OGC Testbed-13

NAS Profiling ER

Table of Contents

Publication Date: YYYY-MM-DD

Approval Date: YYYY-MM-DD

Posted Date: 2017-12-03

Reference number of this document: OGC 17-020

Reference URL for this document: http://www.opengis.net/doc/PER/t13-NG003

Category: Public Engineering Report

Editors: Johannes Echterhoff, Clemens Portele

Title: OGC Testbed-13: NAS Profiling ER

OGC Engineering Report

COPYRIGHT

WARNING

LICENSE AGREEMENT

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER’S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR’s sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to

indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Chapter 1. Summary

The National System for Geospatial-Intelligence (NSG) Application Schema (NAS) is an ISO 19109 compliant application schema that defines the conceptual model for identifying and encoding feature data in the U.S. National System for Geospatial-Intelligence (NSG). NGA utilizes the open source software tool ShapeChange [http://shapechange.net] as an integral piece in NAS development. This tool is used to take NAS-based UML models and create Extensible Markup Language (XML) and Resource Description Framework (RDF) based schemas. Testbed-12 began development of capabilities for extracting profiles supporting specific mission functions from the full NAS content. Testbed-13 further refined the approach to NAS Profiling by investigating how a specific profile ("Urban Military Profile") can be processed in an automated way and used to derive implementation schemas for the OGC standards CDB and CityGML.

This OGC Engineering Report describes:

- The specification of a NAS-based Military Urban Profile as a Unified Modeling Language (UML) model (chapter 5);
- How mission-specific sub-profiles can be specified and maintained using ShapeChange and the new ShapeChange Profile Management Tool (chapter 6); and
- How the model and profile information are processed to derive output for
  - a CDB data store (chapter 7, chapter 8) and
  - a CityGML Application Domain Extension (chapter 9).

This work provides insights into:

- The requirements and constraints on managing profiles of complex ISO 19109 compliant application schemas such as the NAS; and
- Using a model-driven approach to generate implementation schemas of an ISO 19109 compliant application schema profile for different environments.

The target audience of this document is anyone interested in these topics. The implementation environments discussed in this report are the OGC standards CDB and CityGML. The profiled application schema is the NAS.

This report assumes that readers are familiar with the key concepts and technologies discussed in this document. This document does not provide an introduction to them, but the table below provides a brief summary and pointers to more information.

Table 1. Key concepts and technologies discussed in this document

Topic	Overview and additional information
Modeling spatial data	The content and structure of spatial data along with any constraints important for the integrity of the data is typically specified in an **application schema**, a conceptual schema for data required by one or more applications. The fundamental unit of spatial data is the **feature**, which is a digital record of anything in the real-world that matters for the target applications. An application schema, therefore, models features. Application schemas are modeled as specified by ISO 19109 (Rules for application schemas) [https://www.iso.org/standard/59193.html], typically using UML as the formal language to describe the schema. **NOTE** A **feature** modeled according to ISO 19109 can, but does not need to have, geometric properties. This is an important distinction to certain implementation technologies, where a feature must have at least one, and often exactly one, geometric property. **NOTE** ISO has also specified recently how to describe application schemas using OWL, but this approach has limitations since the base types used in application schemas for spatial and temporal geometry or topology, metadata, coordinate reference systems, etc. are only specified in the ISO standards - using UML diagrams. The associated UML model for those types ("ISO/TC 211 Harmonized Model") is incomplete, too, as it, for example, lacks documentation for most elements in the model. OWL ontologies derived from that model exist, but include many UML artifacts and do not reuse existing ontologies used by the semantic web community. Two aspects are essential here: 1. A machine-readable data description is required for automated mechanisms for data management. 2. A schema on the conceptual level greatly helps to interpret data and transform it consistently between different representations, for example, between an XML or JavaScript Object Notation (JSON) file used for data exchange between systems and a tabular database used to store and maintain the data.

Topic	Overview and additional information
Model-driven software development	A software development methodology that focuses on creating and exploiting models on a conceptual level (e.g., an ISO 19109 application schema). As conceptual schemas, these aim at abstract representations of the knowledge and activities that govern a particular application domain, rather than the computing concepts to handle data. Once mature, such models can be used to consistently derive implementation schemas (like XML schemas, JSON schemas, SQL DDL, Esri Geodatabases, etc.), code (like Java class stubs and interfaces) or other representations (like documentation) in an automated way and using consistent rules. The Object Management Group (OMG) [http://www.omg.org/mda/] introduced the terms Platform Independent Model (PIM) and Platform Specific Model (PSM) to distinguish between models that are independent of a particular technology or platform and those that describe how that model could be realized on a particular type of platform. This is related to item 2 in the previous row. For spatial data, the focus in standardization has mostly been on deriving implementation schemas for XML-based data exchange. The concepts are specified in ISO 19118 (Encoding) [https://www.iso.org/standard/44212.html]. The OGC Geography Markup Language (GML) [http://portal.opengeospatial.org/files/?artifact_id=20509] specifies in Annex E standard rules to convert an application schema in UML to a GML application schema (XML Schema).

Topic	Overview and additional information
ShapeChange	ShapeChange [http://shapechange.net] is a tool that supports model driven software development using ISO 19109 application schemas. Originally developed in the early 2000s to support the automated generation of GML application schemas from ISO 19109 application schemas in UML it has since then been extended to support additional targets including • JSON schema • SQL DDL • Esri Geodatabases • RDF/OWL • other XML encoding rules • Schematron • documentation in HTML or DOCX The standard encoding rule like the one in Annex E of GML (see above) have been extended to support additional conversions to implementation schemas, often supporting richer UML profiles used by communities. This is also the case for the NAS, see below. ShapeChange also supports transformations of models. Often the transformations are used to convert a conceptual, platform-independent model to a platform-specific model. A commonly used transformation is the Flattener [http://shapechange.net/transformations/flattener/], which can be used to simplify complex data structures, for example for use in tabular data structures like in SQL, Esri Geodatabases and the GML Simple Feature Level 0 Profile.
NAS	The National System for Geospatial Intelligence (NSG) Application Schema (NAS) [https://nsgreg.nga.mil/nas/] specifies an NSG-wide model for geospatial data. Part 1 of the NAS specifies a technology-neutral logical Platform Independent Model that determines the syntactic structure used to represent the semantics specified by the NSG Entity Catalog (NEC). From it, using Model Driven Architecture (MDA) techniques, technology-tied Platform Specific Models (PSM) may be automatically derived and directly employed in system development. This includes, for example, GML application schemas. ShapeChange has been used in several OGC testbeds to derive and enhance the processes for deriving such PSMs for the NAS. Several additional conversion rules have been specified - and implemented in ShapeChange - to derive adequate PSMs for the NAS.

Topic	Overview and additional information
NAS Profiles	As the NAS specifies an NSG-wide model for geospatial data that supports a wide variety of domains and missions it is advantageous to define subsets of the NAS that meet specific requirements. These NAS profiles may be published and then used to develop a variety of derived artifacts. See here [https://nsgreg.nga.mil/gcsr/profiles.jsp] for more information. In OGC Testbed-12, profiling of ISO 19109 application schemas and especially the NAS was analyzed [1]. See in particular the chapter Profiling [http://docs.opengeospatial.org/per/16-020.html#section_profiling].

Topic	Overview and additional information
Feature types vs. object types	In addition to features, an application schema may model data types (structured data, without identity), unions (data types with choices between a set of options), enumerations (closed list of codes), and code lists (open list of codes). The UML profiles of some application schemas also contain another type of classes, so called "object types", to represent classes that are not feature types. These classes have no stereotype, or a specific stereotype like <<type>>. Typically, such object types represent plain objects with identity. They are used to encode structured data in a referenceable way (since an object has identity). Some application schemas have a more specific characterization of what an object type represents. In the NAS, for example, object types do not have geometry, and support analytical use cases. NAS feature types, on the other hand, do have geometry, and support mapping and visualization use cases. ShapeChange distinguishes between feature and object types. Transformation and conversion rules can treat these two types of classes differently, depending upon the intended outcome. However, ShapeChange does not assume that feature types always have geometry, and that object types do not. When processing an application schema with ShapeChange, it is important to understand which modeling constructs can be represented in a specific output (such as a feature catalogue, an SQL DDL schema, an XML Schema, an ontology, or an ArcGIS workspace), how the representation looks like, and the implications. In an SQL DDL schema, for example, feature and object types can be represented by tables. In an ontology, they are both represented by OWL classes. In an ArcGIS workspace, on the other hand, features are considered to be objects whose spatial extent is given by a single simple feature geometry [http://portal.opengeospatial.org/files/?artifact_id=25355]. Objects that do not have a spatial extent are represented by object classes. When deriving implementation schemas from a given application schema, care must be taken on how the feature and object types from the application schema are processed and how they will eventually be represented in the implementation schema.

Topic	Overview and additional information
CDB	The OGC CDB standard [http://www.opengeospatial.org/standards/cdb] defines a standardized model and structure for a single, "versionable", virtual representation of the earth. A CDB structured data store provides for a geospatial content and model definition repository that is plug-and-play interoperable between database authoring workstations. CDB was approved as an OGC standard in 2016.
CityGML	CityGML [http://www.opengeospatial.org/standards/citygml] is an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is a GML application schema. The aim of the development of CityGML is to reach a common definition of the basic entities, attributes, and relations of a 3D city model. This is especially important with respect to the cost-effective sustainable maintenance of 3D city models, allowing the reuse of the same data in different application fields. A CityGML Application Domain Extension (ADE) specifies additions to the CityGML data model. Such additions comprise the introduction of new properties to existing CityGML classes or the definition of new features types in a new XML namespace. Section 10.13 in CityGML 2.0 specifies rules for ADEs. The relationship between the NAS and CityGML has already been investigated in OGC Testbed-6 in 2009 - using the current versions of the NAS and CityGML at that time. For the NAS, an urban profile called UTDS was used. The findings are documented in the report OWS-6 UTDS-CityGML Implementation Profile.

1.1. Requirements

The following requirements have been addressed by the work documented in this Engineering Report:

- Extend ShapeChange to support generating (NAS-based) output to configure the feature data in a CDB dataset consistent with a NAS profile UML model
- Create the files that specify the structure and content of a vector dataset based on a NAS profile UML model for use in CDB
- Extend ShapeChange to support generating a schema representing a CityGML ADE from a NAS profile UML model
- Create a CityGML ADE from a NAS profile UML model
- Document the requirements for NAS profile UML models to derive the CDB and CityGML outputs

1.2. Prior-After Comparison

1.2.1. Related OGC Working Groups

Work on the NAS has been reported to the Defense & Intelligence (D&I) Domain Working Group (DWG) on several occasions. The NAS Profiling work will be reported to this DWG.

The current situation in relevant OGC Standards Working Groups (SWG) is:

- CDB SWG:
  - The current version OGC CDB 1.0 supports only a fixed set of feature types in Tiled Vector Datasets, supports no schema information and has limitations inherited from the use of the Digital Geographic Information Exchange Standard (DIGEST) [https://www.dgiwg.org/digest/] as the starting point for organizing features and attributes and from the use of Shapefiles for storing feature tiles.
  - Change Requests for CDB 2.0 may be submitted until December 31, 2017.
- CityGML SWG:
  - Section 10.13 in CityGML 2.0 specifies rules for ADEs. An ADE is an XML schema that imports CityGML schemas and extends them in either of two ways:
    - New feature types are defined within the ADE namespace and are based on CityGML feature types (abstract or concrete).
    - Existing CityGML feature types are extended by application specific properties (global property elements in the ADE namespace).
  - Beyond this, CityGML does not specify any constraints on how XML Schema may be used in an ADE.

Since the findings and recommendations may also be of interest to these SWGs, they will also be communicated to them.

1.2.2. Summary of the findings and recommendations

General remarks

This section summarizes the findings and recommendations related to the CDB and CityGML standards and their implementations.

The findings and recommendations related to CDB are more specific than they are for CityGML. The main reason is that the NAS is in some ways closer to CityGML than it is to CDB.

CityGML is conceptually and technically based on the same OGC and ISO standards that also underpin the NAS. This includes the capability for defining application schemas according to the General Feature Model, using encodings supporting such application schemas, etc.

This is not the case for CDB, which uses a different framework, has no support for application schemas and very limited capabilities to support user-defined classification of feature data, etc.

Therefore, not surprisingly the work documented in this ER has identified more clearly what information from the NAS (or similar application schemas) can or cannot be represented properly in CDB. There are far less constraints to consider in the transformation from NAS to CityGML than there are in the transformation from NAS to CDB.

Out-of-scope work items for this testbed activity are:

- What are the typical implementation efforts required to transform NAS data to CDB or CityGML?
- What are the typical implementation efforts required to support such data in CDB implementations?
- What are the typical implementation efforts required to support such data in CityGML implementations (server and clients)?
- What is the impact, if another profile of the NAS is used?
- What are the criteria for transforming NAS into CDB or CityGML and using/developing implementations supporting these standards over using/developing implementations that support NAS data directly?

A future work item recommendation is building on the results of this testbed and investigating the above questions.

CDB

As CDB is now an OGC standard, it seems reasonable to consider if there are potential benefits from improving the alignment between CDB and the OGC standards baseline / the common practices in the OGC community.

One aspect is related to the scope of CDB. If the focus of CDB is not only performant visualization, but also includes support for spatial analytics and feature data, the CDB SWG should consider supporting the ISO/OGC General Feature Model and the concept of application schemas as specified in ISO 19109.

Currently, CDB does not support application schemas and is limited to only a fixed set of feature types and feature properties are limited to simple values (text or numbers).

Potential support for application schemas in CDB 2.0 would imply that implementations would need to support more flexible and potentially richer content. That is, there would also be a cost involved, which needs to be balanced with the benefits from the new capability.

When considering adding support for application schemas in CDB, several aspects need consideration:

- Encoding:
  Currently, CDB uses Shapefiles to encode feature data. In the future, CDB should support other encodings, too, including standardized encodings and encodings that support descriptions of the schema of the data. Potential candidates include GML, Esri Geodatabases, GeoPackage, etc. The NAS specifies encodings for GML and Esri Geodatabases. Since no NAS data was used in CDB in Testbed-13, no recommendations about future encodings were discussed in Testbed-13.

NOTE	GeoJSON is another candidate for a commonly used encoding, but it needs to be understood that GeoJSON is by design essentially schema-less.

- Schema description:
  Each encoding that supports application schemas comes with rules on how to express the application schema in an implementation schema for the encoding. In GeoPackage and Esri Geodatabases, the schema information is stored in tables, for GML the schema is provided as an XML schema ("GML application schema").
  Since no encoding recommendation was made as part of the NAS profiling work in Testbed-13, one approach had to be selected for expressing a schema for NAS data in CDB. It was decided to specify the NAS profile as a GML application schema, because
  - GML is already one of the encodings of NAS data,
  - the conversion of an application schema to a GML application schema is well-defined, and
  - GML application schemas can support simple schemas (GML Simple Feature Level 0 profile) as well as more complex schemas.
- Schema complexity, rich content:
  For Testbed-13, it was decided to limit the complexity to the GML Simple Features Level 0 Profile. This decision was taken for practical reasons: to be compatible with the complexity supported by Shapefiles and, therefore, implementations of OGC CDB 1.0. Chapter 7 has details about the limitations and constraints of such an approach for representing NAS data that makes use of the rich capabilities of the application schema, for example, object metadata or other objects without geometry that are shared between multiple features. If NAS data does not make use of these capabilities, the limitations are less significant.
  If CDB is not mainly about performant visualization, the CDB SWG should consider supporting richer data representations, including, for example, objects that are not features.
- Feature and attribute dictionaries:
  Currently, semantic information about CDB feature data is provided through feature and attribute dictionaries. The feature dictionary is fixed and specified as part of the standard. For attributes, a pre-defined list of attributes is part of the standard, but additional attributes may be defined. See Chapter 8 for details.
  For cases where an application schema is provided, it should be clarified if the feature and attribute dictionaries are still required and, if the answer is "yes", what their relationship is.
  Note that CDB dictionaries currently do not support a capability to express relationships between features.
  Whatever the role of the dictionaries is in a future CDB standard, several changes should be considered:
  - Remove the DIGEST legacy, including
    - the rigid patterns for codes of feature and attribute types,
    - the fixed two-level hierarchy for categories and sub-categories of feature types and their codes
    - the rules for subtypes of feature types
  - Clarify how multiple dictionaries should be handled, for example it is unclear if units of measure need to be redefined in every attribute dictionary and how identifier conflicts are

handled.

See also the OGC Testbed-13 CDB Engineering Report for discussion about additional aspects of CDB.

CityGML

As discussed in Chapter 9 the expectation for a CityGML Application Domain Extension (ADE) for a NAS profile is that it should reflect the conceptual model as well as possible. That is, transformations to simplified structures in the implementation schema should be used only where required by CityGML.

Another aspect to take into account is that it is the expectation in the CityGML community that tool support for an ADE will always require software development.

A consequence of this is that an ADE is an option for a profile of the NAS that is standardized by the community or as part of the NSG, but less for a "mission-specific" profile of the NAS that is created for a specific dataset or data collection effort.

The general question of how ADEs should best be specified seems to be a work-in-progress. Different ADEs under development are taking different approaches.

A related question is, if CityGML would benefit from a conformance class that states which modelling capabilities may be used in an ADE so that CityGML-aware software supporting that ADE conformance class could be expected to handle it without software development (similar to the Simple Feature profiles for GML). This would allow the generation of ADEs for "mission-specific" NAS profiles, too.

NOTE	This is only raised as a question and is not phrased as a recommendation or change request as an analysis of the potential benefits is out-of-scope of the worked documented in this report.

If an ADE has been defined, and software is available that supports this ADE, then defining "mission-specific" profiles of the ADE to strictly subset the content could be possible following an approach similar to GML profiles or metadata profiles. Further details are provided in the future work section on CityGML ADE Profiles.

In any ADE for a NAS profile, due to the broad thematic coverage in the NAS, many features will typically be subtypes of _CityObject. A question that may be debated is, should these be part of a CityGML ADE or to what extent is this still proper CityGML data? Is this more useful than keeping the data in the NAS GML encoding? This discussion is, in particular, relevant for objects that are not features in the NAS and which do not include a geometry property (for example, almost all of Human Geography). Are CityGML implementations able to process / make use of such objects? To answer such questions it would be important to analyze the benefits or problems by using NAS data that has been converted to a NAS-based CityGML ADE in real scenarios.

Finally, it should be pointed out that the question of which information from the NAS maps to which element in the CityGML schemas is to some extent uncertain as the CityGML feature types have vague semantics. The model does not include any definitions for the types or their properties. That is, any mapping relies on an interpretation based exclusively on the names of the model elements. This is a known open issue that was also identified during a similar exercise in OGC

Testbed-6 (see Change Request 09-039, item 1, Lack of definitions [https://portal.opengeospatial.org/files/?artifact_id=33758]).

1.3. What does this ER mean for the Working Group and OGC in general

This ER is mainly informational and is not a proposal for a new OGC standard, but about applying standards - and learning more about how well they fit together. Due to the strong link to NGA's NSG application schema (NAS), the work is expected to be of interest to the Defense & Intelligence DWG.

In addition, it should be of interest to the CityGML SWG and the CDB SWG as this ER discusses how to use profiles of the NAS and data captured for such profiles in applications based on the CDB and CityGML standards.

This ER may also be of interest for others following or considering a model-driven-approach using ISO 19109 application schemas in UML and who are interested in the topics of profiling, CityGML or CDB.

1.4. Document contributor contact points

All questions regarding this document should be directed to the editors:

Table 2. Contacts

Name	Organization
Johannes Echterhoff (editor)	interactive instruments GmbH
Clemens Portele (editor)	interactive instruments GmbH

The editors thank Paul Birkel, Dave Wesloh, Arne Schilling, Terry Idol, and Carl Reed for their reviews and constructive comments.

1.5. Future Work

1.5.1. Use NAS data in CDB and CityGML implementations

General remarks

The NAS profiling work in Testbed-13 was mainly a desk study. Testing rich NAS data, including data that makes use of objects that are not features, in implementations of the CDB and CityGML standards would be helpful to validate - or invalidate - the findings and recommendations in this report and provide firmer recommendations to NGA, the CDB SWG and the CityGML SWG.

In order to provide recommendations to NGA and others with similar application schemas like the NAS, the scope of such future work should include analyzing the practical benefits and obstacles as well as implementation efforts for all parts of the workflow of preparing and using a specific NAS dataset in CDB or CityGML implementations.

A ShapeChange-related aspect in such future work would be to identify and specify clearly the

parameters/options that one wants to vary when deriving an ADE or GML-SF0 profile for a specific NAS profile for a specific use case. This could then be used to design and implement mechanisms that make it simple to select/switch these options when configuring ShapeChange configurations for such workflows.

This report includes sections that describe how the ShapeChange configurations created in this testbed may be adapted to other NAS profiles or use cases: for GML Simple Feature Level 0 Profiles, CDB dictionaries and CityGML ADEs. In particular the process to adapt the configuration for deriving a GML Simple Feature Level 0 Profile currently requires a good understanding of the standards, the NAS and ShapeChange.

CDB

The work documented in the report analyzed how CDB might be improved to support data according to the NSG application schema (NAS), its profiles or - in fact - any other application schema according to ISO 19109. The findings and recommendations should be considered by a future testbed that uses NAS data in CDB implementations and/or CDB SWG.

If the CDB SWG or a future testbed would consider alternative encodings for feature data, then it needs to be agreed how to specify the implementation schema for an application schema like a NAS profile. For example, for GeoPackage a database template with the schema information but no features could be specified. For JSON, a JSON Schema could be specified.

In order to support model-driven workflows, ShapeChange could then be enabled to derive such application schemas - either building on an existing capability (like the targets for JSON Schema or Esri Geodatabases) or developing a new target (e.g., for GeoPackage) to support the technical exploration in the testbed.

CityGML

As documented in the findings related to CityGML generating ADEs for NAS profiles seems to be adequate mainly for standardized profiles. Exploring the use of NAS data in CityGML would allow us to take a closer look at this conclusion as well as at the design decisions taken in this testbed to derive CityGML ADEs for NAS profiles using a repeatable approach.

1.5.2. Evaluation of OCL constraints that restrict place representations

The NAS uses Object Constraint Language (OCL) constraints to restrict how the position(s) of a feature can be represented: by points, curves, surfaces, physical addresses, and location by identifier.

NOTE	OCL is a formal language for defining rules that apply to UML models.

One goal of the derivation of a GML-SF0 NAS profile schema in Testbed-13 was that the schema only contained feature types with a single geometry property. To generate such feature types, restrictions on place representations were extracted from OCL constraints.

Since the implementation of an OCL evaluator to identify such restrictions was not a goal for Testbed-13, an expedient approach was chosen. It relies on specific patterns in the naming and

structure of relevant OCL constraints. If the OCL expressions that restrict place representations did not follow a specific pattern, this approach would not be viable. To support such a scenario, the development of an OCL evaluator to identify place restrictions should be considered.

1.5.3. Analyzing relevance of OCL constraints for GML-SF0 schema

In Testbed-13, only specific information of OCL constraints of the NAS profile is used to generate the GML-SF0 NAS profile schema. The workflow to generate the GML-SF0 schema extracts this information and then transforms all OCL constraints so that they are no longer available for generating Schematron assertions.

It could be useful to perform a detailed analysis of the NAS OCL constraints, to determine which of them would be relevant for the creation of Schematron assertions for the GML-SF0 NAS profile schema. The analysis should also look at how such constraints could be transformed to be valid in the transformed schema.

As an example, consider constraints that limit a codelist-range during inheritance. If transformed appropriately when deriving the GML-SF0 schema, Schematron assertions could be generated to more thoroughly validate GML-SF0 XML instance data.

1.5.4. Extending the feature set for application schema profiling

Chapter 6 documents how application schema profiles can be defined using ShapeChange and the ShapeChange Profile Management Tool (PMT). The functionality was demonstrated in Testbed-13. A number of suggestions for improving the capabilities of the tools were discussed:

- Handling of association classes by the PMT should be improved. For example, if an association class is removed from the profile, the corresponding association (and roles) should be removed as well. Same for adding an association class.
- The PMT should implement the design for *profiling constraints* which has been developed in Testbed 12 (see the Testbed 12 ShapeChange ER, chapter 7). Special cases like generating OCL constraints to specify that a property from a supertype is not allowed for a specific subtype could then be considered as well.
- The profiling design developed in Testbed 12 contained additional profile parameters, which the PMT should implement, such as setting the abstractness of a class.
- Additional views could be developed to simplify the process of defining a profile. For example, specific categories of classes could be hidden. A user would then be able to only work with feature types. Another simplification would be to hide the complex UML package structure within an application schema, instead listing all schema classes directly in the application schema package.

1.5.5. CityGML ADE Profiles

Creating a CityGML ADE for a "mission-specific" profile of an application schema, in an ad-hoc fashion, is considered to not be feasible since there is an expectation that support for an ADE will always require development of ADE specific software.

However, if the purpose of such a profile is just validation of content restrictions, two approaches

could be feasible:

- Create and use an ADE schema defining the subset, and reference it similar to how it is done for GML profile schemas. - A GML profile schema restricts the GML options for representing geometry. The GML profile schema is referenced through an annotation within a GML application schema. Software that understands GML profiles read the annotation, and use the GML profile schema to validate the geometries in XML data that conforms to the application schema. Software that does not understand GML profiles simply ignores the annotation and omits the additional validation step. The difference for CityGML ADE profiles would be that the reference of the ADE profile would not be contained in another ADE schema but in actual CityGML data.
- Create and use a separate Schematron schema to define and check the content restrictions. - This approach would not require the creation of new XML Schemas for "mission-specific" ADE profiles (which would have to have the same target namespace as the full ADE, and can therefore lead to confusion). The approach is similar to how metadata profiles are typically defined.

If CityGML shall support "mission-specific" ADE profiles, these two approaches should be investigated in the future.

1.6. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Chapter 2. References

- ISO 19103:2015, Geographic information - Conceptual schema language, December 2015
- ISO 19107:2003, Geographic information - Spatial schema, May 2003
- ISO 19109:2015, Geographic information - Rules for application schema, December 2015
- ISO 19115-1:2014, Geographic information - Metadata - Part 1: Fundamentals, April 2014
- ISO 19136:2007, Geographic information - Geography Markup Language (GML), September 2007
- OGC 15-113r3, Volume 1: OGC CDB Core Standard: Model and Physical Data Store Structure, Version 1.0, February 2017
- OGC 16-007r3, Volume 11: OGC CDB Core Standard Conceptual Model, Version 1.0, February 2017
- OGC 12-019, OGC City Geography Markup Language (CityGML) Encoding Standard, Version 2.0, April 2012

Chapter 3. Terms

3.1. Abbreviated terms

API

Application Program Interface

ADE

CityGML Application Domain Extension

CDB

Common Database (OGC standard)

DDL

Data Description Language

DIGEST

Digital Geographic Information Exchange Standard

DWG

(OGC) Domain Working Group

EA

Enterprise Architect (UML modeling tool from Sparx Systems)

GML

Geography Markup Language (OGC/ISO standard)

GML-SF

GML Simple Features Profile (OGC standard)

GML-SF0

GML Simple Features Profile, Level 0

GSIP

GEOINT Structure Implementation Profile

HTML

Hypertext Markup Language (W3C standard)

ISO

International Organization for Standardization

JSON

JavaScript Object Notation (IETF standard)

MDA

Model Driven Architecture

NAS

NSG Application Schema

NEC

NSG Entity Catalog

NGA

(U.S.) National Geospatial-Intelligence Agency

NSG

(U.S.) National System for Geospatial-Intelligence

OCL

Object Constraint Language (OMG standard)

OGC

Open Geospatial Consortium

OMG

Object Management Group

OWL

Web Ontology Language (W3C standard)

PIM

Platform Independent Model

PMT

(ShapeChange) Profile Management Tool

PSM

Platform Specific Model

RDF

Resource Description Framework (W3C standard)

SWG

(OGC) Standards Working Group

SQL

Structured Query Language

S-UTDS

Specialized-Urban Topographic Data Store

TDAS

Topographic Data Application Schema

TDS

Topographic Data Store

UML

Unified Modeling Language (OMG standard)

URI

Uniform Resource Identifier

URL

Uniform Resource Locator

UTDS

Urban Topographic Data Store

W3C

World Wide Web Consortium

XML

Extensible Markup Language (W3C standard)

XSD

XML Schema Definition Language (W3C standard)

Chapter 4. Overview

NGA utilizes the open source software tool ShapeChange [http://shapechange.net] as an integral piece in NAS development. This tool is used to take NAS-based UML models and create XML and RDF based schemas. Testbed-12 began development of capabilities for extracting profiles supporting specific mission functions from the full NAS content. Testbed-13 has further refined the approach to profiling the NAS by looking into a repeatable capability to derive implementation schemas for CDB and CityGML for such profiles.

In Testbed-13, an "Urban Military Profile" of the NAS was used as a starting point. This profile defines the vector content requirements for urban centric profiles. This profile is intended to be capable of being used to define the data model requirements for an Urban Military CDB and simultaneously an Urban Military Application Domain Extension (ADE) for use in CityGML.

The following describes the NAS profiling workflow executed in Testbed-13.

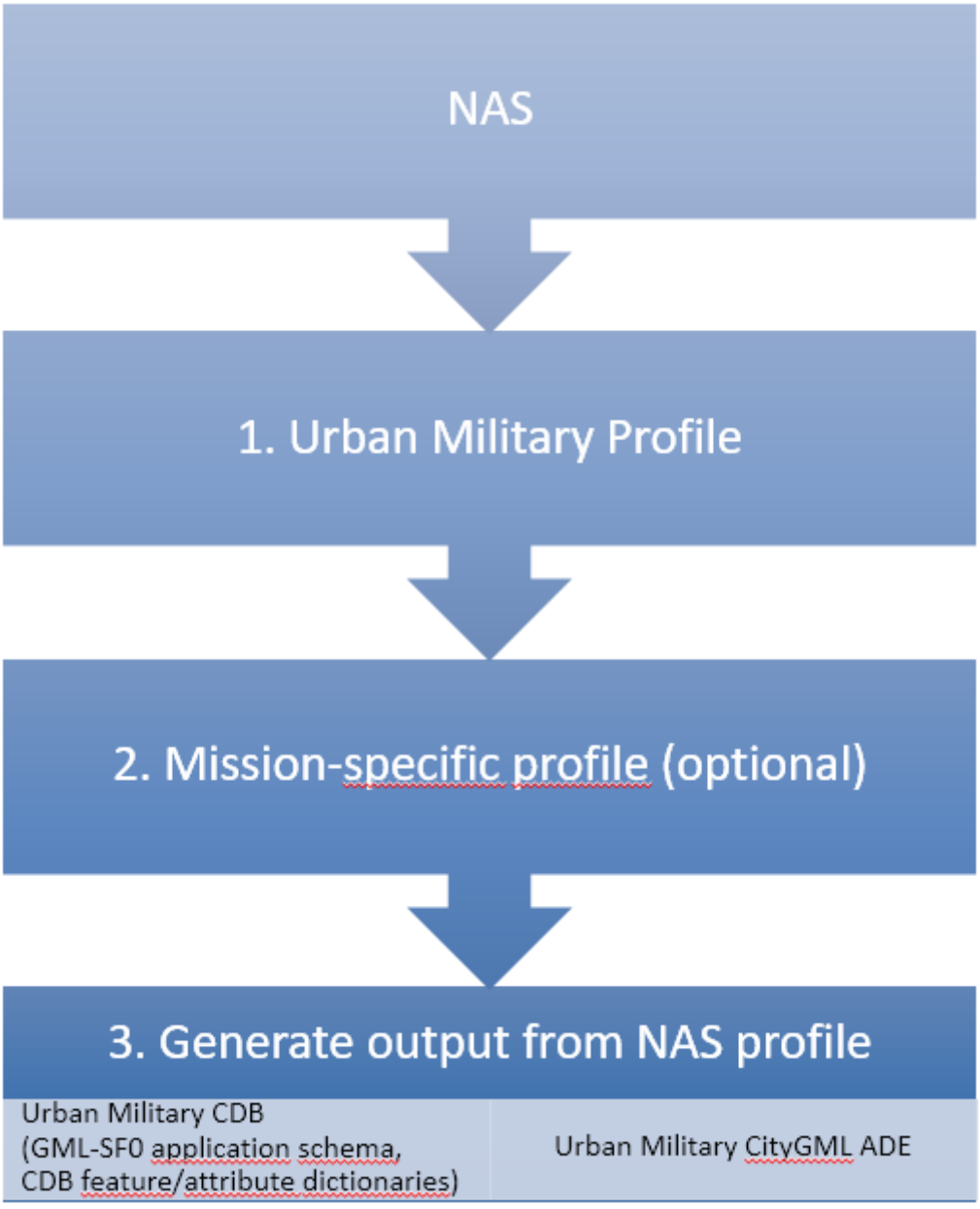

Figure 1. NAS profiling workflow

1. In a first step the NAS-based Military Urban Profile was specified as a UML model, i.e. an Enterprise Architect project file. This is described in chapter 5.
2. In a second step a mission-specific sub-profile can optionally be specified using ShapeChange and the ShapeChange Profile Management Tool. Chapter 6 describes how profiles can be managed and processed using ShapeChange.
3. In the next step, the model and profile definition are processed to derive a CityGML ADE and output for CDB:
    a. For CDB, CDB feature/attribute configurations of the profile and a GML-SF0 application schema of the profile are generated. Chapter 7 and chapter 8 describe how to derive these results for NAS-based profiles and explain the ShapeChange enhancements implemented in Testbed-13.
    b. For CityGML ADE, the model of the profile is transformed to a CityGML ADE implementation model that is used to generate a GML application schema that is a CityGML ADE. Chapter 9 describes how to derive these results for NAS-based profiles and explains the ShapeChange enhancements implemented in Testbed-13.

