



DGIWG - 303
Geography Mark-up Language (GML)
Application Schema for the
Multinational Geospatial Co-production Program (MGCP)

Document Identifier:	TCR-DP-07-023-ed1.2.1-GML_Application_Schema_for_MGCP
Publication Date:	29 August 2007
Edition:	1.2.1
Edition Date:	08 October 2008
Responsible Party:	DGIWG
Audience:	Approved for public release
Abstract:	This study is about the structure of MGCP data as provided in the MGCP feature catalogue, and the application of GML to facilitate data sharing.
Copyright:	(C) Copyright DGIWG, some rights reserved - (CC) (By:) Attribution You are free: <ul style="list-style-type: none">- to copy, distribute, display, and perform/execute the work- to make derivative works- to make commercial use of the work Under the following conditions: <ul style="list-style-type: none">- (By:) Attribution. You must give the original author (DGIWG) credit.- For any reuse or distribution, you must make clear to others the license terms of this work. Any of these conditions can be waived if you get permission from the copyright holder DGIWG. Your fair use and other rights are in no way affected by the above. This is a human-readable summary of the Legal Code (the full license is available from Creative Commons < http://creativecommons.org/licenses/by/2.0/ >).

Executive Summary

A Geography Mark-up Language (GML) application schema was developed to facilitate the sharing of geospatial data generated under the Multinational Geospatial Co-production Program (MGCP) initiative. Military applications that use geospatial intelligence operate in a net-centric environment where geospatial data, products are exchanged using services between coalition nations. Interoperability of data, products and services is enabled through the agreement and adoption of standards by the coalition nations. Dominant standards for net-centric sharing include web-based services and XML-encoded data. The Geography Mark-up Language (GML) is a dialect of XML that supplies standardized data types for geospatial intelligence. Since GML is a generic language for all kinds of geospatial applications, the content and structure of the data to be encoded in GML must be defined in terms of a GML application schema.

A project was undertaken by the Digital Geospatial Information Working Group (DGIWG), funded by DEU and GBR, to develop a GML application schema for MGCP data. This schema can be used to harmonize the geospatial data that is created from variable capture techniques by coalition nations so that these data can be contributed to the International Geospatial Warehouse (IGW) for use in military applications. In addition, this schema will be of value to the NATO Core GIS Project for its access to MGCP data.

The use of GML leads to very large volumes of data, which may be problematic when these data are transmitted over networks; therefore, enhancements to GML were developed that MGCP could employ.

MGCP test data were provided by CAN, DEU, and USA in the form of Shape files and these data were converted into GML. A part of the test data is available in the DGIWG portal in the DATP A05 folder. The test data was used for investigations with respect to coordinate representation and data compression techniques. The results were analyzed in the light of 12 use cases that describe the MGCP requirements for GML.

The analysis of the test data produced statistics on the relationships of the data volumes of various data types. GML and Shape files have the same magnitude in terms of bytes and their compression behaviour is also similar. The data has a high proportion of coordinates, which does not lend itself well to stochastic compression. Whereas point feature data can be LZW-compressed by approximately factor 25, area feature data are only capable of a compression rate of approximately factor 5. The sparseness of binary coordinate strings does not make the introduction of differential coordinates a worthwhile effort.

Complex feature types are not an important current issue for MGCP; however, they likely will be in the future given the myriad data products that must be supported within the military community. Fortunately GML is able to support these structures.

The authors recommend that when GML is to be used for transactions in a web environment where relatively small amounts of data are exchanged; use GML in its conventional XML encoding with tag delimiting. For bulk-exchange, GML data should be fragmented into manageable units of 5-10 MByte. For distribution of GML encoded MGCP data in high volume over the web, apply a systemic compression through a normalized coordinate system and binary encoding using BXML in the short term and the ISO approach in the longer term, and a tag length delimiting mechanism and then apply a stochastic compression using either GZIP or BZIP.

The access of large volumes of data is difficult with the existing bandwidth constraints that confront most systems, regardless of data format; therefore, this problem is not created by

the additional overhead imposed by GML on the data. The benefits offered by GML for interoperability offsets this overhead. The best approach to deal with bandwidth constraints is to have servers ease the burden on clients by delivering the data only in small portions.

The MGCP GML application schema will need to be kept up-to-date with the latest ISO GML standard versions and with profiles produced by DGIWG and NATO. DGIWG is currently working on GML profiles that reflect the DGIWG / TSMAD profiles of the ISO 19107 spatial schema. For the requirements of different applications, these profiles provide different representation of geometry in 2D, 2.5D and 3D, with and without topology. The DGIWG GML profiles will apply the same restrictions to the GML schemas. The MGCP GML application schema that is currently based on the simple feature profile developed by OGC may then be replaced by a suitable DGIWG GML profile. In the DGIWG GML profile development process, the NATO profile is taken into account. It is anticipated that the majority of the DGIWG profiles will subset the NATO profile. The profile that replaces the simple feature profile currently used in the MGCP GML application schema will belong to the DGIWG profiles that are compatible with the NATO profile.

The authors recommend that the MGCP GML schema should be used to harmonize the geospatial data that gets captured from nations using different capturing techniques and to integrate this schema into the IGW. A plan for implementation of GML into MGCP should include the following tasks:

- Encourage the nations that are involved in the MGCP capturing process to participate in the testing procedure and comment officially, so that a final version is constructed through a consensus process. The adopted MGCP GML application schema shall be part of the MGCP product specification and shall be used as a reference for conformance testing.
- Establish maintenance procedures for the MGCP feature catalogue, MGCP GML application schema and GML profiles in order to adapt for changed requirements and to new releases of the base standards. There will be more than one DGIWG GML profile, so the appropriate profile for the application schema has to be chosen.
- Develop a standard operating procedure for GML-encoding of MGCP data as a guide for all member nations. This will include a comprehensive set of GML encoding rules.
- Coordinate closely with DGIWG for the development of standards (military profiles of international standards) for the data structures and service interfaces for MGCP data and services.
- Update the MGCP GML application schema to the latest GML version using the latest version of the feature catalogue based on the DGIWG GML profile for 2D geodata without explicit topology.
- Coordinate closely with NATO Core GIS for schema conformance, software development and testing by providing any results coming out of the DGIWG standardization process at an early stage to NATO and by taking NATO deliverables during the development into account.
- Develop and adopt common tools for GML encoding and compression.
- Adopt standard interface specifications for services for sharing geospatial data to realize the use cases in this document.
- Implement software for these geospatial services in the MGCP International Geospatial Warehouse (IGW) and encourage the implementation of software in MGCP member nations.
- Establish a repository in IGW for GML-encoded MGCP data possibly as a transactional Web Feature Service.
- Test out the transmission of the GML-encoded MGCP data against the various use cases in this document.

- Investigate the possibilities that GML provides for MGCP. The requirement for modelling the relationship between feature instances, complex features and explicit topology for MGCP data should be investigated.

This document represents a proposal for the encoding of MGCP using GML, and should be considered as a base input document for the development of formal specifications in cooperation through the DGIWG and MGCP communities. Further studies are recommended to investigate the compatibility of the MGCP application schema with the NATO core profile, and to test the interoperability of the MGCP GML schema and data with various software packages.

The study results shall be used for further investigations. Acceptance of this report as DGIWG technical report does not imply for the MGCP nations to change their operating procedure for MGCP production.



Contents

1	Acknowledgements	1
2	Introduction	1
3	Study Objective	1
4	MGCP Business Requirements for GML Schema	2
4.1	General Business Requirements	2
4.2	Use Cases.....	2
5	Technical Issues and Approaches.....	7
5.1	Features	7
5.2	Topology.....	9
5.3	Point coordinates.....	10
5.3.1	Precision of point coordinates with floating point representation	10
5.3.2	Absolute coordinate values	11
5.3.3	Coordinates in GML	11
5.3.4	Relative coordinate values	12
5.3.5	Conclusions about precision of point coordinates.....	12
5.4	Data compression.....	13
5.4.1	Reduction of overhead in exchange.....	13
5.4.2	Systemic compression	14
5.4.3	Stochastic compression	20
5.4.4	Binary XML	21
5.5	MGCP Compression Requirements for GML Encoding	23
6	GML Schemas.....	24
6.1	S-57 ENC GML application schema	24
6.2	NGA schema	25
7	Application Schema Development for MGCP	25
7.1	Deriving from feature catalogue.....	26
7.2	Specifications and their versions	26
7.3	Feature-level Metadata.....	27
7.4	Structure of the application schema.....	27
7.5	Design patterns	28
7.6	Naming conventions	29
7.7	Name space	29
8	MGCP Test Data	30
9	MGCP Data Conversion	32
10	Data and Schema Validation	33
10.1	MGCP GML application schema validation.....	34
10.2	Data validation against the MGCP GML application schema	34
11	Analysis of Test Data.....	35
11.1	Analysis results.....	35
11.2	Expected results in other solutions	38
11.3	Inefficiencies of GML	39
12	Open Issues	39
12.1	GML versions	39
12.2	DGIWG spatial schema profile.....	39
12.3	NATO GML schema profile.....	40
12.4	Multi-geometries	40
12.5	Metadata	40
12.6	Conformance testing.....	40
13	Recommendations.....	40
13.1	Recommendations regarding GML encoding of MGCP Data.....	40
13.2	Recommendations specific to MGCP and IGW operations	41

13.2.1	Recommended plan for MGCP implementation of GML.....	42
13.3	Recommendations specific to DGIWG.....	42
13.4	Recommendations for further studies	42
14	References.....	43
15	Bibliography	44
16	Annex A. Terms and Definitions	45
17	Annex B. GML.....	46
18	Annex C. Conversion scripts from MGCP feature catalogue to GML application schema.....	48
18.1	Script 1: transform_catalog v0.1.0.xsl	48
18.2	Script 2: sort_transformed_catalog v0.1.0.xsl	56



List of Figures

Figure 1. General Feature Model	8
Figure 2. Fixed Table Structure.....	14
Figure 3. Inline Tags Structure	15
Figure 4. Tag Length Structure	15
Figure 5. Header Directory Structure.....	16
Figure 6. Normalized Coordinate Space	17
Figure 7. Normalized Area for the Whole World	18
Figure 8. Binary Fraction.....	19
Figure 9. X,Y Interleaved Binary Fractions.....	19
Figure 10. Geometry and topology model from S-57 [Burggraf 2004].....	25
Figure 11: XSL transformation from feature catalogue to GML application schema.....	26
Figure 12: Principal structure of the MGCP GML application schema in UML notation .	28
Figure 13. Visualization of the content of one MGCP data cell	31
Figure 14. Extract from one MGCP data cell	32
Figure 15: Conversion steps from shape file to MGCP GML	33
Figure 16: Validation of application schema and of data	34
Figure 17. Distribution of data volumes by type for MixedPointLineAreaFeature for the average of the MGCP test data cells.....	35
Figure 18. Distribution of data volumes by type for AreaFeature for the average of the MGCP test data cells.	36
Figure 19. Distribution of data volumes by type for LineFeatures for the average of the MGCP test data cells.	36
Figure 20. Distribution of data volumes by type for PointFeatures for the average of the MGCP test data cells.	37
Figure 21. Comparison of file sizes for the different feature types in GML and Shape file for the uncompressed and the LZW-compressed case	38

Tables

Table 1. Cubewerx binary encoding results	23
Table 2. Metadata properties on feature-level	27
Table 3. Frequency of elements by geometric feature for one typical MGCP test data cell.	37
Table 4. Comparison of feature properties of type real	37
Table 5. Results of compression for each geometric feature for one typical MGCP test data cell.....	38

1 Acknowledgements

The authors wish to thank GBR and DEU for providing the funding for this study. The authors also wish to thank the MGCP community and specifically the MGCP members CAN, DEU and USA for the provision of test data for this study.

2 Introduction

Military applications that use geospatial intelligence operate in a net-centric environment where geospatial data, products and services are exchanged between coalition nations. Interoperability of data, products and services is enabled through the agreement and adoption of standards by the coalition nations. Dominant standards for net-centric sharing include web-based services and XML-encoded data. The Geography Mark-up Language (GML) is a dialect of XML that supplies standardized data types for geospatial intelligence. GML is an internationally-standardized implementation specification, recognized in ISO 19136 [Ref: R 18] and by the Open Geospatial Consortium (OGC).

Since GML is a generic language for all kinds of geospatial applications, the content and structure of the data to be encoded in GML must be defined in terms of a GML application schema.

