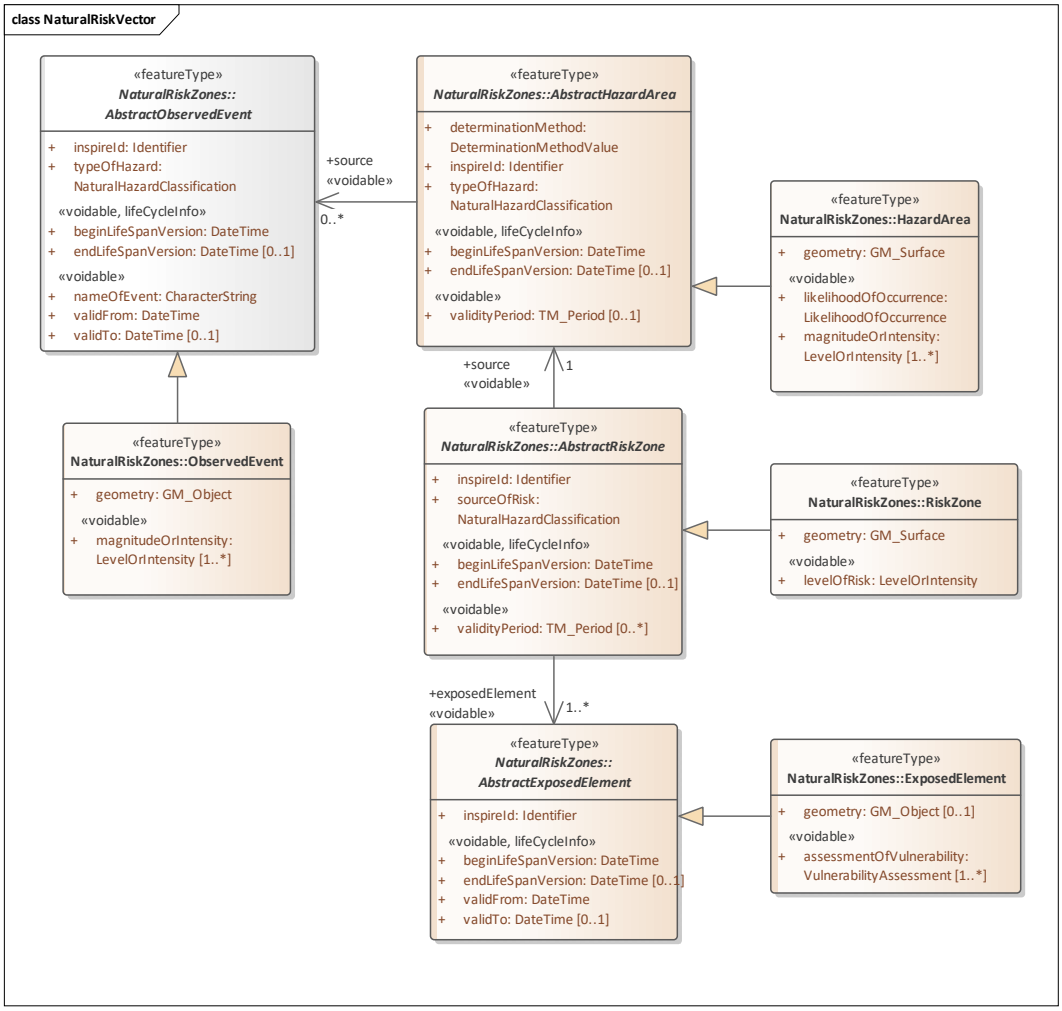
In order to simplify the often painful process of App Schema configuration, we have created a tutorial based on a simple example that serves to illustrate the various options available without burdoning the user with a complex data model as often utilized in such tutorials. The individual sections of this tutorial are described below.

7.1. Example FeatureType and Database

For the purpose of this tutorial, we have created a set of GML featureTypes and dataTypes with various associations between them. These serve to illustrate the various options available within GeoServer App Schema configuration.

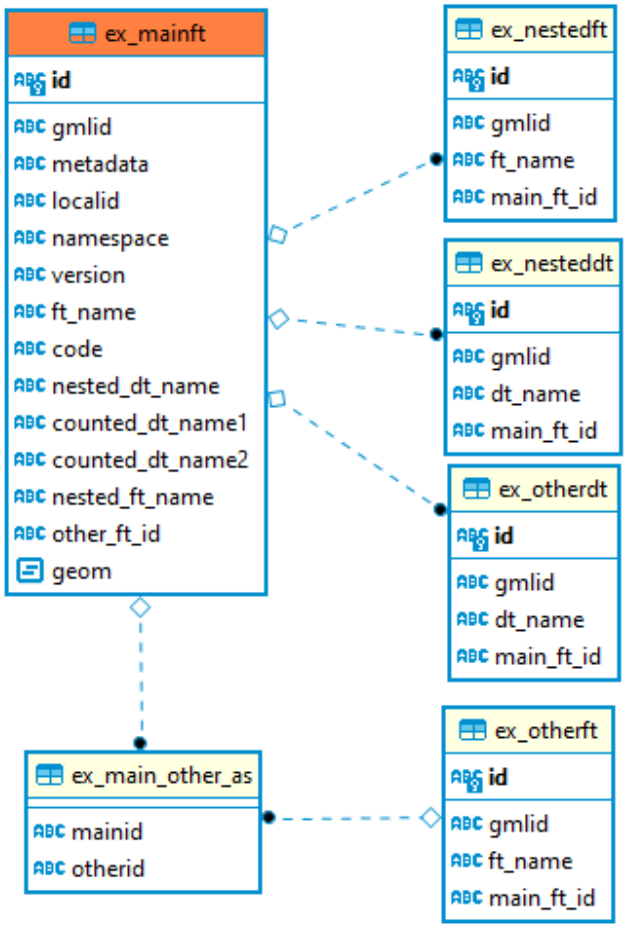
7.1.1. UML for Example FeatureTypes

The UML diagram below shows the featureType and dataTypes defined for this tutorial, together with the associations defined between these types.



7.1.2.ER Diagram of DB Tables

The ER diagram below shows the database tables required for provision of the Example FeatureTypes