The decision to restrict the GML application schema for CDB to the GML Simple Features Level 0 Profile was taken for this testbed for practical reasons: to be compatible with the complexity supported by Shapefiles and, therefore, implementations of OGC CDB 1.0. See 7.1 for more details. Limitations resulting from this approach are documented in 7.4. The CDB SWG and the Testbed-13 CDB NAS Profile ER should consider supporting richer data representations in order to better support complex ISO 19109 application schemas (for example the NAS).

The underlying use case for the work documented in this report is that

- Organization A has defined a NAS profile for a specific purpose and has then captured data based on this profile;
- Organization B now wants to use that NAS data (together with other data) either in its CDB application or in its CityGML application, which requires the transformation of the schema and data according to the rules of CDB or CityGML in order to configure the application so that it can handle the data.

Chapter 5. Definition of the NAS Profile

5.1. Introduction

This chapter describes how the UML model of the NAS profile used in Testbed-13 was created.

A profile for content that is typically used in an urban setting was used in Testbed-13. This was done in order to support CDB and CityGML work that builds on the schemas generated from this profile.

However, from the perspective of the work described in this report, the urban context is mainly used as an example for creating a specific profile of the NAS. The focus is not defining the "best" urban military profile, but on defining a repeatable capability for NAS profiles in general. In other words, the key requirement on the profile is not that it is 100% correct from the point of view of a domain expert, but it should be representative for the NAS and include all of the key modeling elements relevant to an urban setting.

5.2. Identifying the subset of the NAS

At first, the content of the profile had to be identified. The Specialized-Urban Topographic Data Store (S-UTDS) subset of the TDS v6.0 Entity Catalog [https://nsgreg.nga.mil/doc/view?i=82116] (topographic-profile of an older version of the NAS) has been used as the starting point for the "Urban" Military Profile of the NAS for the purposes of this testbed. The subset can be identified from the S-UTDS v6.0 Extraction Guide [https://nsgreg.nga.mil/doc/view?i=82121] by inspecting column H ("S-UTDS EG") on the "Data_Collection_Organization" tab.

Additional documents related to the Topographic Data Store (TDS) profile of the NAS 6.0:

- TDS v6.0 capstone document [https://nsgreg.nga.mil/doc/view?i=82115]
- TDS v6.0 Codelist Resources [https://nsgreg.nga.mil/doc/view?i=82117]

NOTE	A minor derivative of v6.0 is currently in-use in NGA data production. The derivative is known as v6.1 and varies in how it handles metadata. There is no difference in the "feature aspects" from v6.0.

In the next step, the TDS v6.0 subset content has been migrated into a model that is a profile of the NAS v8.0 (published in late 2016).

More information about the NAS v8.0 is available at https://nsgreg.nga.mil/doc/view?i=81111.

5.3. Generating the UML model in EA

Once the content of the NAS profile has been identified, a reference UML model in EA had to be generated for the profile. This was achieved using the tooling that NGA uses to manage the NAS.

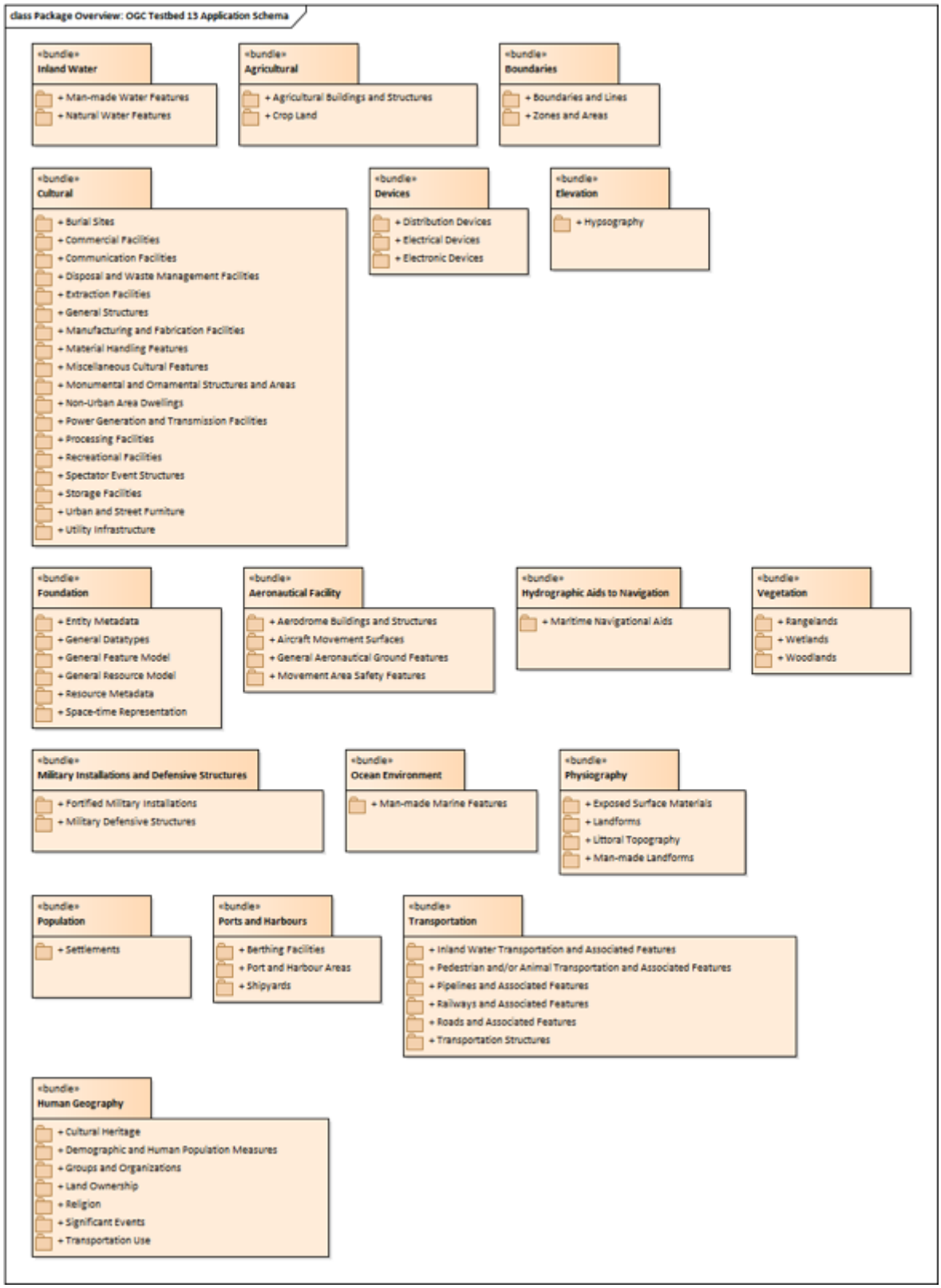

Figure 2. Overview of the NAS profile

The package diagram above provides an overview of the contents of the profile. The diagram shows the top-level packages of the NAS included (i.e., the packages with stereotype `bundle`) as well as the second-level packages that contain the feature types (packages in the `bundle` packages, with a stereotype `leaf`).

The list of feature types in each package that are part of the profile used in Testbed-13 are listed in Annex B.

5.4. Adding CityGML and relevant CityGML ADEs

One testbed task was to explore generating a CityGML ADE for the NAS profile. In order to process the NAS profile and CityGML together in the derivation of the CityGML ADE, the CityGML application schema was added to the UML model for the NAS profile.

When generating a new CityGML ADE, good practice is to re-use existing ADEs in that process. To explore this case, the CityGML UtilityNetwork ADE was selected as an example for the following reasons:

- Utilities are important aspects in an urban setting
- That ADE includes feature types different than just buildings which are the focus of most CityGML datasets; and
- Availability of a UML model for the ADE

The CityGML application schema and the CityGML UtilityNetwork ADE are used when processing the profile to generate a CityGML ADE for the "Urban Military Profile" (that extends CityGML and, potentially, existing ADEs).

Chapter 6. Processing a profile with ShapeChange

This chapter describes the workflow to define and process a profile of a NAS-based application schema using ShapeChange and the ShapeChange Profile Management Tool (PMT).

6.1. Providing the basis

A schema profile is tied to a particular version of a given schema. In order to define such a profile with the PMT, the schema and its dependencies need to be known. This is achieved by loading the model that contains the schema with ShapeChange, and then exporting the model into a ShapeChange-specific XML format (SCXML).

NOTE

The model that is loaded may already contain profile definitions. ShapeChange also supports a transformation to load profiles from external files (with SCXML format) into the model. If profiles are available in the model, ShapeChange can create a model profile and export the result. This would ensure that subprofiles (that are defined for the model profile using the PMT) only contain subsets of the model elements defined by the model profile. For example, a NAS Urban Military profile could be created and exported by ShapeChange. This profile could then be used as the basis to define subprofiles, for example focusing on just transportation or building aspects, in an ad-hoc fashion using the PMT.

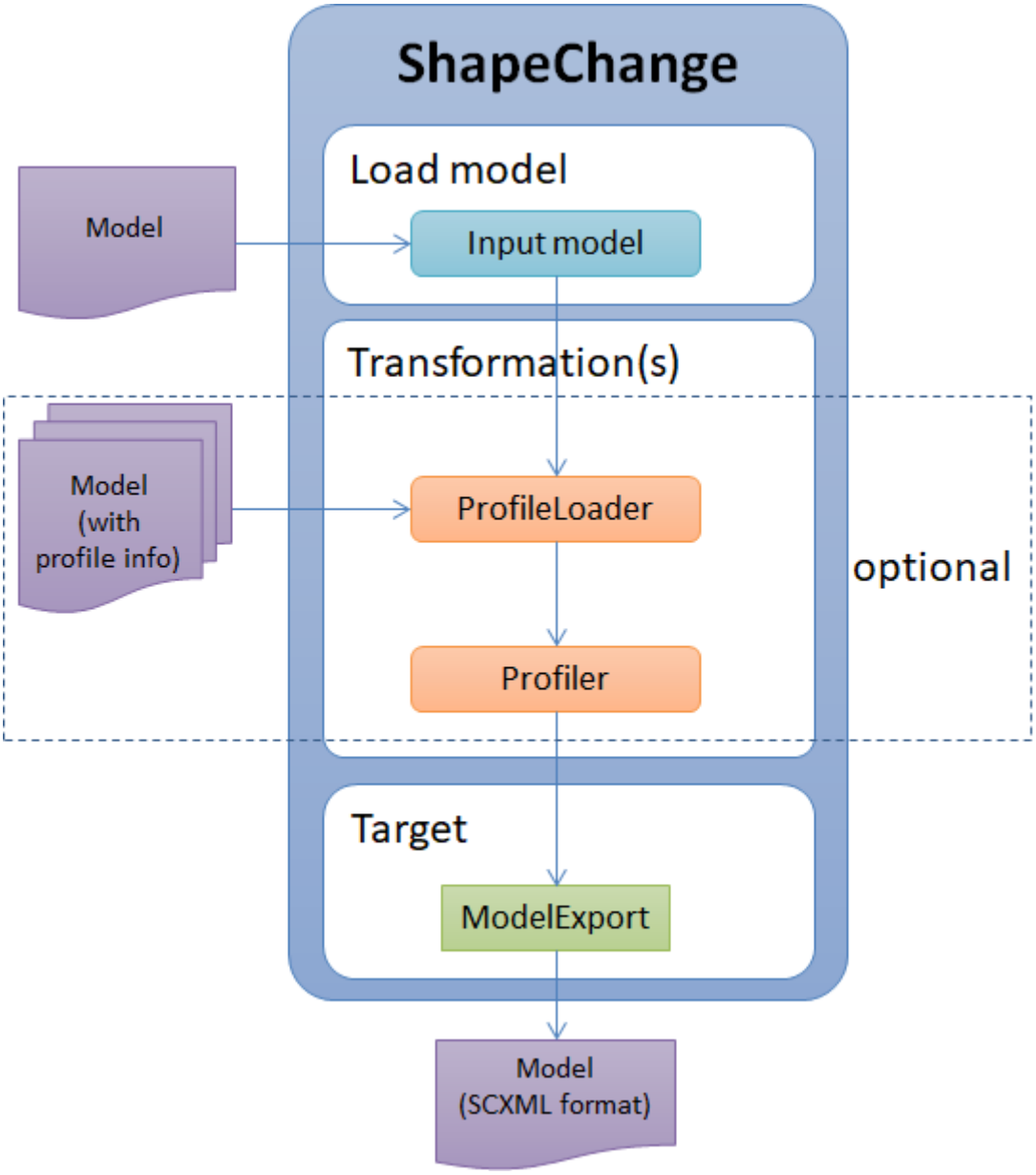

Figure 3. ShapeChange workflow to export a model

NOTE	The ShapeChange configuration to export the NAS profile to SCXML is contained in Annex A.

6.2. Defining a profile

The ShapeChange Profile Management Tool (PMT) is a web application to edit application schema profiles. The definition of a consistent profile for a large application schema like the NAS is a non-trivial task. The PMT provides a number of features to support the user in this task.

The PMT can load UML models from files in SCXML format. Profiles can be added to the model, edited, copied, and deleted.

A model browser is available to navigate through the model. A text-based search can be used to look up specific model elements.

Packages, classes, and their properties can be added to or removed from a profile. The visual

display of a model element changes accordingly. For selected packages and classes, the PMT offers actions for mass processing contained model elements (e.g. adding all properties of a class to the profile).

When adding a model element to the profile, the PMT automatically adds relevant related model elements. For example, when adding a subtype to the profile, its supertypes as well as mandatory properties and their value types are automatically added to the profile. The same applies to automatically added types. Likewise, upon removing a model element from the profile, the PMT would remove related elements. For example, when a supertype is removed, then its subtypes are removed as well. If a property is removed from the profile, then its value type is removed - but only if it is not used as value type by another property.

The consistency of a profile is constantly checked. Consistency issues are reported. An issue report contains a link for the user to jump to the corresponding model element.

In the following screenshot, feature type 'Road' has been selected. The details pane on the right shows available profile actions and profile parameters (e.g. defining the geometry types that are allowed for the feature type). Similar panes are available for packages and properties.

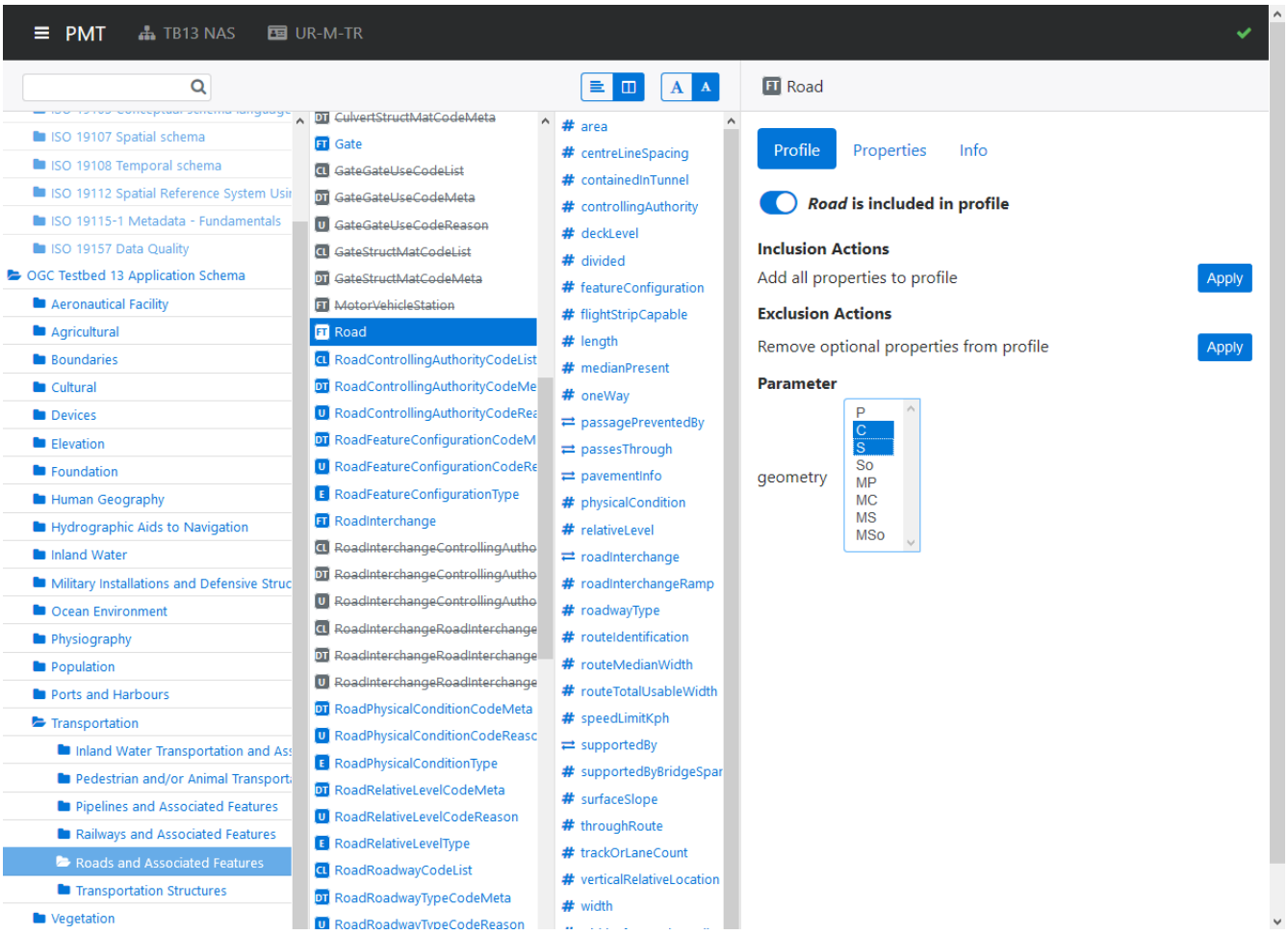

Figure 4. Defining a profile with the Profile Management Tool - feature type 'Road' with profile assignments

For quickly making profile assignments for the properties of a class, a tabular view of the properties is available (see Figure 5).

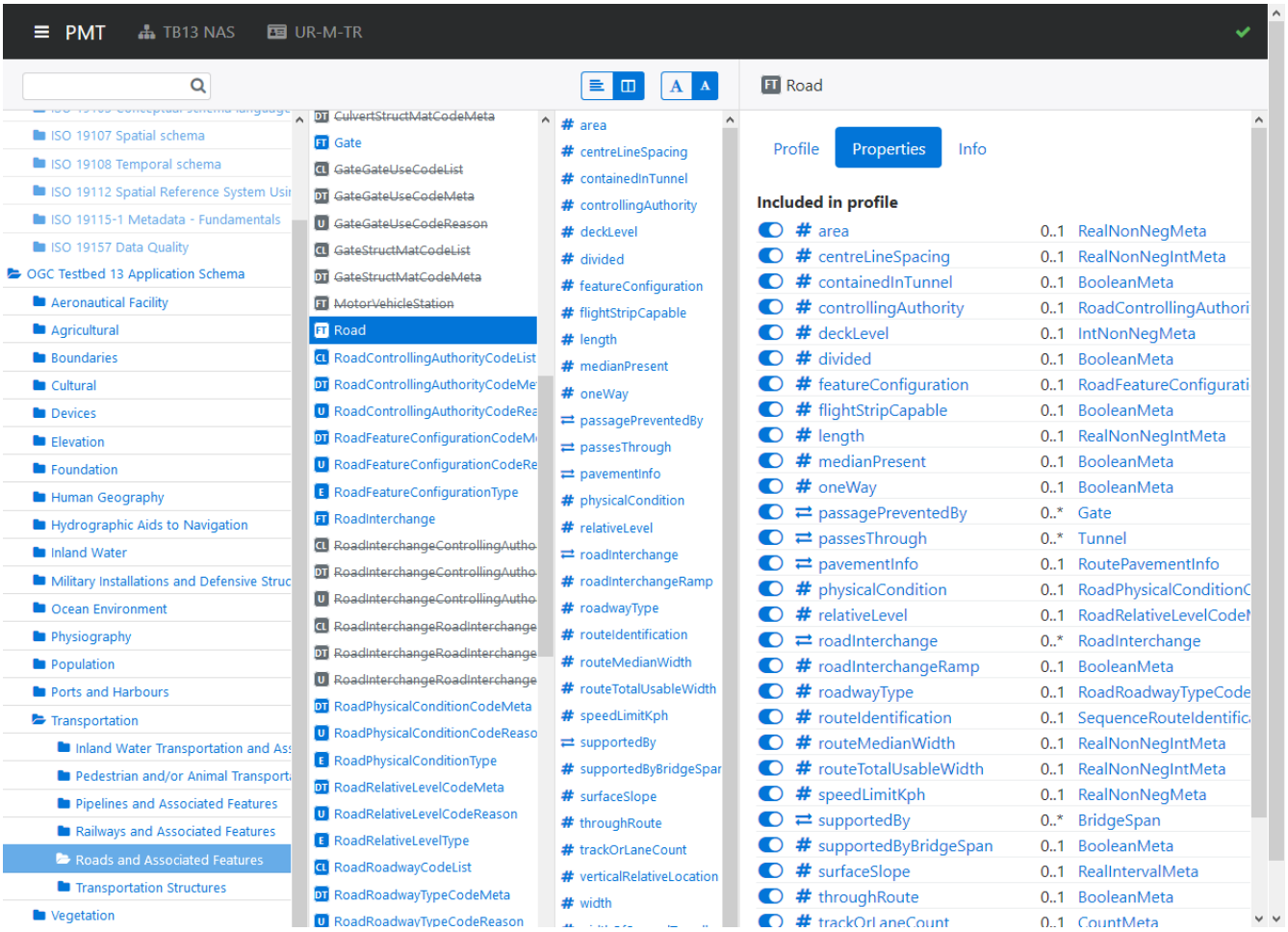

Figure 5. Defining a profile with the Profile Management Tool - properties of feature type 'Road'

When a package is selected, a similar view is available for the classes contained in that package.

Additional information about the selected model element, like its stereotype, super- and subtypes, descriptors (such as alias, definition, and description), and tagged values can be accessed via the info pane (see Figure 6).

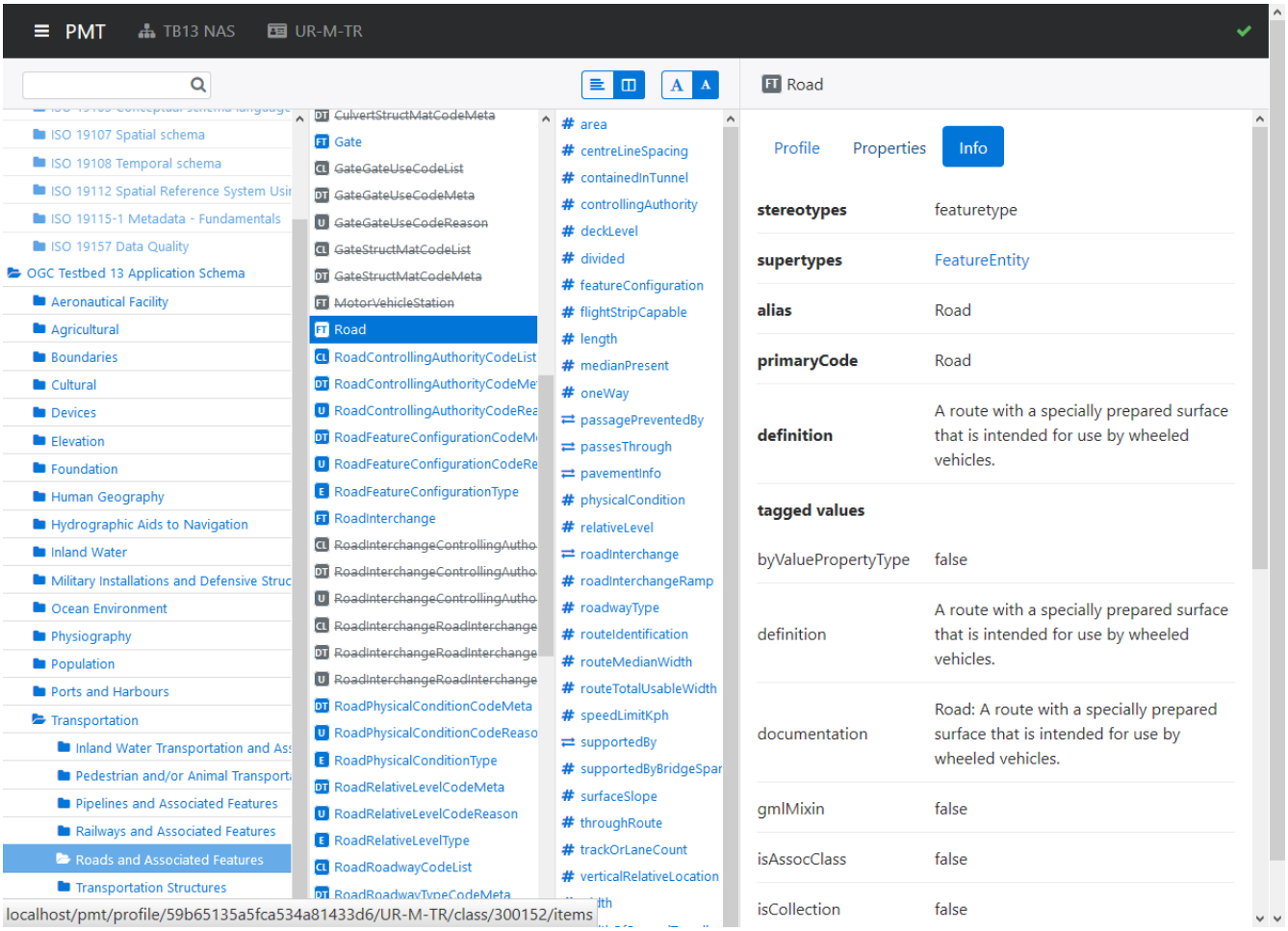

Figure 6. Defining a profile with the Profile Management Tool - additional information on feature type 'Road'

Specific views are available that simplify the model. Inheritance can be flattened. Furthermore, *Meta* and *Reason* classes can be hidden (see Figure 7).

NOTE	The underlying model is not changed, just the presentation to the user.

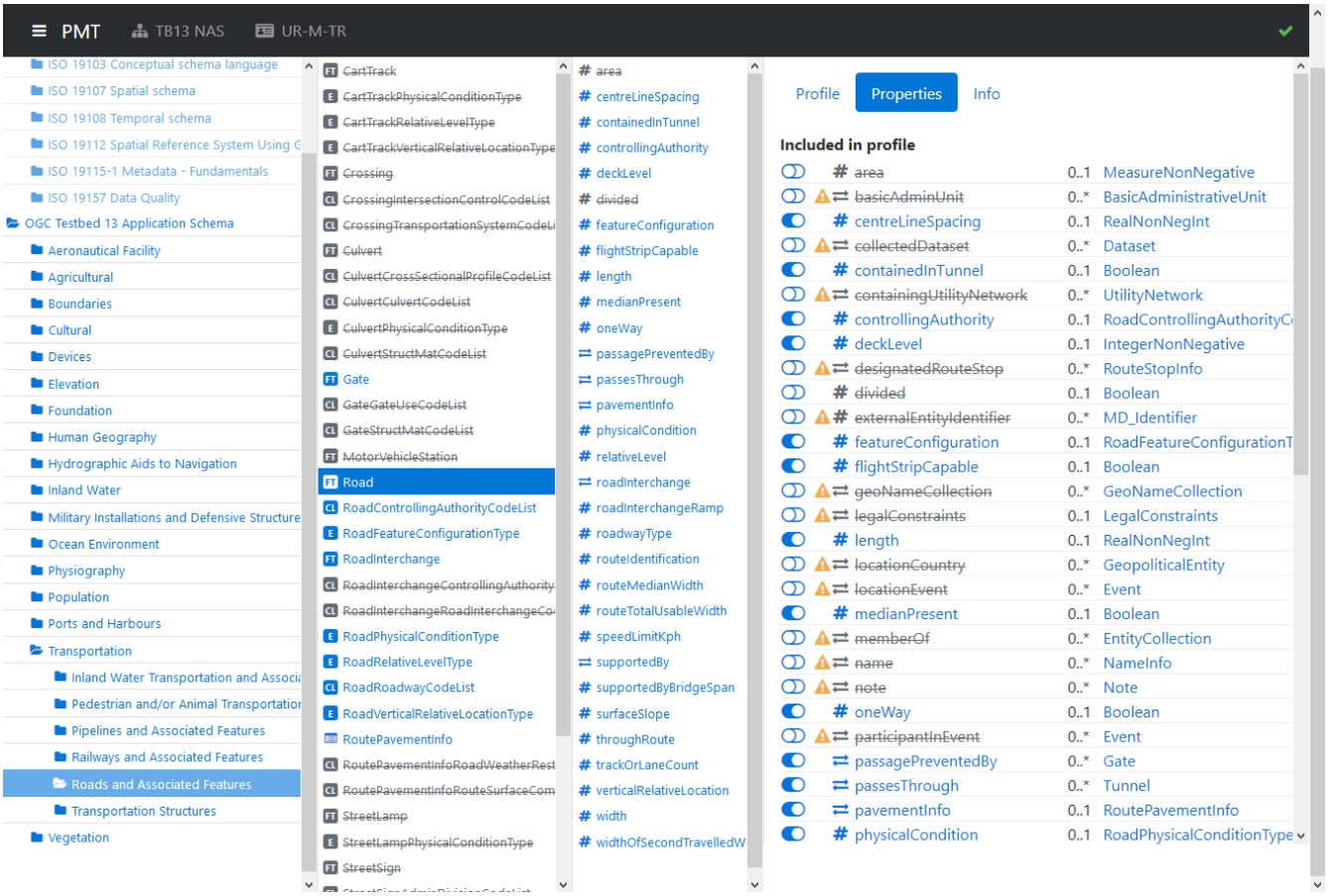

Figure 7. Defining a profile with the Profile Management Tool - special views enabled

NOTE

Since the two views to simplify the model are active in Figure 7, *Meta* and *Reason* classes are no longer shown and the properties pane shows properties that feature type 'Road' inherits from its supertypes.

Yellow warning-symbols are added to inherited properties. They inform the user that changing the profile assignment for these properties will affect other classes, since the profile of such a property will be changed in a supertype. The change may thus also affect other subtypes of that supertype. For the NAS, that is especially the case for all properties inherited from *FeatureEntity*.

When the profile is fully defined, it can be exported (to SCXML) and loaded by ShapeChange to derive implementation schemas (CDB dictionaries, CityGML ADEs, XML Schemas, ontologies, DDL schemas, ArcGIS workspace schemas, feature catalogues, JSON Schemas, etc).

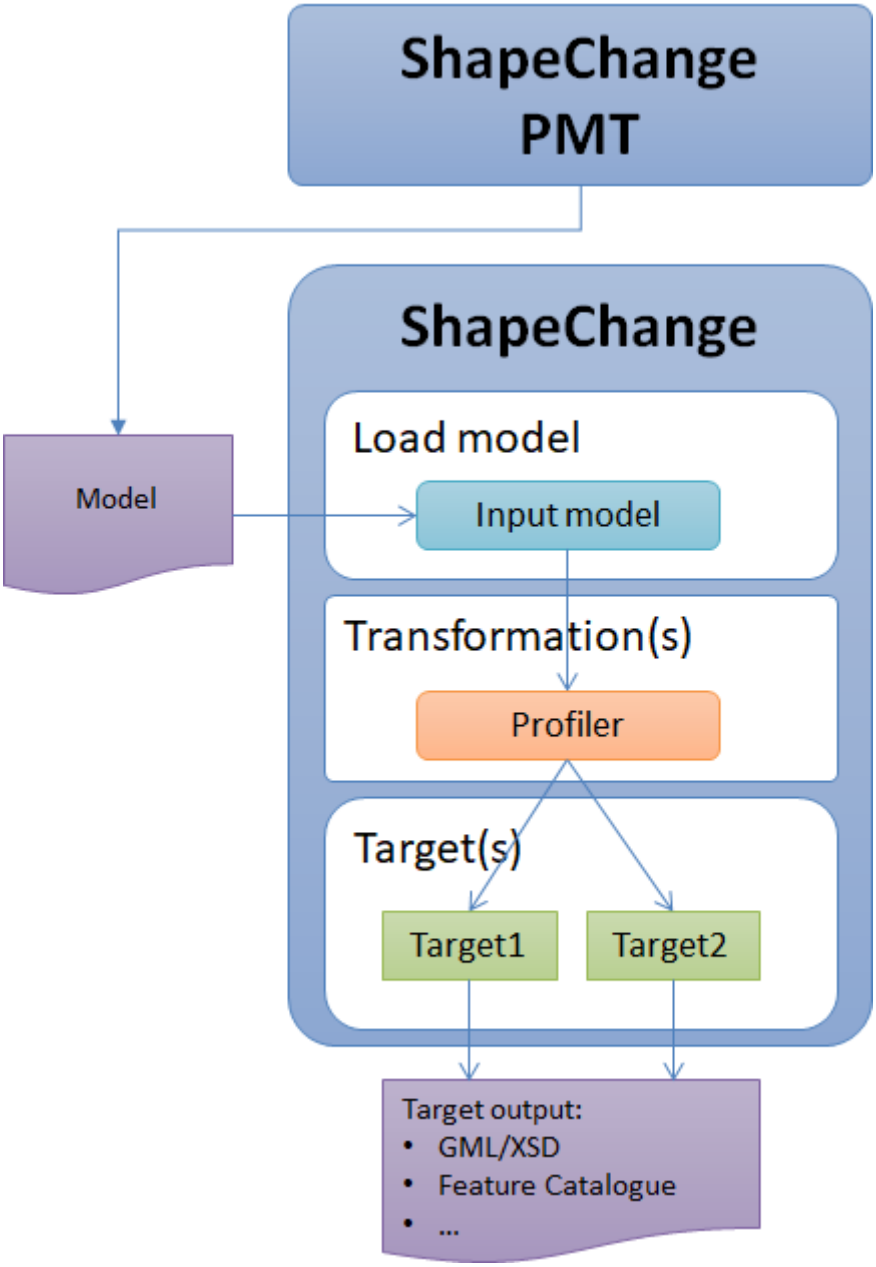

Figure 8. ShapeChange workflow - process profile created using the PMT

NOTE The PMT does not currently support profiling of constraints. ShapeChange itself currently only has a few options to handle constraints while creating a profile. A design for *profiling constraints* has been developed in Testbed 12 (for further details, see the Testbed 12 ShapeChange ER, chapter 7). A future work item documents ideas for extending the functionality of the PMT.