The understanding and importance of the requirements for GML application schemas in the Multinational Geospatial Co-production Program (MGCP) initiative and the NATO Core GIS Project have increased recently to the point where an opportunity now exists to jointly develop a common GML application schema through collaboration between the Digital Geospatial Information Working Group (DGIWG) and these two parties. The NATO Core GIS Project is developing schemas based on DGIWG, ISO and OGC specifications as part of their Data Preparation Phase of the project. They also recognize the importance of being compatible with data products resulting from MGCP. A harmonization of the DGIWG and the NATO core GIS effort on GML application schema development for MGCP is needed.

The production of MGCP data as shapefiles has now started and this process will be applied at least until 2009. The use of GML may then possibly be considered, as long as it is endorsed by the MGCP member nations to address relevant use cases.

The use of GML leads to very large volumes of data, which may be problematic when these data are transmitted over networks. An OGC report [Ref: R 6] by CubeWerx states that "GML is an increasingly important feature encoding format for use in interoperable GIS systems. Unfortunately, the innate text encoding that it uses is both bulky and slow to process. To help GML realize its potential, its encoding efficiency should be made to be comparable with other commonly used feature encoding formats, such as ESRI® Shape file format." Nevertheless, the benefits offered by GML for interoperability offsets the burden of encoding overhead.

3 Study Objective

The objective of this study is to undertake an analysis of the requirements for GML encoding by MGCP and develop a GML application schema based on the MGCP feature catalogue. GML test data that instantiates the MGCP GML application schema was compared to the requirements. The test data was derived from shapefiles, the current format for MGCP data exchange. An analysis of the data in GML and shape format with respect to compressibility was carried out. Limitations of the shape format were not investigated in this study.

Also the possibilities that GML would provide for MGCP are not investigated in detail. Different exchange structures and exchange transactions although considered in the use cases are not the objective of this study. This study is about the structure of MGCP as provided in the MGCP feature catalogue. Services are not investigated.

4 MGCP Business Requirements for GML Schema

4.1 General Business Requirements

The business requirements are to enable MGCP data to be exchanged between MGCP member nations and to insert and withdraw data into and from the International Geospatial Warehouse (IGW).

The purpose of the IGW is to enable the exchange of MGCP data outside of the firewalls of national systems using the World Wide Web. The IGW will not be used as a mechanism for MGCP data distribution. Nations in the role of MGCP data producer will interact with the IGW by validation and uploading finished data products. Validation includes accuracy assessment, source suitability assessment, conformance to geometric and topological rules, compliance with DGIWG Feature Data Dictionary and Metadata specifications, and assurance that minimum levels of data content are met.

Nations, in the role of MGCP data consumer, will access the IGW by downloading these data in units of one-degree cells. Uploading and downloading of data by a nation will be controlled by incrementing credits for contributions and decrementing credits for usage.

4.2 Use Cases

The purpose of these use cases is to depict various situations where GML-encoded MGCP feature data will be sent between systems. It is assumed that the exchange of data within the MGCP community will occur both on a peer-to-peer basis and through IGW. At the present time, the IGW is a repository of Shape files of MGCP data. The use cases assume that IGW will have evolved to a state where it is a central warehouse of GML-encoded cell data with associated data schemas and metadata sufficient to support net-centric input and output of the cell data. Although the IGW provides the service of a central storage of MGCP data, it does not provide a service of brokering requests from consumers to producers.

The actors in the use cases are consumers and producers. MGCP member nations can play the role of consumer and producer and; therefore, be capable of exchanging information between each other on a peer-to-peer basis. This peer-to-peer exchange is described by Use Cases 1-4, 5, 7, and 9-11.

The IGW also plays the role of consumer and producer. There are three specific use cases that highlight IGW's activities:

- Use Case 4 "Feature retrieval in low volume from IGW over the Web" where IGW is in the role of producer.
- Use Case 6 "Feature retrieval in high volume from IGW over the Web" where IGW is in the role of producer.
- Use Case 8 "Transactions for updating the IGW – Push over the Web" where IGW is in the role of consumer.

Because this study report focuses on the behaviour of GML data exchange by low and high volume, by selected or bulk data, and by Web or media, use cases are identified for each relevant situation.

The use cases illustrate the potential usage of GML in different scenarios. The scenarios are not intended to prescribe how MGCP shall evolve. It is up to the MGCP nations to decide how they will handle future handling and distribution of MGCP data.

NOTE: Low volumes of geodata represent an amount of data that can be exchanged over dial up connection. This is e.g. useful for transactions of geodata.

NOTE: High volumes of geodata are considered in this study as bulk data, e.g. a whole MGCP cell.

Use Case 1. Get the capabilities of the producer system feature and service over the Web.

The consumer does not know the feature service capabilities of the producer and; therefore, must request these capabilities (i.e. WFS.getCapabilities) before conducting the retrieval. The producer returns an XML capabilities document that lists the feature-type names plus the description of the operations that are supported on each feature type.

Pre-conditions:

- The consumer is aware of the MGCP application schema and is able to handle geospatial data in MGCP structure.
- The consumer knows the source of the data and has access rights to this source.

Post-conditions:

- The consumer receives an XML capabilities document that lists the feature-type names plus the description of the Web features services. That is, required elements of a data product specification describing the data holding and required elements of the service specification describing the services available to retrieve the data holding are returned.

Use Case 2. Get the GML application schemas of the features from the producer over the Web

The consumer does not have knowledge of the GML application schema and; therefore, must request information on this schema (i.e. WFS.describeFeatureType) before conducting the retrieval. The consumer will have knowledge of the feature types supported by the producer. The consumer uses this information to request GML application schemas for specific features.

Pre-conditions:

- The consumer is aware of the MGCP application schema and is able to handle geospatial data in MGCP structure.
- The consumer knows the source of the data and has access rights to this source.
- The consumer knows the feature service capabilities of the producer.

Post-conditions:

- The consumer receives the GML application schemas for specific features. That is, the schema portion of a data product specification describing the data holding is returned.

Use Case 3. Feature retrieval in low volume for updating -- Pull over the Web.

The generic capabilities of this use case are described first. A consumer requires certain features to be updated in an MGCP product database that the consumer has already established. Because the requirement is for an update of an existing database, the consumer does not expect there to be high data volumes involved in the retrieval; therefore,

the consumer chooses to use a Web-based mechanism for the retrieval. The consumer has knowledge of the producer's feature capabilities and GML schemas for the features of interest and; therefore, is able to conduct the retrieval (i.e. WFS.getFeature). This knowledge could have been gained through consumer-initiated requests for information on the producer's service capabilities and GML application schemas, or the producer may have sent this information to the consumer as part of a notification that data updates were available. In the latter case, the consumer may have subscribed to the producer to receive these update notifications. The data retrieval involves the consumer sending a query to the producer for feature data. This query may select data based on spatial criteria that may define a single cell, an aggregation of cells, or subregions within a cell. The server returns GML data for the requested features. The consumer updates its database with the new feature data.

Pre-conditions:

- The consumer is aware of the MGCP application schema and is able to handle geospatial data in MGCP structure.
- The consumer knows the source of the data and has access rights to this source.
- The consumer does not expect high data volumes in the retrieval.
- The consumer knows that the data of interest has been updated by the MGCP producer.
- The consumer knows the service capabilities and GML applications schemas of the feature data to be requested
- The data elements have identifiers at a sufficient level to support the retrieval required by the consumer.
- Metadata exists to support the spatial criteria of the query.

Post-conditions:

- The consumer receives GML encoded feature data.

Use Case 4. Feature retrieval in low volume from a geoweb service

This use case is a specialization of Use Case 3, where a consumer retrieves a low volume of MGCP data directly from a geoweb service.

Pre-conditions:

- The consumer is aware of the MGCP application schema and is able to handle geospatial data in MGCP structure.
- The consumer has access rights to the service.
- The consumer knows the service capabilities of the service.
- The consumer anticipates a low volume of data from the service.
- The data elements have identifiers at a sufficient level to support the retrieval required by the consumer.
- Metadata exists to support the spatial criteria of the query.

Post-conditions:

- The consumer received a low volume of GML encoded feature data.

Use Case 5. Retrieval in high volume for sending bulk data over the Web

This use case is different from Use Case 3 in that it seeks to retrieve high volumes of data over the Web under circumstances where shipping physical media may be untimely and inefficient. The consumer has likely conducted an initial population of its MGCP database but needs a batch of MGCP cell data to add to its database or replace an existing out-dated batch. The realization of this use case will demonstrate how large volumes of GML-encoded data can be efficiently sent over the Web.

Pre-conditions:

- The consumer is aware of the MGCP application schema and is able to handle geospatial data in MGCP structure.
- The consumer knows the source of the data and has access rights to this source.
- The data elements have identifiers at a sufficient level to support the retrieval required by the consumer.
- Metadata exists to support the spatial criteria of the query.

Post-conditions:

- The consumer receives bulk GML-encoded MGCP data.

Use Case 6. Feature retrieval in high volume from IGW over the Web

The use case is a specialization of Use Case 5, which retrieves a high volume of MGCP data from the IGW.

Pre-conditions:

- The consumer is aware of the MGCP application schema and is able to handle geospatial data in MGCP structure.
- The consumer has access rights to the IGW.
- The consumer knows the service capabilities of the IGW.
- The consumer anticipates a high volume of data from IGW.
- The data elements have identifiers at a sufficient level to support the retrieval required by the consumer.
- Metadata exists to support the spatial criteria of the query.

Post-conditions:

- The consumer receives a high volume of GML encoded feature data from IGW.

Use Case 7. Transactions for updating -- Push over the Web

A producer updates a MGCP database and then needs to push these updates to other producers' MGCP databases, whose custodians have previously subscribed to receive these updates when they occur. These updates may consist of new data, replacement (updated) data, and deleted data (data that no longer exists in the update). The producer sends a transaction to the consumer (receiving producer). The transaction request depends on whether data is to be inserted (i.e. WFS.transaction.insert), updated (i.e. WFS.transaction.update) or deleted (i.e. WFS.transaction.delete).

Pre-conditions:

- The consumer (receiving producer) has subscribed to a producer to receive updated data after updates are created by the producer.
- The producer has *a priori* knowledge of the service capabilities, feature types and feature schemas at the consumer site.
- The producer does not expect high data volumes in the transaction.
- The data elements have identifiers at a sufficient level to support the transactions required by the consumer.

Post-conditions:

- The consumer receives a low volume of transaction requests, which include GML encoded feature data.

Use Case 8. Transactions for updating a central MGCP data warehouse – Push over the Web

This use case is a specialization of Use Case 7 where a producer pushes updates of MGCP data to a central warehouse using web services.

Pre-conditions:

- The producer has *a priori* knowledge of the service capabilities, feature types and feature schemas of the geoweb service.
- The data elements have identifiers at a sufficient level to support the transactions.

Post-conditions:

- The service receives transaction requests for cell data, which includes GML encoded feature data.
- The service will validate the data before committing the transactions to its database.

Use Case 9. Distribution in low volumes over the Web

A producer wishes to distribute a low volume of MGCP data to consumers over the Web.

Pre-conditions:

- The producer does not expect high data volumes in the distribution.
- Metadata exists to support the spatial criteria of the query.

Post-conditions:

- The consumer receives a low volume of GML encoded feature data over the Web.

Use Case 10. Distributions in high volumes over the Web

A producer wishes to distribute a high volume of MGCP data to consumers over the Web.

Pre-conditions:

- The producer expects high data volumes in the distribution.

Post-conditions:

- The consumer receives a high volume of GML encoded feature data over the Web.

Use Case 11. Distribution in high volume via optical or magnetic media

This use case will likely be used certainly as a means for a consumer to conduct an initial population of its MGCP product data. Large volumes of data are placed onto a media such as CD or DVD by the producer. The producer ships the media to a consumer. The consumer loads the data into its MGCP database.

Pre-conditions:

- None

Post-conditions:

- The consumer receives a high volume of GML encoded feature data on digital media.

Use Case 12. Access archived data

A consumer accesses legacy MGCP data that has been archived. This archived data is totally self-contained with all associated metadata and schema information because it cannot be reliant on the availability of this legacy information on networks. The access of this archived data will appear to the consumer to be the same as Use Cases 1 and 2 for

accessing information about the data and services, as Use Cases 3-6 for feature retrieval, and as Use Cases 9-11 for receiving data distributed from the producer (custodian of the archived data).

Pre-conditions:

- The MGCP data is archived with all associated metadata and schema information

Post-conditions:

- The consumer receives the requested data.