7.1.3. Example XML Output MainFT

```
<ex:MainFT gml:id="gmlid1">
  <ex:inspireId>
        <base:Identifier>
          <base:localId>localid 1</base:localId>
          <base:namespace>namespace 1</base:namespace>
          <base:versionId>version 1</base:versionId>
        </base:Identifier>
  </ex:inspireId>
  <ex:name>ft_name</ex:name>
  <ex:code xlink:href="http://codes.datacove.eu/code"/>
  <ex:codeMultiple xlink:href="http://codes.datacove.eu/codeMultiple1"/>
  <ex:codeMultiple xlink:href="http://codes.datacove.eu/codeMultiple2"/>
  <ex:nestedDT>
        <ex:NestedDT>
          <ex:name>nested_dt_name</ex:name>
        </ex:NestedDT>
  </ex:nestedDT>
  <ex:nestedDTMultiple>
        <ex:NestedDT>
```

```
        <ex:name>nesteddt_name1</ex:name>
      </ex:NestedDT>
  </ex:nestedDTMultiple>
  <ex:nestedDTMultiple>
      <ex:NestedDT>
        <ex:name>nesteddt_name2</ex:name>
      </ex:NestedDT>
  </ex:nestedDTMultiple>
  <ex:countedDT>
      <ex:OtherDT>
        <ex:name>counted_dt_name1</ex:name>
      </ex:OtherDT>
  </ex:countedDT>
  <ex:countedDT>
      <ex:OtherDT>
        <ex:name>counted_dt_name2</ex:name>
      </ex:OtherDT>
  </ex:countedDT>
  <ex:nestedFT>
      <ex:NestedFT gml:id="nestedFT_gmlid1">
        <ex:name>nested_ft_name</ex:name>
      </ex:NestedFT>
  </ex:nestedFT>
  <ex:nestedFTMultiple>
      <ex:NestedFT gml:id="nestedft_gmlid1">
        <ex:name>nestedft_name1</ex:name>
      </ex:NestedFT>
  </ex:nestedFTMultiple>
  <ex:nestedFTMultiple>
      <ex:NestedFT gml:id="nestedft_gmlid2">
        <ex:name>nestedft_name2</ex:name>
      </ex:NestedFT>
  </ex:nestedFTMultiple>
  <ex:geometry>
      <gml:Point gml:id="PT_gmlid1" srsDimension="2"
srsName="urn:ogc:def:crs:EPSG::4326">
        <gml:pos>48.022078 -1.505727</gml:pos>
      </gml:Point>
  </ex:geometry>
  <ex:other xlink:href="http://service.datacove.eu/example/other/ID2"/>
</ex:MainFT>
```

7.1.4. Example XML Output MainFT

```
<ex:OtherFT gml:id="ID2">
  <ex:name>OtherFT</ex:name>
  <ex:main xlink:href="http://service.datacove.eu/example/main/ID1"/>
  <ex:main xlink:href="http://service.datacove.eu/example/main/ID3"/>
</ex:OtherFT>
```

7.1.5. Schema file

The Schema file for the types used in this example is provided in the annex to this document, in addition, it is available at https://schema.datacove.eu/Example.xsd

7.1.6. SQL and App Schema Mapping files for Example DB

A simple database has been created for this example, the full SQL is available in the Annex. The same holds true for the required App Schema Mapping file, the creation of which is described in the following sections.

7.2. App Schema Base

The base information within the App Schema configuration always follows the same pattern, providing information on the following concepts:

- Namespaces: all namespaces used within the final output XML must be configured
- Database: location and authentication information for the source database
- Source: source schema for the featureTypes to be provided
- Feature Mapping: mapping information for each individual featureType to be provided

7.2.1. App Schema Schema

While this file should be available from elsewhere, it's currently not, thus this link for the moment: https://schema.datacove.eu/AppSchemaDataAccess.xsd

7.2.2. Namespaces

All namespaces used in the App Schema Mapping must be declared together with their prefixes in the **namespaces** section.

```
<Namespace>
  <prefix>ex</prefix>
  <uri>https://schema.datacove.eu/Example</uri>
</Namespace>
```

*Note: with All we really mean **ALL!** A common error is caused by missing a required namespace, this often happens with nested namespaces such as xlink or gmd*

7.2.3. Database Configuration

GeoServer App Schema allows for various database configuration options. In order to allow great flexibility, the individual **DataStore** sections within the **sourceDataStores** utilize the Parameter Type that provides name/value pairs. The **name** element provides the individual configuration concept, while the **value** element provides the value to be used for this concept.

Regardless of configuration type, the identifier for a **DataStore** is always provided by the **id** element. This name must be provided within subsequent configuration sections to assign the correct data source to the attribute mapping.

Please note that in this tutorial we rely souly on PostGIS databases. For other configuration options, please see the full documentation available for GeoServer.

JNDI/JDBC Data Store

The simplest is to utilize an existing jdbc connection available on the server running GeoServer as a JNDI connection. For this purpose, the following parameters must be set within the **sourceDataStores DataStore**

- **dbtype**: For a JNDI datastore utilizing JDBC on PostGIS, specify the type 'postgisng'
- **jndiReferenceName**: Provide the name of the JDBC connection configured on the server
- **Expose primary keys**: This parameter must be set to true in order to enable getFeatureByID

```
<sourceDataStores>
  <DataStore>
    <id>idDataStoreInsp</id>
    <parameters>
      <Parameter>
        <name>dbtype</name>
        <value>postgisng</value>
      </Parameter>
      <Parameter>
        <name>jndiReferenceName</name>
        <value>java:comp/env/jdbc/tnw</value>
      </Parameter>
      <Parameter>
        <name>Expose primary keys</name>
        <value>true</value>
      </Parameter>
    </parameters>
  </DataStore>
</sourceDataStores>
```

Direct Data Store

Alternatively, one can specify the database connection directly. The values can be either listed directly within the App Schema config file, or taken from a properties file for better security. For this purpose, the following parameters must be set within the **sourceDataStores DataStore**

- **dbtype**: For a datastore on PostGIS, specify the type 'postgisng'
- **host**: Provide the host name or IP of the DB Server
- **port**: Provide the port the DB is listening on. Usual default for PostGres is 5432
- **database**: Provide the name of the database that this connection refers to
- **schema**: Provide the name of the database schema that this connection refers to
- **user**: Provide the name database user that this connection refers to

- **passwd**: Provide the password for the database name of the database user that this connection refers to
- **Expose primary keys**: This parameter must be set to true in order to enable getFeatureByID

Note: in the example below, both options are shown, whereby the direct option has been commented out. More information on setting properties below.

```
<sourceDataStores>
        <DataStore>
                <id>idDataStoreInsp</id>
                <parameters>
                        <Parameter>
                                <name>dbtype</name>
                                <value>postgisng</value>
                        </Parameter>
                        <Parameter>
                                <name>host</name>
                                <!--<value>192.12.34.56</value>-->
                                <value>${inspire.host}</value>
                        </Parameter>
                        <Parameter>
                                <name>port</name>
                                <value>5432</value>
                        </Parameter>
                        <Parameter>
                                <name>database</name>
                                <!--<value>my_inspire_database</value>-->
                                <value>${inspire.database}</value>
                        </Parameter>
                        <Parameter>
                                <name>schema</name>
                                <value>public</value>
                        </Parameter>
                        <Parameter>
                                <name>user</name>
                                <!--<value>my_user</value>-->
                                <value>${inspire.user}</value>
                        </Parameter>
                        <Parameter>
                                <name>passwd</name>
                                <!--<value>my_password</value>-->
                                <value>${inspire.passwd}</value>
                        </Parameter>
                        <Parameter>
                                <name>Expose primary keys</name>
                                <value>true</value>
                        </Parameter>
                </parameters>
        </DataStore>
</sourceDataStores>
```

Properties

Under Debian, the app-schema.properties file can be found under the geoserver directory at the relative path ./WEB-INF/classes/app-schema.properties. The properties variables set in this file can be utilized within the app schema configuration as shown above.

```
inspire.host = 192.12.34.56
inspire.database = my_inspire_database
inspire.user = my_user
inspire.passwd = my_password
```

Configure Source for Feature Types

The location of the xsd file where the description of the Feature Types to be mapped is provided in the **schemaUri** in the **targetTypes** section.

```
<targetTypes>
        <FeatureType>
                <schemaUri>https://schema.datacove.eu/Example.xsd</schemaUri>
        </FeatureType>
</targetTypes>
```

Feature Mapping

For each Feature Type being mapped, basic information on the data source must be provided as follows.