6.3. Migrating a profile to a different schema version

A more specific use case related to schema profiles is transferring a profile to a different version of an application schema. As discussed in Providing the basis, a profile is defined for a specific version of a schema. When the schema is changed, for example by adding new classes, the profile needs to be updated to take the changes (that are relevant for profiling) between the different versions of the schema into account. To migrate a profile:

- The different version of the schema would be loaded with ShapeChange.

- The profile defined for another version of the schema would be loaded. Model differences as well as potential profile inconsistencies can thereby be detected and logged by ShapeChange.
- The resulting schema would be exported by ShapeChange and imported into the PMT. The profile can then be updated using the PMT, to correct any profile inconsistencies and to take into account relevant schema changes.

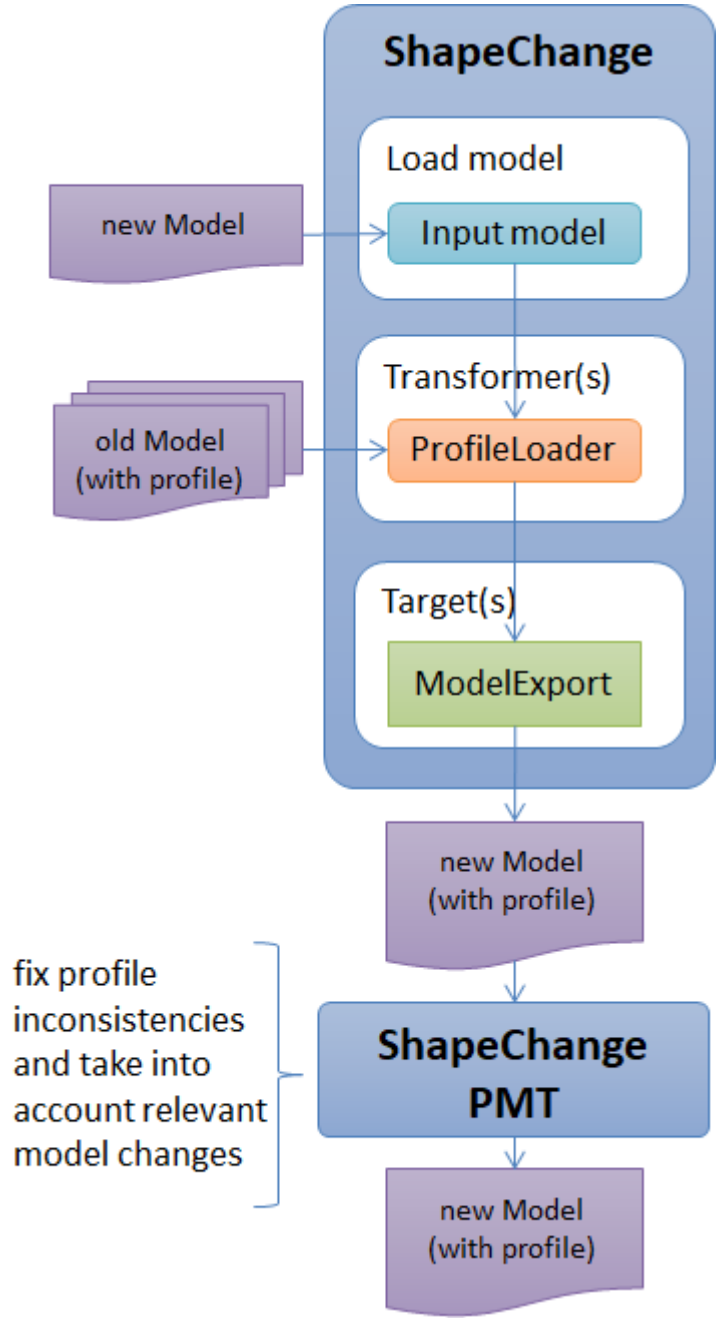

Figure 9. ShapeChange workflow - migrating a profile to a different version of the schema

Chapter 7. Generating a GML-SF0 implementation schema for the profile

7.1. Introduction

As input to the CDB work on a schema-aware CDB (limited to GML-SF0 profiles compatible with Shapefiles), a workflow has been developed to transform the NAS profile into a GML-SF0 schema suitable for implementation as an Esri Geodatabase or Esri Shapefiles (and thus the CDB).

NOTE	The workflow can be applied to other profiles of the NAS, as well as other application schemas.

The workflow takes into account a number of GML-SF0 restrictions as compared to a full GML application schema:

- A GML-SF0 schema may only import or include schemas that also comply with GML-SF. The only exception is the GML schema, which is always imported.

NOTE	GML-SF allows the by reference encoding of properties with metadata as value type, if the metadata schema is not GML-SF compliant. Thus, the GML-SF0 schema may not have an import for the metadata schema, but instance data may still reference metadata objects.

- A GML-SF0 schema may define at most one feature collection, which may not be defined as a feature itself.
- A GML-SF0 schema does not support content models with choices. Only sequences are supported. Thus, <<union>> types are not directly supported. This restriction is relaxed at GML-SF1.
- A GML-SF0 schema does not support element substitution, and thus inheritance hierarchies cannot be represented.
- GML-SF0 only supports encoding of feature types with elements that use one of the allowed basic data types or geometry types as type. Elements with other type, especially with user defined complex types - which in a UML application schema often have stereotype <<dataType>>, <<type>>, or no stereotype -, are not supported.
- GML-SF0 only supports a specific set of basic types: integers, measurements, character strings, date, Boolean, binary data, (URI) references to resources, code lists, enumerations, and reals.
- Only a specific subset of the geometric property types from GML is supported by GML-SF.
- The maximum multiplicity of elements that represent feature properties is 1.
- GML-SF0 does not support nillable elements.

In order for a GML-SF0 schema to be suitable for implementation as an Esri Geodatabase or Esri Shapefiles, an additional restriction needs to be taken into account:

- A feature type cannot have multiple geometric properties with different geometry types.

Furthermore, the following restriction applies to both a GML-SF0 schema and a GML application schema:

- There is no direct implementation of association classes. They need to be transformed into corresponding intermediate classes.

The workflow described in this chapter takes these restrictions into account. The workflow applies several transformations to the input UML application schema. These transformations are described in detail in the Workflow section. They can be grouped as follows:

- Retrieve specific information from model elements and make it available for subsequent transformations. This includes copying tagged values from basic types, but also identifying geometry restrictions from OCL constraints. The transformation of geometry type inheritance is a member of this group as well.
- Remove model elements and structures that shall not or cannot be represented in the GML-SF0 schema. This includes the removal of collection types, inheritance from metadata and basic types, as well as documentation of properties called *valueOrReason* and *valuesOrReason*. Restricting navigability of bi-directional associations is also an important aspect.
- Transform model elements and structures that cannot be represented in the GML-SF0 schema, such as NAS XxxReason <<union>> types, association classes, inheritance, multiplicity, and complex types. In order to be suitable for implementation as an Esri Geodatabase or Esri Shapefiles, feature types that have geometric properties with different geometry types also need to be transformed into geometry specific feature types, like a River with a curve as geometry and a River with a surface as geometry.
- Finally, apply transformations to tidy up the transformed model and prepare it for encoding as a GML-SF0 compliant XML Schema. This includes, for example, removing components of property names that are artifacts of the transformation workflow and not needed in the GML-SF0 schema. Furthermore, tagged values can be set in the whole model, for example to ensure that no property types are generated in the XML Schema, and that properties with a feature type as value type are encoded by reference.

7.2. Workflow

This section describes the workflow for creating a GML-SF0 schema from a NAS profile. The order of the following sections represents the order of the required processing steps. These steps collectively accomplish the specifics outlined in the four groupings described in the Introduction.

NOTE	The ShapeChange configuration to execute the workflow is contained in Annex A. A note at the end of each following section references parts of the configuration that are relevant for the processing described in that section.

7.2.1. Geometry restrictions

In the NAS, the position(s) of any feature can be represented by multiple different types: points, curves, surfaces, physical addresses, and location by identifier. OCL constraints can be defined to restrict these options.

Example 1. Place representation restriction on feature type 'River'

The OCL constraint with name 'Place Representations Disallowed' on feature type 'River' is defined as follows:

```
inv: place->forAll(p| not(p.oclIsKindOf(PhysicalAddressInfo)) and
not(p.oclIsKindOf(PointPositionInfo)))
```

The constraint specifies that if there are associated place representations for a river, then they are neither physical address(es), nor positions established as point(s).

GML-SF0 supports any number of geometry typed elements within a feature type. However, format technologies such as Esri Shapefiles require that a feature have a single geometry. In order to satisfy this requirement in the GML-SF0 schema derived by ShapeChange, NAS OCL constraints are processed to identify which geometry types are allowed for a given feature type. Information about the allowed types is stored in tagged value *geometry* of the feature type, to be used by subsequent processing steps.

NOTE To ensure that a feature in the GML-SF0 NAS profile only has a single geometry, the tagged value *maxOccurs* is set to '1' for all properties with name 'place'. For further details, see Multiplicity.

NOTE When the workflow is applied to an application schema other than the NAS, and the workflow shall produce a GML-SF0 schema suitable for implementation as an Esri Geodatabase or Esri Shapefiles, then the tagged values *geometry* and *maxOccurs* may need to be set explicitly. Tagged value *geometry* should be set if a) the feature types of the application schema can be represented using different geometry types, b) only a subset of these geometry types is allowed for the feature type, and c) the schema does not have OCL constraints as the NAS to restrict the available geometry types. The tagged value *maxOccurs* should be set on properties to ensure that a feature type does not have multiple geometry properties.

NOTE The transformer that executes the processing described in this section is part of the full ShapeChange configuration (see Annex A). It has attribute *id* with value *TRF_GEOMETRY_RESTRICTION_TO_GEOMETRY_TAGGEDVALUE*.

7.2.2. Convert OCL constraints to text constraints

For deriving a GML-SF0 NAS profile, OCL constraints are only relevant for the transformation described in Geometry restrictions. Subsequent processing steps will significantly alter the schema structure, so that most if not all OCL constraints defined in the original model will no longer be valid. The OCL constraints are therefore converted to so called text constraints. Corresponding Schematron assertions can then no longer be created for the GML-SF0 NAS profile.

NOTE	A detailed analysis of the OCL constraints, to determine which would be useful for a GML-SF0 schema, and how the constraints would need to be transformed to be valid in the transformed schema, is deferred to future work (see Analyzing relevance of OCL constraints for GML-SF0 schema).
NOTE	Instead of converting the OCL constraints to text constraints, they could also have been removed. With respect to the derivation of Schematron assertions, the result is the same (Schematron assertions cannot be created from text constraints). By default, ShapeChange validates the OCL constraints of the application schema after each transformation step. Invalid constraints would be reported in the log, and converted to text constraints (which are not validated). By converting OCL constraints to text constraints with an explicit transformation, such log messages can be avoided. Furthermore, text constraints can be included in feature catalogues created by ShapeChange.
NOTE	The transformer that executes the processing described in this section is part of the full ShapeChange configuration (see Annex A). It has attribute *id* with value *TRF_FLATTEN_CONSTRAINTS*.

7.2.3. Geometry type inheritance

Next, NAS position types need to be processed. Position types in the NAS (PointPositionInfo, CurvePositionInfo, and SurfacePositionInfo) inherit from ISO 19107 types (GM_Point, GM_Curve, and GM_Surface). These inheritance relationships are transformed into a 'geometry' property on each NAS position type, with value type being the former supertype.

This transformation is performed since subsequent processing steps remove inheritance relationships of types in the schema (see Inheritance) and rely on the existence of geometry typed properties (see Geometry specific feature types).

NOTE	When processing application schemas that do not use inheritance from ISO 19107 types, and instead have attributes with an ISO 19107 geometry as type, this transformation can be skipped.
NOTE	The transformer that executes the processing described in this section is part of the full ShapeChange configuration (see Annex A). It has attribute *id* with value *TRF_FLATTEN_GEOMETRY_TYPE_INHERITANCE*.

7.2.4. Removing types

The following NAS types are removed to significantly reduce the number of properties of feature types in the GML-SF0 schema:

- Dataset, EntityCollection, FeatureEntityCollection, GeoNameCollection, PhysicalEntityCollection, PropertyMetadata, DataLineage, DataProcessStep, DataSource, DataQuality, PhysicalObjectMetadata

The types 'LocationInfo' and 'PhysicalAddressInfo' are also removed. Keeping these types in the model would result in a choice between LocationInfo properties, PhysicalAddressInfo properties as well as Point-, Curve-, and SurfacePositionInfo properties. It is not clear if being able to provide location by identifier or physical address information is useful for a GML-SF0 schema.

NOTE	Creating a GML-SF0 schema that keeps these types and investigating the impact on systems that consume the schema has been deferred to future studies.

NOTE	The transformer that executes the processing described in this section is part of the full ShapeChange configuration (see Annex A). It has attribute *id* with value *TRF_REMOVE_TYPE*.

7.2.5. Metadata types

Some NAS types inherit from ISO 19115-1 types, for example NAS type 'LegalConstraints'. An attempt to process the ISO 19115-1 schema and pull its contents into the GML-SF0 NAS profile failed (the process ran out of memory, even when executed with 6GB memory). Without further modifications (removal of classes and properties from the profile, also restricting navigability of association roles), processing of ISO 19115-1 leads to an unmanageable increase of property elements in the GML-SF0 NAS profile. Therefore, inheritance relationships to types with name prefix 'MD_' are removed.

NOTE	Incorporating ISO 19115-1 in the GML-SF0 NAS profile requires a detailed analysis and testing, which could be performed as part of a future study.

Another aspect to consider when encoding a GML-SF0 schema is how to encode properties that have a metadata type as value type. The GML 3.2 encoding rule defines the tagged value 'isMetadata' on properties. If this tagged value is 'true' then the value type is assumed to be a metadata type. In addition, map entries of the ShapeChange XML Schema target can indicate if a value type of a property should be encoded as an anonymous complex type that extends gml:AbstractMetadataPropertyType. GML-SF does not allow an encoding with gml:AbstractMetadataPropertyType. Setting tagged value *isMetadata*='false' in the whole model would not suffice. On the one hand, map entries can still result in an encoding with extension of gml:AbstractMetadataPropertyType. On the other hand, metadata types like CI_Address and CI_Contact could still be used as types of property elements, which is not allowed by GML-SF since the XML encoding of these metadata types is not compliant with GML-SF (for further details, see GML-SF section 8.3.3). Therefore, a new conversion rule has been added to ShapeChange: *rule-xsd-prop-metadata-gmlsf-byReference*. It ensures that properties with a metadata type as value type are encoded as elements with type gml:ReferenceType. This approach is in line with the requirements defined in section 7.3 "Metadata handling" of GML-SF.

NOTE	The transformer that executes the processing described in this section is part of the full ShapeChange configuration (see Annex A). It has attribute *id* with value *TRF_REMOVE_MD_INHERITANCE*.

7.2.6. Removing documentation

GML-SF does not appear to allow documentation annotations on schema elements. When encoding the GML-SF0 NAS profile, this restriction can be taken into account with a special encoding rule.

However, the transformation workflow described in this chapter is also used to generate CDB dictionaries (see chapter 8). Documentation of features and their properties is essential there. Some of the transformation steps described in the following sections modify or merge documentation of model elements. In the case of properties, the resulting documentation may not be accurate any more.

Example 2. Merging documentation from valueOrReason property

Take the documentation of Facility.address and CiAddressMeta.valueOrReason as an example:

- The definition of Facility.address is: *An address (for example: postal or electronic) by which a cited party may be contacted.*
- The definition of CiAddressMeta.valueOrReason is: *Either the location of the responsible person and/or organisation or the reason that the value is absent.*

CiAddressMeta is the value type of Facility.address. When flattening complex types, CiAddressMeta will be flattened and the definitions of the two properties will merged. The result would be:

An address (for example: postal or electronic) by which a cited party may be contacted. : Either the location of the responsible person and/or organisation or the reason that the value is absent.

Since *Reason* <<union>>s are removed by the transformation workflow (see section Reason unions), the resulting model no longer provides a choice between providing a value and providing a reason why a value is absent. The workflow also updates property names to remove parts that equal *.valueOReason* and *.valuesOrReason* (see section Naming). Consequently, the documentation of properties in the resulting model should also not include any documentation from *valueOrReason* and *valuesOrReason* properties. Coming back to the example, this means that the documentation of CiAddressMeta.valueOrReason should be ignored.

For the purpose of deriving CDB dictionaries, the definition, description, and alias of *valueOrReason* and *valuesOrReason* properties are removed from the model.

NOTE	The transformer that executes the processing described in this section is part of the full ShapeChange configuration (see Annex A). It has attribute *id* with value *TRF_REMOVE_VALUE_OR_REASON_DESCRIPTORS*.

7.2.7. Reason unions

The NAS uses a specific modeling construct to either provide a (set of) value(s) for a property, or to provide a reason for its absence. This 'choice' is modelled using <<union>> types whose name ends with 'Reason'. The full GML encoding of the NAS would represent these unions as elements that are

nillable and have a *nilReason* attribute.

GML-SF0 does not support the encoding of elements with a choice as content model. Furthermore, elements cannot be nillable.

NOTE	GML-SF1 supports both cases.

Encoding all options given by the Reason <<union>>s in the GML-SF0 NAS profile as optional elements is possible. However, doing so would significantly increase the size of the GML-SF0 NAS profile, which proved to be difficult to handle for typical XML editors. For Testbed-13, Reason <<union>> value types were therefore removed. All properties that use a Reason <<union>> as value type receive the value type of the 'value' or 'values' property contained in the <<union>> as new value type. Furthermore, these properties receive the combination of the maximum multiplicity boundaries of both properties (the property whose value was the <<union>> and the 'value' or 'values' property within the <<union>>) as new maximum multiplicity.

Removing the Reason <<union>> value types comes at the cost of not being able to provide a reason in the GML-SF0 schema why a property value is not given. If this is not acceptable, the processing described in this section can be skipped.

NOTE	The transformer that executes the processing described in this section is part of the full ShapeChange configuration (see Annex A). It has attribute *id* with value *TRF_FLATTEN_ONINAS*.

7.2.8. Basic types

The NAS defines a number of general data types, which are used throughout the application schema. Some of these data types are basic types that extend types from ISO 19103.

Type CharacterString is extended by StrucTextUnconstrained, TextLexUnconstrained, and TextNonLexUnconstrained. Additional subtypes of these types use tagged value *length* to define a restriction for the textual values these types represent.

Likewise, type Measure is extended by, for example, MeasureM180to180 and Measure0to360. These types use tagged values *rangeMinimum* and *rangeMaximum* to restrict the range of the measure value they represent.

In the full GML encoding of the NAS all these types are encoded as global simple types, with constraining facets to restrict their value space depending upon the aforementioned tagged values. However, GML-SF0 requires a simple encoding of simple types, where extension of simple types is not allowed - except for the cases defined by GML-SF (extension of string to create a 'LanguageStringType', and extension for both base64Binary and hexBinary). The NAS model is therefore transformed as follows:

- Tagged values *length*, *rangeMinimum* and *rangeMaximum* are copied from basic types to the properties that use these types as value types.
- Basic types that extend (directly or indirectly) a simple base type - for example CharacterString and Measure - are mapped to that type.

- When deriving the XML Schema, *rule-xsd-prop-constrainingFacets* is used to add constraining facets to elements that represent properties, according to the tagged values *length*, *rangeMinimum* and *rangeMaximum* on these properties.

NOTE	The transformers that will execute the processing described in this section are part of the full ShapeChange configuration (see Annex A). They have attribute *id* with values *TRF_TV_COPY_FROM_VALUE_TYPE* and *TRF_MAP_TO_SIMPLE_BASE_TYPE*.

7.2.9. Navigability

GML-SF0 only supports feature types with a simple structure, i.e. with a sequence of elements that have simple types, are references, etc. Complex types are only supported starting at GML-SF level 1.

As described in Complex types, properties of complex types are recursively copied into the types that have properties with these complex types as value type. This way, feature types will end up having properties with value type either being a feature type, an enumeration, a code list, or a simple type.

Each time a complex type is dissolved in this manner, the model grows (whenever the properties of the type replace properties of other types). In an ideal situation, the model represents a directed graph with cycles only introduced through associations between feature types. However, that is typically not the case. Cycles introduced by other types (e.g. classes with stereotype <<type>>, <<dataType>>, <<union>>, or no stereotype) can be problematic, since properties of dissolved types that are part of such cycles will be copied into the other types. This can lead to a significant increase of the size of the transformed model and the resulting GML-SF0 schema, up to the point that the model or the schema can no longer be processed.

Figure 10 provides an example of cycles contained in the NAS profile. Note that all connector labels have been suppressed to improve readability. What matters for this discussion is the navigability of the associations, which is expressed using arrows. Basically, all associations shown in the figure are bi-directional.

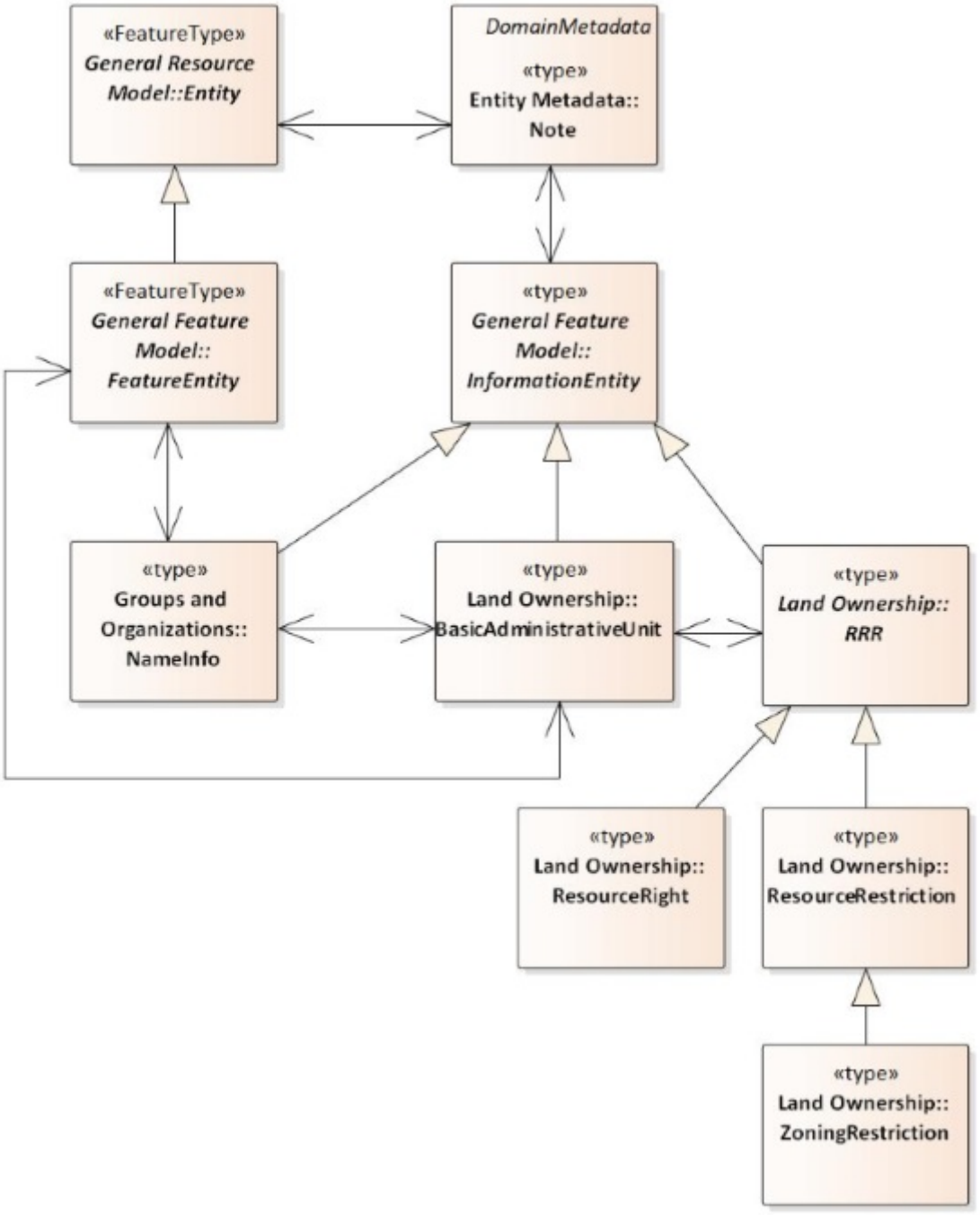

Figure 10. Cycles in the NAS profile

NOTE In Figure 10, cycles that only involve complex types, i.e. not feature types, exist between Note and the subtypes of InformationEntity, also between NameInfo, BasicAdministrativeUnit, and RRR (and its subtypes).

If the complex types of this part of the NAS profile were dissolved (as described in section Complex types), then a feature entity would have properties to represent:

- references from Note to any InformationEntity
- references from NameInfo to any BasicAdministrativeUnit - and from there to RRR (resource rights, restrictions, and responsibilities)
- references from BasicAdministrativeUnit to NameInfo to any FeatureEntity

- ...

Being able to encode these relationships in GML-SF0 may or may not be necessary. It depends on the use case, and therefore requires expert review.

For Testbed-13, the following model transformations were applied:

- Properties of non-feature-types that have a feature type as value type are set to be non-navigable.

> **NOTE** If the model contains "feature-like" types, like the NAS XxxInfo types, that one may wish to retain and use to "collect" properties from related objects, then these types can be transformed into feature types. For further details, see section Object types as feature types.

- Properties of specific object types that have an object type as value type are set to be non-navigable. In Testbed-13, this transformation has been applied to Note, LegalConstraints, ResourceConstraints, LivingQuartersAmenity, RRR, and all XxxInfo types (for example NameInfo). The transformation could also be applied to other object types. For an actual application - i.e. outside of Testbed-13 - a domain expert will need to decide to which object types the transformation is applied. However, the NAS PositionInfo types should not be transformed this way, since they reference object types from ISO 19107 that are essential for creating feature types with a single geometry type (which is explained in Geometry specific feature types).

The result of applying these transformations to the example from Figure 10 is shown with red arrows in Figure 11. In this example, the model graph has no more cycles.

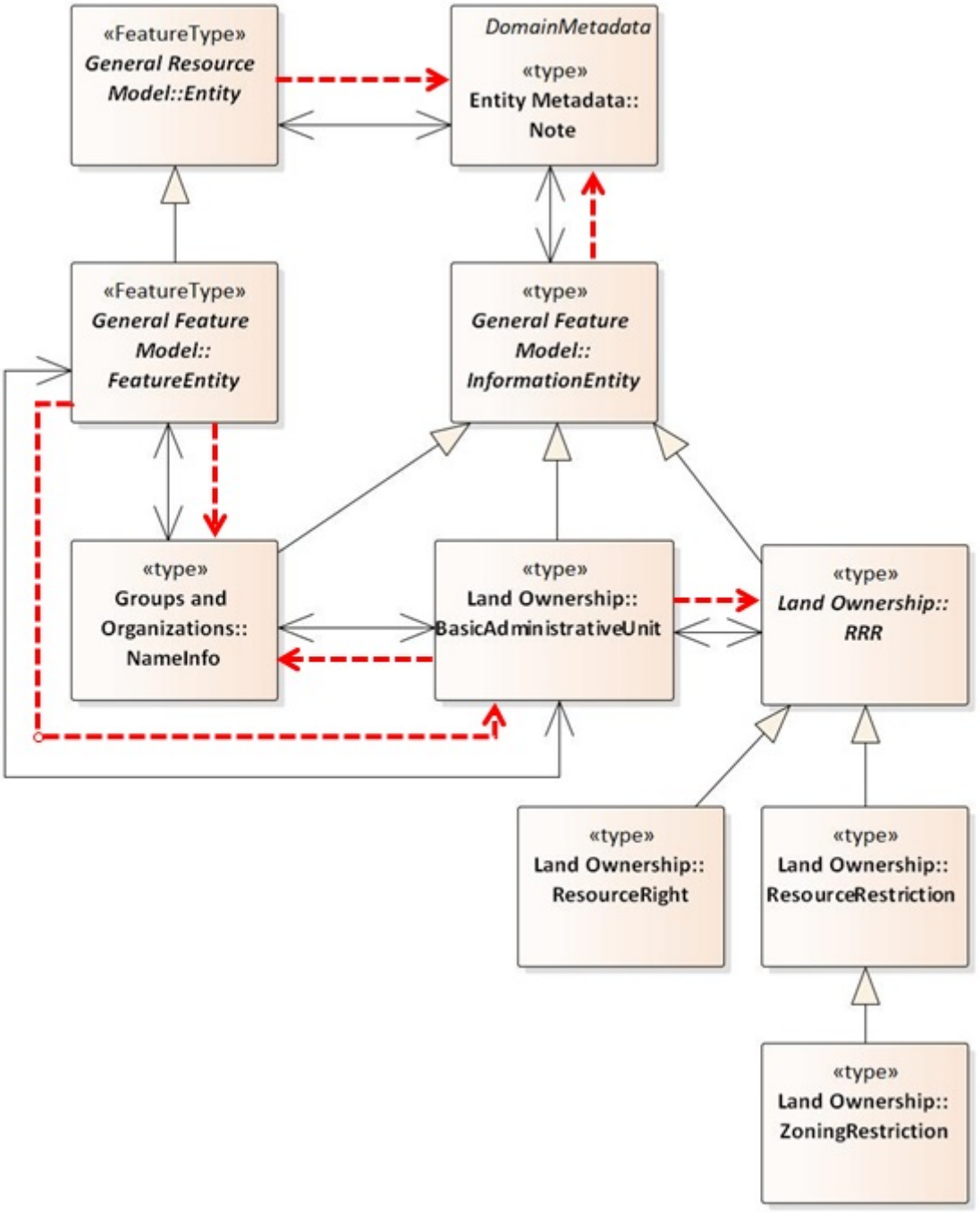

Figure 11. Updating the navigability of association roles to prevent cycles in the NAS profile

ShapeChange provides several configuration options to define which properties shall be set as non-navigable, and thus decreasing the number of cycles in the model.

NOTE Cycles in the model are not eliminated because cycles (circular dependencies) can also coexist between feature types, and those are considered to be non-critical because the properties of feature types are not copied by the transformation of Complex types.

Types where properties with specific categories of value types (feature type or object type) shall be made non-navigable can be identified using regular expressions (with syntax supported by Java [https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html]) that match on the name of the type.

Furthermore, the tagged value *isFlatTarget* can be set to 'true' on one of the roles of a bi-directional association. Doing so will make the role non-navigable.

Example 3. Using tagged value 'isFlatTarget'

Figure 12 contains three classes, one feature type and two object types, where the latter are connected by a bi-directional association. This creates a circular dependency.

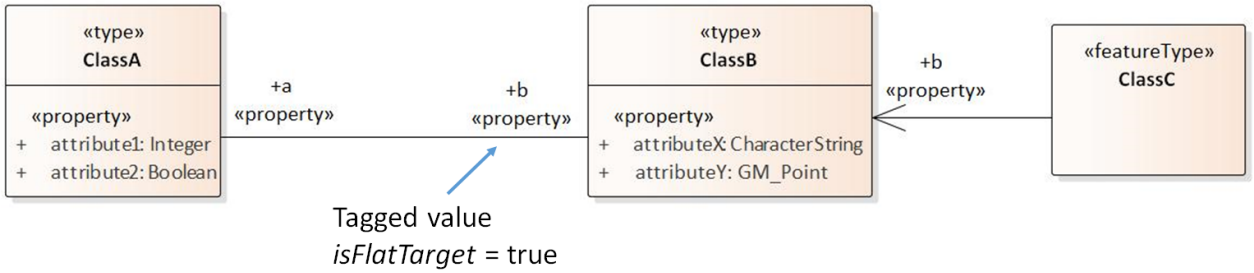

Figure 12. Circular dependency between two <<type>> classes

Tagged value *isFlatTarget* is set on one end of the bi-directional association, more specifically on association role 'b'. When this model is transformed, the association role is no longer navigable, as shown in Figure 13.