5 Technical Issues and Approaches

This section discusses the various issues that are anticipated when the use cases are to be realized in operational implementations. These discussions lead to recommended approaches for operational implementation.

5.1 Features

The representation of MGCP features in GML encoding is an important issue for consideration in operational implementation.

Geographic information described in accordance with the ISO TC211 19100 suite of geographic information standards is feature centric information. That is, the application schema is based on the general feature model described in the ISO standard 19109 Rules for Application Schema [Ref: R 13]. A feature is a representation of real world phenomena. A feature has properties. These include attributes, associations and operations.

The MGCP and the NATO Core GIS projects both are concerned with static data that does not include functions; however, in the future the model may include data on functions. An example of data that includes functions is hydrographic data that includes tidal information where the depth is a harmonic function driven by time.

Features have attributes. That is, a particular definition describes a type of a feature, and attributes further refine the definition so that each feature instance may be unique. One of the most important feature attributes is its position and geometry. This is described in accordance with the DGIWG spatial schema profile of the ISO 19107 spatial schema standard [Ref: R 10].

The MGCP data is relatively flat consisting of a large number of independent features. MGCP data is not organized into complex structures because of the size of the MGCP data collection effort and the fact that structuring features into complex hierarchies adds to the collection costs. The NATO Core GIS system includes data from a number of legacy sources and because these legacy sources were flat data structures, the NATO Core GIS consists also of a flat data structure for these legacy sources. Some special data does need to be structured. An example is a hydrographic buoy that needs to be linked to the light, top mark and other associated features. Feature-to-feature relationships, such as the aggregation of component features to a "master" feature, are rare. Another example is given by shared geometries in aeronautical flight information (DAFIF).

Feature instances may also be related spatially. For example, two line type features may be connected to form a composite line. In general, the topological relationships between the geometric elements related to feature instances can be calculated from those geometric elements. There are a few situations where these calculations are not possible thus requiring constraints to be identified. For example, the no left turn constraints in a highway network need to be carried either as constraints or as a topological directed line graph.

The following figure is an extract from the general feature model as described in ISO 19109.

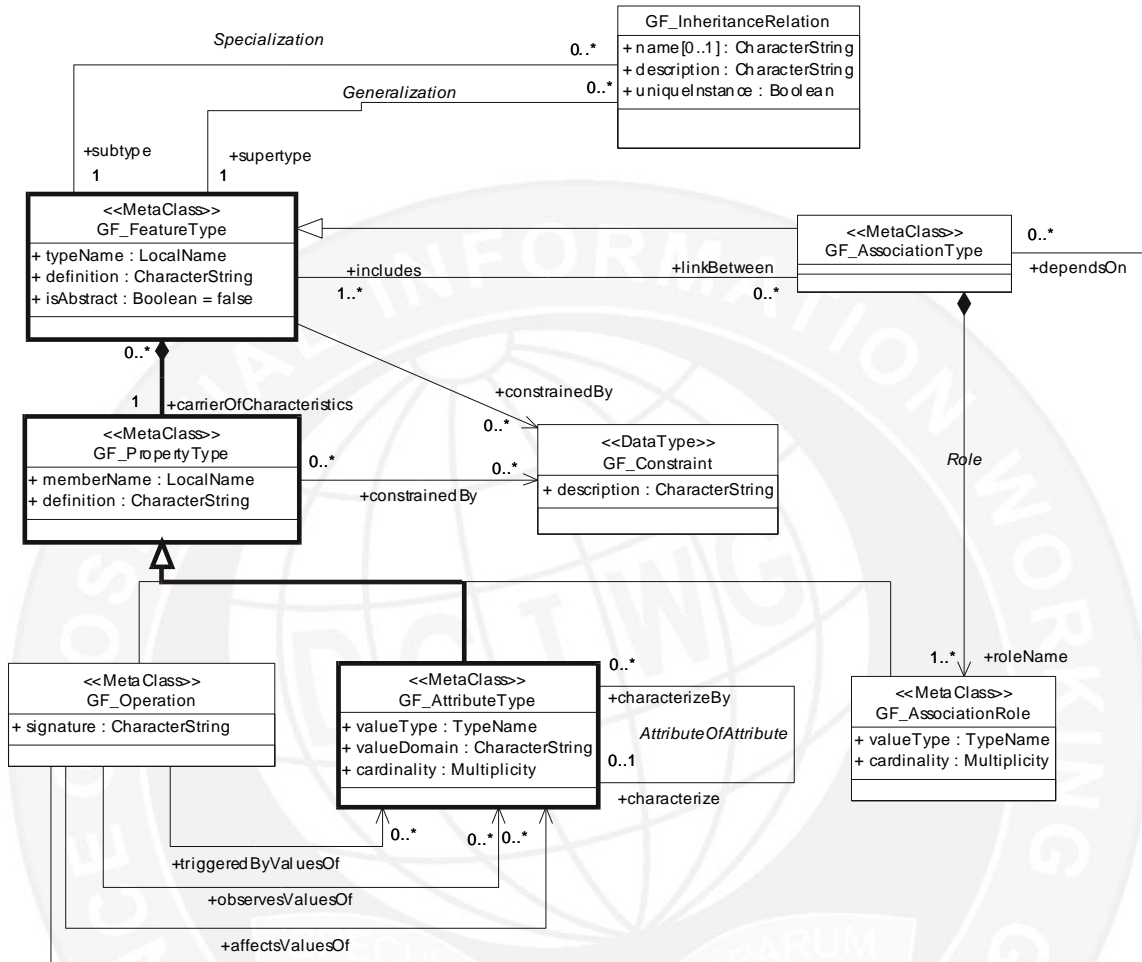


Figure 1. General Feature Model

The MGCP simply uses the Feature Type together with the Attribute Type including spatial attributes. More complex features can be described in extensions but are not part of the simple base model.

Both the MGCP and the NATO Core GIS application schemas make use of the concept of a feature catalogue as described in ISO 19110 Methodology for Feature Cataloguing [Ref: R 14]. A feature catalogue contains definitions and descriptions of the feature types, feature attributes, and feature associations occurring in one or more sets of geographic data, together with any feature operations that may be applied. The catalogue binds features and their attributes. A feature data dictionary contains only the definitions of the feature types, attribute types, enumerants, and association types in isolation. Definitions from the dictionary are combined to produce a feature catalogue for one or more sets of geographic data. The dictionary may also contain codes that may be used to identify the particular feature, attribute or enumerant definition. Multiple equivalent definitions may be held in the dictionary

in different languages. The other way of identifying definitions is by name; however, names may also change in different languages.

The DGIWG Feature Data Dictionary (DFDD) contains the definitions of feature types, attribute types, enumerant types, and associations. This dictionary is managed as a register in accordance with ISO 19135 Procedures for Registration of Geographical Information Items [Ref: R 17]. The register may be maintained continuously and new feature definitions may be added, or definitions may be superseded or indicated as obsolete. The register is managed by date, and nothing is ever deleted so that the status of the register at any particular date can always be determined. A feature catalogue for a particular data product references the data dictionary at a specific date, which is the reference date for the product specification. This makes the feature catalogue stable. It can be maintained, but that entails revising the references into the data dictionary.

The MGCP product specification references the DGIWG DFDD at a particular date. It references the DFDD by feature and attribute code. The situation for the NATO Core GIS is more complex because the Core GIS system holds a number of separate data products from separate sources, each with its own product specification. As part of the Data Preparation Phase of the Core GIS project, data from external sources will be converted into product specifications referencing the DFDD. However, if two products extracted from the Core GIS data base are to be fused, it is necessary to compare the definitions from the DFDD. This comparison can be easily done based on the registered date, but if a definition was changed between the issuance of two products, a mapping is necessary to translate between the old and new feature/attribute/enumerant definition.

The data dictionary allows the feature, attribute and enumerant definitions to be decoupled from the application schema. This indirect approach also allows for the support of multiple languages and the maintenance of the data dictionary over time.

GML requires explicit declaration of feature types and attribute types as elements; therefore, elements cannot be decoupled from the schema. That is, a GML application schema is equivalent to a product specification schema. The data dictionary concept can still exist in the background, but it is not directly represented in the GML application schema. The linkage is the set of feature/attribute codes. A GML application schema can reference the DFDD for example by code.

5.2 Topology

The ISO spatial schema standard ISO 19107 provides for a number of different ways to organize geospatial data. It defines both spatial and topological primitives and the range of allowed relationships between these primitives. Different primitives may be used to represent different types of data, and the relationships between primitives may be expressed or implicit.

Express topological relations may be included in a data set. The primary purpose of expressing such relationships is to assist in the direct use of the data. For example, a data set might include a road network. If the road network included both geometric primitives and topological primitives, then it would be easier for an application to be able to calculate a traversal through the road network. The secondary use of topologically structuring data is to ensure that the data is "clean", ensuring that lines connect and edges of adjacent polygon areas do not overlap producing numerous wedges and slivers, which can be very problematic when the data is used for anything other than display.

In the past some data products, such as VMap and other DIGEST Vector Product Formatted (VPF) data, included explicit topological relationships. These relationships ensured a level of

consistency in the data and allowed the data to be processed on older slow computers. The original VPF product, the Digital Chart of the World, was designed so that it could be directly processed on a 286 class 8 MHz personal computer. More modern geographic application software and Geographic Information Systems (GIS) tend to read in data with explicit topological relations and throw those relations away, generating their own structures. Except for certain special feature-to-feature relationships, topological relationships can always be calculated from the geometric primitives as long as the data is topologically-ready "clean" data. Topology should not be included in a data set as a quality assurance safeguard. Quality assurance should be done at the time that data is prepared, not as part of the structure of the data exchange. That is, data should be checked at data preparation time to resolve wedges, slivers, duplicate objects and other data capture problems.

There is another compelling reason for not having topological primitives included in data. In the past, data sets were monolithic objects. Each data set was stand-alone and it was important that it be self-consistent. With existing net-centric geographic services, data can be combined in many different ways. A geographic query through a service interface may select two objects from a geographic database that contains many additional objects. Two objects may be considered to be connected in one query, but not connected in another query. For example, a park may be adjacent to a river in one query related to open spaces, but have a fence intervening when barriers to transit are selected. Topology varies with the query used to select data; therefore, topology should be generated by the user from the "clean" geometry and not transmitted with the data.

There are three major types of geometric topology: connectivity, adjacency and containment. Connectivity is concerned with those geometric primitives that are connected – a relationship that is different in nature to adjacency (what is beside an object) and containment (whether an object is within, crosses or touches another object). Connectivity is important for networks such as transportation networks and occasionally needs to be calculated and included with data.

MGCP is collecting data in different DFDD-based feature classes. The features of each class are stored in a separate shapefile. However implicit topological relationships or constraints exist between some of the MGCP feature classes. These relationships are described in MGCP Semantic Information Model. The quality assurance process must test the data between Shape files to ensure that edges and relevant boundaries are coincident and that points and other objects are collocated. This type of testing ensures clean data.

Based on the use cases for MGCP data, there is no need to include topological primitives in the GML schema. Geometric primitives are sufficient. The GML expression of the data will lose the segmentation of the data in the separate layers resultant from the Shape files; therefore, the data must be pre-tested at data collection time to ensure that it is topologically consistent (i.e. geometrically clean).

5.3 Point coordinates

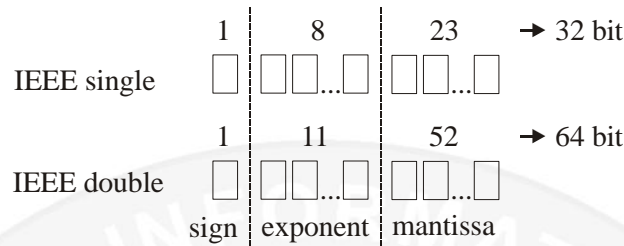
NOTE: This analysis was provided to MGCP early in the study. MGCP has made a choice based on this information.

5.3.1 Precision of point coordinates with floating point representation

With ellipsoidal coordinates (latitude, longitude) it is possible to address each point at the earth's surface in a unique way. The values for latitude range from -90° for the South Pole and $+90^\circ$ for the North Pole. The poles are the only exceptions for the unique representation, since the longitude gets arbitrary in the poles. The values for longitude range from -180° to $+180^\circ$. In a computer, the values for latitude and longitude are treated with a floating point

representation. Compilers provide floating point representations with a distinct number of bits for different precisions.