Under **sourceDataStore**, the name of the data store configured in the Database Configuration must be provided.

Under **sourceType**, the name of the data table in which the data for the Feature Type is stored must be provided.

Under **targetElement**, the name of the Feature Type to be configured must be provided.

Under **defaultGeometry**, the location of the Feature geometry can be be provided. While this is not essential, it is recommended in cases where the geometry is nested within the Feature.

```
<FeatureTypeMapping>
        <sourceDataStore>idDataStoreInsp</sourceDataStore>
        <sourceType>ex_mainft</sourceType>
        <targetElement>ex:MainFT</targetElement>
        <defaultGeometry>ex:geometry</defaultGeometry>
```

Further details on Feature Mapping

7.3. Feature Mapping

Within the Feature Mapping section of the App Schema configuration one must specify the source for each individual element and attribute within the target XML. GeoServer App Schema configuration supports a wide range of options depending on the user requirements, e.g. pertaining to the cardinality of nested types. In the following section, we provide basic configuration options.

For each Feature Type being mapped, basic information on the data source must be provided as follows.

Under **sourceDataStore**, the name of the data store configured in the Database Configuration must be provided.

Under **sourceType**, the name of the data table in which the data for the Feature Type is stored must be provided.

Under **targetElement**, the name of the Feature Type to be configured must be provided.

Under **defaultGeometry**, the location of the Feature geometry can be be provided. While this is not essential, it is recommended in cases where the geometry is nested within the Feature.

```
<FeatureTypeMapping>
        <sourceDataStore>idDataStoreInsp</sourceDataStore>
        <sourceType>ex_mainft</sourceType>
        <targetElement>ex:MainFT</targetElement>
        <defaultGeometry>ex:geometry</defaultGeometry>
```

7.4. AttributeMapping

In the **attributeMappings** section, the data source for each element of the Feature Type being mapped is configured within a **AttributeMapping** section.

```
<attributeMappings>
        <AttributeMapping>
                ...
        </AttributeMapping>
</attributeMappings>
```

7.4.1. Mapping gml:id

Under **targetAttribute**, provide the name of the FeatureType you're mapping to including the namespace

Under **idExpression**, provide the DB Column name which provides the value for the gml:id.

Note of caution pertaining to gml:id, gml:id cannot start with a number. It must be a letter or underscore "_", after this characters may be letters, numbers or one of "_", "-", "."

```
<AttributeMapping>
        <targetAttribute>ex:MainFT</targetAttribute>
        <idExpression>
                <OCQL>gmlid</OCQL>
        </idExpression>
</AttributeMapping>
```

7.4.2. Mapping simple element

Under **targetAttribute**, provide the name of the element you're mapping to including the namespace

Under **sourceExpression**, provide the DB Column name which provides the value for this element.

Note: in this example the type of the attribute is string, the DB column of type varchar. The same principle works for other datatypes, e.g. date, whereby the DB column must be of the appropriate type

```
<AttributeMapping>
        <targetAttribute>ex:name</targetAttribute>
        <sourceExpression>
                <OCQL>ft_name</OCQL>
        </sourceExpression>
</AttributeMapping>
```

7.4.3. Mapping to attributes such as xlink:href

Under **targetAttribute**, provide the name of the element you're mapping to including the namespace

Include **encodeIfEmpty** set to true if only the attributes will contain content. Otherwise GeoServer will only encode this element if the element itself contains content.

Under **ClientProperty**, under **name** provide the name of the attribute to be mapped to, under **value** provide the DB Column name which provides the value for this element.

*Note: both **sourceExpression** and **ClientProperty** can be utilized in the same **AttributeMapping** section if one wants to provide content both to the XML Element itself as well as to attributes of this element*

```
<AttributeMapping>
        <targetAttribute>dm1:code</targetAttribute>
        <encodeIfEmpty>true</encodeIfEmpty>
        <ClientProperty>
```

```
                <name>xlink:href</name>
                <value>strconcat('http://codes.datacove.eu/', code)</value>
        </ClientProperty>
</AttributeMapping>
```

7.4.4.Mapping to geometry

Under **targetAttribute**, provide the name of the geometry element you're mapping to including the namespace. Do **NOT** include the geometry type, Geoserver will figure this out itself depending on the database geometry type.

Under **idExpression**, provide the DB Column name which provides the value for the gml:id of the geometry.

Under **sourceExpression**, provide the DB Column name which provides the value for the geometry.

```
<AttributeMapping>
        <targetAttribute>dm1:geometry</targetAttribute>
        <idExpression>
                <OCQL>strconcat('PT_', id)</OCQL>
        </idExpression>
        <sourceExpression>
                <OCQL>geom</OCQL>
        </sourceExpression>
</AttributeMapping>
```

7.4.5.Mapping to elements of a nested featureType or dataType (singular)

When mapping to elements of featureTypes or dataTypes with a maximum cardinality of one, one can explicitely map to the individual elements of this included type by specifying the full xpath to this element. Both **sourceExpression** for element content as well as **ClientProperty** for mapping to element attributes can be utilized in this context. As already mentioned above under **Mapping to attributes**, if the element itself may be empty, one must set **encodeIfEmpty** to **true**.

This approach to mapping can be utilized regardless of if the included type is a featureType or a dataType.

Under **targetAttribute**, provide the name of the element you're mapping to including the namespace

Under **sourceExpression**, provide the DB Column name which provides the value for this element.

Include **encodeIfEmpty** set to **true** if only the attributes will contain content. Otherwise GeoServer will only encode this element if the element itself contains content.

Under **ClientProperty**, under **name** provide the name of the attribute to be mapped to, under **value** provide the DB Column name which provides the value for this element.

Note: in this example the type of the attribute is string, the DB column of type varchar. The same principle works for other datatypes, e.g. date, whereby the DB column must be of the appropriate type

The following configuration segment assigns the content of the database column nested_dt_name to the ex:name element of the nested dataType ex:NestedDT, that is provided under the ex:nestedDT element of ex:mainFT.

```
<AttributeMapping>
        <targetAttribute>ex:nestedDT/ex:NestedDT/ex:name</targetAttribute>
        <sourceExpression>
                <OCQL>nested_dt_name</OCQL>
          </sourceExpression>
</AttributeMapping>
```

The following configuration section assigns the content of the database column nested_ft_name to the ex:name element of the nested dataType ex:NestedFT, that is provided under the ex:nestedFT element of ex:mainFT. As ex:nestedFT requires the provision of a gml:id for this featureType, an additional **AttributeMapping** section is required to provide this under the gml:id attribute of ex:NestedFT

```
<AttributeMapping>
        <targetAttribute>ex:nestedFT/ex:NestedFT</targetAttribute>
        <ClientProperty>
                <name>gml:id</name>
                <value>strconcat('nestedFT_', id)</value>
        </ClientProperty>
</AttributeMapping>

<AttributeMapping>
        <targetAttribute>ex:nestedFT/ex:NestedFT/ex:name</targetAttribute>
        <sourceExpression>
                <OCQL>nested_ft_name</OCQL>
        </sourceExpression>
</AttributeMapping>
```

7.4.6. Mapping to explicit elements of a nested featureType or dataType (multiple)

When mapping to elements of featureTypes or dataTypes with a maximum cardinality of more than one, one can explicitely map to the individual elements of this included type by specifying the full xpath **including the index** to this element. Both **sourceExpression** for element content as well as **ClientProperty** for mapping to element attributes can be utilized in this context. As already mentioned above under **Mapping to attributes**, if the element itself may be empty, one must set **encodeIfEmpty** to **true**.