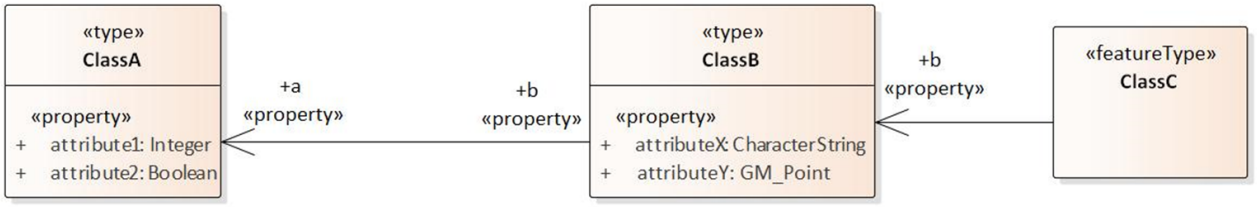

Figure 13. Navigability updated according to setting of 'isFlatTarget'

Transforming the model further to flatten complex types, as discussed in Complex types, will copy the properties of the <<type>> classes into the classes that have properties with such a <<type>> class as value type. ClassB has association role 'a' with ClassA as value type. When ClassA is flattened, the target of this flattening is ClassB, so copies of the properties from ClassA are created in ClassB. The association role 'a' is replaced by these property copies. However, its name will be merged into the names of the new properties. When ClassB is flattened, its properties are copied into ClassC (again merging the name of the association role 'b' that belongs to ClassC and has ClassB as value type). When the transformation is complete, feature type *ClassC* will only have properties that represent the attributes from the two <<type>> classes - see Figure 14.

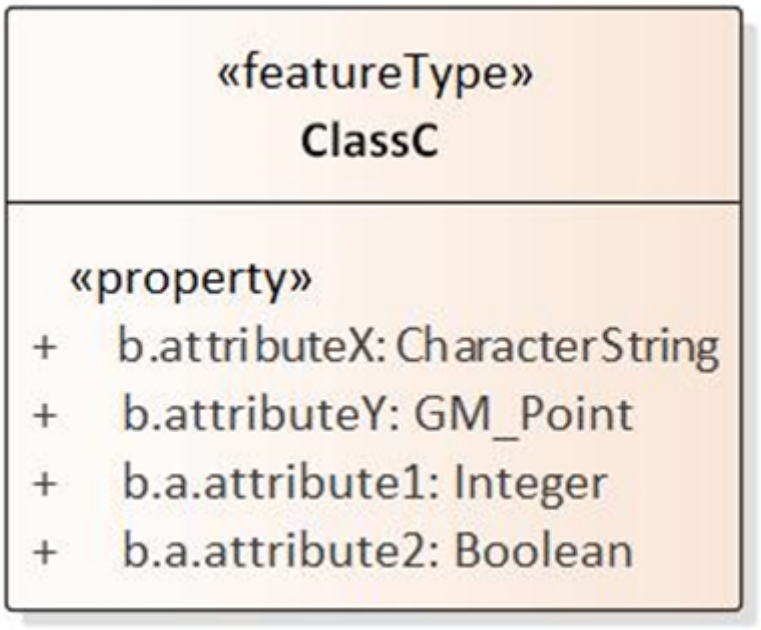

Figure 14. <<type>> classes dissolved

If the tagged value *isFlatTarget* had not been set in this example, the feature type would have additional properties like 'b.a.b.attributeX' and 'b.a.b.attributeY' (to represent the information that ClassA - from the original model shown in Figure 12 - can access via role 'b').

NOTE "Target" in tagged value "isFlatTarget" is not to be confused with a "ShapeChange target". In the context of flattening, tagged value "isFlatTarget" identifies the class to which information, i.e. properties, from the other end (i.e., class) of a bi-directional association shall be transferred.

NOTE The transformers that will execute the processing described in this section are part of the full ShapeChange configuration (see Annex A). They have attribute *id* with value *TRF_REMOVE_OBJECT_TO_FEATURE_TYPE_NAVIGABILITY*, *TRF_REMOVE_OBJECT_TO_FEATURE_TYPE_NAVIGABILITY_2*, and *TRF_REMOVE_NAVIGABILITY_BASED_ON_ISFLATTARGET*.

7.2.10. Association classes

GML-SF specifies how to encode associations between feature types. However, GML-SF does not cover association classes. GML 3.3 section 12.3 defines a conversion rule to transform association classes into corresponding intermediate classes. This transformation is applied on the NAS profile. Consider the example shown in Figure 15.

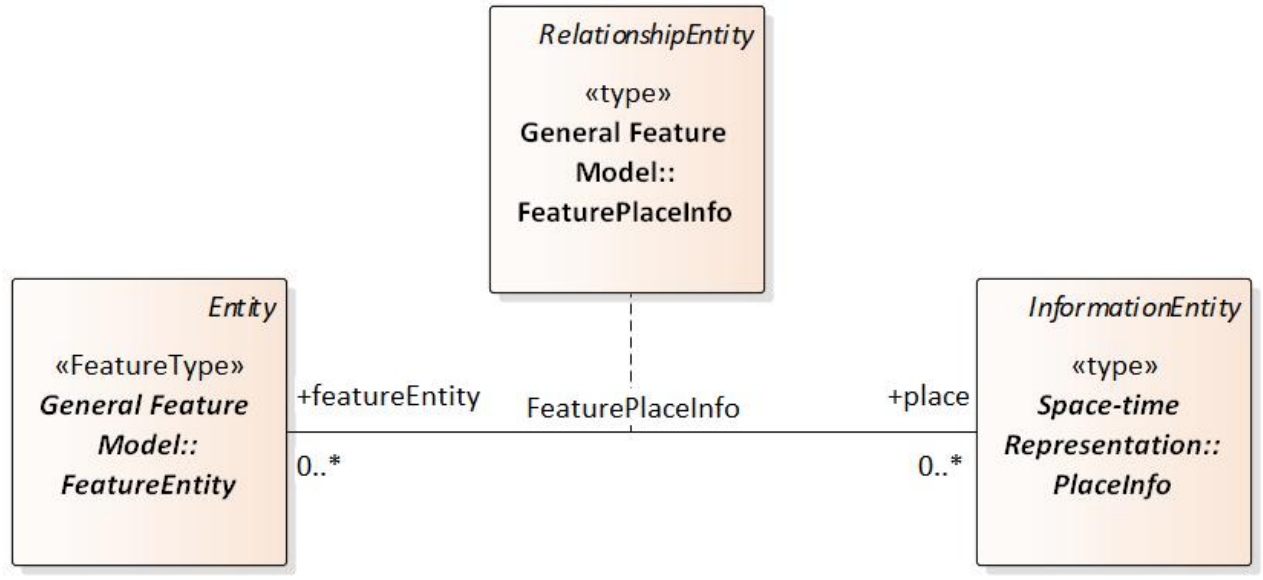

Figure 15. Example of an association class in the NAS

Transforming the association class in this example as defined by GML 3.3 results in a model as shown in Figure 16.

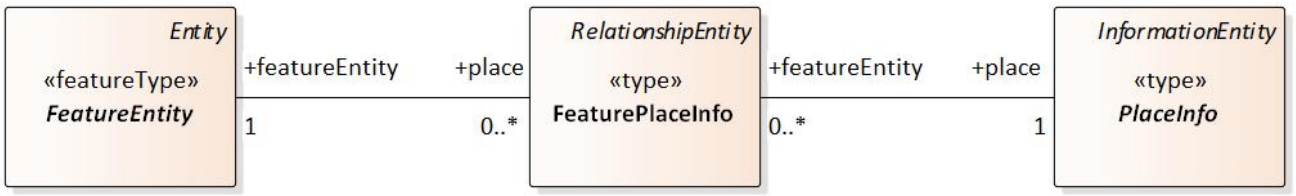

Figure 16. Example of an association class in the NAS, transformed as defined by GML 3.3

The transformed association class is now a normal object type, which can be dissolved as described in Complex types. However, that would result in a FeatureEntity having a property to reference itself (via the property path *place.featureEntity*). That would be unnecessary overhead.

NOTE In the original model (shown in Figure 15) *place.featureEntity* references any number of feature entities, one of which can be the entity itself (i.e., where the path *place.featureEntity* starts). In the transformed model shown in Figure 16, *place.featureEntity* (when starting from FeatureEntity) can only reference the feature entity itself, i.e. it is a self-reference.

While transforming an association class as defined by GML 3.3, ShapeChange adds two specific tagged values to the roles of the two resulting associations: *toAssociationClassFrom* and *fromAssociationClassTo*. The former is added to the association roles that end at the association class (the roles with multiplicity 0..* in Figure 16), the latter is added to the other two roles. The value of both tagged values on the roles of a given association is the name of the class that is at one end of the association and does not represent the association class (e.g. FeatureEntity). This information can be used by the transformation described in Complex types to prevent copying a property from a transformed association class that would only represent a reference to self (for

example to the FeatureEntity).

NOTE	The transformer that executes the processing described in this section is part of the full ShapeChange configuration (see Annex A). It has attribute *id* with value *TRF_ASSOCIATION_CLASS_MAPPER*.

7.2.11. Object types as feature types

When creating a GML-SF0 schema, dissolving complex types as described in Complex types also includes object types.

Depending upon the use case, it may be desirable to treat specific object types as feature types in the GML-SF0 schema. That way, the types would not be dissolved and they could be encoded and referenced as individual objects. It also helps to reduce the size of the resulting schema.

For deriving the GML-SF0 NAS profile in Testbed-13, LegalConstraints and ResourceConstraints have exemplarily been transformed to feature types.

NOTE	The transformer that executes the processing described in this section is part of the full ShapeChange configuration (see Annex A). It has attribute *id* with value *TRF_TO_FEATURE_TYPE*.

NOTE	ShapeChange supports two ways of identifying object types that shall be transformed into feature types. On the one hand, the configuration parameter 'toFeatureTypeNameRegex' can be used to provide a regular expression (with syntax supported by Java [https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html]). Each class whose name matches this expression is converted to a feature type. On the other hand, types to convert can also be identified by setting tagged value 'toFeatureType' to 'true' for them. The name of the tagged value can also be configured via the parameter 'toFeatureTypeTaggedValueName'. This can be useful if the conversion of types to feature types is performed for different goals, and the set of types to convert is different depending upon the goal. For example, a community could introduce a 'gmlsfToFeatureType' tagged value.

7.2.12. Associations

The transformation described in this section addresses a rather specific issue. When dissolving inheritance as described in Inheritance, associations to a supertype are copied, to create a similar relationship to all subtypes of the supertype. Where previously there was only a single relationship to another type, after dissolving inheritance this relationship has been transformed into multiple relationships. Attributes with a supertype as value type are treated differently: their value type is switched to a <<union>> that represents the choice between the supertype (if it is not abstract) and all its non-abstract subtypes.

GML-SF encodes relationships to feature types by reference (see GML-SF section 8.4.4.13). If an association had been multiplied as described before, the GML-SF0 NAS profile would have multiple references to feature types (to each non-abstract member of the set of the supertype and its subtypes) where before there was only one (to the supertype). Since relationships to feature types

are encoded by reference in GML-SF, an element that represents such a relationship can reference any kind of feature. Therefore, encoding multiple elements to represent the relationship is not necessary.

To achieve a GML-SF0 encoding that does not contain additional relationships due to creating association copies while dissolving inheritance, associations that are navigable from types in the application schema are dissolved. Navigable roles of such associations are transformed into attributes. When dissolving inheritance, the value types of such attributes will be changed to <<union>>s that represent feature type sets. These unions are tagged as described in Inheritance, allowing for a specific encoding as described in XML Schema encoding.

NOTE	The transformer that executes the processing described in this section is part of the full ShapeChange configuration (see Annex A). It has attribute *id* with value *TRF_DISSOLVE_ASSOCIATIONS*.

7.2.13. Inheritance

GML-SF does not support the representation of inheritance via XML Schema extension elements. Inheritance structures in the model therefore need to be dissolved. Attributes of a supertype are copied down into its subtypes, and associations to a supertype are copied, resulting in similar associations to all subtypes of the supertype. Unions are created to represent choices between the non-abstract classes in the set of a supertype and its subtype. Attributes with a supertype as value type receive the corresponding union as value type. At the end of the transformation, inheritance relationships and abstract types are removed. More detailed documentation of this existing transformation can be found online [http://shapechange.net/transformations/flattener/#rule-trf-cls-flatten-inheritance].

The transformation has been extended to add the tagged value *representsFeatureTypeSet* to a union created by the transformation, when the union represents a supertype and its subtypes and these types are feature types. This tagging supports a specific encoding as described in XML Schema encoding.

The transformation is configured to exclude various general datatypes from the NAS profile that are basic types: StrucTextUnconstrained, TextLexUnconstrained, and TextNonLexUnconstrained - as well as their subtypes. Not doing so would cause the dissolving of complex types (see Complex types) to split a property with one of these supertypes into multiple properties to represent the choices available for the supertypes. This should be prevented. Instead, the transformation that dissolves complex types simply maps the aforementioned types to their simple base types.

NOTE	The transformer that executes the processing described in this section is part of the full ShapeChange configuration (see Annex A). It has attribute *id* with value *TRF_FLATTEN_INHERITANCE*.

7.2.14. Multiplicity

GML-SF0 does not allow elements with attribute *maxOccurs* greater than 1. Consequently, properties of the conceptual schema that have a maximum multiplicity greater than 1 must be transformed into a set of 1..N properties, each with maximum multiplicity equal to 1. The

properties are copies of the original property. An index number is appended to the property name so that each property has a unique name within the class scope. If a property is not copied (N=1) then the name is not changed.

NOTE To generate suitable documentation in the CDB dictionaries (see chapter 8), the index number would also be appended to the alias of the property, using " - " as separator.

To control how many copies of a property are created, tagged value *maxOccurs* is set on the property. If the tagged value is not set, ShapeChange will use a global default (for further details, see the documentation of the existing rule to flatten multiplicity [http://shapechange.net/transformations/flattener/#rule-trf-prop-flatten-multiplicity]).

Example 4. Restricting the multiplicity of 'resourceConstraints' properties

In the NAS, *ResourceConstraints* are used by *Entity* and *InformationEntity* types. When the NAS profile has been transformed, resource constraints will occur several times in every feature type. Setting the *maxOccurs* tagged value for properties called 'resourceConstraints' to a low number (1, for example) avoids clogging up the schema. The tagged value can be set to a higher number for specific properties, as needed.

NOTE The transformer that executes the processing described in this section is part of the full ShapeChange configuration (see Annex A). It has attribute *id* with value *TRF_FLATTEN_MULTIPLICITY*.

7.2.15. Complex types

As outlined in Navigability, GML-SF0 only supports feature types with a simple structure. Complex types that are not feature types, enumerations or code lists therefore need to be dissolved. This is achieved by recursively copying their properties into the types that have properties with these complex types as value type. Then the complex types are removed. This way, in the GML-SF0 NAS profile, feature types will have properties with value type either being a feature type, an enumeration, a code list, or a simple type. More details on this existing transformation are available online [http://shapechange.net/transformations/flattener/#rule-trf-prop-flatten-types].

NOTE Specific object types can be transformed to feature types as described in Object types as feature types, and would thus not be dissolved.

NOTE When copying properties, the names of the copies are updated to include the name of the property whose value type is the owner of the property that is copied. Example: *address.valueOrReason.value*. To generate suitable documentation in the CDB dictionaries (see chapter 8), the documentation of the copies is also updated. More specifically, the definition, description, alias, and primary code of the property copy and the property that is replaced by the copy are merged, using " : " as separator.

The transformation can also map the value types of properties. This mechanism can be used to map

types from external schemas. For example, MD_Identifier is mapped to CharacterString, with the assumption that the CharacterString represents the codespace and code of the MD_Identifier. In general, the ISO 19115-1 types do not need to be mapped, since they can be encoded by reference as defined by GML-SF. This is described in more detail in XML Schema encoding.

A number of rules for modifying the behavior for dissolving complex types have been added to meet Testbed-13 requirements:

- *rule-trf-prop-flatten-types-ignoreUnionsRepresentingFeatureTypeSets* - Unions that represent feature type sets (see Inheritance and Geometry specific feature types) are not dissolved. When encoding the GML-SF0 NAS profile, these unions are handled by a specific conversion rule (for further details, see XML Schema encoding).
- *rule-trf-prop-flatten-types-removeMappedTypes* - Mapped types are removed from the model. This prevents the encoding of global simple types (that represent basic types) in the GML-SF0 NAS profile. GML-SF does not support global simple types.
- *rule-trf-prop-flatten-types-ignoreSelfReferenceByPropertyWithAssociationClassOrigin* - When dissolving a complex type A by copying its properties into another type B, properties that were originally association roles, where the association had an association class, and that represent a self reference to the type B as explained in Association classes are not copied.

NOTE	The transformer that executes the processing described in this section is part of the full ShapeChange configuration (see Annex A). It has attribute *id* with value *TRF_FLATTEN_TYPES*.

7.2.16. Geometry specific feature types

Information about the geometry types supported by a given feature type has been extracted from OCL constraints in a previous processing step (see Geometry restrictions). If place information of a feature type can be represented by multiple different geometry types, geometry type specific copies of the feature type are created (and the original feature type is removed). A suffix is added to the name of each copy to indicate which geometry type it has, and to make the type name unique in the schema.

Example 5. Creating geometry specific copies of NAS feature type River

The place of the NAS feature type River cannot be represented by a PhysicalAddressInfo or PointPositionInfo. Since PhysicalAddressInfo and LocationInfo have been removed in a previous processing step (see Removing types), that leaves only CurvePositionInfo and SurfacePositionInfo. Two copies of River are created:

- River_C - with place only represented by a GM_Curve - and
- River_S - with place only represented by a GM_Surface

NOTE	To generate suitable documentation in the CDB dictionaries (see chapter 8), the alias of a feature copy is also updated. However, for the alias, the suffix is created by using " : " as separator and then either "Point", "Curve", or "Surface". Example: "River : Curve".

This processing step is included in the workflow to support technologies like Esri Shapefiles that require that a feature has a single geometry.

NOTE	The transformer that executes the processing described in this section is part of the full ShapeChange configuration (see Annex A). It has attribute *id* with value *TRF_HOMOGENEOUS_GEOMETRIES*.

7.2.17. Naming

The model transformations described in previous sections result in long property names. Such a name reflects the origin of a given property, and the processing steps applied to it. Specific name components of a transformed property can be removed to make the property name more readable in the GML-SF0 NAS profile. For Testbed-13, the following regular expressions (with syntax supported by Java [https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html]) were defined to identify name components to remove:

- \.value(s)?OrReason
- \.place-(curve|surface|point)PositionInfo
- \.geometry

Example 6. Removing unnecessary name components of transformed properties

- Property name 'place.place-curvePositionInfo.geometry' is transformed to 'place'.
- Property name 'waterResourceInfo.waterUse.valuesOrReason_2' is transformed to 'waterResourceInfo.waterUse_2'.

Additional regular expressions could be defined. However, they should be defined with care, because removing name components must not result in a class having two properties with same name.

NOTE	The transformer that executes the processing described in this section is part of the full ShapeChange configuration (see Annex A). It has attribute *id* with value *TRF_FLATTEN_REMOVE_NAME_COMPONENT*.

7.2.18. Tagged Values

The last transformation before encoding the model as an XML Schema is to set specific tagged values:

- *gmlsfComplianceLevel* is set to value '0' on the NAS profile application schema package.
- To avoid creation of property types, which are not supported by GML-SF (see GML-SF section

8.4.3), tagged value *noPropertyType* is set to 'true' on all classes of the NAS profile schema.

- To ensure that relationships to feature types are encoded by reference, with an appinfo element as required by GML-SF (see GML-SF section 8.4.4.13), tagged value *inlineOrByReference* is set to 'byReference' on all properties in the NAS profile schema that have a feature type as value type.

NOTE	The transformer that executes the processing described in this section is part of the full ShapeChange configuration (see Annex A). It has attribute *id* with value *TRF_TAGGED_VALUES*.

7.2.19. XML Schema encoding

To encode a GML-SF0 schema, the following conversion rules have been added to ShapeChange:

- *rule-xsd-pkg-gmlsf* - Creates a schema annotation to indicate the GML-SF compliance level as defined by GML-SF section 7.4. The compliance level is read from tagged value *gmlsfComplianceLevel* on the application schema. Furthermore, the rule creates an import for the gmlsfLevels.xsd, as defined by GML-SF section 8.3.2.
- *rule-xsd-cls-codelist-gmlsf* - Encodes a property with codelist value type as specified by GML-SF section 8.4.4.14.
- *rule-xsd-prop-featureType-gmlsf-byReference* - Applies simple byReference encoding of properties with a feature type as value type, as defined by GML-SF section 8.4.4.13.
- *rule-xsd-prop-metadata-gmlsf-byReference* - Applies simple byReference encoding of properties with metadata as value type (determined by tagged value *isMetadata* on the property, and map entries in the ShapeChange configuration).
- *rule-xsd-cls-union-omitUnionsRepresentingFeatureTypeSets* - Do not encode unions that represent feature type sets, i.e. unions with tagged value *representsFeatureTypeSet*='true'. Properties with such unions as type shall be implemented as elements with type gml:ReferenceType. An appinfo annotation is added to properties that have a union that represents a feature type set as type. For each feature type of this set, a GML target element is added to the appinfo annotation.

NOTE	This appinfo annotation would not be added if *rule-xsd-prop-targetElement* was not contained in the encoding rule.

- *rule-xsd-prop-constrainingFacets* - Generate constraining facets based on tagged values *length*, *maxLength*, *size*, *pattern*, *rangeMaximum*, and *rangeMinimum*.

A specific encoding rule has been defined to derive the GML-SF0 NAS profile schema. The encoding rule contains the aforementioned conversion rules. In addition, it uses conversion rules from the encoding rule defined by ISO 19136:2007 and the encoding rule to derive the full GML schema for the NAS.

Conversion rules from the ISO 19136:2007 encoding rule that are NOT included in the encoding rule for the GML-SF0 NAS profile schema:

- *rule-xsd-cls-global-enumeration* - This rule has been replaced by *rule-xsd-cls-local-enumeration*. To be compliant with GML-SF, a schema must encode enumerations as anonymous types with

local scope, instead of named simple types with global scope.

- *rule-xsd-cls-byValuePropertyType* - Encoding of by-value property types is not allowed by GML-SF.
- *rule-xsd-pkg-gmlProfileSchema* - Declaration of a GML profile schema is not covered by GML-SF. GML-SF itself already defines a number of restrictions for elements with geometric content (see GML-SF section 8.4.4.11).
- *rule-xsd-prop-reverseProperty* - Encoding an appinfo element to document the reverse property (of an association role) is not allowed by GML-SF.
- *rule-xsd-prop-defaultCodeSpace* - Encoding an appinfo element with a gml:defaultCodespace is not allowed by GML-SF.

Conversion rules from the encoding rule to derive the full GML schema for the NAS that are NOT included in the encoding rule for the GML-SF0 NAS profile schema:

- *rule-xsd-cls-union-asCharacterString*, *rule-xsd-cls-union-asGroup*, *rule-xsd-cls-union-direct* - <<union>> types from the model have been dissolved during the model transformation (see Complex types).
- *rule-xsd-cls-codelist-constraints*, *rule-xsd-pkg-schematron* - OCL constraints have been converted to text constraints in a previous transformation (see Convert OCL constraints to text constraints), and thus no schematron will be derived for the GML-SF0 NAS profile.
- *rule-xsd-prop-length-size-pattern* - This rule has been replaced by the new *rule-xsd-prop-constrainingFacets*, which also takes into account tagged values *rangeMinimum* and *rangeMaximum* that have been copied from basic types (see Basic types).
- *rule-xsd-prop-xsdAsAttribute* - The encoding of properties as attributes is not allowed by GML-SF.
- *rule-xsd-prop-nillable*, *rule-xsd-prop-nilReasonAllowed* - Attribute *nillable* is prohibited by GML-SF on compliance level 0 (see GML-SF section 8.4.4.4).
- *rule-xsd-prop-initialValue* - Default or fixed attributes to represent initial values defined for properties are not allowed by GML-SF.
- *rule-xsd-prop-att-map-entry* - Reuse of global attribute and attributeGroup schema components is not allowed by GML-SF.
- *rule-xsd-all-tagged-values* - The addition of annotations with information on specific tagged values of model elements is not allowed by GML-SF.
- *rule-xsd-rel-association-classes* - Association classes have been dissolved (see Association classes).

NOTE	*rule-xsd-all-no-documentation* prevents the generation of annotations with documentation of model elements. The rule is included since GML-SF does not appear to allow such documentation of schema elements.

NOTE	The new encoding rule is part of the full ShapeChange configuration shown in Annex A. Look for the element `<EncodingRule name="gmlsf">`.

7.3. Adaptation to other NAS Profiles

This section discusses how to adapt the workflow for creating a GML-SF0 implementation schema to another NAS profile. The following list covers each of the processing steps of the workflow, and documents which - if any - changes are required. Changes to the ShapeChange configuration (see Annex A) and the UML model of the NAS profile may be necessary.

1. Geometry restrictions - No changes are required
2. Convert OCL constraints to text constraints - No changes are required
3. Geometry type inheritance - No changes are required
4. Removing types - The set of types to remove by this processing step can be modified. Depending on the requirements of the actual use case, the ShapeChange configuration parameter 'removeType' of the transformation with ID 'TRF_REMOVE_TYPE' would be updated to list the types that shall be removed. As documented for the processing step, at least the types 'LocationInfo' and 'PhysicalAddressInfo' should be removed, until the impact of keeping these types on systems that consume the GML-SF0 schema has been analyzed. Other types such as 'Dataset' and the various collection types have been removed, since a GML-SF0 schema may contain no more than one feature collection. Additional types can be removed to produce a GML-SF0 schema with a manageable size (both from the perspective of schema creation by ShapeChange and the consumption of the schema - as well as actual data - by other software). The size of the schema is also influenced by the decisions made in subsequent processing steps, namely 'Navigability', 'Object types as feature types', and 'Multiplicity'. If in doubt, keep the value of the configuration parameter 'removeType' as defined in Testbed-13.
5. Metadata types - No changes required
6. Removing documentation - No changes required
7. Reason unions - No changes required
8. Basic types - No changes required
9. Navigability - As documented for this processing step, cycles caused by complex types (object types, data types, and unions) can be problematic and should be avoided if possible. ShapeChange provides a number of configuration options to decrease the number of cycles in the model:
    - In the ShapeChange transformation with ID 'TRF_REMOVE_OBJECT_TO_FEATURE_TYPE_NAVIGABILITY', use the parameter 'removeObjectToFeatureNavRegex' to identify the object types whose properties with a feature type as type shall become non-navigable. The parameter value is a regular expression that matches on the name of an object type from the NAS profile. You could add all object types with the exception of those object types that will be transformed into feature types later on by the transformation with ID 'TRF_TO_FEATURE_TYPE'. If in doubt, keep the value set in Testbed-13 (to apply this transformation to all object types).
    - The transformation 'TRF_REMOVE_OBJECT_TO_FEATURE_TYPE_NAVIGABILITY_2' was introduced to make object typed properties of specific object types non-navigable. Again, use the parameter 'removeObjectTyoFeatureNavRegex' to identify the object types that shall be processed by the transformation. For further details, see the examples given for this processing step. Keep in mind that the NAS PositionInfo types should not be matched by the

regular expression defined by the configuration parameter value.

- Fine grained reduction of cycles can also be achieved by setting the tagged value 'isFlatTarget' to 'true' on specific association ends in the model. For further details, see the examples in the documentation of this processing step. If prevention of cycles is fully accomplished by setting this tagged value in the model, the previous two transformations are not needed. In that case, set the XML attribute 'input' on the transformation with ID 'TRF_REMOVE_NAVIGABILITY_BASED_ON_ISFLATTARGET' to 'TRF_MAP_TO_SIMPLE_BASE_TYPE' and remove the two transformations with IDs 'TRF_REMOVE_OBJECT_TO_FEATURE_TYPE_NAVIGABILITY' and 'TRF_REMOVE_OBJECT_TO_FEATURE_TYPE_NAVIGABILITY_2' in the ShapeChange configuration.

10. Association classes - No changes are required
11. Object types as feature types - As documented for this processing step, it may be desirable to keep object types from the NAS profile as individual objects in the GML-SF0 schema, by transforming them into feature types. Which object types should be transformed must be decided by a domain expert. Use the configuration parameter 'toFeatureTypeNameRegex' from the transformation with ID 'TRF_TO_FEATURE_TYPE' to identify these object types (via a regular expression that matches on the name of a type).
12. Associations - No changes required
13. Inheritance - No changes required
14. Multiplicity - Since the GML-SF0 schema does not support properties with a multiplicity greater than 1, copies of such properties from the NAS profile must be created, as documented for this processing step. The number of copies for a particular property can be controlled by setting the tagged value 'maxOccurs' on that property in the UML model to an according integer value. If the tagged value is not defined for a property, ShapeChange will use a default given by the configuration parameter 'maxOccurs' of the transformation with ID 'TRF_FLATTEN_MULTIPLICITY'. In Testbed-13, that parameter has been set to 2. You may want to configure a different value.
15. Complex types - If desired, this processing step can map value types of properties to other types. This mapping facility should be used for cases in which complex types from external schemas occur as property types, and these types would not be encoded by reference in the GML-SF0 schema. In Testbed-13 an according mapping has exemplarily been defined in the configuration - see the transformation with ID 'TRF_FLATTEN_TYPES', and in particular the `<ProcessMapEntry>` defined by it. The XML attribute 'rule' should be kept as is, only the 'type' and 'targetType' attributes should be set, to define the source and the target of the mapping.
16. Geometry specific feature types - No changes required
17. Naming - The transformation of this processing step can be used to remove unnecessary components of property names that were generated by the previous processing steps. You can use parameter 'removePropertyNameAndCodeComponent' of the transformation with ID 'TRF_FLATTEN_REMOVE_NAME_COMPONENT' to remove such name components. The parameter contains a comma-separated list of regular expressions that match unnecessary name components. Be careful when adding a regular expression: it shall not lead to a class having two properties with the same name. If in doubt, keep the value of the configuration parameter 'removePropertyNameAndCodeComponent' as defined in Testbed-13.

18. Tagged values - No changes required
19. XML Schema encoding - The parameters and rules of the XML Schema target can be kept as is. If the NAS profile used other types from external schemas (especially the ISO standards) than the NAS profile from Testbed-13, XSD map entries would need to be added for these types. For further details on the configuration of such map entries, see the ShapeChange documentation [http://shapechange.net/targets/xsd/#XSD_Map_Entries].

7.4. Conclusion

The Workflow developed in Testbed-13 can transform a NAS profile into a GML-SF0 schema suitable for implementation as an Esri Geodatabase or Esri Shapefiles (and thus use in the CDB data store).

The workflow contains processing steps that are applicable for transforming any UML application schema into a GML-SF0 schema. Some processing steps have specifically been developed in Testbed-13 to derive a GML-SF0 schema from the NAS (or a profile thereof): Geometry restrictions, Geometry type inheritance, Metadata types, Reason unions, and Basic types. However, the transformations are reusable, and can therefore be used in workflows that produce other implementation schemas.

A GML-SF0 schema can serve as input for deriving CDB feature and attribute dictionaries (see chapter 8). However, deriving CDB dictionaries from a fully compliant GML-SF0 schema can have the following limitations:

- A GML-SF0 schema can only define a single feature collection. An application schema may contain several collection types, which may be relevant for building a CDB compliant data store. The NAS, for example, defines collection types such as *Dataset*, *EntityCollection*, and *FeatureEntityCollection*. In order to have relevant collection types in the CDB feature dictionary, the GML-SF0 workflow would be adapted (to keep these types and to convert them to feature types). The resulting schema - and thus the CDB feature dictionary derived from it - would then have multiple collection types.

Chapter 8. Generating CDB output from a NAS profile

8.1. Introduction

Tiled Vector Datasets are one dataset type defined in the OGC CDB 1.0 standard. The CDB standard provides a set of pre-defined feature types ("Feature Data Dictionary") and attribute types ("CDB Attributes").

In OGC CDB 1.0 the list of feature types is fixed and the standard does not provide an extension mechanism for feature types.

The situation is different for attributes. In addition to CDB Attributes, the standard also supports "Geomatics Attributes" and "Vendor Attributes", where the definition of the Geomatics Attributes group fits the characteristics of attribute types from the NAS.

> Geomatics attributes are attributes whose semantics, data type, length, format, range, usage, and units, are governed by various governmental/industrial specifications and standards. Such attributes are generally found in source data that conforms to such standards and specifications. While the CDB standard itself does not define and govern the usage of these attributes, it nonetheless accommodates their storage within the repository structure of a CDB compliant dataset/data store.
>
> — CDB Volume 1 - section 5.7.1.2.6.2

To define Geomatics Attributes, a file `Geomatics_Attributes.xml` has to be placed in the global metadata directory. The file has to validate against Vector_Attributes.xsd [http://schemas.opengis.net/cdb/1.0/Vector_Attributes.xsd] which is the same information as has to be provided for the CDB Attributes. More information on the attribute metadata is provided in OGC CDB 1.0 Volume 1, section 5.1.7.

CDB feature data may include any feature-attribute combination as long as both the feature type and the attribute type are specified in the metadata, i.e. the fixed Feature Data Dictionary and one of the attribute type dictionaries.