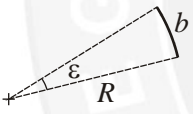
IEEE has standardized two representations named single and double precision. They differ in the range for the largest and smallest possible number and in the number of significant digits.



The significant digits are determined by the number of bits for the mantissa. For the single precision, the mantissa is represented by 23 bits. That means that this mantissa can represent 2^{23} or 8 388 608 different values. This determines the resolution of a single precision floating point number to 1:8,388,608.

5.3.2 Absolute coordinate values

The smallest distinguishable values for the latitude in a single precision floating point representation is approximately $\epsilon = 90^\circ / 8\,388\,608 = 10^{-5}^\circ$. An estimation of the minimal distinguishable distance on the earth surface with a radius of 6,380 km along a meridian and in East-West-direction for different latitudes is given by:

	$b_{lat, single} = 90^\circ / 2^{23} \cdot \pi / 180 \cdot R = 1.2\text{m}$ $b_{long, single} = 180^\circ \cos \varphi / 2^{23} \cdot \pi / 180 \cdot R = \begin{cases} 2.4\text{m} @ equator \\ 1.7\text{m} @ \varphi = 45^\circ \\ 1.2\text{m} @ \varphi = 60^\circ \end{cases}$
---	--

An estimation of the point location precisions for double precision floating point numbers can be obtained by the same method with a 52 bit mantissa.

single : $2^{23} = 8\,388\,608$

double : $2^{52} = 4\,503\,599\,627\,370\,496$

Ellipsoidal or geographic coordinates: latitude $-90^\circ < \varphi < 90^\circ$, longitude $-180^\circ < \lambda < 180^\circ$

$b_{lat, double} = 90^\circ / 2^{52} \cdot \pi / 180 \cdot R = 2.2 \cdot 10^{-9} \text{ m}$

$b_{long, double} = 180^\circ \cos \varphi / 2^{52} \cdot \pi / 180 \cdot R = \begin{cases} 4.4 \cdot 10^{-9} \text{ m} @ equator \\ 3.1 \cdot 10^{-9} \text{ m} @ \varphi = 45^\circ \\ 2.2 \cdot 10^{-9} \text{ m} @ \varphi = 60^\circ \end{cases}$

5.3.3 Coordinates in GML

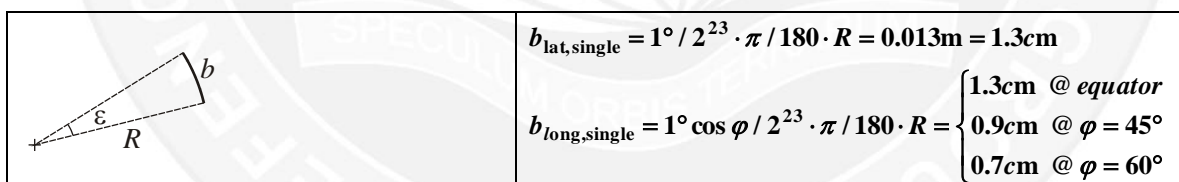
GML is using absolute coordinates and it is treating them as xs:double that is encoded in formatted text, i.e. a type cast takes place whenever data is encoded or decoded, since the

internal representation of xs:double is binary. With that respect, it is possible to store coordinates with an arbitrary number of digits and hence in an arbitrary precision. The integer and fractional parts of a decimal number are separated by a decimal separator. Since coordinates are stored as text, each digit and the decimal point and the delimiters are using one Byte each. In order to achieve millimetre accuracy for values of longitude along the equator, the decimal degree values are required to have 9 fractional decimals. Theoretically, the required number of Bytes for a textual representation of the latitude or longitude can be calculated by 1 Byte for the sign, 3 Bytes for the integer part in front of the separator, 1 Byte for the separator, and 8 for the fractional digits. That sums up to 13 Bytes. In GML implementations, the format usually uses the full amount of available digits, which is approximately 17 Bytes. This can be compared to a binary representation where each of the coordinate tuple values needs 4 Bytes for single precision floating point number in a less precise representation as explained above and 8 Bytes as double with a much higher precision. The textual representation of the number as in GML requires theoretically less than twice storage consumption than a binary representation, but in practise the coordinate values are stored with many more digits and, hence, ends up with much more storage volume than in a double binary representation.

5.3.4 Relative coordinate values

If only a certain region such as an MGCP cell is of interest, it is possible to operate with relative coordinates. These coordinates express the difference to a fixed coordinate tuple that can be the coordinates of one of the corners of the cell or rounded coordinate values, which makes the calculation of the difference very easy since in the latter case only leading digits have to be truncated. With relative coordinates and the knowledge of the origin where the relative coordinates refer to, it is possible to reconstruct the absolute coordinates with respect to a global coordinate reference system.

Relative coordinates in a $1^\circ \times 1^\circ$ cell require the 90^{th} or 180^{th} part of significant digits. That means theoretically for the above calculations that the achievable precision with relative coordinates in a $1^\circ \times 1^\circ$ cell increases by a factor of 90 or 180 respectively. The resolution of a point position on the earth's surface in single precision floating point representation will then be approximately 1 centimetre.



5.3.5 Conclusions about precision of point coordinates

With single precision floating point representation it is only possible to globally locate a feature on the earth's surface with a precision of 1 to 2 m. When changing to double precision the resolution is fine enough to separate the location of 2 adjacent water molecules in the Atlantic Ocean.

Relative coordinates bring a slight improvement for the precision but have the disadvantage of additional calculation whenever absolute coordinates are required. Also, relative coordinate values are not standardised procedures for data exchange methods like with GML, although SVG supports relative coordinate values. In order to support relative coordinates, the GML application schema does not have to be changed because the CRS in which the coordinates are provided are labelled with a CRS attribute in the GML data file.

NOTE: the use of relative coordinates in the MGCP GML schema would require a modification of MGCP technical reference documentation.

Double precision floating point numbers require twice as much storage space than single precision. This requirement pertains to main memory and for disk storage or file exchange when stored in binary format. When stored in ASCII, such as in GML, the number of digits can be regulated by the output format for floating point numbers, where the number of significant decimals can be fixed either by rounding or truncation. In practise, the number of decimals in a textual representation, such as in GML, are many more than the number of significant decimals. This situation leads to a much higher data volume.

5.4 Data compression

5.4.1 Reduction of overhead in exchange

A stand-alone product must carry all the information about the product including the application schema, all of which adds to the overhead of data in an exchange. One of the principal issues with any XML encoding and so also for GML is that it is inefficient with respect to file size. Numeric data is encoded in a character form and the data is delimited by tags. For example, the cost of an item might be encoded in an XML schema for price data as: `<cost currency="USD">243.77</cost>`. This data stream occupies 34 bytes of data, whereas the number itself could be represented in a four byte real number in a computer system. The advantage of XML is that it is easily interpretable by a human with minimum tools or simple software, while the packed binary real number is only understood within the computer program that is processing the data.

Information theorists use a measure called an "Erlang" to measure information density. The original use of this measure was in the telecommunications industry where it was used to assess the capability of analogue phone networks. A so-called phantom phone circuit that carries three signals on two pairs of phone wires measures 3/2 or 1.5 Erlangs. By playing analogue tricks, it is sometimes possible to have systems with an Erlang measure of greater than one. In the GML encoding example above, we only get an efficiency of 4/20 or 0.2 Erlangs. Complex schema in XML are much worse where the overhead greatly dominates the actual information exchanged. Of course the overhead also varies with the data. For example, the encoding `<cost> $1,843,679. 42 </cost>` is less efficient than our previous example just because the number is longer. There is both structural overhead related to how the data is organized and statistical overhead related to the characteristics of a particular instance of data. Both can be reduced with different, complementary techniques.

In using GML to express MGCP data, there will be a significant overhead introduced. This is balanced by the flexibility and broad level of implementation of GML; however, techniques to reduce the overhead will need to be used. An initial test¹ has shown that a direct conversion of some initial sample MGCP data into GML is larger than the data volume of expressing the same data in the ESRI Shape file format, and the ESRI Shape file format is itself not efficient.

A number of theoretical approaches for the reduction of overhead are described in the next three sections.

¹ This test was undertaken by Alan Ramanus of the Directorate of Intelligence Information Management, Canadian Department of National Defence. An accurate comparison of GML and Shape would require a larger volume of data than was used in the test.

5.4.2 Systemic compression

Compression techniques can be used to reduce data volumes, but this must be done in a lossless manner for geographic data. It is best to structure the data to eliminate unnecessary data to achieve high compressions, such as by filtering out extraneous data.

Eliminating unnecessary data is known as systemic compression. Data may be organized so that only the relevant data needs to be handled. Stochastic compression looks at the probability of the occurrence of a particular data pattern in a bit stream. Systemic compression should be applied first thereby eliminating unwanted data and then stochastic compression can be applied to reduce the already smaller data set. If something is eliminated, it does not need to be compressed.

References - The first and most obvious systemic data reduction is the use of references to eliminate the need to exchange the schema. The definition of features and other information that remains constant and can be known by both the sender and receiver. As already stated, a stand-alone product must carry all the information about the product including the application schema; however, the application schema can be included within a product by reference. The GML schema and all of the data types can be well known by both ends of a communication channel. In fact, this is a good reason for using GML. The GML standard ensures interoperability within the military geospatial community, whereas if one were to build a unique encoding, it would be necessary to ensure that each receiver had sufficient information to be able to understand the format. References to predefined elements such as feature definitions and other registered elements also eliminate the need to carry the description of these elements in the exchange.

Identifying data elements - In any data structure, it is necessary to be able to identify the individual data elements. There are several ways to do this. In a fixed structure (Figure 2. Fixed Table Structure), such as a table, the data can be known by its location in the table. The table structure does not need to be exchanged, but all the table cells, whether empty or populated, need to be exchanged. This technique is appropriate for small fixed data sets, such as the output of a sensor, and is not appropriate for general geographic information; however, such a structure can be very efficient when it is used.

entry 1
entry 2
entry 3
blank entry (no data)
entry 5

Figure 2. Fixed Table Structure

Because of the variability of geographic information, a variable structure is required. This means that data must in some ways be delimited so that the start and end of each variable entity can be identified. Three delimiting schemes are in common use: inline tags, tag length pairs and header directories.

Inline tags – XML, using its normal encoding rules, makes use of inline tags. Some special character or other flag is designated as the tag identifier. The entire data stream needs to be parsed to identify these tags and end tags marking the end of the domain of an element.

There also needs to be a mechanism to allow the tag identifier to be used in regular data when it is needed. In XML, the tag delimiter is the character "<" and a tag is structures as <tag> ... </tag>. The character "<" may be included in a data stream as < and the & character, which is also special, can be represented as &. There are other basic rules, but the foregoing is sufficient for the following illustration (

Figure 3. Inline Tags Structure). Inline tags are inefficient to manage and require that a receiving program examine every character in a data stream to identify the end of a tagged entity; however, they are easy to interpret.

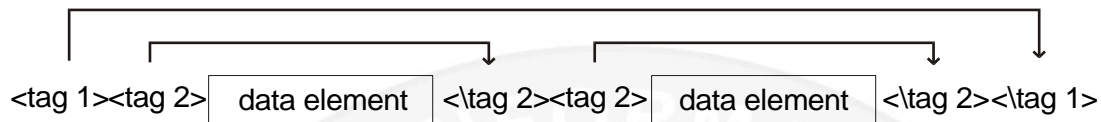


Figure 3. Inline Tags Structure

Tag length - A tag length structure (

Figure 4. Tag Length Structure) includes a delimiting tag that identifies an entity together with a length indicator, which describes the length of the entity. Such a structure requires no special characters and the data does not need to be fully scanned to identify each tagged element; however, the length variable on each tag length element is critical and the structure is fragile, because if a single tag length count is corrupted the entire data set may become unreadable. This situation means that additional redundancy may need to be added on top of the structure to protect or correct the data. Tag length delimiting is appropriate for binary data structures where the data elements and the tag lengths are all binary. Clever binary encoding tricks are often used to make the encoding of variable length tags easy and to handle the large tag lengths that may span the whole data set.