This approach to mapping can be utilized regardless of if the included type is a featureType or a dataType. All elements of **AttributeMapping** are to be provided as described in the sections above.

The following configuration segment assigns the content of the database columns counted_dt_name1 and counted_dt_name2 to the ex:name element of the nested dataType ex:OtherDT, that is provided under the ex:countedDT element of ex:mainFT. In contrast to the examples above, the XPath provided within the **targetAttribute** element includes the index of the **ex:countedDT** to be mapped to. Thus, the value of the database column counted_dt_name1 is provided as ex:name in the first instance of ex:OtherDT (specified by the index value of '1'), while the value of the database column counted_dt_name2 is provided as ex:name in the second instance of ex:OtherDT (specified by the index value of '2')

```
<AttributeMapping>
  <targetAttribute>ex:countedDT[1]/ex:OtherDT/ex:name</targetAttribute>
  <sourceExpression>
        <OCQL>counted_dt_name1</OCQL>
  </sourceExpression>
</AttributeMapping>

<AttributeMapping>
  <targetAttribute>ex:countedDT[2]/ex:OtherDT/ex:name</targetAttribute>
  <sourceExpression>
        <OCQL>counted_dt_name2</OCQL>
  </sourceExpression>
</AttributeMapping>
```

7.5. Feature Chaining

Feature Chaining is one of the more complex aspects of GeoServer App Schema configuration. It is utilized in those cases where a nested featureType or dataType can have a cardinality of more than one. In such cases, an additional database table will be required for the provision of the data pertaining to the nested featureType or dataType. Feature Chaining allows one to configure the mapping for each individual featureType or dataType separately within a **FeatureTypeMapping** section of the configuration file, then reference this complete feature mapping for the content of the attribute providing the nested type.

7.5.1. Feature Chaining of dataTypes

Provision of a chained dataType

In our example, the dataType **NestedDT** is provided by the nestedDtMultiple attribute of **MainFT** with a cardinality of 0..* . The data for the dataType **NestedDT** is stored in the database table ex_nesteddt. The **FeatureTypeMapping** for this nested dataType has the same structure

as that done for the **MainFT**, with a few additions. In the header a **mappingName** is provided that allows us to reference this feature mapping.

In the example below, the **mappingName** is defined as '_Nested_DT'

```
<FeatureTypeMapping>
  <mappingName>_Nested_DT</mappingName>
  <sourceDataStore>idDataStoreInsp</sourceDataStore>
  <sourceType>ex_nesteddt</sourceType>
  <targetElement>ex:NestedDT</targetElement>
```

In addition, we must provide an **AttributeMapping** section that specifies the database column to be utilized as a foreign key to the main table; the name provided under **targetAttribute** for this section must be the same as used when referencing this dataType. In the example below the database column main_ft_id references the id column of the main table as specified under **sourceExpression**, while the name for the **targetAttribute** is provided as 'FEATURE_LINK'

```
<AttributeMapping>
  <targetAttribute>FEATURE_LINK</targetAttribute>
  <sourceExpression>
    <OCQL>main_ft_id</OCQL>
  </sourceExpression>
</AttributeMapping>
```

*Note: in cases where the same dataType is referenced from multiple locations, whereby different foreign keys should be utilized in associating the correct data, multiple such **AttributeMapping** sections can be provided, each with a different name for **targetAttribute**, e.g. 'FEATURE_LINK[1]' and 'FEATURE_LINK[2]'

Referencing of a chained dataType

Once the dataType to be nested has been configured as described above, one must configure the reference to it within the MainFT as shown below. The **targetAttribute** is the name of the XML element as in basic feature mapping. In the **sourceExpression** the linkage to the chained dataType is provided via the following elements:

- OCQL: the column of the main database table referenced by the foreign key of the linked feature, in our example 'id'
- linkElement: the **mappingName** of the **FeatureTypeMapping** to be linked, in our example '_Nested_DT'
- linkField: the name provided as **targetAttribute** of the **AttributeMapping** providing the correct foreign key association, in our example 'FEATURE_LINK'

```
<AttributeMapping>
  <targetAttribute>ex:nestedDTMultiple</targetAttribute>
  <sourceExpression>
    <OCQL>id</OCQL>
    <linkElement>_Nested_DT</linkElement>
    <linkField>FEATURE_LINK</linkField>
  </sourceExpression>
```

```
</AttributeMapping>
```

7.5.2. Feature Chaining of featureTypes

The same principle as described above can also be applied to featureTypes. The only addition is the provision of an **idExpression** within the **FeatureTypeMapping** of the linked featureType.

```
<AttributeMapping>
  <targetAttribute>ex:NestedFT</targetAttribute>
  <idExpression>
    <OCQL>gmlid</OCQL>
  </idExpression>
</AttributeMapping>
```

7.6. Linking Features and Multiple xlinks

Usually, feature chaining is used to include additional features with a multiplicity above one within a feature. However, this functionality can also be used to provide a list of xlinks referencing a related feature or alternative resource such as a codelist. As with Feature Chaining, two database tables are required:

- the table providing the data for the main feature, in our example ex_mainft
- the table providing the data for the linked feature, in our example ex_main_other_as, the association table providing the link between MainFT and OtherFT

7.6.1. Setting up the Link

In our example, links to the featureType **OtherFT** (data contained in *ex_mainft*) are provided by xlinks within the ex:other attribute of the **MainFT** (data contained in *ex_otherft*), with a cardinality of 0..* . The association table *ex_main_other_as* provides the links between between the tables, (2 attributes, mainid and otherid).

The header section of the **FeatureTypeMapping** is the same as for normal chained features, described in the previous section in more detail. In this example, the **mappingName** _main_to_other is provided. The association table *ex_main_other_as* providing information on the link between **MainFT** and **OtherFT** is provided in the **sourceType** element. The **targetElement** is the featureType being linked, in this case **OtherFT**.

Within the **attributeMappings**, two **AttributeMapping** blocks must be provided as follows:

- First **AttributeMapping**
  - sourceExpression/OCQL: in this element, the column providing the foreign key to the main table must be entered. In this example “mainid”.
  - targetAttribute: this element only needs the content “FEATURE_LINK”.
- Second **AttributeMapping**

- targetAttribute: the element from the main featureType that will be providing the xlink (*Note: this is the one tricky bit as usually **FeatureTypeMapping** only contains elements from the **nested** feature, whereas here the element provided is from the **main** feature*)

```
<FeatureTypeMapping>
  <mappingName>_main_to_other</mappingName>
  <sourceDataStore>idDataStoreInsp</sourceDataStore>
  <sourceType>ex_main_other_as</sourceType>
  <targetElement>ex:OtherFT</targetElement>
  <attributeMappings>
        <encodeIfEmpty>true</encodeIfEmpty>
        <AttributeMapping>
          <targetAttribute>FEATURE_LINK</targetAttribute>
          <!-- FK in linked table -->
          <sourceExpression>
                <OCQL>mainid</OCQL>
          </sourceExpression>
        </AttributeMapping>
        <AttributeMapping>
          <targetAttribute>ex:other</targetAttribute>
        </AttributeMapping>
  </attributeMappings>
</FeatureTypeMapping>
```

7.6.2. Embedding the Link

Once the link has been set up as described above, it only needs to be inserted as a **AttributeMapping** into the mapping of the **MainFT** as as shown below. The **targetAttribute** is the name of the XML element as in basic feature mapping, in this example ex:other. The elements **encodeIfEmpty** and **isMultiple** should be set to true.