Testbed-13 investigated how CDB could support:

- Schemas (a specification of the feature types that may occur in a dataset along with their attributes and additional information like multiplicity)
- Feature types and attribute types other than those specified in the OGC CDB 1.0 standard

The NAS-based Urban Military profile was used for this purpose.

To be consistent with the level of complexity supported by CDB applications, schemas for NAS-profiles were constrained to those suitable for an Esri Geodatabase and a GML-SF0 application

schema. As a specification of the schema, a GML-SF0 application schema was used in Testbed-13.

To derive the necessary artifacts using ShapeChange, the following steps are required:

1. Transform the conceptual UML model (NAS-based Urban Military profile) into an implementation model suitable for an Esri Geodatabase or GML-SF0 application schema;
2. Generate a GML-SF0 application schema from that implementation model;
3. Generate CDB feature/attribute dictionaries from that implementation model.

The first two steps are described in chapter 7, the last step is described in this chapter.

8.2. Generating the CDB feature and attribute type dictionaries

WARNING	An application schema may contain properties with identical names, but different semantics (in application schemas in UML, properties are defined in the scope of the classifier). If multiple application schemas are used to generate a feature data dictionary, the same feature type name may be used in more than one application schema. These cases would create conflicts. The following discussion assumes that all feature and property names have a consistent definition and only the first occurrence of a feature type or attribute name will be written to the dictionary.

NOTE	The ShapeChange configuration to derive the CDB dictionaries is contained in Annex A.

8.2.1. The CDB Feature Data Dictionary

As stated, the OGC CDB 1.0 standard includes a fixed set of feature types that may be used. These are specified in the CDB Feature Data Dictionary, in the form of an XML file. The feature type definitions are compiled from different sources including DIGEST/FACC, NGFCD or SEDRIS.

If future revisions of CDB are intended to support the NAS (and other application schemas), the CDB standard needs to support a capability to define the feature types used by the application schema.

The easiest approach for existing CDB implementations seems to be to simply replace the CDB Feature Data Dictionary with a new one containing the feature types from the application schema. In the following, it is assumed that this approach is taken.

NOTE	Since the codes in the dictionary are used to define paths and file names to resources in a CDB data store, this has implications for the links to files in the data store. The assumption is that the data store file structure would continue to use the codes from the user-defined dictionary - and the links to the files would, therefore, use the user-defined codes, too.

IMPORTANT	This would require a Change Request to the OGC CDB 1.0 standard, but for the purpose of the exercise in Testbed-13 it is assumed that the next version of CDB will support that the schema and its associated feature types may be specified along with a feature dataset. The Testbed-13 CDB Engineering Report is expected to analyze the options in more detail and submit a Change Request that will support datasets based on the NAS or a NAS profile (or some other application schema that conforms to ISO 19109).

The contents of a CDB feature data dictionary are defined in OGC CDB 1.0 Volume 1 section 3.3.8.1 and OGC CDB 1.0 Volume 11 chapter 5.

The following XML snippet is an excerpt of the dictionary file with one feature type ("Well").

```
<Feature_Data_Dictionary version="1.1">
  <Category code="A">
    <Label>Culture</Label>
    <Subcategory code="A">
      <Label>Extraction</Label>
      ...
      <Feature_Type code="050">
        <Label>Well</Label>
        <Subcode code="000">
          <Label>Well</Label>
          <Concept_Definition>A hole drilled or dug into the earth or sea bed for the
extraction of liquids or gases. (See also BH170)</Concept_Definition>
          <Recommended_Dataset_Component>GSFeature
Point</Recommended_Dataset_Component>
          <Origin>DIGEST 2.1</Origin>
        </Subcode>
      </Feature_Type>
      ...
```

There are several assumptions hard-coded in this dictionary file that complicate the representation of NAS feature types in a CDB Feature Data Dictionary. These are partly a result of the DIGEST legacy, partly because of early CDB design decisions:

- Feature types are organized into a two-level hierarchy (category and sub-category). This information is used in the tiling scheme.
- Categories and sub-categories are identified by a single character code.
- Feature types are identified by two three-digit codes ("code" and "subcode").
- The feature type can be used in one or more of the pre-defined CDB dataset components for vector feature data:
  - GSFeature
  - GTFeature
  - GeoPolitical

- RoadNetwork
- RailRoadNetwork
- PowerLineNetwork
- HydrographyNetwork

The geometric dimension is stated as "Point", "Lineal" or "Areal".

The entry in the feature data dictionary includes one or more recommended pairs of dataset and geometric dimension.

The NAS-based Urban Military Profile also contains a feature type "Well". The NAS also uses a two-level grouping hierarchy ("Cultural" > "Extraction Facilities" > "Well"). I.e., the NAS fits with this fixed structure in general, but this will not be the case for other application schemas.

IMPORTANT	Extending the CDB standard to support a more flexible grouping of feature types (1..n category levels) would be beneficial.

However, the packages in the NAS do not have a single-character code. The codes of the relevant elements (tagged values "primaryCode" / "secondaryCode") are:

- Cultural: "Cultural" / "CUL"
- Extraction Facilities: "ExtractionFacility" / "EXTR"
- Well: "Well" / "AA050"

There are two differences here:

1. The length constraints of the codes in CDB.
2. The assumption that the full code of the feature is a concatenation of the category code, sub-category code and the feature type code.

IMPORTANT	The CDB standard should reconsider the constraints on the allowed values of codes.

Assuming that the constraints are removed, and that the primary codes can be used, Feature_Data_Dictionary.xml would contain:

```
<Feature_Data_Dictionary version="v1.0">
  <Category code="Cultural">
    <Label>Cultural</Label>
    <Subcategory code="ExtractionFacility">
      <Label>Extraction Facilities</Label>
      ...
      <Feature_Type code="Well">
        <Label>Well</Label>
        <Subcode code="000">
          <Label>Well</Label>
          <Concept_Definition>An excavation drilled or dug into the ground (for
example: the sea bed) for the extraction of liquids and/or gases.</Concept_Definition>
          <Recommended_Dataset_Component></Recommended_Dataset_Component>
          <Origin>OGC Testbed 13 Application Schema v1.0</Origin>
        </Subcode>
      </Feature_Type>
      ...
```

`Subcode` does not have a corresponding concept in the General Feature Model, which underpins ISO 19109 application schemas like the NAS. Therefore having `Subcode` as optional in the CDB dictionary would be a positive enhancement. For now exactly one subcode is included per feature type with a (dummy) code of "000", to validate against the CDB 1.0 schema.

As the dataset component is "just" recommended and there is no information in the NAS profile that could be used to make a proper recommendation, the `Recommended_Dataset_Component` element is left empty.

NOTE

Well is abstract in the NAS and, therefore, would not appear in a CDB Feature Data Dictionary for the NAS-based Urban Military Profile. Only the instantiable subtype WaterWell would be included as there will never be any Well features in the dataset. But ignore this for now, since this discussion is only concerned with how the values of the XML elements in the CDB Feature Data Dictionary can be derived from the profile UML model.

Use the following template and substitution values:

```
<Feature_Data_Dictionary version="{version}">
  <Category code="{bundle-primaryCode}">
    <Label>{bundle-name}</Label>
    <Subcategory code="{leaf-primaryCode}">
      <Label>{leaf-name}</Label>
      <Feature_Type code="{feature-type-model-name}">
        <Label>{feature-type-name}</Label>
        <Subcode code="000">
          <Label>{feature-type-name}</Label>
          <Concept_Definition>{feature-type-definition} [desc] {feature-type-
description}</Concept_Definition>
          <Recommended_Dataset_Component></Recommended_Dataset_Component>
          <Origin>{application-schema-name} {version}</Origin>
        </Subcode>
      </Feature_Type>
      ... additional feature types in the same leaf package
    </Subcategory>
    ... additional leaf packages / sub-categories in the same bundle package
  </Category>
  ... additional bundle packages / categories
</Feature_Data_Dictionary>
```

NOTE	The prefix "[desc] " and {feature-type-description} are only used if a description exists.

Substitution values:

- {application-schema-name} = Name of the application schema package (stereotype <<ApplicationSchema>>)
- {version} = Tagged value `version` of the application schema package
- {bundle-name} = "Default" if the package that owns the feature types listed under this category is a schema; otherwise it is the alias of the parent of that package (or the name of the parent package, if it has no alias). For the NAS, this is typically the alias of a package with stereotype <<bundle>>.
- {bundle-primaryCode} = Tagged value `primaryCode` of the bundle package; the name of the package is used as fallback
- {leaf-name} = "Default" if the package that owns the feature types listed under this subcategory - or its parent package - is a schema; otherwise it is the alias of the package (or the name of the package, if it has no alias). For the NAS, this is typically the alias of a package with stereotype <<Leaf>>.
- {leaf-primaryCode} = Tagged value `primaryCode` of the leaf package; the name of the package is used as fallback
- {feature-type-name} = Human readable name of the feature type class (stereotype <<FeatureType>>), for NAS it is given by tagged value `name` of a feature type class.

NOTE	ShapeChange uses the alias, a particular descriptor [http://shapechange.net/get-started/config/input/#Descriptor_sources], of a feature type, to identify its human readable name. If no alias is available, the name of the feature type, as defined in the UML model, is used as fallback.

- {feature-type-model-name} = Name of the UML class of the feature type
- {feature-type-definition} = Definition of the feature type class, in case of the NAS it is given by tagged value `definition`
- {feature-type-description} = Description of the feature type class, in case of the NAS it is given by tagged value `description`

For the code of the feature type (`Feature_Type/@code`), use the name of the UML class as this is the name of the feature element in the GML-SF0 application schema. Both must be identical so that the entry in the feature data dictionary can be found for each feature.

WARNING	The resulting dictionary will be valid against the XML schema http://schemas.opengis.net/cdb/1.0/Feature_Data_Dictionary.xsd, but breaks other requirements from the OGC CDB 1.0 standard as described above.

8.2.2. The CDB Geomatics Attributes Dictionary

As stated in the introduction, the OGC CDB 1.0 standard supports the definition of attribute types from an application schema in a dictionary for Geomatics Attributes. The contents of this dictionary are defined in OGC CDB 1.0 Volume 1 section 5.1.7 and OGC CDB 1.0 Volume 11 section 3.9.

The following XML snippet is an excerpt of the dictionary file with one attribute type.

```
<Vector_Attributes version="3.2">
  <Attributes>
    ...
    <Attribute code="3" symbol="AO1">
      <Name>Angle of Orientation with greater than 1 degree resolution</Name>
      <Description>
        The angular distance measured from true north (0 deg) clockwise to the major
(Y) axis of the feature.
        If the feature is square, the axis 0 through 89.999 deg shall be recorded.
        If the feature is circular, 360.000 deg shall be recorded.
        Recommended Usage. CDB readers should default to a value of 0.000 if AO1 is
missing.
        Applicable to Point, Light Point, Moving Model Location and Figure Point
features.
        When used in conjunction with the PowerLine dataset, AO1 corresponds to the
orientation of the Y-axis of the modeled pylon.
        The modeled pylon should be oriented (in its local Cartesian space) so that
the wires nominally attach along the Y-axis.
        Refer to Appendix A - "Creating a 3D Model for a Powerline Pylon" for
additional usage guidelines.
      </Description>
      <Level>
        <Instance>Preferred</Instance>
        <Class>Not Supported</Class>
        <Extended>Supported</Extended>
      </Level>
      <Value>
        <Type>Numeric</Type>
        <Format>Floating-Point</Format>
        <Precision>3.3</Precision>
        <Range interval="Right-Open">
          <Min>0</Min>
          <Max>360</Max>
        </Range>
        <Unit>2</Unit>
      </Value>
    </Attribute>
    ...
  </Attributes>
  <Units>
    ...
    <Unit code="2" symbol="deg">
      <Name>degree</Name>
      <Description>To measure an angle.</Description>
    </Unit>
  </Units>
  ...
  ... scalars may be added, too
</Vector_Attributes>
```

The following listing shows how one of the attributes of the NAS feature Well ("length") could be mapped to this structure.

```
<Vector_Attributes version="v1.0">
  <Attributes>
    ...
    <Attribute code="123" symbol="length">
      <Name>Length</Name>
      <Description>
        The dimension of a feature taken along its primary alignment of use and
generally in the horizontal plane. [desc] The primary alignment of a feature is its
established direction of flow or use (for example: a road, a power line, a river, a
rapid, and/or a bridge). A feature-specific rule may apply. In the case of a bridge,
the length is the distance between the bridge abutments along the bridge centreline.
In the case of a dam, the length is the distance along the dam crest. If no
established direction of flow or use exists then (1) if the feature is irregular in
shape its length is its greatest horizontal dimension (see Attribute: 'Greatest
Horizontal Extent'), else (2) if the feature is regular in shape then a shape-specific
rule may apply: for a rectangular feature, the length of the longer axis; for a round
feature, the diameter.
      </Description>
      <Level>
        <Instance>Preferred</Instance>
        <Class>Not Supported</Class>
        <Extended>Not Supported</Extended>
      </Level>
      <Value>
        <Type>Numeric</Type>
        <Format>Floating-Point</Format>
        <Unit>1</Unit>
      </Value>
    </Attribute>
    ...
  </Attributes>
  <Units>
    ...
    <Unit code="1" symbol="m">
      <Name>metre</Name>
    </Unit>
  </Units>
  ...
</Vector_Attributes>
```

The following assumptions were used in this mapping:

- The code XML attributes are integer values that are assigned incrementally and used only for internal referencing.
- In the feature data dictionary the XML element-name is "definition" and for attributes it is "description". It is assumed that these elements are meant to include both the definition and the

description of the model element.

- The primary codes (in this case: "length") are used instead of the secondary code ("LZN").
- The level "instance" results in the values being stored in the Shapefile (or whatever format will be used in the future - e.g., GML-SFO, Geodatabase, GeoJSON, etc.) - as it should be. So the value has been set to "Preferred" and both other levels have been set to "Not Supported".
- Requirement 77 in OGC CDB 1.0 Volume 1 section 5.1.7.1 is unclear whether `<Precision/>` has to be provided for floating point values, but based on discussion on the CDB SWG mailing list this is not the case. The element has, therefore, been omitted. Note that the model of the NAS profile does not include this information.

This example used the following template and substitution values:

```
<Vector_Attributes version="v1.0">
  <Attributes>
    ...
    <Attribute code="{attribute-counter}" symbol="{attribute-model-name}">
      <Name>{attribute-name}</Name>
      <Description>{attribute-definition} [desc] {attribute-description}</Description>
      <Level>
        <Instance>Preferred</Instance>
        <Class>Not Supported</Class>
        <Extended>Not Supported</Extended>
      </Level>
      <Value>
        <Type>{value-type}</Type>
        <Format>{numeric-format}</Format>
        <Range>
            <Min>{range-minimum}</Min>
            <Max>{range-maximum}</Max>
        <Unit>{unit-code}</Unit>
      </Value>
    </Attribute>
    ...
  </Attributes>
  <Units>
    ...
    <Unit code="{unit-counter}" symbol="{unit-symbol}">
      <Name>{unit-name}</Name>
    </Unit>
  </Units>
  ...
</Vector_Attributes>
```

Some of the elements in the "template" are conditional:

- `<Attribute/>` is only provided, if the dictionary does not yet include an attribute with the same symbol (model name).
- The prefix "[desc] " and {attribute-description} are only used if a description exists.

- `<Format/>` is only provided for `Type=Numeric`
- `<Range/>` is only provided if the attribute has non-empty tagged value `rangeMinimum` or `rangeMaximum`.
- `<Unit/>` is only provided if the attribute has a non-empty tagged value `recommendedMeasure`
  - A new `<Unit/>` declaration is provided if no declaration for the `recommendedMeasure` exists yet.

NOTE

There does not appear to be a mechanism that supports referencing units across dictionaries in CDB 1.0. In Testbed-13, it was therefore decided to add the "standard CDB units" (defined in the CDB 1.0 standard attribute dictionary [http://schemas.opengis.net/cdb/1.0/Metadata/CDB_Attributes.xml]) to the CDB target in the ShapeChange configuration (see the ShapeChange configuration example in Annex A).

Whenever the `<name>` or `<alias>` elements of a corresponding CDBUnit definition from the ShapeChange configuration matches a `recommendedMeasure`, the unit definition from the ShapeChange configuration is used. The codes and symbols of standardized units can thus be used consistently across different CDB dictionaries. I.e., the unit with code "1" would use the symbol "m" for meter in all dictionaries.

Note that the unit definitions from the ShapeChange configuration are only used as needed. If none of the attributes from the application schema refer to a unit definition from the configuration, none of these definitions are added to the attribute dictionary.

The following substitution values are used:

- {attribute-counter} = Counter that is incremented by one for each new attribute
- {attribute-name} = Human readable name of the attribute (from a type with stereotype <<FeatureType>>), for NAS it is given by tagged value `name` of the attribute.

NOTE

ShapeChange uses the alias, a particular descriptor [http://shapechange.net/get-started/config/input/#Descriptor_sources], of the attribute, to identify its human readable name. If no alias is available, the name of the attribute, as defined in the UML model, is used as fallback.

- {attribute-model-name} = Name of the UML attribute in the model
- {attribute-definition} = Tagged value `definition` of the attribute
- {attribute-description} = Tagged value `description` of the attribute
- {value-type} = one of "Text" (CharacterString, Character, enumerations, code lists, references to resources [objects, webpages, etc.]), "Numeric" (Real, Integer, Decimal, Measure, etc.) or "Boolean" (Boolean); text is also the default value for any other type
- {numeric-format} = "Floating-Point" unless the data type is Integer, then it is "Integer"

NOTE

The numeric format can be defined in map entries of the CDB target in the ShapeChange configuration.

- {range-minimum} / {range-maximum} = Tagged value `rangeMinimum` / `rangeMaximum` of the attribute

NOTE	The transformation workflow described in chapter 7 has copied the tagged value from basic types to attributes, so that the tagged value is available there instead of only on the classifier.

- {unit-code} = value of {unit-counter} for the unit declaration where the `Name` is the same as the tagged value `recommendedMeasure`

NOTE	The name of the unit declaration may be different in the case that the value of tag `recommendedMeasure` matches the alias of a CDB unit definition from the ShapeChange configuration. Example: "metre" vs "meter".

- {unit-counter} = Counter that is incremented by one for each new unit

NOTE	If a unit definition from the ShapeChange configuration was used, and that unit definition declared a code, the code is used as-is (allowing re-use across CDB dictionaries).

- {unit-name} = Name of the unit definition from the ShapeChange configuration that matches the value of tag `recommendedMeasure` of the attribute
- {unit-symbol} has to be configured for each {unit-name} in the ShapeChange configuration

For the symbol of the attribute (`Attribute/@symbol`), the name of the UML attribute has to be used as this is the name of the feature property element in the GML-SF0 application schema. Both must be identical so that the entry in the attribute data dictionary can be found for each property.

WARNING	The resulting dictionary will be valid against the XML schema http://schemas.opengis.net/cdb/1.0/Vector_Attributes.xsd, but breaks other requirements from the OGC CDB 1.0 standard as described above.

NOTE	The "length" example is simplified as the GML application schema conforming to the Simple Features profile Level 0 may not include an attribute `length` in the `Well` feature, but may have a different name as a result of the various transformation steps.

8.3. Adaptation to other NAS Profiles

This section discusses how to adapt the workflow for creating CDB dictionaries to another NAS profile.

The first step of the workflow, the generation of the GML-SF0 implementation model, is covered in chapter 7, including the potentially required adaptations when working with another NAS profile.

The ShapeChange target that derives the actual CDB dictionaries from that implementation model has a number of configuration options. The configuration used in Testbed-13 is shown in Annex A.

- If additional common CDB unit definitions are available, add them to the `advancedProcessConfiguration` element. The configuration in Annex A contains a list of examples that should be straightforward to adapt. Also, define `<alias>` elements for each CDB unit definition where the `<name>` does not match the spelling used by the NAS (in recommended measures). An example is the unit definition for meter.
- Most of the target parameters can be kept in the current form. Only the parameter 'unitsToIgnore' may need to be updated. It should contain the list of recommended measures (from the 'recommendedMeasure' tagged values in the profile) that shall be ignored, typically because no mapping to a CDB unit is available.
- The rules can be kept as-is.
- The list of map entries defined in Testbed-13 can most likely be re-used for other NAS profiles. However, it can be extended to cover mappings for property types in the GML-SF0 implementation model which should not simply map to 'Text'. Mappings for additional numeric types would be the most likely use case for extension. The configuration in Annex A contains a list of examples that should be straightforward to adapt.

Chapter 9. Generating a CityGML ADE from a NAS profile

9.1. Mapping the NAS profile to CityGML

9.1.1. Overview

As described in the section Adding CityGML and relevant CityGML ADEs, a key difference between generating output for CDB and CityGML is that in the CityGML case the existing types and properties of CityGML 2.0 and the Utility Network ADE should be re-used, where possible.

That means that in a first step the model needs to be transformed so that:

- Feature types in the NAS profile extend existing CityGML feature types, where applicable, and
- Properties in the NAS profile are removed where they are covered by existing properties of the CityGML feature types.

First, it was analyzed how NAS features (related to utilities) relate to the CityGML UtilityNetwork ADE. Then the analysis focused on the mapping from each of these NAS features to CityGML features.

It is important to understand that it is not the intention of this exercise to identify the most correct mapping from the point of view of a domain expert. For this, the mapping should be done by a domain expert! Instead, the focus is on understanding the process well enough - and implement the required mechanisms in ShapeChange - so that a domain expert could perform the mapping from a NAS profile to CityGML and configure ShapeChange to derive an ADE for the NAS profile.

Therefore the mapping discussion below analyzes a subset of all feature types in the NAS profile. I.e., the feature types discussed in this chapter do not constitute the entirety of the profile in any topic area. The selection is intended to be broad enough to understand different **types** of mappings, not to map all of the feature types in the NAS profile to CityGML.

The mapping to CityGML is to some extent uncertain as the CityGML feature types have vague semantics. The CityGML model does not include any definitions for the types or their properties. In other words any mapping relies on an interpretation based on the names of the model elements.

When the descriptions in this chapter state that a NAS feature type is "mapped to" an existing CityGML/ADE feature type, this means that the NAS feature type will be transformed to a subtype of the CityGML/ADE feature type; i.e. an inheritance relationship is added in the transformation to the CityGML ADE implementation model.

9.1.2. Mapping the NAS profile to the CityGML Utility Network ADE

The NAS profile includes a package "utility infrastructure", which seemed like a good candidate for NAS features that would be "mapable" to a Utility Network ADE.

Utility Infrastructure

Buildings, non-building structures and equipment that form a set of interconnected elements supporting a utility network.

The following figure shows the feature types included in the NAS profile.

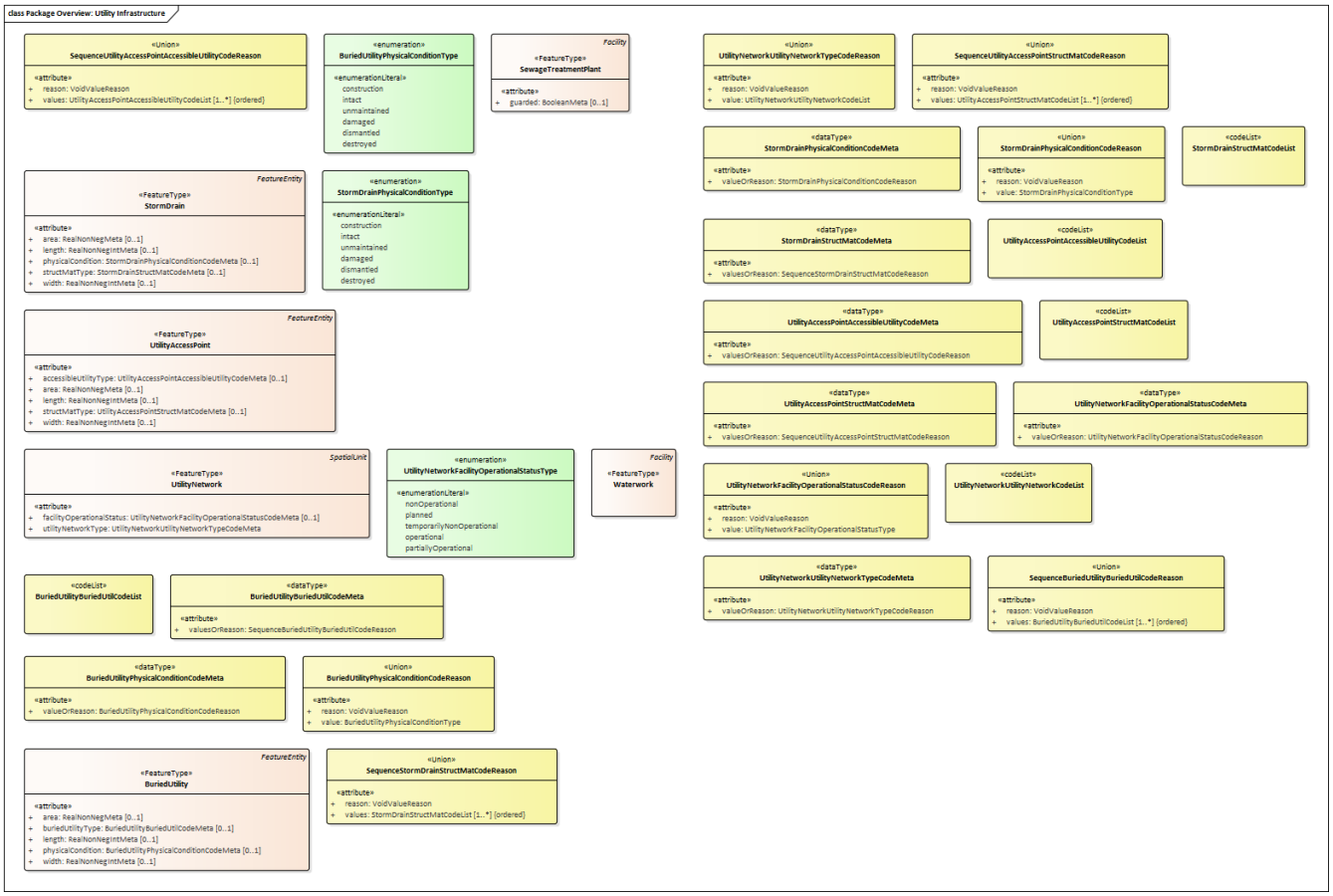

Figure 17. Overview of the "utility infrastructure" feature types in the NAS profile

The feature types, and their definitions, are:

BuriedUtility

Utilities or pipeline network that is located underground or below a waterbody.

SewageTreatmentPlant

An operational area with buildings and other facilities for the purification of wastewater.

StormDrain

A collector opening into a pipe or channel to allow the removal of excess runoff water or in some cases sewage.

UtilityAccessPoint

A location that provides access to underground utility tunnels, distribution lines, or drainage systems.

UtilityNetwork

A system of spatially dispersed but interconnected utility nodes (for example: facilities, buildings, non-building structures, equipment, and/or distribution devices) that support a utility service.

Waterwork

An establishment for storing, purifying, and supplying an area or town with water.

As it turns out, relating the NAS features to the CityGML Utility Network ADE is difficult. The main reason is that the ADE models a generic view to utility networks with a certain focus on the network topology while the NAS profile has more concrete feature types without a focus on the network characteristics.

NOTE	Another problem is that the current version of the ADE does not include documentation for the elements in the application schema. The ADE is still a draft and this is likely the reason. The lack of documentation in the application schema is a problem for mappings as the semantics must be derived just from the names - or maybe from studying additional documentation.

The following figure shows the core of the ADE.

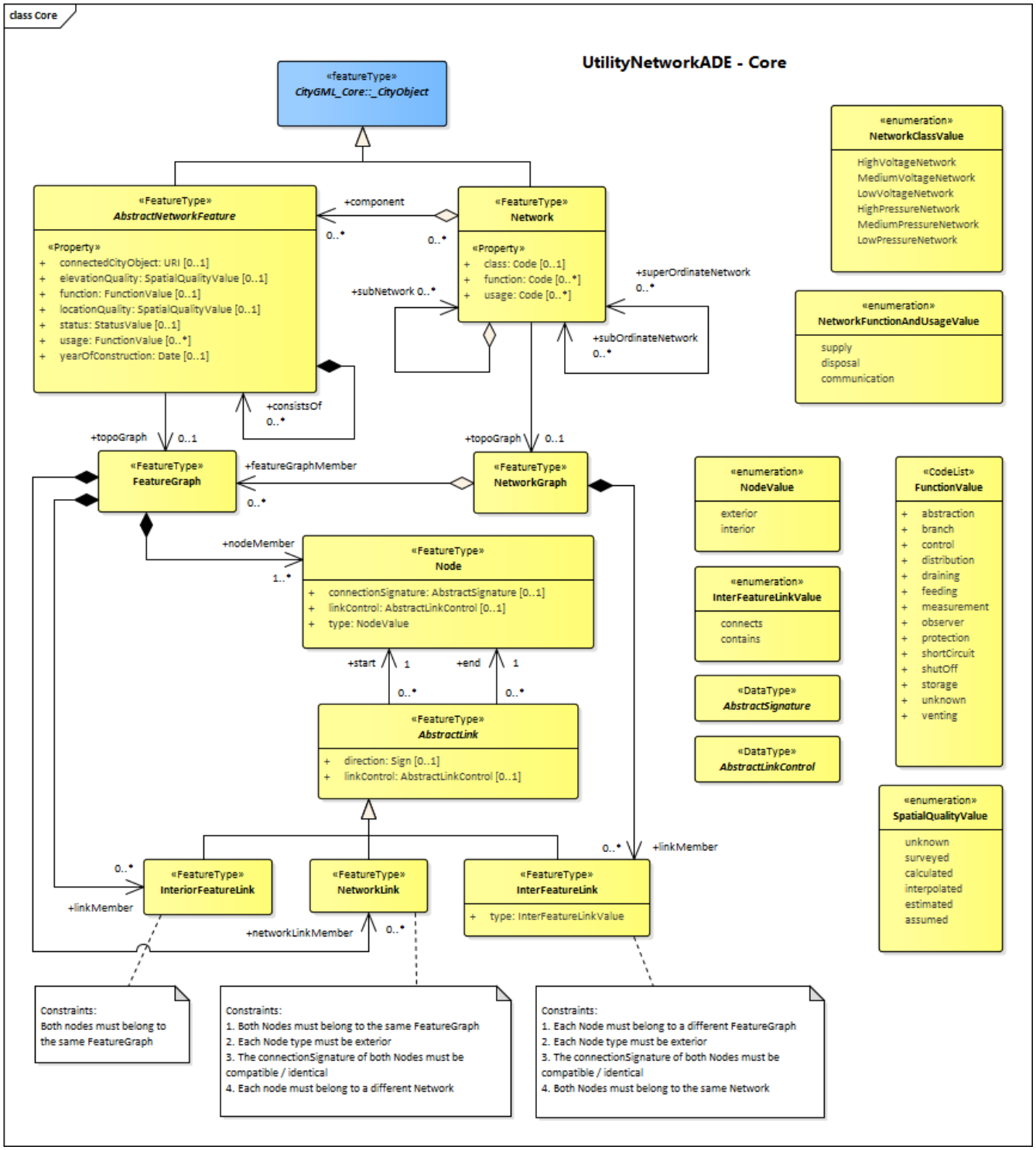

Figure 18. Utility Network ADE - Core

Table 3. Utility Infrastructure (Part 1)

NAS feature type	Mapping to Utility Network ADE
`UtilityNetwork`	Very close to `Network`. However, `facilityOperationalStatus` (NAS) is not related to the properties `class`, `function` and `usage` (all ADE). In the ADE the status is a property of the individual devices/elements in the network, not of the network itself. And the values of `utilityNetworkType` (NAS) are not known, so it is difficult to assess, if/how this property maps to any of the three ADE properties.

For the other feature types, there is no obvious match, except that they are probably all some

subtype of `AbstractNetworkFeature`. The ADE includes several subtypes, which are shown in the figure below. The mapping is analyzed in the table below the figure.