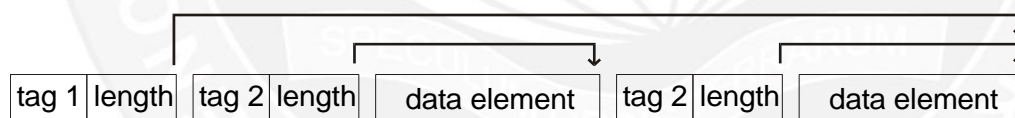


Figure 4. Tag Length Structure

Directory - A directory header-based delimiting scheme (Figure 5. Header Directory Structure) is the most efficient delimiting scheme that accommodates variable length and repeating data elements. A directory is included at the beginning of the data file with pointers into the data set based on data counts. The directory may be hierarchical and contain the implicit structure of the data. Of course, the more complex the directory, the more it becomes a data element itself with its own overhead. Directory header structures are easy to maintain, but they must be established before any data is exchanged. They are best suited to the bulk transfer of larger blocks of data.

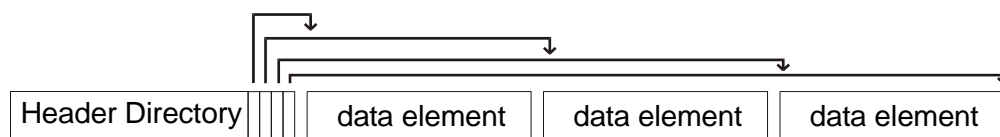


Figure 5. Header Directory Structure

Binary data representation - The representation of data elements in binary, rather than using textual representations, is one of the major systemic compressions that can be achieved. Binary can be used to represent certain data types as well as the delimiting structure.

At the basic level there are certain data types that need to be carried. These are:

Textual data - including text coded labels and identifiers, and text intended to be read by humans. Of course, text data needs to be text encoded. There are several standardized repertoires and encodings for text. For text that is intended to be machine readable, the repertoire should be restricted to ASCII (American Standard Code for Information Interchange), which corresponds to ISO standard 646 using the \$ as the monetary symbol. This is the basic computer alphabet used by virtually all computer systems. For lexical text that is intended to be read by humans, the ISO 10646 repertoire should be used and the encoding should be UTF-8, which allows for western European alphabet characters to be represented using one byte and extended character sets and iconographic characters to be represented using a standardized multi-byte string. Some database implementations prefer that all characters be represented by 2 or 4 bytes to facilitate searching, but for data exchange and most processing, UTF-8 is preferred.

Time and date - There are relatively few time data elements recorded in geographic data, so time can either be represented in a textual manner or in binary. The representation of time is described in ISO 19108 [Ref: R 12].

Numeric data types - Boolean, Integer and Real numbers. Since there may be a large number of numeric data types used in geographic information data sets, especially for attribute values, these data types should be represented as binary data. The real number standard is the IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754-1985) [Ref: R 8], and it is also known as IEC 60559:1989, Binary floating-point arithmetic for microprocessor systems. This standard provides for a "not a number" state so it can be used as padding in fixed field data structures to indicate that no data has been entered. Integer and Boolean numbers are more bit efficient and have no "not a number" state so they cannot be used in fixed field data structures for padding. In the past standards such as DIGEST [Ref: R 4] have resorted to non-standard tricks, such as using the maximum negative two's complement binary integer value (1111...1111) as a "not a number". Using non-standard numeric tricks creates incompatibilities and should be avoided.

Coordinate data - Coordinates may dominate a set of geographic information and the proper representation of coordinates can provide significant efficiency. In some encodings that are not tuned to represent geographic information, coordinates are handled as pairs of real numbers. If only a small range of numbers is valid, this is a potential waste of "code space" and; therefore, it is inefficient. For example, if a number could validly range from 0 to 10,000, then values such as 25,000 or 2.94×10^{43} would be illegal; however, the real number coding system would assign

potential bit combinations to handle these potential values. Since these values would never be encountered in an actual communication, the encoding of the real numbers would be larger than required to handle the actual data.

If a number has a fixed range, such as the 0 to 10,000 in our example, then it can always be normalized to a corresponding 0 to 1 range. All values in the range would be fractions of 0 to 1. This can be efficiently encoded as a binary fraction. A number in the range 0 (inclusive) to 1 (exclusive) is equivalent to the mantissa part of a real number, and can be termed a fraction number. In an actual implementation, there are always a fixed number of bits of precision assigned; therefore, a fraction number can easily be implemented by scaling the number up by the number of bits of precision assigned. That is, the decimal point (or more correctly binary point) can be shifted by multiplying by 2^n where n is the number of bits of precision available in the number field; therefore, the number can be represented as an integer number in a fixed-length binary field. Since integer numbers are encoded using fewer bits, this provides a significant improvement in efficiency for equivalent precision. Also, since the number range is a better fit to the value range of the data, it is sometimes possible to use short integers rather than long integers; therefore, a four-fold increase in the efficiency of encoding a particular real number can be achieved.

Coordinate data is normally handled as a Cartesian pair (or triple) of real numbers. Latitude and longitude are normally represented as degrees however projected coordinates such as northings and eastings in terms of meters from the equator and central meridian are at times communicated. Great efficiencies can be achieved by normalizing the coordinate data to the possible number range and carrying that as a pair of normalized binary fractions (Figure 6. Normalized Coordinate Space). For example, if we were interested in a map area with a minimum bounding rectangle of 63° to 62° longitude and 44° to 45° latitude, then we could represent that as a unit square from 0 to 1 and from 0 to 1. Note that both positive and negative normalized number ranges are permitted. Normalized unit areas are always square to maintain the aspect ratio of the projected data; therefore, if the map area of interest is not square, it is circumscribed within the minimum bounding square area.

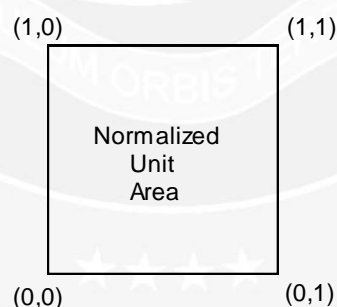


Figure 6. Normalized Coordinate Space

Normalized coordinates may be used to handle most of the spatial information in the body of the data exchange. This covers the vast majority of the spatial data elements in a data set. The specification of the “minimum bounding” in the metadata effectively defines the extent of the normalized coordinate space. In effect, all the coordinates are defined as fractions of the minimum bounding square.

For most coordinate-oriented data, such as point positions, the dimensions of the Minimum Bounding Rectangles around features, etc., Fraction numbers in a

normalized coordinate space are adequate; however, the curve geometric primitive is a special case. A curve carries a string of vertices as a spatial attribute defining the shape of the edge. Due to the nature of geographical data, each vertex tends to be "near" its neighbour in a mathematical sense; therefore, an additional efficiency can be achieved for strings of vertices. Three special types of coordinate representations may be defined for strings of vertices. These are **absolute**, **relative** and **differential** coordinate strings.

Absolute-coordinate-string(s) are sets of X,Y or X,Y,Z coordinates represented as pairs or triplets of numbers in normalized coordinate space. The number of bits of precision required must be defined. Absolute coordinates are fractions of the normalized whole world map represented as Lat/Long coordinates (Figure 7. Normalized Area for the Whole World). Note that 64 bits of precision (8 bytes), including both X and Y interleaved, are required to represent a position to 1 millimetre resolution (at the equator).

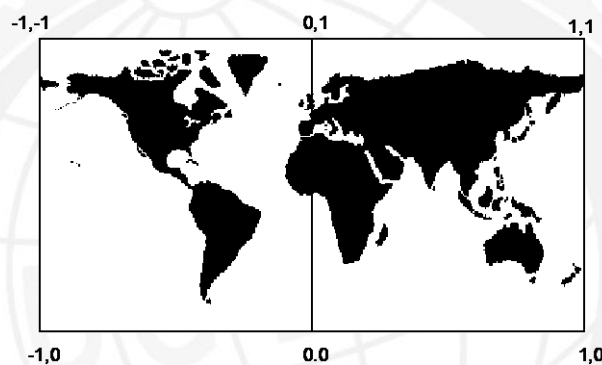


Figure 7. Normalized Area for the Whole World

Relative-Coordinate-String(s) are a set of X,Y coordinates specified as relative to a local coordinate system. The origin of the local coordinate system is defined as part of the metadata and may be a local coordinate system for a tile. Since the possible range of values in a local coordinate system is less than the full absolute coordinate system, it requires fewer bits to encode coordinates. A relative coordinate string would begin with a sub-string length indicator, which specifies the number of bits of each coordinate element. This is followed by binary numbers containing the bit packed coordinate information. All of these values are in terms of the normalized coordinate space. For example, a curve might consist of 299 X,Y vertices. This curve might be restricted to a local region that can be represented in 14 bits (7 bits for X and 7 bits for Y). The reference position and scale factor of the coordinate system would be set as part of the metadata. The relative coordinate string would consist of an integer number indicating that the following relative coordinate sub-strings are each 14 bits long, followed by a string of 4186 bits, packed into 524 bytes, representing the 299 vertices. Note that there are 6 remaining leftover bits in the octet string that would need to be padded with 0 and ignored.

Differential-Coordinate-String(s) provide an even higher level of compaction for that class of curves in which there is both a maximum and minimum vectorial displacement between vertices. If the edge is "well behaved" and varies only locally, then it can be represented very efficiently in terms of short vectors that are defined as relative to the preceding vector in a chain of such vectors. Actually, due to the manner in which geographic data is often gathered, this is a very common situation.

A differential coordinate defines a differential factor that indicates the increment step size. For example, if each relative coordinate displacement is a minimum of 3-bit increments, for the numerical precision used in the Normalized Unit Coordinate system, then the differential factor of 3 can be divided out of each of the coordinate positions. Also, if the maximum increment size is 3 bits, this would mean in the example used above, that each coordinate could be represented as 6 bits (3 bits for X and 3 bits for Y). Therefore, the string would consist of an initial relative or absolute coordinate position, followed by an integer number specifying a differential factor of 3, followed by an integer number indicating that the following differential coordinate sub-strings are each 6 bits long. This is followed by a string of 1794 bits, packed into 225 bytes, representing the 299 vertices. Note that there are 6 remaining leftover bits in the octet string that are padded with 0 and ignored.

Packed binary coordinates are stored as interleaved normalized binary fractions. That is, a number ranging from 0 (inclusive) to 1 (exclusive) may be stored as a binary fraction encoded in an integer field with the implied binary point at the most significant position. A single binary fraction is shown in

Figure 8. Binary Fraction and an interleaved normalised binary fraction, representing a coordinate, is illustrated in

Figure 9. X,Y Interleaved Binary Fractions.

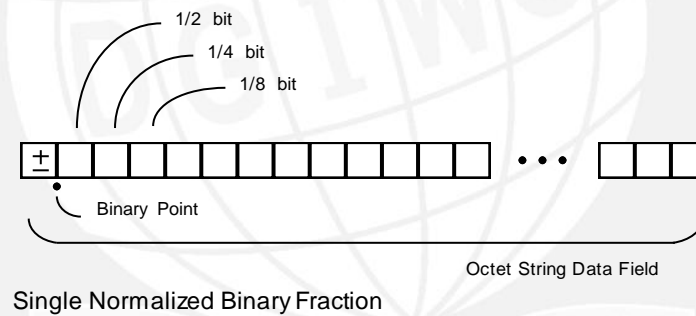
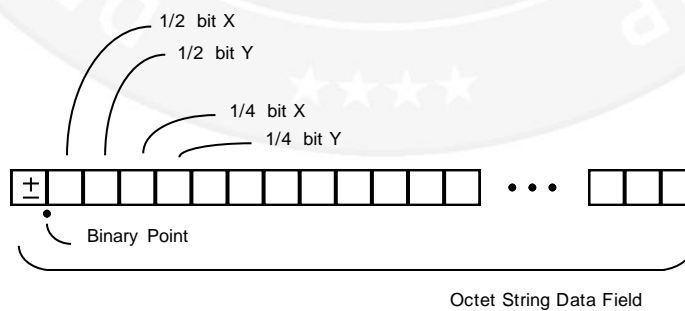


Figure 8. Binary Fraction



X,Y Coordinate as an Interleaved Normalized Binary Fraction

Figure 9. X,Y Interleaved Binary Fractions

Use case relationship All of the above delimiting structures can be used with GML. Data is translatable from one structure to another, and any of the delimiting structures can be applied to the same GML schema. Different data structures are appropriate to the various use cases identified earlier in this document. For small sets of data, such as those obtained through a Web Feature Service [Use Cases 3 and 4], it is best to make use of GML in its conventional XML encoding with tag delimiting. Efficiency is not the primary concern in this situation, but flexibility is important, and since such responses to web queries tend to be transmitted without extensive error correction, the ruggedness of character encoded tag delimited XML is important. In this case, the schema will be known by both ends of the data exchange.