In the **sourceExpression** the linkage to the chained dataType is provided via the following elements:

- OCQL: the primary key of the main table, column the foreign key of the linked table is referencing. In this example “gmlid”
- linkElement: the **mappingName** of the link **FeatureTypeMapping**
- linkField: this element only needs the content “FEATURE_LINK”.

In the **ClientProperty**, we provide the XML attribute to be mapped to, as well as the content of this attribute as follows:

- name: the name of the XML attribute to be mapped to, in this example xlink:href
- value: the content of the xlink:href attribute. There are two tricky bits in this entry:
  - In contrast to other App Schema modalities, the column names being referenced stem from the linked table, “otherid” is a column of the linked association table *ex_main_other_as*
  - In order to make the link actionable, the API URL has been prepended to the identifier of the linked feature. In addition, in this example the application type has

been added as a suffix for easier exploration within a browser - this suffix should **NOT** be provided in real data endpoints, only provided for illustrative purposes here.

```
<AttributeMapping>
  <targetAttribute>ex:other</targetAttribute>
  <encodeIfEmpty>true</encodeIfEmpty>
  <sourceExpression>
    <!-- PK in main table, what the FK in the linked table links to -->
    <OCQL>gmlid</OCQL>
    <linkElement>_main_to_other</linkElement>
    <linkField>FEATURE_LINK</linkField>
  </sourceExpression>
  <isMultiple>true</isMultiple>
  <ClientProperty>
    <name>xlink:href</name>
    <!-- here we can now access fields from the linked table. -->
    <!-- otherid is from the table ex_main_other_as referenced in the
FeatureTypeMapping _main_to_other -->

<value>strconcat(strconcat('https://service.datacove.eu/geoserver/ogc/features
/collections/ex:OtherFT/items/', otherid),
'?f=application%2Fgeo%2Bjson')</value>
  </ClientProperty>
</AttributeMapping>
```

7.6.3. A general note to the “FEATURE_LINK”

In most cases where we use the string “FEATURE_LINK”, this simple string suffices. However, there are cases where different types are nested under the same element. In those cases, the FEATURE_LINK can be formulated as an array, the first instance as “FEATURE_LINK[1]”, the second as “FEATURE_LINK[2]”. Note that the index must be included in both mentions of FEATURE_LINK (so both in the main as well as in the nested feature mapping)

7.7. Resources

7.7.1.Schema File

Available online at:

https://schema.datacove.eu/Example.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2019 rel. 3 (x64) (http://www.altova.com) by
Katharina Schleidt (DataCove e.U.) -->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:ex="https://schema.datacove.eu/Example"
xmlns:base="http://inspire.ec.europa.eu/schemas/base/3.3"
xmlns:base2="http://inspire.ec.europa.eu/schemas/base2/1.0"
xmlns:gml="http://www.opengis.net/gml/3.2"
targetNamespace="https://schema.datacove.eu/Example"
elementFormDefault="qualified" version="1.0.7">
        <import namespace="http://inspire.ec.europa.eu/schemas/base/3.3"
schemaLocation="http://inspire.ec.europa.eu/schemas/base/3.3/BaseTypes.xsd"/>
        <import namespace="http://inspire.ec.europa.eu/schemas/base2/1.0"
schemaLocation="http://inspire.ec.europa.eu/schemas/base2/1.0/BaseTypes2.xsd"/
>
        <import namespace="http://www.opengis.net/gml/3.2"
schemaLocation="http://schemas.opengis.net/gml/3.2.1/gml.xsd"/>
        <element name="MainFT" type="ex:MainFTType"
substitutionGroup="gml:AbstractFeature">
                <annotation>
                        <documentation>-- Name --
                                Main FeatureType
                        </documentation>
                </annotation>
        </element>
        <complexType name="MainFTType">
                <complexContent>
                        <extension base="gml:AbstractFeatureType">
                                <sequence>
                                        <element name="inspireId"
type="base:IdentifierPropertyType">
                                                <annotation>
                                                        <documentation>--
Definition --
External object identifier.</documentation>
                                                </annotation>
                                        </element>
                                        <element name="name" nillable="true"
minOccurs="1" maxOccurs="1">
                                                <annotation>
                                                        <documentation>--
Definition --
Name of the Main FeatureType</documentation>
                                                </annotation>
                                                <complexType>
                                                        <simpleContent>
                                                                <extension
base="string">

        <attribute name="nilReason" type="gml:NilReasonType"/>
                                                                </extension>
                                                        </simpleContent>
                                                </complexType>
                                        </element>
                                        <element name="code"
type="gml:ReferenceType" minOccurs="1" maxOccurs="1">
```

```
                                                <annotation>
                                                        <documentation>--
Definition --
Code from a reference list</documentation>
                                                </annotation>
                                        </element>
                                        <element name="codeMultiple"
type="gml:ReferenceType" minOccurs="1" maxOccurs="unbounded">
                                                <annotation>
                                                        <documentation>--
Definition --
Multiple Codes from a reference list</documentation>
                                                </annotation>
                                        </element>
                                        <element name="nestedDT"
type="ex:NestedDTPropertyType" minOccurs="1" maxOccurs="1">
                                                <annotation>
                                                        <documentation>--
Definition --
A nested dataType</documentation>
                                                </annotation>
                                        </element>
                                        <element name="nestedDTMultiple"
type="ex:NestedDTPropertyType" minOccurs="1" maxOccurs="unbounded">
                                                <annotation>
                                                        <documentation>--
Definition --
A nested multiple dataType</documentation>
                                                </annotation>
                                        </element>
                                        <element name="countedDT"
type="ex:OtherDTPropertyType" minOccurs="1" maxOccurs="unbounded">
                                                <annotation>
                                                        <documentation>--
Definition --
An additional nested multiple dataType. Added to the example to show how to
explicitly address individual positions within the array of provided OtherDT
instances</documentation>
                                                </annotation>
                                        </element>

                                        <element name="nestedFT"
type="ex:NestedFTPropertyType" minOccurs="1" maxOccurs="1">
                                                <annotation>
                                                        <documentation>--
Definition --
A nested featureType</documentation>
                                                </annotation>
                                        </element>
                                        <element name="nestedFTMultiple"
type="ex:NestedFTPropertyType" minOccurs="1" maxOccurs="unbounded">
                                                <annotation>
                                                        <documentation>--
Definition --
A nested multiple featureType</documentation>
                                                </annotation>
                                        </element>
```