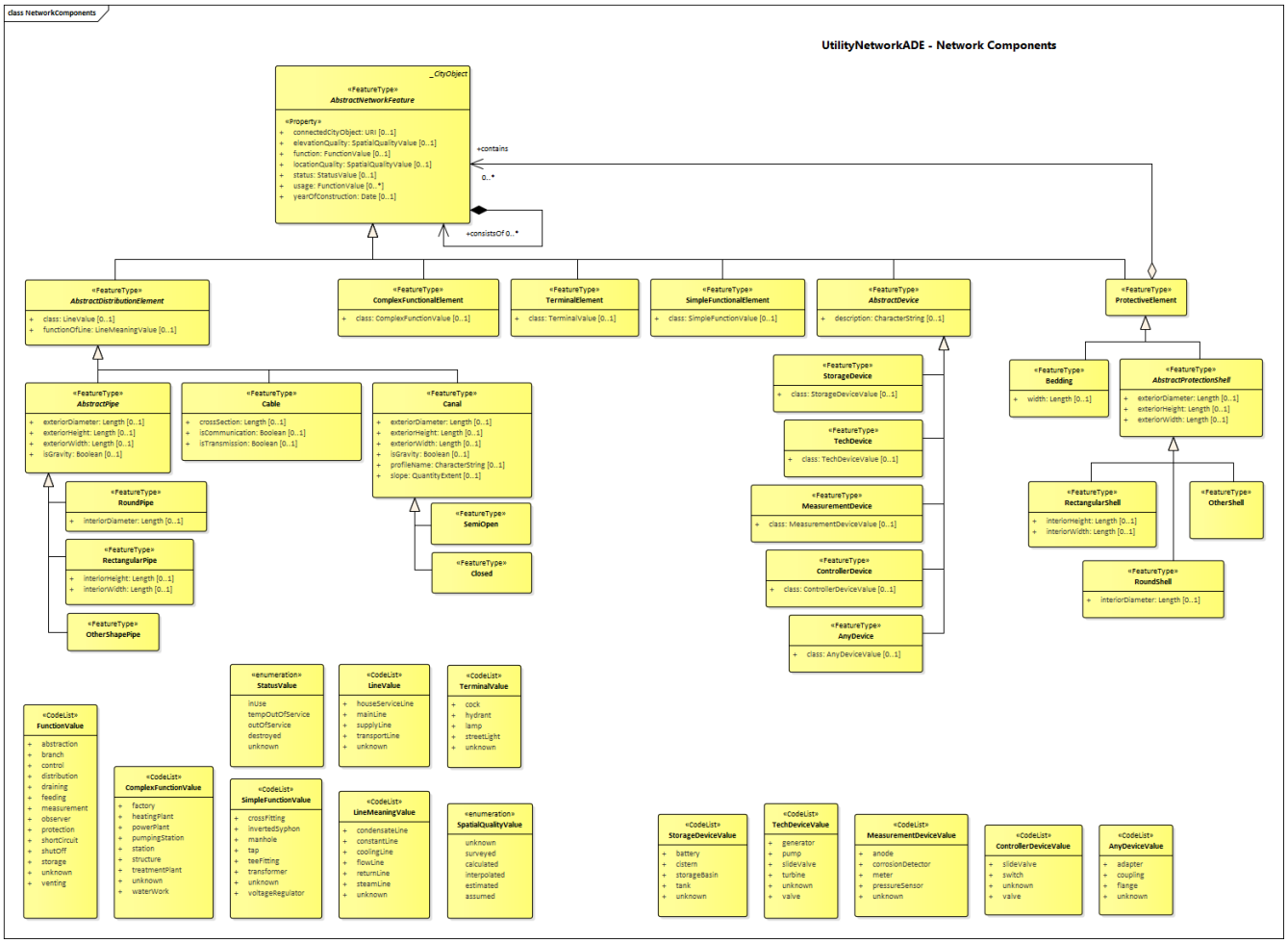

Figure 19. Utility Network ADE - Network Components

Table 4. Utility Infrastructure (Part 2)

NAS feature type	Mapping to Utility Network ADE
`BuriedUtility`	Any `AbstractNetworkFeature` where the property `relativeToTerrain` inherited from `_CityObject` (CityGML) has the value `entirelyBelowTerrain`. The NAS property `buriedUtilityType` provides additional information about the type of utility, but there is no obvious mapping of these features to the Utility Network ADE. The values [https://nsgreg.nga.mil/ir/view?i=100205&type=dlvb] are: [2: To access the NSG code list register, you have to install the US DoD root certificates on your system. Follow the instructions at http://militarycac.com/dodcerts.htm.] • District Heating or Cooling • Gas • Power • Sewage • Telecommunication • Water The attribute should be mapped to the CityGML property `usage`.
`SewageTreatmentPlant`	`ComplexFunctionalElement` with `class=treatmentPlant`.
`StormDrain`	`TerminalElement` with `class=unknown` - or `stormDrain` (as `TerminalValue` is a code list, i.e., it can be extended).

NAS feature type	Mapping to Utility Network ADE
`UtilityAccessPoint`	Unclear, map to `_CityObject` (CityGML). The NAS property `accessibleUtilityType` provides additional information about the type of utility, but again there is no obvious mapping of these features to the Utility Network ADE. The values [https://nsgreg.nga.mil/ir/view?i=100043&type=dlvb] are: [2: To access the NSG code list register, you have to install the US DoD root certificates on your system. Follow the instructions at http://militarycac.com/dodcerts.htm.] • Cable Television • Cooling Fluid Circulation • Digital Fibre-optic System • Electric Power Distribution • Heating Fluid Circulation • Natural Gas Distribution • Sewage • Storm Sewer • Street Light • Telegraph • Telephone • Traffic Light • Water Distribution The attribute would be mapped to the CityGML property `usage`.
`Waterwork`	`ComplexFunctionalElement` with `class=waterwork`.

9.1.3. Mapping the NAS profile to the CityGML application schema

This section documents the potential mapping of selected feature types from selected packages in the NAS profile to CityGML feature types.

Package: Aeronautical Ground Features

Aeronautical Ground Features

Areas of land and/or water as well as structures set aside for the take-off, landing and/or accommodation of both civilian and military aircraft, rockets, missiles and spacecraft.

The feature types, and their definitions, are:

Aerodrome

A defined area on land or water (including any buildings, installations and equipment) intended

to be used either wholly or in part for the arrival, departure and surface movement of aircraft.

Heliport

An aerodrome intended to be used for the arrival, landing, takeoff or departure of vertical takeoff and landing aircraft/helicopters.

LandAerodrome

An aerodrome on land intended to be used either wholly or in part for the arrival, departure and surface movement of aircraft.

`Aerodrome` is abstract and does not need to be mapped.

Table 5. General Aeronautical Ground Features

NAS feature type	Mapping to CityGML
`Heliport`	`TransportationComplex` with `usage`=(`aircraft`, `helicopter`), `function`=`heliport`
`LandAerodrome`	`TransportationComplex` with `usage`=`aircraft`, `function`=`aerodrome`

Package: Aerodrome Buildings and Structures

Aerodrome Buildings and Structures

Buildings and structures that support aerodrome operations and the protection, storage, maintenance and/or repair of aircraft.

The feature types, and their definitions, are:

AircraftHangar

A building for housing aircraft.

ControlTower

A structure that houses personnel and equipment used to control the flow of traffic within a specified range of an installation.

HardenedAircraftShelter

A hardened structure built above or partially above ground that encloses aircraft to provide protection from enemy attack.

All three feature types seem to be `Building` features in CityGML, where `class` has a value `traffic` and `function` reflects the name of the NAS feature type.

Table 6. Aerodrome Buildings and Structures

NAS feature type	Mapping to CityGML
`AircraftHangar`	`Building` with `class`=`traffic`, `function`=`aircraftHangar`
`ControlTower`	`Building` with `class`=`traffic`, `function`=`controlTower`
`HardenedAircraftShelter`	`Building` with `class`=`traffic`, `function`=`hardenedAircraftShelter`

Package: Aircraft Movement Surfaces

Aircraft Movement Surfaces

Defined areas on an aerodrome prepared for: the landing and take-off of aircraft; the orderly movement of aircraft to and from the runway, helipad, or seaplane run; and for aircraft to park while involved in a variety of non-flying functions.

The feature types, and their definitions, are:

AerodromeMoveArea

That part of an aerodrome to be used for the take-off, landing and taxiing of aircraft, consisting of the manoeuvring area and the apron(s).

Apron

A defined area, on a land aerodrome/heliport, intended to accommodate aircraft/helicopters for purposes of loading and unloading passengers, mail or cargo, and for fuelling, parking or maintenance.

Helipad

A designated area, usually with a prepared surface, used for the take-off, landing, or parking of helicopters.

Runway

A defined rectangular area on a land aerodrome prepared for the landing and take-off of aircraft.

Taxiway

A defined path at an aerodrome established for the taxiing of aircraft and intended to provide a ground movement link between one part of the aerodrome and another.

`AerodromeMoveArea` is abstract and does not need to be mapped.

Table 7. Aircraft Movement Surfaces

NAS feature type	Mapping to CityGML
`Apron`	`TrafficArea` with `usage`=(`aircraft`, `helicopter`), `function`=`apron`
`Helipad`	`TrafficArea` with `usage`=`helicopter`, `function`=`helipad`
`Runway`	`TrafficArea` with `usage`=`aircraft`, `function`=`runway`
`Taxiway`	`TrafficArea` with `usage`=`aircraft`, `function`=`taxiway`

Package: Area Safety Features

Area Safety Features

Structures and/or markings serving as safety measures used as a guidance to aircraft on the aerodrome surface or approach for landing, and equipment and areas on the ground used in case of emergency situations.

The single feature type, and its definitions, is:

Stopway

A defined rectangular area on the ground at the end of the take-off run available that has been prepared as a suitable area in which an aircraft can be stopped in the case of an abandoned take-off.

Table 8. Movement Area Safety Features

NAS feature type	Mapping to CityGML
`Stopway`	`TrafficArea` with `usage=aircraft`, `function=stopway`

Package: General Structures

General Structures

Buildings and their components, non-building structures and man-made barriers (for example: walls and fences).

The feature types, and their definitions, are:

Building

A free-standing self-supporting construction that is roofed, usually walled, and is intended for human occupancy (for example: a place of work or recreation) and/or habitation.

BuildingOverhang

A canopy or ledge attached to the front of a building and protruding beyond the perimeter wall.

BuildingSuperstructure

A supplemental portion of a building which rises from the roof but is not considered to be a portion of the roof.

EntranceExit

A location of entrance and/or exit.

Facility

An area that has been developed to perform a specific principal function, consisting of one or more vertical constructions (for example: structures or buildings), horizontal constructions (for example: pavements, roads, rail tracks, or bridges), and/or supporting utilities (for example: power lines, water supply, or sewerage), plus the underlying land.

FortifiedBuilding

A building that is specifically designed or reinforced to provide for defence from armed attack.

Installation

A grouping of facilities, located in the same vicinity, which support particular functions.

NonBuildingStructure

A free-standing self-supporting construction (for example: a large piece of equipment) designed to support human activities (for example: agriculture, manufacturing, or mining) but not intended for human occupancy and/or habitation (for example: a house, a bank, an office, or a stadium).

OverheadObstruction

An overhead obstruction (for example: an underpass, an overhead pipeline, and/or the overhang of a building) on a transportation route.

Shed

A small building, generally of light construction, that usually has one or more open sides.

Stair

A series of fixed steps leading from one level to another, especially such a series leading from one floor level to another inside a structure.

Wall

A solid man-made barrier of generally heavy material used as an enclosure, boundary, or for protection.

Table 9. General Structures

NAS feature type	Mapping to CityGML
`Building`	`Building`, `function`=value(s) of `featureFunction` (NAS), `roofType`=value of `roofShape`, `measuredHeight`=value of `heightAboveSurfaceLevel`. All other attributes do not match to a sufficient level (CityGML distinguishes between "storeys above/below ground" where the NAS just has the floor count).
`BuildingOverhang`	`BuildingInstallation` with `class=overhang`
`BuildingSuperstructure`	`BuildingInstallation` with `class=superstructure`
`EntranceExit`	`Door` (although the NAS semantics is broader and includes cave openings, but in an urban setting, these should be doors)
`Facility`	`_CityObject` (no match)
`FortifiedBuilding`	See `Building`, with `class=fortified`
`Installation`	`_CityObject` (no match)
`NonBuildingStructure`	In CityGML probably also a `Building`, `function`=value(s) of `featureFunction` (NAS), `measuredHeight`=value of `heightAboveSurfaceLevel`.

NAS feature type	Mapping to CityGML
OverheadObstruction	The NAS property overheadObstructionType provides additional information about the type. The values are defined in a code list [https://nsgreg.nga.mil/ir/view?i=100736&type=dlvb]. [2: To access the NSG code list register, you have to install the US DoD root certificates on your system. Follow the instructions at http://militarycac.com/dodcerts.htm.] These seem to map to a range of CityGML concepts, including but not limited to BuildingInstallation, Building and CityFurniture depending on the type. It is not fully clear (to us) how to best map this feature to CityGML. It is therefore mapped to the default (_CityObject); the type is mapped to the CityGML property class.
Shed	See Building, with class=shed
Stair	BuildingInstallation with class=stair
Wall	_CityObject (no match)

Package: Urban and Street Furniture

Urban and Street Furniture

Structures and/or equipment used for various purposes usually associated with urban areas and sometimes specifically associated with a road (street).

The feature types, and their definitions, are:

Bench

A long hard seat for two or more persons.

Billboard

A large outdoor board for advertisements.

DisplaySign

An upright panel or similar structure used to convey visual information.

Flagpole

A permanently affixed or emplaced staff or pole on which a flag is raised.

LightSupportStructure

A structure serving as a support for one or more lights.

PicnicTable

A table used for picnics.

Planter

A container for growing or displaying plants, usually decorative and permanent.

`DisplaySign` is abstract and does not need to be mapped.

Table 10. Urban and Street Furniture

NAS feature type	Mapping to CityGML
`Bench`	`CityFurniture` with `usage=pedestrian`, `function=bench`
`Billboard`	`CityFurniture` with `function=billboard`
`Flagpole`	`CityFurniture` with `function=flagpole`
`LightSupportStructure`	`CityFurniture` with `function=lightSupportStructure`
`PicnicTable`	`CityFurniture` with `usage=pedestrian`, `function=picnicTable`
`Planter`	`CityFurniture` with `function=planter`

Package: Ports and Harbours

Ports and Harbours

Information about ports (settlements with installations for handling waterborne shipping) or harbours (areas where the anchorage and shore are protected from the sea and storms by natural or man-made barriers).

The feature types, and their definitions, are:

ShorelineRamp

A ramp-like structure on a shoreline that is intended to facilitate the movement of vessels and/or materials (for example: logs) into or out of the water.

ShorelineConstruction

An artificial structure attached to land bordering a body of water and fixed in position.

Dock

An artificially enclosed body of water within which vessels may moor and which may have gates used to regulate the interior water level.

Harbour

A natural or artificial improved body of water providing protection for vessels and generally anchorage and docking facilities.

SmallCraftFacility

A place at which a service generally of interest to small craft or pleasure boats is available.

Shipyard

A large enclosed area adjoining the sea or a major river, including facilities in which ships are built or repaired.

`SmallCraftFacility` is a subtype of `Facility` and does not need to be mapped (`Facility` has been mapped above already).

Table 11. Ports and Harbours

NAS feature type	Mapping to CityGML
ShorelineRamp	_CityObject (no match)
ShorelineConstruction	_CityObject (no match)
Dock	_CityObject (no match)
Harbour	_CityObject (no match)
Shipyard	_CityObject (no match)

9.2. Derivation of an ADE from the NAS profile

9.2.1. Overview

Now that a mapping from (parts of) the NAS profile to CityGML and the Utility Network ADE has been defined, this mapping needs to be expressed in a ShapeChange configuration so that the XML Schema can be derived for the ADE, corresponding to the NAS profile.

NOTE The ShapeChange configuration developed in Testbed-13 is included in Annex A.

For simplicity all feature types not covered by the mapping in this chapter are mapped to _CityObject.

A CityModel feature collection can only consist of feature types. Object types can be included by encoding an object inline once, and referencing it wherever else it occurs as value. Object types do not become subtypes of _CityObject; i.e., no inheritance relationship will be added.

9.2.2. Transformation: use generic application property elements?

Property elements in the ADE could be defined using two approaches:

1. As standard GML properties, locally defined in the context of the feature type. This requires that each NAS feature type becomes a subtype of the CityGML or Utility Network ADE feature type that the NAS feature type has been mapped to. The new NAS feature types and their properties are created in the ADE namespace.
2. As new global property elements in the ADE namespace, which are in the substitution group of the corresponding CityGML _GenericApplicationPropertyOf<Featuretypename> property element.

The first option seems appropriate for an ADE for NAS profiles. Reasons:

- The second option will require the definition of NAS feature types in the ADE, too, in all cases where a NAS feature type has been mapped to an abstract CityGML type, typically _CityObject, as a feature type in the ADE is needed that can be instantiated.
- In the second option where global property elements are used, there is a risk of name clashes for the global property elements, if different NAS feature types that are mapped to the same CityGML feature type have properties with the same name (unless the NAS feature type name is included somehow in the property names).
- ADEs, including the Utility Network ADE, typically use the global application property elements

for cases where new properties are meant to be available for existing CityGML features, for example, all `_CityObject` instances. This does not apply in this use case where it is desired to enable that existing NAS data can be transformed to a CityGML representation.

The typical ADE patterns can also be seen in the generic test ADE [https://github.com/yaozhihang/module-test-ade]. Most feature types, e.g. `IndustrialBuilding`, follow the first approach described above. This test ADE also has an example of the second approach where `ownerName` is implemented as a global application property element, which is injected into the existing AbstractBuilding type using a substitution group. The test ADE also includes a feature type (`LightingFacilities`) that is directly based on `gml:_Feature` without reusing any schema components from CityGML.

9.2.3. Transformation: schema complexity

CityGML has no limitations on the schema complexity of the feature types and their properties in the ADE. In other words choosing both of the following options - or anything in-between - is possible:

- Do not flatten the schema of the NAS profile, just add the inheritance relationships to the corresponding CityGML feature type and make the root elements of the NAS class hierarchy "mix-in" classes. Object types are encoded as defined by the GML encoding rule.
- Use the flattened GML-SF0 implementation schema as a starting point - with the exception of the transformations for
  - Multiplicity - as CityGML frequently uses maximum multiplicities larger than 1 and
  - Geometry specific feature types - as splitting the feature types by geometry type would not really be consistent with the use of geometries in CityGML.

  Then add the inheritance relationships to the corresponding CityGML feature type for each of the feature types in the implementation schema.

In general, the choice should depend on the capabilities of software libraries intended to use the data.

One objective of this testbed activity was to explore, if/how clients could make use of an ADE for a NAS profile. The discussion in the testbed came to the following conclusions:

- In general, the assumption is that an ADE is used if more than just simple key-value-pairs are needed (in that case the generic object and properties could be used).

NOTE	This does not involve any schema definitions, so data cannot be validated using XML schema validation.

- CityGML itself makes use of many modelling elements both in the CityGML UML and XML schema that go beyond GML-SF0 (abstract classes, class hierarchies/inheritance, associations including reflexive associations, association classes, multiple geometry representations, sharing of geometries, etc.). That is, the CityGML core modules already require that software has support for schemas that significantly go beyond the GML-SF0 profile and the community is already used to this kind of complexity. Any ADE-aware software is expected to be able to

handle complex schemas.

- A result is that CityGML-aware software will typically require customization (software development) to support an ADE.

9.2.4. Conclusion

A CityGML ADE for a profile of the NAS should reflect the conceptual model as exactly as possible and only use transformations to simplified structures in the implementation schema where required by CityGML.

For example, some limitations exist regarding the geometry types used in CityGML. Free form surfaces are not allowed, only polyhedrons and triangle meshes. This, however, is not a restriction for NAS data, which does not include these geometry types.

A consequence is that an ADE is an option for a profile of the NAS that is standardized by the community or as part of the NSG, but less for a "mission-specific" profile of the NAS that is created for a specific dataset or data capturing effort.

If the CityGML standard included a conformance class that clarifies which modelling capabilities may be used in an ADE so that CityGML-aware software supporting that ADE conformance class could be expected to handle it without special code for the ADE (similar to the Simple Feature profiles for GML), this would allow software to generate ADEs for "mission-specific" NAS profiles, too.

NOTE	This Wiki [http://en.wiki.quality.sig3d.org/index.php/Modeling] includes some modelling guidelines, but mainly related to the use of geometries and LoD.

9.3. Configuring ShapeChange

9.3.1. Transforming a NAS profile into an implementation model suitable for a CityGML ADE

A new ShapeChange transformation performs the mapping of NAS feature types to CityGML types.

Feature types for which a mapping to a CityGML type is defined (either via tagged value `cityGmlTargetType` or via map entries of the transformer configuration - both options use QNames to identify a CityGML type) receive that type as new supertype. All other feature types receive the CityGML type `_CityObject` as new supertype. Any previously existing supertype is transformed to a mixin type.

NOTE	All CityGML types that are used as mapping targets must be available in the model. The `_CityObject` type is determined by looking for a type with according name that does NOT have stereotype <<ADEElement>>.

NOTE	Object types are encoded as individual schema elements, like in the complex NAS GML application schema, and can be used as such.

In addition to this mapping, the target namespace of the NAS profile application schema is updated to reflect that the transformed schema is an ADE. This can be as simple as adding a suffix like "/ade" to the original target namespace.

9.3.2. Generating a GML application schema from the implementation model

The encoding rule to create a CityGML ADE from the transformed NAS profile is quite similar to the encoding rule used to derive the GML application schema for the NAS. The differences are:

- The encoding rule extends the `citygml-ade` encoding rule, instead of `iso19136_2007`. `citygml-ade` extends `iso19136_2007` and adds CityGML specific conversion rules.
  - The rule definitions are available online [http://shapechange.net/resources/config/StandardRules.xml] (in form of a ShapeChange configuration file).
  - The conversion rules are documented on http://shapechange.net/targets/xsd/
  - The `citygml-ade` encoding rule existed before Testbed-13.
- *rule-xsd-pkg-schematron* has been removed since Schematron does not appear to play a role for the CityGML ADE.
- GML 3.1.1 is used as base instead of GML 3.2.1, since CityGML is based on GML 3.1.1.

The encoding rule results in an ADE where NAS feature types are subtypes of CityGML types. The complex structures of the NAS are kept.

9.3.3. Adaptation to other NAS Profiles

This section discusses how to adapt the workflow for creating a CityGML ADE for another NAS profile. The configuration of the workflow used in Testbed-13 is shown in Annex A.

The first step of the workflow is the mapping of NAS feature types to CityGML feature types. The transformation in the ShapeChange configuration that executes this mapping has ID 'TRF_CREATE_ADE'. A domain expert needs to define the mapping. Then, a `<ProcessMapEntry>` must be configured for each feature type mapping. The XML attribute 'rule' should be kept as in the examples from Testbed-13 ('rule-trf-CityGML-createADE'). The 'type' and 'targetType' identify the source and target of the mapping, where 'type' is the name of the NAS feature type, and 'targetType' is the QName of the CityGML feature type. The namespace prefixes used in these QNames must be covered by `<XmlNamespace>` elements of the XML Schema target configuration. That is especially of interest if the mapping was made to feature types from another CityGML ADE.

WARNING	The UML model must contain all CityGML schemas that contain feature types that are mapping targets for features types of the NAS profile.

The user may also want to define a target namespace that is specific to the given NAS profile. As shown in the ShapeChange configuration from Testbed-13, this can be achieved by modifying the 'targetNamespace' tagged value of the NAS profile schema (see the according `<TaggedValue>` element in the transformation with ID 'IDENTITY').

The second step of the workflow is deriving the actual CityGML from the implementation model

produced by the transformation. The XML Schema target configuration can mostly be kept as-is:

- The target parameters do not need to be changed.
- The encoding rule can be kept as-is.
- It should not be necessary to extend the XSD map entries. However, if the NAS profile made use of property types that are not covered by the map entries, new map entries would need to be added. Further details can be found in the ShapeChange documentation [http://shapechange.net/targets/xsd/#XSD_Map_Entries].
- The list of XML namespaces in the ShapeChange configuration from Testbed-13 is already quite extensive. Additional map entries may require additional namespaces. Also, as mentioned before, they may be required by the mapping to CityGML feature types. Further details on the declaration of XML namespaces can be found in the ShapeChange documentation [http://shapechange.net/targets/xsd/#Namespace_Identifiers].

Appendix A: ShapeChange Configurations

This annex contains the ShapeChange configurations that were used to generate the different implementation schemas in Testbed 13.

NOTE	In the HTML representation of this report, you can copy the configurations by starting the selection at the XML declaration (<?xml...) - not at the line number.

ShapeChange Configuration to generate SCXML for the NAS Profile

The following listing contains the ShapeChange configuration that was used to export the model that contains the NAS profile and CityGML schemas.

```
<?xml version="1.0" encoding="UTF-8"?>
<ShapeChangeConfiguration
 xmlns="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1"
 xmlns:sc="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1"
 xmlns:xi="http://www.w3.org/2001/XInclude"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.interactive-
instruments.de/ShapeChange/Configuration/1.1
http://shapechange.net/resources/schema/ShapeChangeConfiguration.xsd">
 <input id="INPUT">
  <parameter name="inputModelType" value="EA7"/>
  <parameter name="inputFile" value="models/TB13AS_v1.2_FULL_with_CityGML_ADE.eap"/>
  <parameter name="publicOnly" value="true"/>
  <parameter name="checkingConstraints" value="enabled"/>
  <parameter name="sortedOutput" value="true"/>
  <parameter name="sortedSchemaOutput" value="true"/>
  <parameter name="appSchemaName" value="OGC Testbed 13 Application Schema"/>
  <xi:include href="config/StandardAliases.xml"/>
  <descriptorSources>
   <DescriptorSource descriptor="documentation" source="tag" tag="documentation"/>
   <DescriptorSource descriptor="alias" source="tag" tag="name"/>
   <DescriptorSource descriptor="primaryCode" source="tag" tag="primaryCode"/>
   <DescriptorSource descriptor="definition" source="tag" tag="definition"/>
   <DescriptorSource descriptor="description" source="tag" tag="description"/>
   <DescriptorSource descriptor="example" source="none"/>
   <DescriptorSource descriptor="legalBasis" source="none"/>
   <DescriptorSource descriptor="dataCaptureStatement" source="none"/>
   <DescriptorSource descriptor="language" source="none"/>
  </descriptorSources>
  <tagAliases>
   <TagAlias wellknown="length" alias="characterLength"/>
  </tagAliases>
 </input>
```

```
 <log>
  <parameter name="reportLevel" value="INFO"/>
  <parameter name="logFile" value="results/export/log_export.xml"/>
 </log>

 <transformers>
  <Transformer id="IDENTITY" input="INPUT"

class="de.interactive_instruments.ShapeChange.Transformation.Identity.IdentityTransfor
m"/>
 </transformers>

 <targets>
  <TargetXmlSchema
class="de.interactive_instruments.ShapeChange.Target.XmlSchema.XmlSchema"
   mode="enabled" inputs="IDENTITY">
   <targetParameter name="outputDirectory" value="results/export/xsd"/>
   <targetParameter name="sortedOutput" value="true"/>
   <targetParameter name="defaultEncodingRule" value="gsip"/>
   <targetParameter name="includeDocumentation" value="true"/>
   <targetParameter name="documentationTemplate" value="[[documentation]]"/>
   <rules>
    <EncodingRule name="gsip" extends="citygml-ade">
     <rule name="req-xsd-cls-codelist-no-supertypes"/>
     <rule name="rule-xsd-cls-union-asCharacterString"/>
     <rule name="rule-xsd-cls-union-asGroup"/>
     <rule name="rule-xsd-cls-enum-supertypes"/>
     <rule name="rule-xsd-cls-enum-subtypes"/>
     <rule name="rule-xsd-cls-basictype"/>
     <rule name="rule-xsd-cls-union-direct"/>
     <rule name="rule-xsd-cls-codelist-constraints"/>
     <rule name="rule-xsd-cls-mixin-classes-as-group"/>
     <rule name="rule-xsd-cls-mixin-classes"/>
     <rule name="rule-xsd-cls-mixin-classes-non-mixin-supertypes"/>
     <rule name="req-xsd-cls-mixin-supertypes-overrule"/>
     <rule name="rule-xsd-prop-exclude-derived"/>
     <rule name="rule-xsd-prop-length-size-pattern"/>
     <rule name="rule-xsd-prop-xsdAsAttribute"/>
     <rule name="rule-xsd-prop-nillable"/>
     <rule name="rule-xsd-prop-nilReasonAllowed"/>
     <rule name="rule-xsd-prop-initialValue"/>
     <rule name="rule-xsd-prop-att-map-entry"/>
     <rule name="rule-xsd-all-tagged-values"/>
     <rule name="rule-xsd-rel-association-classes"/>
     <rule name="rule-xsd-cls-codelist-constraints-codeAbsenceInModelAllowed"/>
    </EncodingRule>
   </rules>
   <xi:include href="config/StandardRules.xml"/>
   <xi:include href="config/StandardNamespaces_local.xml"/>
   <xi:include href="config/StandardMapEntries_iso19136_2007_GSIP.xml"/>
```

```
   <xi:include href="config/StandardMapEntries_sweCommon.xml"/>
   <xi:include href="config/StandardMapEntries_gsip.xml"/>
   <xi:include href="config/StandardMapEntries_iso19107.xml"/>
   <xi:include href="config/StandardMapEntries_iso19108_GSIP.xml"/>
   <xi:include href="config/StandardMapEntries_iso19111.xml"/>
   <xi:include href="config/StandardMapEntries_iso19112.xml"/>
   <xi:include href="config/StandardMapEntries_iso19115-1.xml"/>
   <xi:include href="config/StandardMapEntries_iso19123.xml"/>
   <xi:include href="config/StandardMapEntries_iso19156.xml"/>
   <xi:include href="config/StandardMapEntries_iso19157.xml"/>
   <xi:include href="config/StandardMapEntries_gmlcov.xml"/>
   <xsdMapEntries>
    <XsdMapEntry type="URN" xmlPropertyType="anyURI" xmlType="anyURI"
xmlTypeContent="simple"
     xmlTypeType="simple" xsdEncodingRules="*"/>
    <XsdMapEntry type="URI" xmlPropertyType="anyURI" xmlType="anyURI"
xmlTypeContent="simple"
     xmlTypeType="simple" xsdEncodingRules="*"/>
    <XsdMapEntry type="URL" xmlPropertyType="anyURI" xmlType="anyURI"
xmlTypeContent="simple"
     xmlTypeType="simple" xsdEncodingRules="*"/>
    <XsdMapEntry type="CharacterString" xmlElement="gco:CharacterString"
     xmlPropertyType="gco:CharacterString_PropertyType"
xmlType="gco:CharacterString_Type"
     xsdEncodingRules="*"/>
    <XsdMapEntry type="SecurityAttributesGroupType" xmlPropertyType="_P_"
xmlTypeNilReason="false"
     xmlAttributeGroup="icism:SecurityAttributesOptionGroup" xsdEncodingRules="*"/>
    <XsdMapEntry type="ISM_Notice" xmlPropertyType="icism:NoticeType"
xmlType="icism:NoticeType"
     xmlTypeContent="complex" xmlTypeNilReason="false" xsdEncodingRules="*"/>
    <XsdMapEntry type="NTKAccess" xmlPropertyType="ntk:RequiresType"
xmlType="ntk:RequiresType"
     xmlTypeContent="complex" xmlTypeNilReason="false" xsdEncodingRules="*"/>
    <XsdMapEntry type="RevisionRecall" xmlPropertyType="rr:RevisionRecallType"
     xmlType="rr:RevisionRecallType" xmlTypeContent="complex" xmlTypeNilReason="false"
     xsdEncodingRules="*"/>
   </xsdMapEntries>
   <xmlNamespaces>
    <XmlNamespace nsabr="icism" ns="urn:us:gov:ic:ism"
location="../../../ic/ism/V13/IC-ISM.xsd"/>
    <XmlNamespace nsabr="ntk" ns="urn:us:gov:ic:ntk" location="../../../ic/ntk/V10/IC-
NTK.xsd"/>
    <XmlNamespace nsabr="rr" ns="urn:us:gov:ic:revrecall"
     location="../../../ic/RevRecall/V1/RevRecall_XML.xsd"/>
   </xmlNamespaces>
  </TargetXmlSchema>
  <Target
class="de.interactive_instruments.ShapeChange.Target.ModelExport.ModelExport"
   mode="enabled" inputs="IDENTITY">
   <targetParameter name="outputDirectory" value="results/export"/>
```

```
    <targetParameter name="outputFilename" value="TB13AS_v1.2_FULL_with_CityGML_ADE"/>
    <targetParameter name="sortedOutput" value="true"/>
    <targetParameter name="zipOutput" value="true"/>
   </Target>
  </targets>
</ShapeChangeConfiguration>
```

ShapeChange Configuration to generate GML-SF0 Schema and CDB dictionaries for the NAS Profile