For distribution in high volume over the web [Use Cases 5, 6 and 10], a binary encoding and a tag length delimiting mechanism is the best choice. This approach can significantly reduce the data volume, but some of the efficiency is lost because a data error detection and correction overhead must be introduced. Error detection and correction is a whole separate topic, but an overhead of 5% on a large data set can detect errors of 1 in 10^9 and correct errors of 1 in 10^6 .

For data archival applications on fixed storage media [Use Case 12], a directory structure is the best solution. The data volume can be so large that parsing the entire data holdings is impractical; however, for archival purposes the schema and the referenced information such as an extract from the referenced registries should be carried with the data. This material can be lost over time; therefore, for archival purposes the data media should be self-contained.

The bottleneck for the transmission of GML data is not the initial file size, since these data compress well. By using more sophisticated techniques that take the specific, highly redundant structure of XML into account, then even a higher compression and parsing rate could be achieved. With more sophisticated compression techniques, the amount of data can be reduced closer to Kolmogorov complexity ("absolute information"). GML-reading or transformation involves parsing of the document. This processing is, by some parsers, done via a DOM-object whereby the parser builds in memory a tree-structure with all GML information. This process is slow and memory-consuming. In fact, processing complexity is the real limit on using the tag-oriented encoding schemes. This is why a limit of file size is proposed. For very large static archival data, the directory structure can be a pre-defined parsing tree.

5.4.3 Stochastic compression

Geographic information is naturally variable. Some parts of the earth are relatively uniform whereas others contain a lot of detail. This means that any particular data set can have very different characteristics than another. For example, there may be a lot of curve primitives in a data set that covers a mountainous region due to the number of contours whereas another region containing large bodies of water might have very few curve primitives.

Stochastic compression is based on the statistical composition of a particular data set. There are many commercial tools that can be used to compress data. Some are inherent within specific encoding formats. This is especially true in image encoding formats where data volumes are great.

The amount of data compression can vary with the type and instance of data. A simple run-length encoding algorithm will on average result in a 2:1 compression, but on certain data sets, it can increase the volume of data. Compression algorithms with high compression ratios (sometimes up to 40:1) exist, but are tuned to specific types of data, such as

photographic colour images, or high definition television. Applying a compression algorithm to previously-compressed data often results in an expansion of the volume of data, especially if the first stage of compression was efficient; however, stochastic compression can be applied to data that has been thinned by systemic compression because the characteristics of the original data remain – all that has been done is the removal of meaningless data.

Stochastic compression is the responsibility of the encoding level of an exchange format. There are a number of commercial choices such as Huffman coding, Lempel-Ziv, or ZIP that are dependent on the probability of detecting patterns within textual data, web pages, etc... that are appropriate for use with vector based geographic information complying with a GML schema.

5.4.4 Binary XML

The inefficiency of character-coded XML is well recognized, but XML does not need to be character-encoded. The XML/GML schema describes the data elements and the relation between the data elements; however, different encodings may be used to carry the XML schema structured information.

The principal benefit of the basic character encoding for XML is that XML is easy to read by humans, text processing, and other simple data processing tools. XML read into some web browsers displays as a structured indented tag delimited set of data.

A binary encoding, including one using special data types such as those identified above, can be directly translated to and from a character-oriented XML encoding. There are a number of commercial and standards efforts that are developing a binary XML encoding, although none make use of the special data types identified above.

Galdos made the following statement about data compression with GML [Lake et al 2004 Ref: R 5]:

GML is a text-based language and, without compression, the data volumes can be substantial. Many GML-based applications will require gigabytes of disk storage. If you use data compression, you can expect significant compression ratios from 5:1 to 10:1. Data compression, however, is not always the best solution, given that extensive memory and CPU resources are still required to store large volumes of data on the client application. Coder-decoders (Codecs) offer a very promising solution to this problem. Although Codecs express the GML data in a binary format for transport that is similar to standard data-compression techniques, they also provide standard XML APIs to navigate the data. This provides an effective data-compression ratio of approx. 4:1 to 10:1 and significantly reduces the client application's processing and memory resource requirements. ExpWay's BinXML is one example of a product that uses this approach.

The W3C consortium describes a number of binary XML use cases [<http://www.w3.org/TR/xbc-use-cases/>]. One of the use cases is for X3D Graphic Model Compression, Serialization and Transmission, which is one that may closely resemble some of the requirements of MGCP. The relevant requirements in this use case are:

- Interoperability. The compressed binary encoding shall contain identical information to their encodings and support identical round-trip conversion between encodings.
- Multiple, separable data types.
- Processing Performance. The compressed binary encoding shall be easy and efficient to process in a runtime environment.

- Ease of Implementation. Binary compression algorithms shall be easy to implement.
- Streaming. Compressed binary encoding will operate in a variety of network-streaming environments.
- Compression. Compressed binary encoding algorithms will together enable effective compression of diverse data types.
- Security. Compressed binary encoding will optionally enable security, content protection, privacy preferences and metadata such as encryption, conditional access, and watermarking.

There is still significant debate in W3C and elsewhere with respect to the best approach for encoding XML in binary. The use cases have not been fully resolved. There is a trade-off between the speed in which the data can be parsed and the compaction of the data; however, the two aspects are not directly opposed. Tag length delimiting improves parsing speed and binary encoding improves compaction, but the tighter that data is encoded, the more processing is involved in decoding it.

An effort is ongoing jointly between ISO and the International Telegraphic Union ITU (The UN telecommunications treaty body) to establish a binary encoding for XML using the binary encoding rules of ISO 8825 [Ref: R 9] for the Abstract Syntax Notation. This joint effort represents XML-defined data using the value notation for ASN.1 types. XML Schemas may be mapped to ASN.1 modules and use ASN.1 standardized encoding rules such as DER (a canonical encoding that allows digital signatures and encryption) or PER (to very efficiently transmit data over a radio channel). [Ref: XML Cover Pages online resource by OASIS <<http://xml.coverpages.org>>]. The mapping from XML to ASN.1 encoding is the subject of ISO standard ISO/IEC 8825-5 ITU-T Rec X.694 "Mapping W3C XML Schema definitions into ASN.1. [Ref: R 16]. Studies in Telecom France have shown a compaction of 5:1 for XML text and a performance increase in parsing of 100:1. [<http://asn1.elibel.tm.fr>]

There are a number of efforts that the OGC [Ref: R 7] has undertaken to test the compactness and system throughput capacity of GML as encoded in numerous different ways, including binary encoding in BXML and BinXML™ and general compression formats such as GZIP and BZIP2. Two studies are of particular importance, one done by Galdos Inc, and another done by Cubewerx.

Bruce [Ref: R 1] of Cubewerx states that "The binary encoding used in BXML is greatly faster than XML encoding for processing general data and is enormously faster for dense numeric data. The uncompressed files are also significantly smaller, especially for numeric data, and are more efficient to compress and uncompress. A BXML file is also a direct stand-alone drop-in replacement for an XML file..." Their study looked at images, text documents such as Open Office documents and Web mapping Service capabilities and looked at XML, binary XML and then the results further compressed by a stochastic compression technique, GZIP and BZIP2. The following table summarises their results.

Table 1. Cubewerx binary encoding results

<i>File</i>	<i>XML</i>	<i>BXML</i>	<i>XML +GZIP</i>	<i>BXML +GZIP</i>	<i>XML +BZIP2</i>	<i>BXML +BZIP2</i>
WMS Capabilities	935 KB	521 KB	79.1 KB	76.6 KB	53.6 KB	56.4 KB
OpenOffice.org Doc	702 KB	427 KB	76.6 KB	75.5 KB	52.1 KB	57.6 KB

They conclude that "The uncompressed BXML representation is significantly smaller than the XML representation. This is to be expected, as just about every BXML structure is almost always smaller than the corresponding XML markup. ... The compressed results are similar in size. BZIP2 does a good job on XML data for a generic compression method."

Cubewerx and Galdos looked at GML in particular. Each used a different binary encoding of GML. Cubewerx compared efficiencies to BXML, a publicly available binary encoding format for XML developed by CubeWerx, whereas Galdos compared to BinXML™ a commercial product for which they are a reseller. Cubewerx then used a stochastic compression technique on top of the systemic binary conversion. They compared GZIP, which achieved a 5:1 compression ratio, and BZIP2, which achieved a 6:1 compression ratio for their sample. Cubewerx states that "“compaction” of the raw information before the “compression” step can have a large impact on overall system performance." They also compared ESRI Shape files, which are initially somewhat more compact than GML files but which contain less structure. The test data that they used was VMap0 data. While MGCP data will have different characteristics, the VMap0 data is more representative than the text documents and images used in some studies.

Galdos Inc. prepared a report as part of the OGC Web Services initiative [Ref: R 7] OWS 3 GML Investigations—Performance Experiment. They studied the retrieval time of GML features from a Web Feature Service (WFS) to a Web Feature Client including different methods of encoding and compression. The major purpose of the experiment was to address performance, but performance and data volume are closely linked. They found that GZIP compression offers only a small performance increase because the processing time required to perform the compression counterbalanced the benefits of a smaller file size. They also found that binary encoding with and without compression resulted in a performance loss because the binary encoding process was slow; however, this may just indicate that the "native" character-based XML encoding of GML is wrong and that the native form should be binary. The benefit of character coded XML is that it is easily readable, but machines prefer binary. The test data used by Galdos was also VMap0 data.

The study showed that there are different ways of compression for XML data. Binary XML is just one way of minimizing the data volume. Experiments with other compression techniques are carried out. For interoperability the compression has to be standardized and well known. None of the presented compression techniques for XML are mature enough to set the standard.

5.5 MGCP Compression Requirements for GML Encoding

Based on the work done by Cubewerx, it appears that it is desirable to apply both systemic compression and then stochastic compression to GML data for distribution of high volumes of data over the Web (Use Cases 5, 6 and 10).

The systemic compression should consist of encoding in binary. The BXML format seems preferable because it is an open specification; however, BXML does not appear to be widely adopted by industry. The work of ISO on 8825-4 seems to offer a standardized solution in the near future. The DER and PER encoding rules of ISO 8825-4 need future study because they offer even higher levels of compaction. The current recommendation is for MGCP to use BXML, but to watch the developments in ISO for a long term solution

Developing specialized numeric encodings for coordinates offer significant compaction by eliminating meaningless data. The use of normalized binary fractions seems beneficial, but it is not clear whether the complexity of converting to structures such as differential coordinates is worthwhile. The results have been investigated and are reported in Section 11.

The GZIP and BZIP2 stochastic compressions seem to work well with GML and binary GML data – which one is better for MGCP may depend upon the characteristics of the MGCP data; however, the difference seems small enough that other factors such as availability of tools that can work with GZIP and BZIP2 will dominate the decision of selecting a compression format.

6 GML Schemas

The MGCP GML application schema is not the first GML application schema that has been developed for the military community. Nonetheless, each application schema is different and fulfils different requirements. For the realization of the MGCP schema, the experiences in these projects as far as they were published or communicated by the developers in OGC or other meetings have been taken into account.

The comparison to GML application schemas of the legacy products S-57 [Burggraf 2004 Ref: R 2] and the NGA schema that represents Mission Specific Data Sets – like VMap usually provided in VPF for different levels of detail – is conducted in the following sections.

6.1 S-57 ENC GML application schema

S-57 feature classes are related to the marine environment and it serves mainly navigational purposes. This application leads to a fundamentally different way of modelling the geometric and topologic properties of the features. Since topology is required for routing in topological graphs, the features in S-57 GML are described by geometric and topological primitives (See Figure 10. Geometry and topology model from S-57 [Burggraf 2004]). Each topology primitive (Node, Edge) in the topology complex is realized by a geometry primitive (Point, LineString) using a backpointer (xlink:href reference) from topology to geometry. Topology was not a requirement for MGCP and, therefore, not implemented. Note that explicit topology will no longer be included in the new S-101 that will replace S-57 ENC.