```
                                        <element name="geometry"
type="gml:GeometryPropertyType" minOccurs="1" maxOccurs="1">
                                                <annotation>
                                                        <documentation>--
Definition --
Geometry associated to the Dummy.</documentation>
                                                </annotation>
                                        </element>
                                        <element name="other" nillable="true"
minOccurs="0" maxOccurs="unbounded">
                                                <annotation>
                                                        <documentation>--
Definition --
A link pointing to the OtherFT FeatureType. In contrast to the other
associations in this example, this one is bidirectional, both FeatureTypes are
first class citizens</documentation>
                                                        <appinfo>

        <reversePropertyName
xmlns="http://www.opengis.net/gml/3.2">ex:main</reversePropertyName>
                                                        </appinfo>
                                                </annotation>
                                                <complexType>
                                                        <sequence minOccurs="0">
                                                                <element
ref="ex:OtherFT"/>
                                                        </sequence>
                                                        <attributeGroup
ref="gml:AssociationAttributeGroup"/>
                                                        <attributeGroup
ref="gml:OwnershipAttributeGroup"/>
                                                </complexType>
                                        </element>
                                </sequence>
                        </extension>
                </complexContent>
        </complexType>
        <complexType name="MainFTPropertyType">
                <sequence minOccurs="0">
                        <element ref="ex:MainFT"/>
                </sequence>
                <attributeGroup ref="gml:AssociationAttributeGroup"/>
                <attributeGroup ref="gml:OwnershipAttributeGroup"/>
        </complexType>
        <element name="NestedFT" type="ex:NestedFTType"
substitutionGroup="gml:AbstractFeature">
                <annotation>
                        <documentation>-- Name --
                                Nested FeatureType
                        </documentation>
                </annotation>
        </element>
        <complexType name="NestedFTType">
                <complexContent>
                        <extension base="gml:AbstractFeatureType">
                                <sequence>
                                        <element name="name" nillable="true"
minOccurs="1" maxOccurs="1">
```

```
                                                <annotation>
                                                        <documentation>--
Definition --
Name of the Nested FeatureType</documentation>
                                                </annotation>
                                                <complexType>
                                                        <simpleContent>
                                                                <extension
base="string">

        <attribute name="nilReason" type="gml:NilReasonType"/>
                                                                </extension>
                                                        </simpleContent>
                                                </complexType>
                                        </element>
                                </sequence>
                        </extension>
                </complexContent>
        </complexType>
        <complexType name="NestedFTPropertyType">
                <sequence minOccurs="0">
                        <element ref="ex:NestedFT"/>
                </sequence>
                <attributeGroup ref="gml:AssociationAttributeGroup"/>
                <attributeGroup ref="gml:OwnershipAttributeGroup"/>
        </complexType>
        <element name="NestedDT" type="ex:NestedDTType"
substitutionGroup="gml:AbstractObject">
                <annotation>
                        <documentation>-- Definition --
                                Nested DataType
                        </documentation>
                </annotation>
        </element>
        <complexType name="NestedDTType">
                <sequence>
                        <element name="name" nillable="true" minOccurs="1"
maxOccurs="1">
                                <annotation>
                                        <documentation>-- Definition --
Name of the Nested DataType</documentation>
                                </annotation>
                                <complexType>
                                        <simpleContent>
                                                <extension base="string">
                                                        <attribute
name="nilReason" type="gml:NilReasonType"/>
                                                </extension>
                                        </simpleContent>
                                </complexType>
                        </element>
                </sequence>
        </complexType>
        <complexType name="NestedDTPropertyType">
                <sequence>
                        <element ref="ex:NestedDT"/>
                </sequence>
        </complexType>
```

```
        <element name="OtherDT" type="ex:OtherDTType"
substitutionGroup="gml:AbstractObject">
                <annotation>
                        <documentation>-- Definition --
                                Other DataType
                        </documentation>
                </annotation>
        </element>
        <complexType name="OtherDTType">
                <sequence>
                        <element name="name" nillable="true" minOccurs="1"
maxOccurs="1">
                                <annotation>
                                        <documentation>-- Definition --
Name of the Other DataType</documentation>
                                </annotation>
                                <complexType>
                                        <simpleContent>
                                                <extension base="string">
                                                        <attribute
name="nilReason" type="gml:NilReasonType"/>
                                                </extension>
                                        </simpleContent>
                                </complexType>
                        </element>
                </sequence>
        </complexType>
        <complexType name="OtherDTPropertyType">
                <sequence>
                        <element ref="ex:OtherDT"/>
                </sequence>
        </complexType>
        <element name="OtherFT" type="ex:OtherFTType"
substitutionGroup="gml:AbstractFeature">
                <annotation>
                        <documentation>-- Name --
                                Other FeatureType
                        </documentation>
                </annotation>
        </element>
        <complexType name="OtherFTType">
                <complexContent>
                        <extension base="gml:AbstractFeatureType">
                                <sequence>
                                        <element name="name" nillable="true"
minOccurs="1" maxOccurs="1">
                                                <annotation>
                                                        <documentation>--
Definition --
Name of the Other FeatureType</documentation>
                                                </annotation>
                                                <complexType>
                                                        <simpleContent>
                                                                <extension
base="string">

        <attribute name="nilReason" type="gml:NilReasonType"/>
                                                                </extension>
```

```
                                                        </simpleContent>
                                                </complexType>
                                        </element>
                                        <element name="main" nillable="true"
minOccurs="0" maxOccurs="unbounded">
                                                <annotation>
                                                        <documentation>--
Definition --
A link pointing to the MainFT.</documentation>
                                                        <appinfo>

        <reversePropertyName
xmlns="http://www.opengis.net/gml/3.2">ex:other</reversePropertyName>
                                                        </appinfo>
                                                </annotation>
                                                <complexType>
                                                        <sequence minOccurs="0">
                                                                <element
ref="ex:MainFT"/>
                                                        </sequence>
                                                        <attributeGroup
ref="gml:AssociationAttributeGroup"/>
                                                        <attributeGroup
ref="gml:OwnershipAttributeGroup"/>
                                                </complexType>
                                        </element>
                                </sequence>
                        </extension>
                </complexContent>
        </complexType>
        <complexType name="OtherFTPropertyType">
                <sequence minOccurs="0">
                        <element ref="ex:OtherFT"/>
                </sequence>
                <attributeGroup ref="gml:AssociationAttributeGroup"/>
                <attributeGroup ref="gml:OwnershipAttributeGroup"/>
        </complexType>
</schema>
```

7.7.2.SQL for Example Database

```
CREATE TABLE public.ex_mainft (
        id varchar(100) NOT NULL,
        gmlid varchar(100) NULL,
        metadata varchar(100) NULL,
        localid varchar(100) NULL,
        "namespace" varchar(100) NULL,
        "version" varchar(100) NULL,
        ft_name varchar(100) NULL,
        code varchar(100) NULL,
        nested_dt_name varchar(100) NULL,
        counted_dt_name1 varchar(100) NULL,
```

```
        counted_dt_name2 varchar(100) NULL,
        nested_ft_name varchar(100) NULL,
        other_ft_id varchar(100) NULL,
        geom geometry(POINT, 4326) NULL,
        CONSTRAINT ex_mainft_gmlid UNIQUE (gmlid),
        CONSTRAINT ex_mainft_pk PRIMARY KEY (id)
);