The following listing contains the ShapeChange configuration that was used to create the GML-SF0 as well as CDB dictionaries in Testbed-13.

```
<?xml version="1.0" encoding="UTF-8"?>
<ShapeChangeConfiguration
 xmlns="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1"
 xmlns:sc="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1"
 xmlns:xi="http://www.w3.org/2001/XInclude"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.interactive-
instruments.de/ShapeChange/Configuration/1.1
http://shapechange.net/resources/schema/ShapeChangeConfiguration.xsd">
 <input id="INPUT">
  <parameter name="inputModelType" value="SCXML"/>
  <parameter name="inputFile" value="models/TB13AS_minimalist_profile.zip"/>
  <parameter name="ignoreEncodingRuleTaggedValues" value="true"/>
  <parameter name="publicOnly" value="true"/>
  <parameter name="checkingConstraints" value="enabled"/>
  <parameter name="sortedOutput" value="true"/>
  <parameter name="sortedSchemaOutput" value="true"/>
  <parameter name="appSchemaNameRegex" value="OGC Testbed 13 Application
Schema|DoD/IC"/>
  <xi:include href="http://shapechange.net/resources/config/StandardAliases.xml"/>
  <descriptorSources>
   <DescriptorSource descriptor="documentation" source="tag" tag="documentation"/>
   <DescriptorSource descriptor="alias" source="tag" tag="name"/>
   <DescriptorSource descriptor="primaryCode" source="tag" tag="primaryCode"/>
   <DescriptorSource descriptor="definition" source="tag" tag="definition"/>
   <DescriptorSource descriptor="description" source="tag" tag="description"/>
   <DescriptorSource descriptor="example" source="none"/>
   <DescriptorSource descriptor="legalBasis" source="none"/>
   <DescriptorSource descriptor="dataCaptureStatement" source="none"/>
   <DescriptorSource descriptor="language" source="none"/>
  </descriptorSources>
 </input>
 <log>
  <parameter name="reportLevel" value="INFO"/>
  <parameter name="logFile" value="results/log_gmlsf0_and_cdb.xml"/>
 </log>
 <transformers>
```

```
  <Transformer id="IDENTITY" input="INPUT"

class="de.interactive_instruments.ShapeChange.Transformation.Identity.IdentityTransfor
m">
   <taggedValues>
    <TaggedValue name="inlineOrByReference" value="inline" modelElementType="Property"
     applicationSchemaName="OGC Testbed 13 Application Schema"/>
    <TaggedValue name="inlineOrByReference" value="inline" modelElementType="Property"
     applicationSchemaName="DoD/IC"/>
    <TaggedValue name="maxOccurs" value="1" modelElementType="Property"
     modelElementName="resourceConstraints"
     applicationSchemaName="OGC Testbed 13 Application Schema"/>
    <TaggedValue applicationSchemaName="OGC Testbed 13 Application Schema"
modelElementName="place"
     name="maxOccurs" value="1"/>
   </taggedValues>
  </Transformer>
  <Transformer

class="de.interactive_instruments.ShapeChange.Transformation.Constraints.ConstraintCon
verter"
   id="TRF_GEOMETRY_RESTRICTION_TO_GEOMETRY_TAGGEDVALUE" input="IDENTITY"
mode="enabled">
   <parameters>
    <ProcessParameter name="geometryRepresentationTypes"
     value="PointPositionInfo = P; CurvePositionInfo = C; SurfacePositionInfo = S"/>
    <ProcessParameter name="geometryRepresentationConstraintRegex"
     value=".*Place Representations Disallowed.*"/>
    <ProcessParameter name="geometryRepresentationValueTypeRegex" value="PlaceInfo"/>
   </parameters>
   <rules>
    <ProcessRuleSet name="trf">
     <rule name="rule-trf-cls-constraints-geometryRestrictionToGeometryTV-exclusion"/>
     <rule
      name="rule-trf-cls-constraints-geometryRestrictionToGeometryTV-
typesWithoutRestriction-byValueTypeMatch"
     />
    </ProcessRuleSet>
   </rules>
  </Transformer>
  <Transformer
class="de.interactive_instruments.ShapeChange.Transformation.Flattening.Flattener"
   id="TRF_FLATTEN_CONSTRAINTS"
input="TRF_GEOMETRY_RESTRICTION_TO_GEOMETRY_TAGGEDVALUE"
   mode="enabled">
   <rules>
    <ProcessRuleSet name="trf">
     <rule name="rule-trf-all-flatten-constraints"/>
    </ProcessRuleSet>
   </rules>
  </Transformer>
```

```
  <Transformer
class="de.interactive_instruments.ShapeChange.Transformation.Flattening.Flattener"
   id="TRF_FLATTEN_GEOMETRY_TYPE_INHERITANCE" input="TRF_FLATTEN_CONSTRAINTS"
mode="enabled">
   <rules>
    <ProcessRuleSet name="trf">
     <rule name="rule-trf-cls-flatten-geometryTypeInheritance"/>
    </ProcessRuleSet>
   </rules>
  </Transformer>
  <Transformer
class="de.interactive_instruments.ShapeChange.Transformation.Flattening.Flattener"
   id="TRF_REMOVE_TYPE" input="TRF_FLATTEN_GEOMETRY_TYPE_INHERITANCE" mode="enabled">
   <parameters>
    <ProcessParameter name="removeType"

value="Dataset,EntityCollection,FeatureEntityCollection,GeoNameCollection,PhysicalEnti
tyCollection,PropertyMetadata,DataLineage,DataProcessStep,DataSource,DataQuality,Physi
calObjectMetadata,LocationInfo,PhysicalAddressInfo"
    />
   </parameters>
   <rules>
    <ProcessRuleSet name="flattener">
     <rule name="rule-trf-all-removeType"/>
    </ProcessRuleSet>
   </rules>
  </Transformer>
  <Transformer
class="de.interactive_instruments.ShapeChange.Transformation.Flattening.Flattener"
   id="TRF_REMOVE_MD_INHERITANCE" input="TRF_REMOVE_TYPE" mode="enabled">
   <parameters>
    <ProcessParameter name="removeInheritanceIncludeRegex" value=
"^(MD|CI|EX|LI)_.*$"/>
   </parameters>
   <rules>
    <ProcessRuleSet name="trf">
     <rule name="rule-trf-cls-remove-inheritance-relationship"/>
    </ProcessRuleSet>
   </rules>
  </Transformer>
  <Transformer id="TRF_REMOVE_VALUE_OR_REASON_DESCRIPTORS"
input="TRF_REMOVE_MD_INHERITANCE"

class="de.interactive_instruments.ShapeChange.Transformation.Descriptors.DescriptorTra
nsformer">
   <advancedProcessConfigurations>
    <DescriptorValue descriptorName="alias" modelElementType="Property"
     modelElementName="value(s)?OrReason"/>
    <DescriptorValue descriptorName="definition" modelElementType="Property"
     modelElementName="value(s)?OrReason"/>
    <DescriptorValue descriptorName="description" modelElementType="Property"
```

```
      modelElementName="value(s)?OrReason"/>
    </advancedProcessConfigurations>
    <rules>
     <ProcessRuleSet name="trf">
      <rule name="rule-trf-all-updateDescriptors"/>
     </ProcessRuleSet>
    </rules>
   </Transformer>
   <Transformer
class="de.interactive_instruments.ShapeChange.Transformation.Flattening.Flattener"
    id="TRF_FLATTEN_ONINAS" input="TRF_REMOVE_VALUE_OR_REASON_DESCRIPTORS"
mode="enabled">
    <rules>
     <ProcessRuleSet name="trf">
      <rule name="rule-trf-prop-flatten-ONINAs"/>
      <rule name="rule-trf-prop-flatten-ONINAs-onlyRemoveReasons"/>
     </ProcessRuleSet>
    </rules>
   </Transformer>
   <Transformer

class="de.interactive_instruments.ShapeChange.Transformation.TaggedValues.TaggedValueT
ransformer"
    id="TRF_TV_COPY_FROM_VALUE_TYPE" input="TRF_FLATTEN_ONINAS" mode="enabled">
    <parameters>
     <ProcessParameter name="taggedValuesToCopy"
value="length,pattern,rangeMinimum,rangeMaximum"/>
    </parameters>
    <rules>
     <ProcessRuleSet name="trf">
      <rule name="rule-trf-taggedValue-copyFromValueType"/>
     </ProcessRuleSet>
    </rules>
   </Transformer>
   <Transformer
class="de.interactive_instruments.ShapeChange.Transformation.Flattening.Flattener"
    id="TRF_MAP_TO_SIMPLE_BASE_TYPE" input="TRF_TV_COPY_FROM_VALUE_TYPE" mode=
"enabled">
    <parameters>
     <ProcessParameter name="simpleBaseTypes"
value="CharacterString,Integer,Measure,Real"/>
    </parameters>
    <rules>
     <ProcessRuleSet name="trf">
      <rule name="rule-trf-all-flatten-type-mapToSimpleBaseType"/>
     </ProcessRuleSet>
    </rules>
   </Transformer>
   <Transformer
class="de.interactive_instruments.ShapeChange.Transformation.Flattening.Flattener"
    id="TRF_REMOVE_OBJECT_TO_FEATURE_TYPE_NAVIGABILITY"
```

```
input="TRF_MAP_TO_SIMPLE_BASE_TYPE"
   mode="enabled">
   <parameters>
    <ProcessParameter name="removeObjectToFeatureNavRegex" value=".*"/>
   </parameters>
   <rules>
    <ProcessRuleSet name="flattener">
     <rule name="rule-trf-prop-removeObjectToFeatureTypeNavigability"/>
    </ProcessRuleSet>
   </rules>
  </Transformer>
  <Transformer
class="de.interactive_instruments.ShapeChange.Transformation.Flattening.Flattener"
   id="TRF_REMOVE_OBJECT_TO_FEATURE_TYPE_NAVIGABILITY_2"
   input="TRF_REMOVE_OBJECT_TO_FEATURE_TYPE_NAVIGABILITY" mode="enabled">
   <parameters>
    <ProcessParameter name="removeObjectToFeatureNavRegex"

value="^(Note|LegalConstraints|ResourceConstraints|LivingQuartersAmenity|RRR)$|^(?!.*?
Position).*Info$"/>
    <ProcessParameter name="includeObjectToObjectNavigability" value="true"/>
   </parameters>
   <rules>
    <ProcessRuleSet name="flattener">
     <rule name="rule-trf-prop-removeObjectToFeatureTypeNavigability"/>
    </ProcessRuleSet>
   </rules>
  </Transformer>
  <Transformer
class="de.interactive_instruments.ShapeChange.Transformation.Flattening.Flattener"
   id="TRF_REMOVE_NAVIGABILITY_BASED_ON_ISFLATTARGET"
   input="TRF_REMOVE_OBJECT_TO_FEATURE_TYPE_NAVIGABILITY_2" mode="enabled">
   <rules>
    <ProcessRuleSet name="flattener">
     <rule name="rule-trf-prop-removeNavigabilityBasedOnIsFlatTarget"/>
    </ProcessRuleSet>
   </rules>
  </Transformer>
  <Transformer id="TRF_ASSOCIATION_CLASS_MAPPER"
   input="TRF_REMOVE_NAVIGABILITY_BASED_ON_ISFLATTARGET"

class="de.interactive_instruments.ShapeChange.Transformation.Flattening.AssociationCla
ssMapper"
   mode="enabled"/>
  <Transformer

class="de.interactive_instruments.ShapeChange.Transformation.TypeConversion.TypeConver
ter"
   id="TRF_TO_FEATURE_TYPE" input="TRF_ASSOCIATION_CLASS_MAPPER" mode="enabled">
   <parameters>
    <ProcessParameter name="toFeatureTypeNameRegex" value="^.*Constraints$"/>
```

```
    </parameters>
    <rules>
     <ProcessRuleSet name="trf">
      <rule name="rule-trf-toFeatureType"/>
     </ProcessRuleSet>
    </rules>
   </Transformer>
   <Transformer

class="de.interactive_instruments.ShapeChange.Transformation.TypeConversion.TypeConver
ter"
    input="TRF_TO_FEATURE_TYPE" id="TRF_DISSOLVE_ASSOCIATIONS" mode="enabled">
    <rules>
     <ProcessRuleSet name="typeConversion">
      <rule name="rule-trf-dissolveAssociations"/>
      <rule name="rule-trf-dissolveAssociations-keepType"/>
     </ProcessRuleSet>
    </rules>
   </Transformer>
   <Transformer
class="de.interactive_instruments.ShapeChange.Transformation.Flattening.Flattener"
    id="TRF_FLATTEN_INHERITANCE" input="TRF_DISSOLVE_ASSOCIATIONS" mode="enabled">
    <rules>
     <ProcessRuleSet name="trf">
      <rule name="rule-trf-cls-flatten-inheritance"/>
     </ProcessRuleSet>
    </rules>
   </Transformer>
   <Transformer
class="de.interactive_instruments.ShapeChange.Transformation.Flattening.Flattener"
    id="TRF_FLATTEN_MULTIPLICITY" input="TRF_FLATTEN_INHERITANCE" mode="enabled">
    <parameters>
     <ProcessParameter name="maxOccurs" value="2"/>
     <ProcessParameter name="descriptorModification_propertyIndexNumberSeparator"
value="alias{ - }"
     />
    </parameters>
    <rules>
     <ProcessRuleSet name="trf">
      <rule name="rule-trf-prop-flatten-multiplicity"/>
     </ProcessRuleSet>
    </rules>
   </Transformer>
   <Transformer
class="de.interactive_instruments.ShapeChange.Transformation.Flattening.Flattener"
    id="TRF_FLATTEN_TYPES" input="TRF_FLATTEN_MULTIPLICITY" mode="enabled">
    <parameters>
     <ProcessParameter name="descriptorModification_nonUnionSeparator"
      value="documentation{ : }, alias{ : }, definition{ : }, description{ : },
primaryCode{ : }"/>
     <ProcessParameter name="descriptorModification_unionSeparator"
```

```
     value="documentation{ : }, alias{ : }, definition{ : }, description{ : },
primaryCode{ : }"/>
   </parameters>
   <rules>
    <ProcessRuleSet name="trf">
     <rule name="rule-trf-prop-flatten-types"/>
     <rule name="rule-trf-prop-flatten-types-
ignoreUnionsRepresentingFeatureTypeSets"/>
     <rule name="rule-trf-prop-flatten-types-removeMappedTypes"/>
     <rule
      name="rule-trf-prop-flatten-types-
ignoreSelfReferenceByPropertyWithAssociationClassOrigin"/>
    </ProcessRuleSet>
   </rules>
   <mapEntries>
    <ProcessMapEntry rule="rule-trf-prop-flatten-types" type="MD_Identifier"
     targetType="CharacterString"/>
   </mapEntries>
  </Transformer>
  <Transformer
class="de.interactive_instruments.ShapeChange.Transformation.Flattening.Flattener"
   id="TRF_HOMOGENEOUS_GEOMETRIES" input="TRF_FLATTEN_TYPES" mode="enabled">
   <parameters>
    <ProcessParameter name="separatorForGeometryTypeSuffix" value="_"/>
    <ProcessParameter name="descriptorModification_geometryTypeSuffixSeparator"
value="alias{ : }"/>
    <ProcessParameter name="descriptorModification_geometryTypeAlias"
     value="alias{P=Point,C=Curve,S=Surface}"/>
   </parameters>
   <rules>
    <ProcessRuleSet name="trf">
     <rule name="rule-trf-prop-flatten-homogeneousgeometries"/>
    </ProcessRuleSet>
   </rules>
   <mapEntries>
    <ProcessMapEntry param="P" rule="rule-trf-prop-flatten-homogeneousgeometries"
     targetType="GM_Point" type="GM_Point"/>
    <ProcessMapEntry param="C" rule="rule-trf-prop-flatten-homogeneousgeometries"
     targetType="GM_Curve" type="GM_Curve"/>
    <ProcessMapEntry param="S" rule="rule-trf-prop-flatten-homogeneousgeometries"
     targetType="GM_Surface" type="GM_Surface"/>
   </mapEntries>
  </Transformer>
  <Transformer
class="de.interactive_instruments.ShapeChange.Transformation.Flattening.Flattener"
   id="TRF_FLATTEN_REMOVE_NAME_COMPONENT" input="TRF_HOMOGENEOUS_GEOMETRIES"
mode="enabled">
   <parameters>
    <ProcessParameter name="removePropertyNameAndCodeComponent"
     value="\.value(s)?OrReason,\.place-
(curve|surface|point)PositionInfo,\.geometry"/>
```

```
    </parameters>
    <rules>
     <ProcessRuleSet name="trf">
      <rule name="rule-trf-prop-remove-name-and-code-component"/>
     </ProcessRuleSet>
    </rules>
   </Transformer>
   <Transformer id="TRF_TAGGED_VALUES" input="TRF_FLATTEN_REMOVE_NAME_COMPONENT"

class="de.interactive_instruments.ShapeChange.Transformation.Identity.IdentityTransfor
m"
    mode="enabled">
    <taggedValues>
     <TaggedValue name="gmlsfComplianceLevel" value="0" modelElementType="Package"
      applicationSchemaName="OGC Testbed 13 Application Schema"/>
     <TaggedValue name="noPropertyType" value="true" modelElementType="Class"
      applicationSchemaName="OGC Testbed 13 Application Schema"/>
     <TaggedValue name="inlineOrByReference" value="byReference"
modelElementType="Property"
      propertyValueTypeStereotype="(?i:FeatureType)"
      applicationSchemaName="OGC Testbed 13 Application Schema"/>
    </taggedValues>
   </Transformer>
  </transformers>
  <targets>
   <TargetXmlSchema
class="de.interactive_instruments.ShapeChange.Target.XmlSchema.XmlSchema"
    mode="enabled" inputs="TRF_TAGGED_VALUES">
    <targetParameter name="appSchemaName" value="OGC Testbed 13 Application Schema"/>
    <targetParameter name="outputDirectory" value="results/xsd_gmlsf0"/>
    <targetParameter name="sortedOutput" value="true"/>
    <targetParameter name="defaultEncodingRule" value="gmlsf"/>
    <targetParameter name="documentationTemplate" value="[[definition]]"/>
    <rules>
     <EncodingRule name="gmlsf">
      <rule name="req-xsd-cls-generalization-consistent"/>
      <rule name="rule-xsd-all-naming-gml"/>
      <rule name="rule-xsd-cls-codelist-asDictionary"/>
      <rule name="rule-xsd-cls-standard-gml-property-types"/>
      <rule name="rule-xsd-cls-noPropertyType"/>
      <rule name="rule-xsd-prop-targetElement"/>
      <rule name="rule-xsd-prop-inlineOrByReference"/>
      <rule name="req-xsd-cls-codelist-no-supertypes"/>
      <rule name="rule-xsd-cls-codelist-constraints-codeAbsenceInModelAllowed"/>
      <rule name="rule-xsd-cls-enum-supertypes"/>
      <rule name="rule-xsd-cls-enum-subtypes"/>
      <rule name="rule-xsd-cls-local-enumeration"/>
      <rule name="rule-xsd-cls-basictype"/>
      <rule name="rule-xsd-cls-mixin-classes-as-group"/>
      <rule name="rule-xsd-cls-mixin-classes"/>
      <rule name="rule-xsd-cls-mixin-classes-non-mixin-supertypes"/>
```

```
      <rule name="req-xsd-cls-mixin-supertypes-overrule"/>
      <rule name="rule-xsd-prop-exclude-derived"/>
      <rule name="rule-xsd-prop-constrainingFacets"/>
      <rule name="rule-xsd-all-no-documentation"/>
      <rule name="rule-xsd-pkg-gmlsf"/>
      <rule name="rule-xsd-cls-codelist-gmlsf"/>
      <rule name="rule-xsd-prop-featureType-gmlsf-byReference"/>
      <rule name="rule-xsd-prop-metadata-gmlsf-byReference"/>
      <rule name="rule-xsd-cls-union-omitUnionsRepresentingFeatureTypeSets"/>
     </EncodingRule>
    </rules>
    <xsdMapEntries>
     <XsdMapEntry type="CharacterString" xsdEncodingRules="iso19136_2007 gmlsf"
      xmlPropertyType="string" xmlType="string" xmlTypeType="simple"
xmlTypeContent="simple"/>
     <XsdMapEntry type="Integer" xsdEncodingRules="iso19136_2007 gml33 gmlsf"
      xmlPropertyType="integer" xmlType="integer" xmlTypeType="simple"
xmlTypeContent="simple"/>
     <XsdMapEntry type="Boolean" xsdEncodingRules="iso19136_2007 gml33 gmlsf"
      xmlPropertyType="boolean" xmlType="boolean" xmlTypeType="simple"
xmlTypeContent="simple"/>
     <XsdMapEntry type="Real" xsdEncodingRules="iso19136_2007 gml33 gmlsf"
xmlPropertyType="double"
      xmlType="double" xmlTypeType="simple" xmlTypeContent="simple"/>
     <XsdMapEntry type="Date" xsdEncodingRules="iso19136_2007 gml33 gmlsf"
xmlPropertyType="date"
      xmlType="date" xmlTypeType="simple" xmlTypeContent="simple"/>
     <XsdMapEntry type="DateTime" xsdEncodingRules="iso19136_2007 gml33 gmlsf"
      xmlPropertyType="dateTime" xmlType="dateTime" xmlTypeType="simple"
xmlTypeContent="simple"/>
     <XsdMapEntry type="Decimal" xsdEncodingRules="iso19136_2007 gml33 gmlsf"
      xmlPropertyType="double" xmlType="double" xmlTypeType="simple"
xmlTypeContent="simple"/>
     <XsdMapEntry type="Measure" xsdEncodingRules="iso19136_2007 gml33 gmlsf"
      xmlPropertyType="gml:MeasureType" xmlType="gml:MeasureType"
xmlTypeContent="simple"
      xmlTypeNilReason="false"/>
     <XsdMapEntry type="GM_Object" xsdEncodingRules="iso19136_2007 gmlsf gml33"
      xmlType="gml:AbstractGeometryType" xmlElement="gml:AbstractGeometry"
      xmlPropertyType="gml:GeometryPropertyType"/>
     <XsdMapEntry type="GM_Point" xsdEncodingRules="iso19136_2007 gmlsf gml33"
      xmlType="gml:PointType" xmlElement="gml:Point"
xmlPropertyType="gml:PointPropertyType"/>
     <XsdMapEntry type="GM_Curve" xsdEncodingRules="iso19136_2007 gmlsf gml33"
      xmlType="gml:CurveType" xmlElement="gml:Curve"
xmlPropertyType="gml:CurvePropertyType"/>
     <XsdMapEntry type="GM_Surface" xsdEncodingRules="iso19136_2007 gmlsf gml33"
      xmlType="gml:SurfaceType" xmlElement="gml:Surface"
xmlPropertyType="gml:SurfacePropertyType"/>
     <XsdMapEntry type="URN" xmlPropertyType="anyURI" xmlType="anyURI"
xmlTypeContent="simple"
```

```
     xmlTypeType="simple" xsdEncodingRules="*"/>
    <XsdMapEntry type="URI" xmlPropertyType="anyURI" xmlType="anyURI"
xmlTypeContent="simple"
     xmlTypeType="simple" xsdEncodingRules="*"/>
    <XsdMapEntry type="URL" xmlPropertyType="anyURI" xmlType="anyURI"
xmlTypeContent="simple"
     xmlTypeType="simple" xsdEncodingRules="*"/>
    <XsdMapEntry type="MD_Metadata" xsdEncodingRules="*"
xmlType="mdb:MD_Metadata_Type"
     xmlElement="mdb:MD_Metadata" xmlPropertyType="_MP_"/>
    <XsdMapEntry type="CI_Address" xsdEncodingRules="*" xmlType="cit:CI_Address_Type"
     xmlElement="cit:CI_Address" xmlPropertyType="_MP_"/>
    <XsdMapEntry type="CI_Citation" xsdEncodingRules="*"
xmlType="cit:CI_Citation_Type"
     xmlElement="cit:CI_Citation" xmlPropertyType="_MP_"/>
    <XsdMapEntry type="CI_Contact" xsdEncodingRules="*" xmlType="cit:CI_Contact_Type"
     xmlElement="cit:CI_Contact" xmlPropertyType="_MP_"/>
    <XsdMapEntry type="CI_OnlineResource" xsdEncodingRules="*"
xmlType="cit:CI_OnlineResource_Type"
     xmlElement="cit:CI_OnlineResource" xmlPropertyType="_MP_"/>
    <XsdMapEntry type="CI_Party" xsdEncodingRules="*" xmlType="cit:CI_Party_Type"
     xmlElement="cit:CI_Party" xmlPropertyType="_MP_"/>
    <XsdMapEntry type="CI_PresentationFormCode" xsdEncodingRules="*"
     xmlType="cit:CI_PresentationFormCode_Type"
xmlElement="cit:CI_PresentationFormCode"
     xmlPropertyType="_MP_"/>
    <XsdMapEntry type="CI_Responsibility" xsdEncodingRules="*"
xmlType="cit:CI_Responsibility_Type"
     xmlElement="cit:CI_Responsibility" xmlPropertyType="_MP_"/>
    <XsdMapEntry type="CI_Series" xsdEncodingRules="*" xmlType="cit:CI_Series_Type "
     xmlElement="cit:CI_Series" xmlPropertyType="_MP_"/>
    <XsdMapEntry type="CI_Telephone" xsdEncodingRules="*"
xmlType="cit:CI_Telephone_Type"
     xmlElement="cit:CI_Telephone" xmlPropertyType="_MP_"/>
    <XsdMapEntry type="EX_Extent" xsdEncodingRules="*" xmlType="gex:EX_Extent_Type"
     xmlElement="gex:EX_Extent" xmlPropertyType="_MP_"/>
    <XsdMapEntry type="EX_GeographicExtent" xsdEncodingRules="*"
     xmlType="gex:AbstractEX_GeographicExtent_Type"
xmlElement="gex:AbstractEX_GeographicExtent"
     xmlPropertyType="_MP_"/>
    <XsdMapEntry type="EX_BoundingPolygon" xsdEncodingRules="*"
     xmlType="gex:EX_BoundingPolygon_Type" xmlElement="gex:EX_BoundingPolygon"
     xmlPropertyType="_MP_"/>
    <XsdMapEntry type="MD_BrowseGraphic" xsdEncodingRules="*"
xmlType="mcc:MD_BrowseGraphic_Type"
     xmlElement="mcc:MD_BrowseGraphic" xmlPropertyType="_MP_"/>
    <XsdMapEntry type="MD_Identifier" xsdEncodingRules="*"
xmlType="mcc:MD_Identifier_Type"
     xmlElement="mcc:MD_Identifier" xmlPropertyType="_MP_"/>
    <XsdMapEntry type="MD_Constraints" xsdEncodingRules="*"
xmlType="mco:MD_Constraints_Type"
```

```
      xmlElement="mco:MD_Constraints" xmlPropertyType="_MP_"/>
     <XsdMapEntry type="MD_DigitalTransferOptions" xsdEncodingRules="*"
      xmlType="mrd:MD_DigitalTransferOptions_Type"
xmlElement="mrd:MD_DigitalTransferOptions"
      xmlPropertyType="_MP_"/>
     <XsdMapEntry type="MD_DataIdentification" xsdEncodingRules="*"
      xmlType="mri:MD_DataIdentification_Type" xmlElement="mri:MD_DataIdentification"
      xmlPropertyType="_MP_"/>
     <XsdMapEntry type="MD_Resolution" xsdEncodingRules="*"
xmlElement="mri:MD_Resolution"
      xmlPropertyType="_MP_"/>
     <XsdMapEntry type="LI_Lineage" xsdEncodingRules="*" xmlType="mrl:LI_Lineage_Type"
      xmlElement="mrl:LI_Lineage" xmlPropertyType="_MP_"/>
     <XsdMapEntry type="LI_ProcessStep" xsdEncodingRules="*"
xmlType="mrl:LI_ProcessStep_Type"
      xmlElement="mrl:LI_ProcessStep" xmlPropertyType="_MP_"/>
     <XsdMapEntry type="MD_CharacterSetCode" xsdEncodingRules="*"
      xmlPropertyType="lan:MD_CharacterSetCode_PropertyType"/>
     <XsdMapEntry type="LanguageCode" xsdEncodingRules="*"
      xmlPropertyType="lan:LanguageCode_PropertyType"/>
     <XsdMapEntry type="CountryCode" xsdEncodingRules="*"
xmlPropertyType="lan:Country_PropertyType"/>
     <XsdMapEntry type="PT_FreeText" xsdEncodingRules="*"
      xmlPropertyType="lan:PT_FreeText_PropertyType"/>
     <XsdMapEntry type="PT_Locale" xsdEncodingRules="*"
xmlPropertyType="lan:PT_Locale_PropertyType"/>
     <XsdMapEntry type="LocalisedCharacterString" xsdEncodingRules="iso19136_2007
iso19139_2007"
      xmlPropertyType="lan:LocalisedCharacterString_PropertyType"/>
     <XsdMapEntry type="LocalisedCharacterString" xsdEncodingRules="gml33"
      xmlPropertyType="gmlxbt:LanguageStringType"/>
    </xsdMapEntries>
    <xmlNamespaces>
     <XmlNamespace nsabr="gmlsf" ns="http://www.opengis.net/gmlsf/2.0"
      location="http://schemas.opengis.net/gmlsfProfile/2.0/gmlsfLevels.xsd"/>
     <XmlNamespace nsabr="gml" ns="http://www.opengis.net/gml/3.2"
      location="../../../ogc/gml/3.2.1/gml.xsd"/>
    </xmlNamespaces>
   </TargetXmlSchema>
   <Target class="de.interactive_instruments.ShapeChange.Target.CDB.CDB" mode="enabled"
    inputs="TRF_TAGGED_VALUES">
    <advancedProcessConfigurations>
     <CDBUnitDefinition code="1" symbol="m">
      <name>meter</name>
      <alias>metre</alias>
      <description>To measure a length.</description>
     </CDBUnitDefinition>
     <CDBUnitDefinition code="2" symbol="deg">
      <name>degree</name>
      <description>To mesure an angle.</description>
     </CDBUnitDefinition>
```

```
    <CDBUnitDefinition code="3" symbol="rad">
     <name>radian</name>
     <description>To mesure an angle.</description>
    </CDBUnitDefinition>
    <CDBUnitDefinition code="4" symbol="kph">
     <name>Kilometer per hour</name>
     <description>To measure a speed.</description>
    </CDBUnitDefinition>
    <CDBUnitDefinition code="5" symbol="g">
     <name>gram</name>
     <description>To measure a mass.</description>
    </CDBUnitDefinition>
    <CDBUnitDefinition code="6" symbol="s">
     <name>second</name>
     <description>To measure time.</description>
    </CDBUnitDefinition>
    <CDBUnitDefinition code="7" symbol="%">
     <name>Percentage</name>
     <description>A value between 0 and 100.</description>
    </CDBUnitDefinition>
   </advancedProcessConfigurations>
   <targetParameter name="outputDirectory" value="results/cdb"/>
   <targetParameter name="outputFilename" value="Test"/>
   <targetParameter name="sortedOutput" value="true"/>
   <targetParameter name="defaultEncodingRule" value="cdb"/>
   <targetParameter name="unitsToIgnore" value="unitless"/>
   <rules>
    <EncodingRule name="cdb" extends="*">
     <rule name="rule-cdb-all-valueTypeTextForUnionRepresentingFeatureSet"/>
    </EncodingRule>
   </rules>
   <mapEntries>
    <MapEntry type="Numeric" targetType="Numeric" rule="*"
param="numericFormat{Floating-Point}"/>
    <MapEntry type="Real" targetType="Numeric" rule="*" param="numericFormat{Floating-
Point}"/>
    <MapEntry type="Decimal" targetType="Numeric" rule="*"
param="numericFormat{Floating-Point}"/>
    <MapEntry type="Measure" targetType="Numeric" rule="*"
param="numericFormat{Floating-Point}"/>
    <MapEntry type="Integer" targetType="Numeric" rule="*"
param="numericFormat{Integer}"/>
    <MapEntry type="Boolean" targetType="Boolean" rule="*"/>
    <MapEntry type="CharacterString" targetType="Text" rule="*"/>
    <MapEntry type="Character" targetType="Text" rule="*"/>
    <MapEntry type="Date" targetType="Text" rule="*"/>
    <MapEntry type="DateTime" targetType="Text" rule="*"/>
    <MapEntry type="LocalName" targetType="Text" rule="*"/>
    <MapEntry type="ScopedName" targetType="Text" rule="*"/>
    <MapEntry type="Any" targetType="Text" rule="*"/>
    <MapEntry type="GM_Point" targetType="Text" rule="*"/>
```

```
    <MapEntry type="GM_Surface" targetType="Text" rule="*"/>
    <MapEntry type="GM_Curve" targetType="Text" rule="*"/>
    <MapEntry type="CI_Address" targetType="Text" rule="*"/>
    <MapEntry type="CI_Contact" targetType="Text" rule="*"/>
   </mapEntries>
  </Target>
 </targets>
</ShapeChangeConfiguration>
```