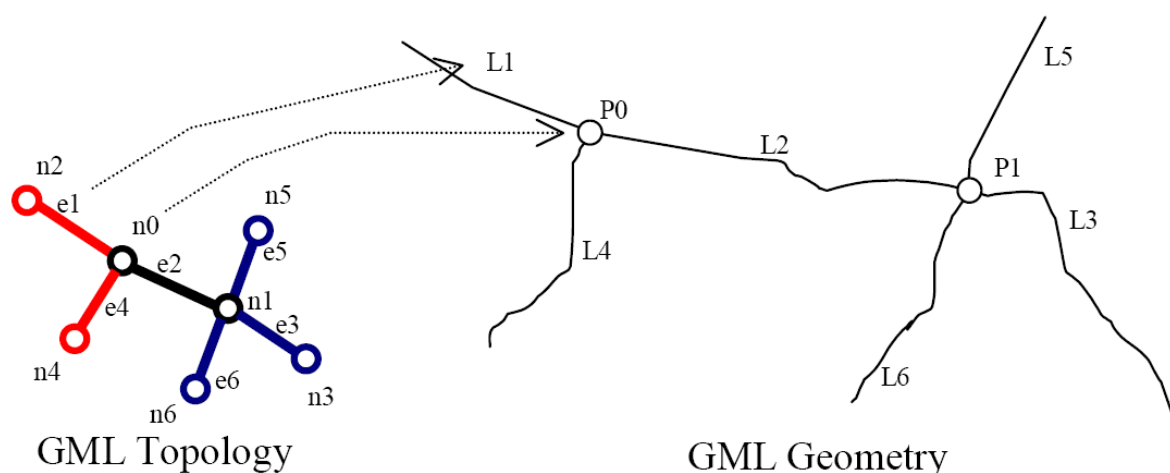


Figure 10. Geometry and topology model from S-57 [Burggraf 2004]

S-57 schema contains time series that are not necessary for MGCP.

The names of the feature classes in S-57 are given by a meaningful expression. The MGCP implementation uses the geometry character code (P, L, A) plus the 5-character DFDD code as feature class name. This usage can be shown by the following comparison.

S-57:

```
<element name="AirportAirfield" type="s57:AirportAirfieldType"
substitutionGroup="gml:_Feature">
```

MGCP:

```
<xs:element name="AGB005" type="MGCP:AGB005Type"
substitutionGroup="gml:_Feature"/>
```

The IHO is in the process of revising S-57 and is producing a new standard S-100 that is based on the ISO/TC211 suite of standards including the use of a register for feature types and a lessened dependence on topology in the data, and the future S-100 schema will probably look closer to that for MGCP.

GML allows for any name of the feature class. The short code or the full name can both be used for the feature class. In terms of human readability the long version is preferable.

6.2 NGA schema

Even if the NGA schemas are following a different approach in development, the resulting application schemas are structured with many similarities. Similar to MGCP, the NGA schemas are derived from a feature catalogue. The difference is that NGA derived the GML application schema from a UML model. The MGCP GML application schema is directly derived from the feature catalogue by an XSL transformation. This approach is possible and straight forward since the MGCP feature catalogue is already available as an XML document.

7 Application Schema Development for MGCP

All work with GML is based on an application schema, since it defines the structure, content and semantics of GML data files. A GML application schema is written in XML schema, a

formal XML language specified by W3C to express shared vocabularies and allow machines to carry out rules. It uses the data types provided by the GML specification and it has to adhere to the rules for application schemas as defined in ISO 19109 Geographic information - Rules for application schema.

7.1 Deriving from feature catalogue

The names and codes of the feature classes for MGCP and their attributes are defined by the MGCP feature catalogue. The catalogue is based on the DGIWG Feature Data Dictionary (DFDD). Since the features of MGCP must explicitly model relations among themselves, it is possible to derive the GML application schema directly from the feature catalogue.

To provide a source for the transformation from MGCP feature catalogue to GML application schema, the feature catalogue was encoded as an XML file. The version of the MGCP feature catalogue was still in a draft stage. The actual file used is MGCP_FeatureCatalogue_050721.xml

The MGCP feature catalogue was transformed using an XSLT transformation, which entailed the extraction of the feature classes and their attributes with proper attribute types and conversion into a GML application schema (See Figure 11: XSL transformation from feature catalogue to GML application schema). Extensible Stylesheet Language Transformations (XSLT) is a language for transforming XML documents into other XML documents. Since the GML application schema is an XML schema specified in XML, the source XML catalogue and the target GML application schema are specified in XML. For that reason, it is possible to use XSLT for the transformation. The script consists of two parts that are provided as Annex C. Conversion scripts from MGCP feature catalogue to GML application schema.

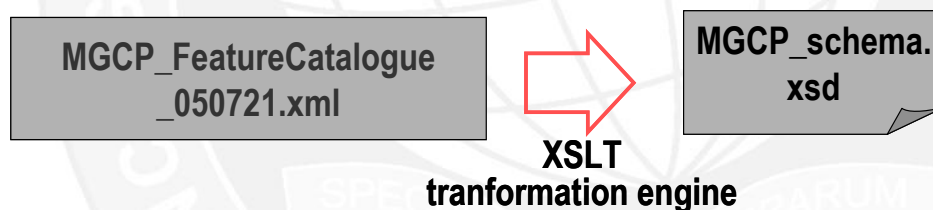


Figure 11: XSL transformation from feature catalogue to GML application schema

7.2 Specifications and their versions

The development of the MGCP GML application is based on the latest current version of GML, which is Version 3.1.1. It will be superseded by GML 3.2, which is identical to ISO 19136. At the time of the development, ISO 19136 is not yet finalized and, therefore, the official OGC implementation specification GML 3.1.1 was used. The OGC profile for simple features is also based on this version. The OGC profile for simple features is used for the schema development. The OGC profile for simple features consists of different levels. It can be shown that a Level 1 profile is sufficient for MGCP. The OGC simple feature profile of GML restricts the GML specification to features with simple geometries. Some of these restrictions are formally specified as XML schema in the document sfgml.xsd. This schema was used to verify the MGCP application schema. Further details on GML and the simple feature profile are provided in Annex B. GML.

NOTE: Areas with holes are included in the geometry types.

The MGCP GML application schema developed in this project was labelled as version 0.1.0 since it has not been approved by DGIWG and by MGCP. After a rigorous practical testing, it

should be revised and then published officially as version 1.0.0 of the application schema. It is recommended that interested nations involved in the MGCP capturing process should participate in the testing procedure and comment officially. The final version should then be released through a consensus process.

7.3 Feature-level Metadata

For the project, no other metadata than feature-level metadata was used. Metadata at a dataset-level should be encoded using the XML schema for metadata of ISO 19139 [Ref: R 15]. The feature-level metadata was modelled in the application schema as properties to the feature in the same way as it is provided by the MGCP feature catalogue. Every feature has the same metadata properties regardless of the feature class and of the geometric property. A list of the metadata properties is given in Table 2. Metadata properties on feature-level.

Table 2. Metadata properties on feature-level

Code	Metadata Property Name		Type	Unit
ACC	Accuracy Category	(M)	Enumeration	
ACE	Absolute Horizontal Accuracy	(M)	Real	Metre
ACE_EVAL	Absolute Horizontal Accuracy Evaluation Method	(M)	Enumeration	
ALE	Absolute Vertical Accuracy	(O)	Real	Metre
ALE_EVAL	Absolute Vertical Accuracy Evaluation Method	(O)	Enumeration	
CPYRT_NOTE	Commercial Copyright Notice	(O)	CharacterString	
SRC_DATE	Source Date and Time	(M)	CharacterString	
SRC_INFO	Source Description	(O)	CharacterString	
SRC_NAME	Source Type	(M)	Enumeration	
TIER_NOTE	Commercial Distribution Restriction	(O)	CharacterString	
TXT	Associated Text	(O)	CharacterString	
UPD_DATE	Review Source Date and Time	(M)	CharacterString	
UPD_INFO	Review Source Description	(O)	CharacterString	
UPD_NAME	Review Source Type	(M)	Enumeration	
ZVAL_TYPE	Vertical Source Category	(M)	Enumeration	

NOTE: UID was not part of the MGCP feature catalogue at that time.

7.4 Structure of the application schema

The MGCP GML application schema contains one feature collection and 225 feature classes. The feature collection is called "cell" and it is used as the root element for all MGCP GML datasets, thereby ensuring that all features in an instance document belong to this feature collection. The feature collection cell itself has the standard object properties of all GML objects. Metadata information on a cell-level could be referenced from this feature collection. The feature collection cell contains all feature classes of the MGCP feature catalogue. Point, line and area features are represented as separate feature classes. The GML feature name is composed of the geometric type and the feature code as it is used in the catalogue. For example, regarding AA010 for Extraction Mine, the full feature class name is

AAA010 ... A for area feature & AA010 DFDD code for Extraction Mine

The geometric type can take 3 different values:

- A ... area feature
- L ... line feature
- P ... point feature

In this version of the MGCP GML application schema, there are no other geometric types supported. Multi geometries, which are principally possible in the simple feature profile, are not part of the application schema. If features with multiple geometries are to be realized, they must be defined in a product specification.

The application schema is abstract. The non-instanciable, feature class called "MGCP_feature" carries all metadata properties described in the previous section. All the instanciable feature classes inherit the same metadata properties from this abstract class.

The structure of the GML application schema can be visualized in an UML notation.

Figure 12: Principal structure of the MGCP GML application schema in UML notation is only used to demonstrate the principal structure. UML was not used for developing the GML application schema.

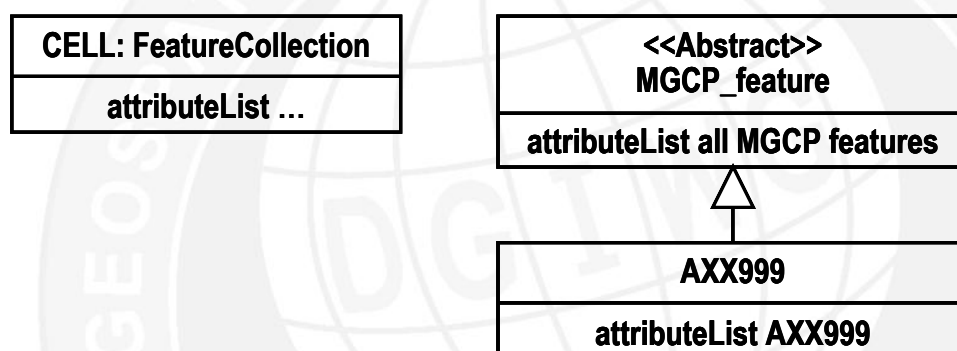


Figure 12: Principal structure of the MGCP GML application schema in UML notation

7.5 Design patterns

Many of the attribute properties to the feature classes are of type code list – only a defined list of attribute values is allowed. The list of allowed attribute values may change from feature class to feature class; therefore, it is necessary to define a separate list of possible values for each feature class. This definition has been done inline within the feature definition part of the feature properties.

There are attributes where their values have a unit. These entire attribute types are mapped to the type `measureType` that is provided by GML. This `measureType` requires for each occurrence of the attribute in the data file to specify a unit for the given attribute value. In GML, the numerical value of the attribute is of type `real`.

There is one exception where the unit from the feature catalogue is neglected. This exception is the attribute track or lane count with the attribute code LTN. Since only integer numbers of lanes or tracks are possible and since this attribute implies the unit for the attribute value, it was decided to use `integer` as the type for this attribute. The authors recommend to the DFDD maintenance team to change this also in DFDD and especially in the MGCP feature catalogue.

NOTE: The attribute type for LTN has been changed since the analysis has been done.

The handling of optional attributes is implemented in the MGCP GML application schema so that for an instance, MGCP GML document tags for the attribute can either be left out, or that they can be provided but with empty values. In the application schema, this is realized with XML attributes for the definition of an optional attribute element.

For optional feature properties where the tags may be missing in the GML instance file, then set the XML attribute `minOccurs="0"`. For optional feature properties where the tags are present in the GML instance file but without value, then set the XML attribute `nillable="true"` (required especially for enumeration feature property types)

7.6 Naming conventions

The ISO standards for geographical information ISO 19103 conceptual schema language [Ref: R 11] and ISO 19136 Geography Markup Language (GML) [Ref: R 18] stipulate naming conventions for the syntax and spelling of feature class names, attributes, operations, associations and parameters. They introduce camel case spelling of combined words. Camel case means that phrases of several words are added to one word with no spaces in between so that each word starts with a capital letter. The first letter of the whole word is either capitalized for UpperCamelCase or starts with a lower case letter for lowerCamelCase. With respect to ISO 19103, as written in its Section 6.10, feature attribute names are spelled as lowerCamelCase. ISO 19136 gives a strong recommendation to also spell feature attribute names as in ISO 19103 regulated in lowerCamelCase.