CREATE TABLE public.ex_nesteddt (
        id varchar(100) NOT NULL,
        gmlid varchar(100) NULL,
        dt_name varchar(100) NULL,
        main_ft_id varchar(100) NULL,
        CONSTRAINT ex_nesteddt_pk PRIMARY KEY (id),
        CONSTRAINT ex_nesteddt_main_ft_id_fkey FOREIGN KEY (main_ft_id)
REFERENCES ex_mainft(id)
);

CREATE TABLE public.ex_nestedft (
        id varchar(100) NOT NULL,
        gmlid varchar(100) NULL,
        ft_name varchar(100) NULL,
        main_ft_id varchar(100) NULL,
        CONSTRAINT ex_nestedft_pk PRIMARY KEY (id),
        CONSTRAINT ex_nestedft_main_ft_id_fkey FOREIGN KEY (main_ft_id)
REFERENCES ex_mainft(id)
);

CREATE TABLE public.ex_otherdt (
        id varchar(100) NOT NULL,
        gmlid varchar(100) NULL,
        dt_name varchar(100) NULL,
        main_ft_id varchar(100) NULL,
        CONSTRAINT ex_otherdt_pk PRIMARY KEY (id),
        CONSTRAINT ex_otherdt_main_ft_id_fkey FOREIGN KEY (main_ft_id)
REFERENCES ex_mainft(id)
);

CREATE TABLE public.ex_otherft (
        id varchar(100) NOT NULL,
        gmlid varchar(100) NULL,
        ft_name varchar(100) NULL,
        main_ft_id varchar(100) NULL,
        CONSTRAINT ex_otherft_gmlid UNIQUE (gmlid),
        CONSTRAINT ex_otherft_pk PRIMARY KEY (id)
);

CREATE TABLE public.ex_main_other_as (
        mainid varchar(100) NOT NULL,
        otherid varchar(100) NOT NULL,
        CONSTRAINT ex_main_ft_id_fkey FOREIGN KEY (mainid) REFERENCES
ex_mainft(gmlid),
        CONSTRAINT ex_other_ft_id_fkey FOREIGN KEY (otherid) REFERENCES
ex_otherft(gmlid)
);
```

7.7.3.App Schema Mapping for Example DB

```
<?xml version="1.0" encoding="UTF-8"?>
<as:AppSchemaDataAccess xmlns:as="http://www.geotools.org/app-schema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.geotools.org/app-schema
https://schema.datacove.eu/AppSchemaDataAccess.xsd">
        <namespaces>
                <Namespace>
                        <prefix>ex</prefix>
                        <uri>https://schema.datacove.eu/Example</uri>
                </Namespace>
                <Namespace>
                        <prefix>xlink</prefix>
                        <uri>http://www.w3.org/1999/xlink</uri>
                </Namespace>
                <Namespace>
                        <prefix>base</prefix>
                        <uri>http://inspire.ec.europa.eu/schemas/base/3.3</uri>
                </Namespace>
                <Namespace>
                        <prefix>xsi</prefix>
                        <uri>http://www.w3.org/2001/XMLSchema-instance</uri>
                </Namespace>
                <Namespace>
                        <prefix>gml</prefix>
                        <uri>http://www.opengis.net/gml/3.2</uri>
                </Namespace>
        </namespaces>
        <includedTypes>
        </includedTypes>
        <sourceDataStores>
                <DataStore>
                        <id>idDataStoreInsp</id>
                        <parameters>
                                <Parameter>
                                        <name>dbtype</name>
                                        <value>postgisng</value>
                                </Parameter>
                                <Parameter>
                                        <name>jndiReferenceName</name>
                                        <value>java:comp/env/jdbc/tnw</value>
                                        <!--          <value>jdbc/tnw</value> --
>
                                </Parameter>
                                <Parameter>
                                        <name>Expose primary keys</name>
                                        <value>true</value>
                                </Parameter>
                        </parameters>
                </DataStore>
        </sourceDataStores>
        <!--    <catalog>...</catalog>-->
        <targetTypes>
                <FeatureType>

        <schemaUri>https://schema.datacove.eu/Example.xsd</schemaUri>
                </FeatureType>
```

```
</targetTypes>
<typeMappings>
        <FeatureTypeMapping>
                <mappingName>_Nested_FT</mappingName>
                <sourceDataStore>idDataStoreInsp</sourceDataStore>
                <sourceType>ex_nestedft</sourceType>
                <targetElement>ex:NestedFT</targetElement>
                <defaultGeometry/>
                <attributeMappings>
                        <AttributeMapping>

<targetAttribute>FEATURE_LINK[1]</targetAttribute>
                                <sourceExpression>
                                        <OCQL>main_ft_id</OCQL>
                                </sourceExpression>
                        </AttributeMapping>
                        <AttributeMapping>

<targetAttribute>ex:NestedFT</targetAttribute>
                                <idExpression>
                                        <OCQL>gmlid</OCQL>
                                </idExpression>
                        </AttributeMapping>
                        <AttributeMapping>

<targetAttribute>ex:name</targetAttribute>
                                <sourceExpression>
                                        <OCQL>ft_name</OCQL>
                                </sourceExpression>
                        </AttributeMapping>
                </attributeMappings>
        </FeatureTypeMapping>
        <FeatureTypeMapping>
                <mappingName>_Nested_DT</mappingName>
                <sourceDataStore>idDataStoreInsp</sourceDataStore>
                <sourceType>ex_nesteddt</sourceType>
                <targetElement>ex:NestedDT</targetElement>
                <defaultGeometry/>
                <attributeMappings>
                        <AttributeMapping>

<targetAttribute>FEATURE_LINK</targetAttribute>
                                <sourceExpression>
                                        <OCQL>main_ft_id</OCQL>
                                </sourceExpression>
                        </AttributeMapping>
                        <AttributeMapping>

<targetAttribute>ex:name</targetAttribute>
                                <sourceExpression>
                                        <OCQL>dt_name</OCQL>
                                </sourceExpression>
                        </AttributeMapping>
                </attributeMappings>
        </FeatureTypeMapping>
        <FeatureTypeMapping>
                <sourceDataStore>idDataStoreInsp</sourceDataStore>
                <sourceType>ex_mainft</sourceType>
```

```
                        <targetElement>ex:MainFT</targetElement>
                        <defaultGeometry>ex:geometry</defaultGeometry>
                        <attributeMappings>
                                <AttributeMapping>

        <targetAttribute>ex:MainFT</targetAttribute>
                                        <idExpression>
                                                <OCQL>gmlid</OCQL>
                                        </idExpression>
                                </AttributeMapping>
                                <AttributeMapping>

        <targetAttribute>ex:inspireId/base:Identifier/base:localId</targetAttr
ibute>
                                        <sourceExpression>
                                                <OCQL>localid</OCQL>
                                        </sourceExpression>
                                </AttributeMapping>
                                <AttributeMapping>

        <targetAttribute>ex:inspireId/base:Identifier/base:namespace</targetAt
tribute>
                                        <sourceExpression>
                                                <OCQL>namespace</OCQL>
                                        </sourceExpression>
                                </AttributeMapping>
                                <AttributeMapping>