ShapeChange Configuration to generate the CityGML ADE for the NAS Profile

The following listing contains the ShapeChange configuration that was used to create the CityGML ADE NAS profile in Testbed-13.

```
<?xml version="1.0" encoding="UTF-8"?>
<ShapeChangeConfiguration
 xmlns="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1"
 xmlns:sc="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1"
 xmlns:xi="http://www.w3.org/2001/XInclude"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.interactive-
instruments.de/ShapeChange/Configuration/1.1
http://shapechange.net/resources/schema/ShapeChangeConfiguration.xsd">
 <input id="INPUT">
  <parameter name="inputModelType" value="SCXML"/>
  <parameter name="inputFile" value="models/TB13AS_v1.2_FULL_with_CityGML_ADE.zip"/>
  <parameter name="ignoreEncodingRuleTaggedValues" value="true"/>
  <parameter name="publicOnly" value="true"/>
  <parameter name="checkingConstraints" value="enabled"/>
  <parameter name="sortedOutput" value="true"/>
  <parameter name="sortedSchemaOutput" value="true"/>
  <parameter name="appSchemaNameRegex" value="OGC Testbed 13 Application Schema"/>
  <xi:include href="http://shapechange.net/resources/config/StandardAliases.xml"/>
  <descriptorSources>
   <DescriptorSource descriptor="documentation" source="tag" tag="documentation"/>
   <DescriptorSource descriptor="alias" source="tag" tag="name"/>
   <DescriptorSource descriptor="primaryCode" source="tag" tag="primaryCode"/>
   <DescriptorSource descriptor="definition" source="tag" tag="definition"/>
   <DescriptorSource descriptor="description" source="tag" tag="description"/>
   <DescriptorSource descriptor="example" source="none"/>
   <DescriptorSource descriptor="legalBasis" source="none"/>
   <DescriptorSource descriptor="dataCaptureStatement" source="none"/>
   <DescriptorSource descriptor="language" source="none"/>
  </descriptorSources>
 </input>
 <log>
  <parameter name="reportLevel" value="INFO"/>
  <parameter name="logFile" value="results/ade/log.xml"/>
 </log>
 <transformers>
```

```
  <Transformer id="IDENTITY" input="INPUT"

class="de.interactive_instruments.ShapeChange.Transformation.Identity.IdentityTransfor
m">
   <taggedValues>
    <TaggedValue name="targetNamespace"
value="http://metadata.dod.mil/mdr/ns/GSIP/6.0/tdas/6.0/ade"
     modelElementName="OGC Testbed 13 Application Schema" modelElementType="Package"/>
   </taggedValues>
  </Transformer>
  <Transformer

class="de.interactive_instruments.ShapeChange.Transformation.CityGML.CityGMLTransforme
r"
   id="TRF_CREATE_ADE" input="IDENTITY" mode="enabled">
   <rules>
    <ProcessRuleSet name="trf">
     <rule name="rule-trf-CityGML-createADE"/>
    </ProcessRuleSet>
   </rules>
   <mapEntries>
    <ProcessMapEntry type="AircraftHangar" targetType="bldg:Building"
     rule="rule-trf-CityGML-createADE"/>
    <ProcessMapEntry type="Apron" targetType="tran:TrafficArea" rule="rule-trf-
CityGML-createADE"/>
    <ProcessMapEntry type="Bench" targetType="frn:CityFurniture" rule="rule-trf-
CityGML-createADE"/>
    <ProcessMapEntry type="Billboard" targetType="frn:CityFurniture"
     rule="rule-trf-CityGML-createADE"/>
    <ProcessMapEntry type="Building" targetType="bldg:Building" rule="rule-trf-
CityGML-createADE"/>
    <ProcessMapEntry type="BuildingOverhang" targetType="bldg:BuildingInstallation"
     rule="rule-trf-CityGML-createADE"/>
    <ProcessMapEntry type="BuildingSuperstructure"
targetType="bldg:BuildingInstallation"
     rule="rule-trf-CityGML-createADE"/>
    <ProcessMapEntry type="ControlTower" targetType="bldg:Building"
     rule="rule-trf-CityGML-createADE"/>
    <ProcessMapEntry type="EntranceExit" targetType="bldg:Door" rule="rule-trf-
CityGML-createADE"/>
    <ProcessMapEntry type="Flagpole" targetType="frn:CityFurniture"
     rule="rule-trf-CityGML-createADE"/>
    <ProcessMapEntry type="FortifiedBuilding" targetType="bldg:Building"
     rule="rule-trf-CityGML-createADE"/>
    <ProcessMapEntry type="HardenedAircraftShelter" targetType="bldg:Building"
     rule="rule-trf-CityGML-createADE"/>
    <ProcessMapEntry type="Helipad" targetType="tran:TrafficArea" rule="rule-trf-
CityGML-createADE"/>
    <ProcessMapEntry type="Heliport" targetType="tran:TransportationComplex"
     rule="rule-trf-CityGML-createADE"/>
    <ProcessMapEntry type="LandAerodrome" targetType="tran:TransportationComplex"
```

```
     rule="rule-trf-CityGML-createADE"/>
    <ProcessMapEntry type="LightSupportStructure" targetType="frn:CityFurniture"
     rule="rule-trf-CityGML-createADE"/>
    <ProcessMapEntry type="NonBuildingStructure" targetType="bldg:Building"
     rule="rule-trf-CityGML-createADE"/>
    <ProcessMapEntry type="PicnicTable" targetType="frn:CityFurniture"
     rule="rule-trf-CityGML-createADE"/>
    <ProcessMapEntry type="Planter" targetType="frn:CityFurniture" rule="rule-trf-
CityGML-createADE"/>
    <ProcessMapEntry type="Runway" targetType="tran:TrafficArea" rule="rule-trf-
CityGML-createADE"/>
    <ProcessMapEntry type="Shed" targetType="bldg:Building" rule="rule-trf-CityGML-
createADE"/>
    <ProcessMapEntry type="Stair" targetType="bldg:BuildingInstallation"
     rule="rule-trf-CityGML-createADE"/>
    <ProcessMapEntry type="Stopway" targetType="tran:TrafficArea" rule="rule-trf-
CityGML-createADE"/>
    <ProcessMapEntry type="Taxiway" targetType="tran:TrafficArea" rule="rule-trf-
CityGML-createADE"
    />
   </mapEntries>
  </Transformer>
 </transformers>
 <targets>
  <TargetXmlSchema inputs="TRF_CREATE_ADE"
   class="de.interactive_instruments.ShapeChange.Target.XmlSchema.XmlSchema"
mode="enabled">
   <targetParameter name="outputDirectory" value="results/ade"/>
   <targetParameter name="sortedOutput" value="true"/>
   <targetParameter name="defaultEncodingRule" value="nas_ade"/>
   <rules>
    <EncodingRule name="nas_ade" extends="citygml-ade">
     <rule name="req-xsd-cls-codelist-no-supertypes"/>
     <rule name="rule-xsd-cls-union-asCharacterString"/>
     <rule name="rule-xsd-cls-union-asGroup"/>
     <rule name="rule-xsd-cls-enum-supertypes"/>
     <rule name="rule-xsd-cls-enum-subtypes"/>
     <rule name="rule-xsd-cls-basictype"/>
     <rule name="rule-xsd-cls-union-direct"/>
     <rule name="rule-xsd-cls-codelist-constraints"/>
     <rule name="rule-xsd-cls-mixin-classes-as-group"/>
     <rule name="rule-xsd-cls-mixin-classes"/>
     <rule name="rule-xsd-cls-mixin-classes-non-mixin-supertypes"/>
     <rule name="req-xsd-cls-mixin-supertypes-overrule"/>
     <rule name="rule-xsd-prop-exclude-derived"/>
     <rule name="rule-xsd-prop-length-size-pattern"/>
     <rule name="rule-xsd-prop-xsdAsAttribute"/>
     <rule name="rule-xsd-prop-nillable"/>
     <rule name="rule-xsd-prop-nilReasonAllowed"/>
     <rule name="rule-xsd-prop-initialValue"/>
     <rule name="rule-xsd-prop-att-map-entry"/>
```

```
      <rule name="rule-xsd-all-tagged-values"/>
      <rule name="rule-xsd-all-no-documentation"/>
      <rule name="rule-xsd-rel-association-classes"/>
     </EncodingRule>
    </rules>
    <targetParameter name="includeDocumentation" value="true"/>

    <xi:include href="config/StandardRules.xml"/>

    <xi:include href="config/StandardMapEntries_gml31.xml"/>
    <xi:include href="config/StandardMapEntries_iso19107-v31.xml"/>
    <xi:include href="config/StandardMapEntries_iso19108-v31.xml"/>
    <xi:include href="config/StandardMapEntries_iso19111-v31.xml"/>
    <xi:include href="config/StandardMapEntries_iso19112.xml"/>
    <xi:include href="config/StandardMapEntries_iso19115-1.xml"/>
    <xi:include href="config/StandardMapEntries_iso19123-v31.xml"/>
    <xi:include href="config/StandardMapEntries_iso19156.xml"/>
    <xi:include href="config/StandardMapEntries_iso19157.xml"/>
    <xi:include href="config/StandardMapEntries_gmlcov.xml"/>
    <xsdMapEntries>
     <XsdMapEntry type="URN" xmlPropertyType="anyURI" xmlType="anyURI"
xmlTypeContent="simple"
      xmlTypeType="simple" xsdEncodingRules="*"/>
     <XsdMapEntry type="URI" xmlPropertyType="anyURI" xmlType="anyURI"
xmlTypeContent="simple"
      xmlTypeType="simple" xsdEncodingRules="*"/>
     <XsdMapEntry type="URL" xmlPropertyType="anyURI" xmlType="anyURI"
xmlTypeContent="simple"
      xmlTypeType="simple" xsdEncodingRules="*"/>

     <XsdMapEntry type="IntegerNonNegativeType" xsdEncodingRules="*"
xmlPropertyType="integer"
      xmlType="integer" xmlTypeType="simple" xmlTypeContent="simple"/>
     <XsdMapEntry type="RealNonNegativeType" xsdEncodingRules="*"
xmlPropertyType="double"
      xmlType="double" xmlTypeType="simple" xmlTypeContent="simple"/>
     <XsdMapEntry type="RealUnconstrainedType" xsdEncodingRules="*"
xmlPropertyType="double"
      xmlType="double" xmlTypeType="simple" xmlTypeContent="simple"/>
     <XsdMapEntry type="CharacterString" xsdEncodingRules="*" xmlPropertyType="string"
      xmlType="string" xmlTypeType="simple" xmlTypeContent="simple"/>

     <XsdMapEntry type="SecurityAttributesGroupType" xmlPropertyType="_P_"
xmlTypeNilReason="false"
      xmlAttributeGroup="icism:SecurityAttributesOptionGroup" xsdEncodingRules="*"/>
     <XsdMapEntry type="ISM_Notice" xmlPropertyType="icism:NoticeType"
xmlType="icism:NoticeType"
      xmlTypeContent="complex" xmlTypeNilReason="false" xsdEncodingRules="*"/>
     <XsdMapEntry type="NTKAccess" xmlPropertyType="ntk:RequiresType"
xmlType="ntk:RequiresType"
      xmlTypeContent="complex" xmlTypeNilReason="false" xsdEncodingRules="*"/>
```

```
    <XsdMapEntry type="RevisionRecall" xmlPropertyType="rr:RevisionRecallType"
     xmlType="rr:RevisionRecallType" xmlTypeContent="complex" xmlTypeNilReason="false"
     xsdEncodingRules="*"/>
   </xsdMapEntries>

   <xsdMapEntries>
    <XsdMapEntry type="_CityObject" xsdEncodingRules="iso19136_2007"
xmlElement="core:_CityObject"
     xmlType="core:AbstractCityObjectType"/>
    <XsdMapEntry type="Building" xsdEncodingRules="iso19136_2007"
xmlElement="bldg:Building"
     xmlType="bldg:BuildingType"/>
    <XsdMapEntry type="BuildingInstallation" xsdEncodingRules="iso19136_2007"
     xmlElement="bldg:BuildingInstallation" xmlType="bldg:BuildingInstallationType"
     xmlPropertyType="bldg:BuildingInstallationPropertyType"/>
    <XsdMapEntry type="CityFurniture" xsdEncodingRules="iso19136_2007"
     xmlElement="frn:CityFurniture" xmlType="frn:CityFurnitureType"/>
    <XsdMapEntry type="Door" xsdEncodingRules="iso19136_2007" xmlElement="bldg:Door"
     xmlType="bldg:DoorType"/>
    <XsdMapEntry type="TrafficArea" xsdEncodingRules="iso19136_2007"
xmlElement="tran:TrafficArea"
     xmlType="tran:TrafficAreaType" xmlPropertyType="tran:TrafficAreaPropertyType"/>
    <XsdMapEntry type="TransportationComplex" xsdEncodingRules="iso19136_2007"
     xmlElement="tran:TransportationComplex" xmlType=
"tran:TransportationComplexType"/>
   </xsdMapEntries>

   <xmlNamespaces>

    <!-- Standard namespaces -->
    <XmlNamespace nsabr="sc" ns="http://www.interactive-
instruments.de/ShapeChange/AppInfo"
     location="../../../schema/ShapeChangeAppinfo.xsd"/>
    <!--   -->
    <XmlNamespace nsabr="xlink" ns="http://www.w3.org/1999/xlink"
     location="../../../w3/1999/xlink/xlinks.xsd"/>

    <XmlNamespace nsabr="iso19112" ns="http://standards.iso.org/iso/19112/BP_2012"
     location="../../../iso/19112/BP_2012/si.xsd"/>
    <XmlNamespace nsabr="gmlsf" ns="http://www.opengis.net/gmlsf/2.0"
     location="http://schemas.opengis.net/gmlsfProfile/2.0/gmlsfLevels.xsd"/>
    <XmlNamespace nsabr="gml" ns="http://www.opengis.net/gml"
     location="../../../ogc/gml/3.1.1/base/gml.xsd"/>
    <XmlNamespace nsabr="gml32" ns="http://www.opengis.net/gml/3.2"
     location="../../../ogc/gml/3.2.1/gml.xsd"/>
    <XmlNamespace nsabr="gmlxbt" ns="http://www.opengis.net/gml/3.3/xbt"
     location="../../../ogc/gml/3.3/extdBaseTypes.xsd"/>
    <XmlNamespace nsabr="gmlce" ns="http://www.opengis.net/gml/3.3/ce"
     location="../../../ogc/gml/3.3/geometryCompact.xsd"/>
    <XmlNamespace nsabr="gmltin" ns="http://www.opengis.net/gml/3.3/tin"
     location="../../../ogc/gml/3.3/tin.xsd"/>
```

```
<XmlNamespace nsabr="gmllr" ns="http://www.opengis.net/gml/3.3/lr"
 location="../../../ogc/gml/3.3/linearRef.xsd"/>
<XmlNamespace nsabr="gmllrtr" ns="http://www.opengis.net/gml/3.3/lrtr"
 location="../../../ogc/gml/3.3/linearRefTowardsReferent.xsd"/>
<XmlNamespace nsabr="gmllro" ns="http://www.opengis.net/gml/3.3/lro"
 location="../../../ogc/gml/3.3/linearRefOffset.xsd"/>
<XmlNamespace nsabr="gmllrov" ns="http://www.opengis.net/gml/3.3/lrov"
 location="../../../ogc/gml/3.3/linearRefOffsetVector.xsd"/>
<XmlNamespace nsabr="gmlrgrid" ns="http://www.opengis.net/gml/3.3/rgrid"
 location="../../../ogc/gml/3.3/referencableGrid.xsd"/>
<XmlNamespace nsabr="gmlexr" ns="http://www.opengis.net/gml/3.3/exr"
 location="../../../ogc/gml/3.3/extdEncRule.xsd"/>
<XmlNamespace nsabr="gmlcov" ns="http://www.opengis.net/gmlcov/1.0"
 location="../../../ogc/gmlcov/1.0/gmlcovAll.xsd"/>
<XmlNamespace nsabr="om" ns="http://www.opengis.net/om/2.0"
 location="../../../ogc/om/2.0/observation.xsd"/>
<XmlNamespace nsabr="swe" ns="http://www.opengis.net/swe/2.0"
 location="../../../ogc/sweCommon/2.0/swe.xsd"/>
<XmlNamespace nsabr="xsi" ns="http://www.w3.org/2001/XMLSchema-instance"/>
<XmlNamespace nsabr="cat" ns="http://standards.iso.org/iso/19115/-3/cat/1.0"
 location="../../../iso/19115/-3/cat/1.0/cat.xsd"/>
<XmlNamespace nsabr="cit" ns="http://standards.iso.org/iso/19115/-3/cit/1.0"
 location="../../../iso/19115/-3/cit/1.0/cit.xsd"/>
<XmlNamespace nsabr="gco" ns="http://standards.iso.org/iso/19115/-3/gco/1.0"
 location="../../../iso/19115/-3/gco/1.0/gco.xsd"/>
<XmlNamespace nsabr="gcx" ns="http://standards.iso.org/iso/19115/-3/gcx/1.0"
 location="../../../iso/19115/-3/gcx/1.0/gcx.xsd"/>
<XmlNamespace nsabr="gex" ns="http://standards.iso.org/iso/19115/-3/gex/1.0"
 location="../../../iso/19115/-3/gex/1.0/gex.xsd"/>
<XmlNamespace nsabr="lan" ns="http://standards.iso.org/iso/19115/-3/lan/1.0"
 location="../../../iso/19115/-3/lan/1.0/lan.xsd"/>
<XmlNamespace nsabr="mas" ns="http://standards.iso.org/iso/19115/-3/mas/1.0"
 location="../../../iso/19115/-3/mas/1.0/mas.xsd"/>
<XmlNamespace nsabr="mcc" ns="http://standards.iso.org/iso/19115/-3/mcc/1.0"
 location="../../../iso/19115/-3/mcc/1.0/mcc.xsd"/>
<XmlNamespace nsabr="mco" ns="http://standards.iso.org/iso/19115/-3/mco/1.0"
 location="../../../iso/19115/-3/mco/1.0/mco.xsd"/>
<XmlNamespace nsabr="mex" ns="http://standards.iso.org/iso/19115/-3/mex/1.0"
 location="../../../iso/19115/-3/mex/1.0/mex.xsd"/>
<XmlNamespace nsabr="mmi" ns="http://standards.iso.org/iso/19115/-3/mmi/1.0"
 location="../../../iso/19115/-3/mmi/1.0/mmi.xsd"/>
<XmlNamespace nsabr="mpc" ns="http://standards.iso.org/iso/19115/-3/mpc/1.0"
 location="../../../iso/19115/-3/mpc/1.0/mpc.xsd"/>
<XmlNamespace nsabr="mrc" ns="http://standards.iso.org/iso/19115/-3/rco/1.0"
 location="../../../iso/19115/-3/mrc/1.0/mrc.xsd"/>
<XmlNamespace nsabr="mrd" ns="http://standards.iso.org/iso/19115/-3/mrd/1.0"
 location="../../../iso/19115/-3/mrd/1.0/mrd.xsd"/>
<XmlNamespace nsabr="mri" ns="http://standards.iso.org/iso/19115/-3/mri/1.0"
 location="../../../iso/19115/-3/mri/1.0/mri.xsd"/>
<XmlNamespace nsabr="mrl" ns="http://standards.iso.org/iso/19115/-3/mrl/1.0"
 location="../../../iso/19115/-3/mrl/1.0/mrl.xsd"/>
```

```
    <XmlNamespace nsabr="mrs" ns="http://standards.iso.org/iso/19115/-3/mrs/1.0"
     location="../../../iso/19115/-3/mrs/1.0/mrs.xsd"/>
    <XmlNamespace nsabr="msr" ns="http://standards.iso.org/iso/19115/-3/msr/1.0"
     location="../../../iso/19115/-3/msr/1.0/msr.xsd"/>
    <XmlNamespace nsabr="srv" ns="http://standards.iso.org/iso/19115/-3/srv/2.0"
     location="../../../iso/19115/-3/srv/2.0/srv.xsd"/>
    <XmlNamespace nsabr="mda" ns="http://standards.iso.org/iso/19115/-3/mda/1.0"
     location="../../../iso/19115/-3/mda/1.0/mda.xsd"/>
    <XmlNamespace nsabr="mdb" ns="http://standards.iso.org/iso/19115/-3/mdb/1.0"
     location="../../../iso/19115/-3/mdb/1.0/mdb.xsd"/>
    <XmlNamespace nsabr="mds" ns="http://standards.iso.org/iso/19115/-3/mds/1.0"
     location="../../../iso/19115/-3/mds/1.0/mds.xsd"/>
    <XmlNamespace nsabr="mdt" ns="http://standards.iso.org/iso/19115/-3/mdt/1.0"
     location="../../../iso/19115/-3/mdt/1.0/mdt.xsd"/>
    <XmlNamespace nsabr="md1" ns="http://standards.iso.org/iso/19115/-3/md1/1.0"
     location="../../../iso/19115/-3/md1/1.0/md1.xsd"/>
    <XmlNamespace nsabr="mex" ns="http://standards.iso.org/iso/19115/-3/mex/1.0"
     location="../../../iso/19115/-3/mex/1.0/mex.xsd"/>
    <XmlNamespace nsabr="dqc" ns="http://standards.iso.org/iso/19157/-2/dqc/1.0"
     location="../../../iso/19157/-2/dqc/1.0/dqc.xsd"/>
    <XmlNamespace nsabr="dqm" ns="http://standards.iso.org/iso/19157/-2/dqm/1.0"
     location="../../../iso/19157/-2/dqm/1.0/dqm.xsd"/>
    <XmlNamespace nsabr="mdq" ns="http://standards.iso.org/iso/19157/-2/mdq/1.0"
     location="../../../iso/19157/-2/mdq/1.0/mdq.xsd"/>

    <!-- NAS specific namespaces -->
    <XmlNamespace nsabr="icism" ns="urn:us:gov:ic:ism"
location="../../../ic/ism/V13/IC-ISM.xsd"/>
    <XmlNamespace nsabr="ntk" ns="urn:us:gov:ic:ntk" location="../../../ic/ntk/V10/IC-
NTK.xsd"/>
    <XmlNamespace nsabr="rr" ns="urn:us:gov:ic:revrecall"
     location="../../../ic/RevRecall/V1/RevRecall_XML.xsd"/>

    <!-- CityGML namespaces -->
    <XmlNamespace nsabr="core" ns="http://www.opengis.net/citygml/2.0"
     location="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
    <XmlNamespace nsabr="app" ns="http://www.opengis.net/citygml/appearance/2.0"
     location="http://schemas.opengis.net/citygml/appearance/2.0/appearance.xsd"/>
    <XmlNamespace nsabr="brid" ns="http://www.opengis.net/citygml/bridge/2.0"
     location="http://schemas.opengis.net/citygml/bridge/2.0/bridge.xsd"/>
    <XmlNamespace nsabr="bldg" ns="http://www.opengis.net/citygml/building/2.0"
     location="http://schemas.opengis.net/citygml/building/2.0/building.xsd"/>
    <XmlNamespace nsabr="frn" ns="http://www.opengis.net/citygml/cityfurniture/2.0"
     location=
"http://schemas.opengis.net/citygml/cityfurniture/2.0/cityFurniture.xsd"/>
    <XmlNamespace nsabr="grp" ns="http://www.opengis.net/citygml/cityobjectgroup/2.0"

location="http://schemas.opengis.net/citygml/cityobjectgroup/2.0/cityObjectGroup.xsd"/
>
    <XmlNamespace nsabr="gen" ns="http://www.opengis.net/citygml/generics/2.0"
     location="http://schemas.opengis.net/citygml/generics/2.0/generics.xsd"/>
```

```
    <XmlNamespace nsabr="luse" ns="http://www.opengis.net/citygml/landuse/2.0"
     location="http://schemas.opengis.net/citygml/landuse/2.0/landUse.xsd"/>
    <XmlNamespace nsabr="dem" ns="http://www.opengis.net/citygml/relief/2.0"
     location="http://schemas.opengis.net/citygml/relief/2.0/relief.xsd"/>
    <XmlNamespace nsabr="tran" ns="http://www.opengis.net/citygml/transportation/2.0"

location="http://schemas.opengis.net/citygml/transportation/2.0/transportation.xsd"/>
    <XmlNamespace nsabr="tun" ns="http://www.opengis.net/citygml/tunnel/2.0"
     location="http://schemas.opengis.net/citygml/tunnel/2.0/tunnel.xsd"/>
    <XmlNamespace nsabr="veg" ns="http://www.opengis.net/citygml/vegetation/2.0"
     location="http://schemas.opengis.net/citygml/vegetation/2.0/vegetation.xsd"/>
    <XmlNamespace nsabr="wtr" ns="http://www.opengis.net/citygml/waterbody/2.0"
     location="http://schemas.opengis.net/citygml/waterbody/2.0/waterBody.xsd"/>
   </xmlNamespaces>
  </TargetXmlSchema>
 </targets>
</ShapeChangeConfiguration>
```

Appendix B: Information on the NAS Profile

This annex provides further information on the content of the NAS profile. It lists the feature types contained in the profile, grouped by first and second level packages within the schema.

- Aeronautical Facility
  - Aerodrome Buildings and Structures
    - AircraftHangar
    - ControlTower
    - HardenedAircraftShelter
  - Aircraft Movement Surfaces
    - AerodromeMoveArea
    - Apron
    - Helipad
    - Runway
    - Taxiway
  - General Aeronautical Ground Features
    - Aerodrome
    - Heliport
    - LandAerodrome
  - Movement Area Safety Features
    - Stopway
- Agricultural
  - Agricultural Buildings and Structures
    - GrainElevator
    - Greenhouse
  - Crop Land
    - CropLand
    - Orchard
- Boundaries
  - Boundaries and Lines
    - AdministrativeBoundary
    - AdministrativeSubdivision
    - SpecialAdminDivision

    - Zones and Areas
        - ConservationArea
        - ContaminatedRegion
        - GeopoliticalEntity
- Cultural
    - Burial Sites
        - Cemetery
        - Tomb
    - Commercial Facilities
        - OfficePark
        - RetailStand
        - ShoppingComplex
    - Communication Facilities
        - Cable
        - Pylon
        - Tower
    - Disposal and Waste Management Facilities
        - DisposalSite
        - WasteHeap
    - Extraction Facilities
        - Well
    - General Structures
        - Building
        - BuildingOverhang
        - BuildingSuperstructure
        - EntranceExit
        - Facility
        - FortifiedBuilding
        - Installation
        - NonBuildingStructure
        - OverheadObstruction
        - Shed
        - Stair
        - Wall
    - Material Handling Features

        - Cableway
    - Miscellaneous Cultural Features
        - Checkpoint
        - Fence
        - FiringRange
        - RegulatorySign
        - TrainingSite
    - Monumental and Ornamental Structures and Areas
        - BotanicGarden
        - Fountain
        - MemorialMonument
        - Park
        - StatuePedestal
    - Non-Urban Area Dwellings
        - Hut
    - Power Generation and Transmission Facilities
        - CoolingFacility
        - CoolingTower
        - ElectricPowerStation
        - HeatingFacility
        - PowerSubstation
        - Smokestack
        - SolarFarm
        - SolarPanel
        - WindFarm
        - WindTurbine
    - Processing Facilities
        - HydrocarbonProdFacility
    - Recreational Facilities
        - AmusementPark
        - AmusementParkAttraction
        - Bandshell
        - Courtyard
        - DriveInTheatre
        - Fairground

        - GolfCourse
        - OutdoorTheatreScreen
        - PublicSquare
        - Racetrack
        - SportsGround
        - SwimmingPool
        - TennisCourt
        - Zoo
    - Spectator Event Structures
        - Grandstand
        - Scoreboard
        - Stadium
    - Storage Facilities
        - Cistern
        - FuelStorageFacility
        - MunitionStorageFacility
        - ParkingGarage
        - ShippingContainer
        - StorageDepot
        - StorageTank
        - TankFarm
        - VehicleLot
        - WaterTower
    - Urban and Street Furniture
        - Bench
        - Billboard
        - DisplaySign
        - Flagpole
        - LightSupportStructure
        - PicnicTable
        - Planter
    - Utility Infrastructure
        - BuriedUtility
        - SewageTreatmentPlant
        - StormDrain

        - UtilityAccessPoint
        - UtilityNetwork
        - Waterwork
- Devices
    - Distribution Devices
        - DistributionDevice
        - FireHydrant
    - Electrical Devices
        - ElectricalDevice
        - PowerGeneratingUnit
    - Electronic Devices
        - Aerial
        - DishAerial
        - ElectronicCommsDevice
        - ElectronicDevice
- Elevation
    - Hypsography
        - ElevationContour
        - SpotElevation
- Foundation
    - General Feature Model
        - ActorEntity
        - DeviceEntity
        - FeatureEntity
        - TemporalEntity
    - General Resource Model
        - Entity
- Human Geography
    - Groups and Organizations
        - Organisation
        - Person
    - Land Ownership
        - BuildingUnit
        - SpatialUnit
        - SpatialUnitGroup

    - Significant Events
        - Event
    - Transportation Use
        - TransportationStop
- Hydrographic Aids to Navigation
    - Maritime Navigational Aids
        - Buoy
        - MaritimeNavigationLight
        - MaritimeNavLightSupport
- Inland Water
    - Man-made Water Features
        - Aqueduct
        - Dam
        - Ditch
        - Lock
        - StructuralPile
    - Natural Water Features
        - InlandWaterbody
        - InlandWaterbodyBank
        - NaturalPool
        - River
        - WaterWell
- Military Installations and Defensive Structures
    - Fortified Military Installations
        - InstallationBoundary
        - MilitaryInstallation
    - Military Defensive Structures
        - DefensiveRevetment
        - Minefield
        - SurfaceBunker
- Ocean Environment
    - Man-made Marine Features
        - ShorelineConstruction
- Physiography
    - Exposed Surface Materials

        - SoilSurfaceRegion
    - Landforms
        - Island
    - Littoral Topography
        - Beach
        - LandWaterBoundary
    - Man-made Landforms
        - Cut
        - CutLine
        - Embankment
- Population
    - Settlements
        - Camp
        - CaravanPark
        - Neighbourhood
        - PopulatedPlace
        - ShantyTown
- Ports and Harbours
    - Berthing Facilities
        - ShorelineRamp
    - Port and Harbour Areas
        - Dock
        - Harbour
        - SmallCraftFacility
    - Shipyards
        - Shipyard
- Transportation
    - Inland Water Transportation and Associated Features
        - FerryStation
        - NavigableCanal
    - Pedestrian and/or Animal Transportation and Associated Features
        - Sidewalk
        - Trail
    - Pipelines and Associated Features
        - Pipeline

        - PumpingStation
    - Railways and Associated Features
        - Railway
        - RailwaySidetrack
        - RailwayYard
        - TransportationStation
    - Roads and Associated Features
        - CartTrack
        - Crossing
        - Culvert
        - Gate
        - MotorVehicleStation
        - Road
        - RoadInterchange
        - StreetLamp
        - StreetSign
        - TrafficLight
    - Transportation Structures
        - Bridge
        - BridgePier
        - BridgeSpan
        - BridgeSuperstructure
        - BridgeTower
        - CausewayStructure
        - Tunnel
        - TunnelMouth
- Vegetation
    - Rangelands
        - Grassland
    - Wetlands
        - Marsh
        - Swamp
        - Wetland
    - Woodlands
        - Forest

- Tree

Appendix C: Revision History

Table 12. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
April 27, 2017	C. Portele	0.1	all	initial version
May 12, 2017	C. Portele	0.2	all	update, mainly summary and CDB
August 3, 2017	J. Echterhoff	0.3	chapter 7	document GML-SF0 generation
August 9, 2017	C. Portele	0.4	chapters 8, 9	document CityGML analysis, update CDB chapter
August 15, 2017	J. Echterhoff, C. Portele	0.5	chapters 1, 7	Add introduction, update GML-SF0 chapter
August 22, 2017	J. Echterhoff	0.6	chapter 7	Update GML-SF0 chapter
August 23, 2017	C. Portele	0.7	chapters 8, 9	Work-in-progress, updates after more analysis of the processing for CDB dictionaries and the ADE
August 30, 2017	J. Echterhoff, C. Portele	0.8	chapters 1, 7, 8, 9	Updates after review by Paul Birkel
September 06, 2017	C. Portele	0.9	bibliography	Editorial update
September 11, 2017	J. Echterhoff	0.10	chapter 6	Document processing a profile with ShapeChange

Date	Editor	Release	Primary clauses modified	Descriptions
September 18, 2017	J. Echterhoff	0.11	throughout	Editorial updates and description of ShapeChange features for generation of CDB dictionaries and CityGML ADE
September 19, 2017	C. Portele	0.12	chapters 4, 9	Updates based on feedback by Paul Birkel and Arne Schilling
September 20, 2017	C. Portele, J. Echterhoff	0.13	chapters 2, 3, 5	New Annex B, updated ShapeChange configurations in Annex A, editorial updates and other minor edits
September 24, 2017	C. Portele	0.14	chapters 1, 6, 7, 8, 9	Document findings and recommendations, editorial updates
September 25, 2017	J. Echterhoff	0.15	chapters 1 and 6	Future work items for profiling
October 5, 2017	C. Portele	0.16	chapter 1	Detail added to a future work item
October 17, 2017	J. Echterhoff	0.17	throughout	Incorporated editorial changes and feedback from reviewers

Date	Editor	Release	Primary clauses modified	Descriptions
October 19, 2017	C. Portele, J. Echterhoff	0.18	chapter 1, 7, 8, 9	Added text based on reviews, added sections to describe adaptations of workflows when processing a different NAS profile
November 2, 2017	C. Portele, J. Echterhoff	0.18	throughout	Incorporated editorial changes and feedback from Terry Idol
November 20, 2017	J. Echterhoff	0.19	throughout	Incorporated editorial changes and feedback from Carl Reed
November 23, 2017	C. Portele	0.20	throughout	Address comments from Carl Reed

Appendix D: Bibliography

[1] Echterhoff, J.: OGC Testbed-12 ShapeChange Engineering Report, http://docs.opengeospatial.org/per/16-020.html

[2] Reed, C. (Editor): Volume 1: OGC CDB Core Standard: Model and Physical Data Store Structure, Version 1.0, http://www.opengeospatial.org/standards/cdb

[3] Saeedi, S. (Editor): Volume 11: OGC CDB Core Standard Conceptual Model, Version 1.0, http://www.opengeospatial.org/standards/cdb

[4] NN: OGC CDB Core Standard, Feature Data Dictionary, Version 1.0, http://schemas.opengis.net/cdb/1.0/Metadata/Feature_Data_Dictionary.xml

[5] NN: OGC CDB Core Standard, CDB Attributes, Version 1.0, http://schemas.opengis.net/cdb/1.0/Metadata/CDB_Attributes.xml

[6] Kutzner, T.: CityGML UtilityNetwork ADE, https://github.com/TatjanaKutzner/CityGML-UtilityNetwork-ADE

[7] Gröger, Kolbe, Nagel, Ha fele (Editors): OGC City Geography Markup Language (CityGML) Encoding Standard, Version 2.0, http://www.opengeospatial.org/standards/citygml

[8] Saeedi, S. (Editor): OGC Testbed-13: CDB Engineering Report, Work in progress, https://portal.opengeospatial.org/wiki/pub/Testbed13/ConvertDocsT13Output/T13/NG001-CDB.html, to be published at http://docs.opengeospatial.org/per/17-042.html

[9] Portele, C.: OWS-6 UTDS-CityGML Implementation Profile Engineering Report, http://portal.opengeospatial.org/files/?artifact_id=34098