        <targetAttribute>ex:inspireId/base:Identifier/base:versionId</targetAt
tribute>
                                        <sourceExpression>
                                                <OCQL>version</OCQL>
                                        </sourceExpression>
                                </AttributeMapping>
                                <AttributeMapping>

        <targetAttribute>ex:name</targetAttribute>
                                        <sourceExpression>
                                                <OCQL>ft_name</OCQL>
                                        </sourceExpression>
                                </AttributeMapping>
                                <AttributeMapping>

        <targetAttribute>ex:code</targetAttribute>
                                        <encodeIfEmpty>true</encodeIfEmpty>
                                        <ClientProperty>
                                                <name>xlink:href</name>

        <value>strconcat('http://codes.datacove.eu/', code)</value>
                                        </ClientProperty>
                                </AttributeMapping>
                                <AttributeMapping>

        <targetAttribute>ex:geometry</targetAttribute>
                                        <idExpression>
                                                <OCQL>strconcat('PT_',
gmlid)</OCQL>
                                        </idExpression>
```

```
                                        <sourceExpression>
                                                <OCQL>geom</OCQL>
                                        </sourceExpression>
                                </AttributeMapping>
                                <AttributeMapping>

        <targetAttribute>ex:nestedDT/ex:NestedDT/ex:name</targetAttribute>
                                        <sourceExpression>
                                                <OCQL>nested_dt_name</OCQL>
                                        </sourceExpression>
                                </AttributeMapping>
                                <AttributeMapping>

        <targetAttribute>ex:countedDT[1]/ex:OtherDT/ex:name</targetAttribute>
                                        <sourceExpression>
                                                <OCQL>counted_dt_name1</OCQL>
                                        </sourceExpression>
                                </AttributeMapping>
                                <AttributeMapping>

        <targetAttribute>ex:countedDT[2]/ex:OtherDT/ex:name</targetAttribute>
                                        <sourceExpression>
                                                <OCQL>counted_dt_name2</OCQL>
                                        </sourceExpression>
                                </AttributeMapping>
                                <AttributeMapping>

        <targetAttribute>ex:nestedFT/ex:NestedFT</targetAttribute>
                                        <ClientProperty>
                                                <name>gml:id</name>
                                                <value>strconcat('nestedFT_',
id)</value>
                                        </ClientProperty>
                                </AttributeMapping>
                                <AttributeMapping>

        <targetAttribute>ex:nestedFT/ex:NestedFT/ex:name</targetAttribute>
                                        <sourceExpression>
                                                <OCQL>nested_ft_name</OCQL>
                                        </sourceExpression>
                                </AttributeMapping>
                                <AttributeMapping>

        <targetAttribute>ex:nestedFTMultiple</targetAttribute>
                                        <sourceExpression>
                                                <OCQL>id</OCQL>

        <linkElement>_Nested_FT</linkElement>

        <linkField>FEATURE_LINK[1]</linkField>
                                        </sourceExpression>
                                </AttributeMapping>
                                <AttributeMapping>

        <targetAttribute>ex:nestedDTMultiple</targetAttribute>
                                        <sourceExpression>
                                                <OCQL>id</OCQL>
```

```
        <linkElement>_Nested_DT</linkElement>

        <linkField>FEATURE_LINK</linkField>
                                    </sourceExpression>
                                </AttributeMapping>

                                <AttributeMapping>
                                  <targetAttribute>ex:other</targetAttribute>
                                  <encodeIfEmpty>true</encodeIfEmpty>
                                  <sourceExpression>
                                        <OCQL>gmlid</OCQL>

        <linkElement>_main_to_other_</linkElement>
                                        <linkField>FEATURE_LINK</linkField>
                                  </sourceExpression>
                                  <isMultiple>true</isMultiple>
                                  <ClientProperty>
                                        <name>xlink:href</name>

        <value>strconcat(strconcat('https://service.datacove.eu/geoserver/ogc/
features/collections/ex:OtherFT/items/', otherid),
'?f=application%2Fgeo%2Bjson')</value>
                                  </ClientProperty>
                                </AttributeMapping>


                        </attributeMappings>
                </FeatureTypeMapping>


    <FeatureTypeMapping>
      <mappingName>_main_to_other_</mappingName>
      <sourceDataStore>idDataStoreInsp</sourceDataStore>
      <sourceType>ex_main_other_as</sourceType>
      <targetElement>ex:OtherFT</targetElement>
      <attributeMappings>
        <encodeIfEmpty>true</encodeIfEmpty>
        <AttributeMapping>
          <targetAttribute>FEATURE_LINK</targetAttribute>
          <!-- FK in linked table -->
          <sourceExpression>
            <OCQL>mainid</OCQL>
          </sourceExpression>
        </AttributeMapping>
        <AttributeMapping>
          <targetAttribute>ex:other</targetAttribute>
        </AttributeMapping>
      </attributeMappings>
    </FeatureTypeMapping>


    <FeatureTypeMapping>
      <mappingName>_other_to_main</mappingName>
      <sourceDataStore>idDataStoreInsp</sourceDataStore>
      <sourceType>ex_main_other_as</sourceType>
      <targetElement>ex:MainFT</targetElement>
      <attributeMappings>
```

```
        <encodeIfEmpty>true</encodeIfEmpty>
        <AttributeMapping>
          <targetAttribute>FEATURE_LINK</targetAttribute>
          <!-- FK in linked table -->
          <sourceExpression>
            <OCQL>otherid</OCQL>
          </sourceExpression>
        </AttributeMapping>
        <AttributeMapping>
          <targetAttribute>ex:main</targetAttribute>
        </AttributeMapping>
      </attributeMappings>
    </FeatureTypeMapping>

                <FeatureTypeMapping>
                        <mappingName>_Other_FT</mappingName>
                        <sourceDataStore>idDataStoreInsp</sourceDataStore>
                        <sourceType>ex_otherft</sourceType>
                        <targetElement>ex:OtherFT</targetElement>
                        <defaultGeometry/>
                        <attributeMappings>
                                <AttributeMapping>

        <targetAttribute>ex:OtherFT</targetAttribute>
                                        <idExpression>
                                                <OCQL>gmlid</OCQL>
                                        </idExpression>
                                </AttributeMapping>
                                <AttributeMapping>

        <targetAttribute>ex:name</targetAttribute>
                                        <sourceExpression>
                                                <OCQL>ft_name</OCQL>
                                        </sourceExpression>
                                </AttributeMapping>
                                <AttributeMapping>
                                  <targetAttribute>ex:main</targetAttribute>
                                  <encodeIfEmpty>true</encodeIfEmpty>
                                  <sourceExpression>
                                        <OCQL>gmlid</OCQL>

        <linkElement>_other_to_main</linkElement>
                                        <linkField>FEATURE_LINK</linkField>
                                  </sourceExpression>
                                  <isMultiple>true</isMultiple>
                                  <ClientProperty>
                                        <name>xlink:href</name>

        <value>strconcat(strconcat('https://service.datacove.eu/geoserver/ogc/
features/collections/ex:MainFT/items/', mainid),
'?f=application%2Fgeo%2Bjson')</value>
                                  </ClientProperty>
                                </AttributeMapping>
                        </attributeMappings>
                </FeatureTypeMapping>

        </typeMappings>
</as:AppSchemaDataAccess>
```