The feature attribute names in DFDD and, therefore, also in the MGCP feature catalogue contradict this naming convention. In the MGCP GML application schema, the feature attribute names are, therefore, implemented in lower case. This has been done also in analogy to the NGA application schema as developed in OWS-2, the Open Web Services OGC testbed Phase 2. Since the attribute names consist of the attribute code, not only the first letter of the name is spelled as lower case, but also the whole code (e.g. `ltn` instead of `LTN` or `ace_eval` instead of `ACE_EVAL` – 2nd example itself shows a slight deviation from the naming convention in which strictly the attribute name should be `aceEval`). The sample MGCP data files were all using the DFDD attribute codes in all capital letters. A direct implementation with those names in the application schema would contradict this naming convention.

It is recommended for DFDD to change all feature attribute codes from upper to lower case letters in order to make a direct mapping from the catalogue to the application schema.

In the application schema development, the upper case attribute codes had to be mapped to lower case attribute names. The XSLT script takes this into consideration. The case conversion had to be performed also during the GML encoding of the test data. This requires a customizing of the GML encoding, whereby the encoding is not generic any more, since additional information rules for the transformation have to be introduced. In the processing of the test data that was provided by different nations for the statistical analysis, the attribute name conversion required additional process steps. It was performed with an XSLT script. The experience showed that this process was time intensive (several minutes per megabyte of data) and it was very sensitive to file size. It was not possible to transform large datasets. The data had to be split into smaller units in order to perform the conversion.

7.7 Name space

According to the GML base specification, a name space is defined as a collection of names, identified by a URI reference, which are used in XML documents as element names and attribute names. A schema, according to the base specification, is defined as a collection of

schema components within the same target namespace. An application schema in this respect has to define its own name space.

In the MGCP GML application schema the target name space is defined by the URI <http://www.dgiwg.org/schemas/MGCP/0.1.0> with a prefix "MGCP", so that the whole namespace statement reads:

```
xmlns:MGCP="http://www.dgiwg.org/schemas/MGCP/0.1.0"
```

In the process of revision of the DGIWG web site, the directory schemas with a sub-directory MGCP and a second level sub-directory for the version numbering system should be established as schema location for application schemas. New versions or application schemas for other datasets can be located there under the same structure.

8 MGCP Test Data

The MGCP test data was delivered from the defence mapping agencies of USA, CAN and DEU as cells in the Shape file format – the current format for data exchange in MGCP.

The Shape file primitives are point, line, area, surface patches and multiple types.

The uncompressed data volume per MGCP cell ranged from 100 MB to more than 200 MB.

Figure 13. Visualization of the content of one MGCP data cell visualizes the high detail of content with an MGCP cell.

Figure 14. Extract from one MGCP data cell is an extract from this cell, which more clearly distinguish the geospatial features.

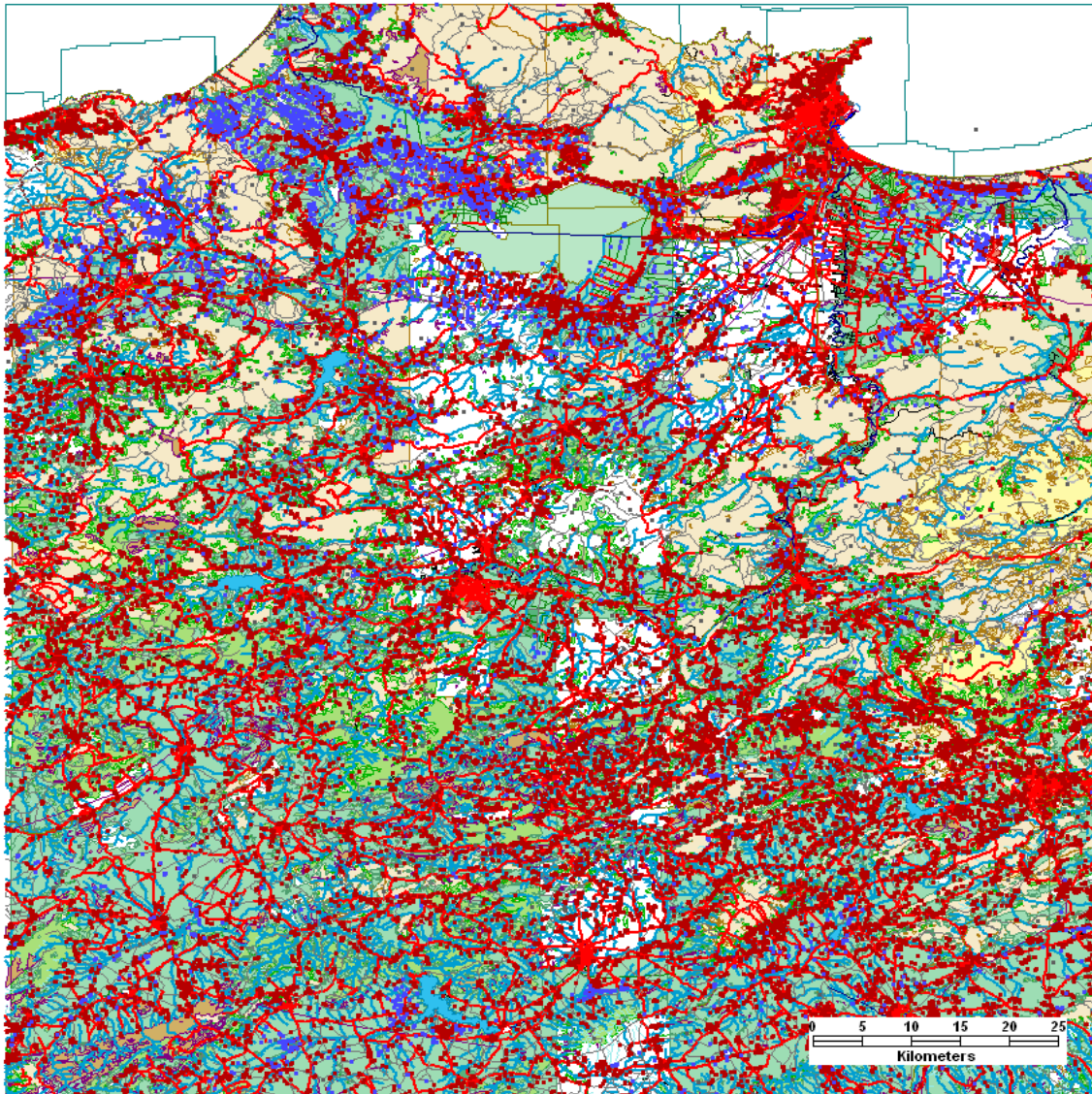


Figure 13. Visualization of the content of one MGCP data cell

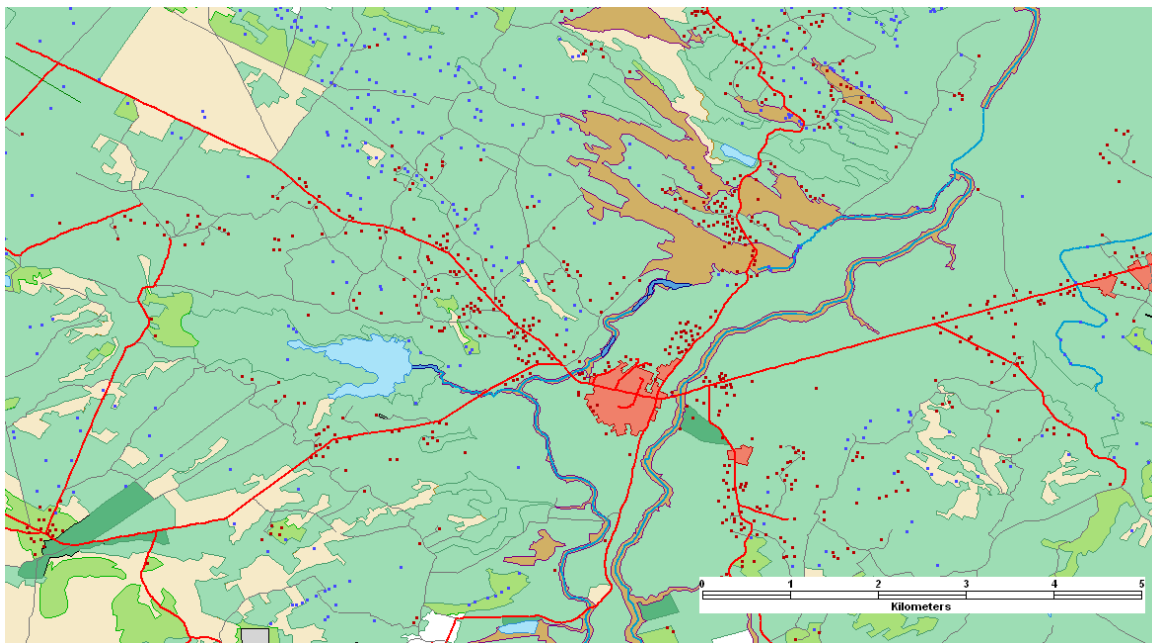


Figure 14. Extract from one MGCP data cell

9 MGCP Data Conversion

The conversion from Shape file to GML was performed in two steps. For Step 1, the commercial software, FME of Safe Software Inc., Canada was used. This format transformation tool produces GML data corresponding to the schema implied by the Shape data, which is not necessarily in conformance to the GML application schema as derived from the MGCP feature catalogue. Therefore, Step 2 is required to adjust the data in order to make it conformant to the application schema. If the data would be captured in the structure of the application schema, this additional step would not be required. Depending on the level of conformance, more or less adjustments were required. Step 2 was carried out by a XSL transformation. The XSLT scripts are dataset specific. With an application schema that is accepted by all MGCP contributing nations as delivered as part of this project and with capturing MGCP data adhering to this schema, Step 2 will be superfluous. This Step 2 is, as discussed in Section 7.6, also time consuming and relies on the amount of available computer memory. For that reason, the usage of a common application schema is required.

The steps used for the data conversion, in order to align the test data with the MGCP application schema, are shown in

Figure 15: Conversion steps from shape file to MGCP GML.

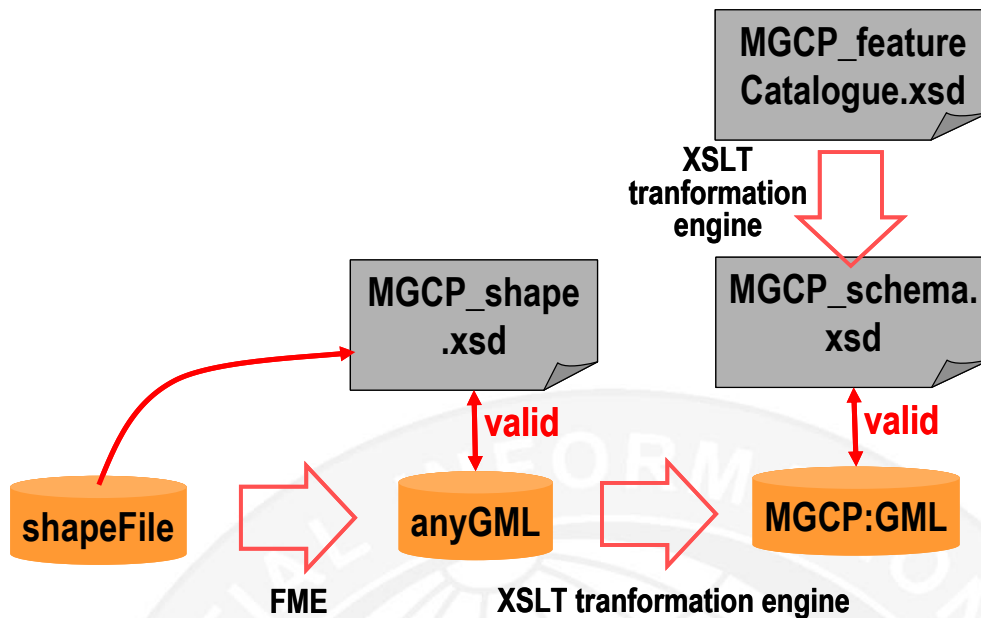


Figure 15: Conversion steps from shape file to MGCP GML

The MGCP datasets in GML form are used for statistical analysis in order to estimate file size and possible non-lossy data compression techniques.

10 Data and Schema Validation

This implementation claims conformance with the base standards. In order to claim conformance, the implementation has to pass conformance tests. This implementation utilizes the model-based testing approach.

The validity of the GML application schema is tested against the simple feature profile of the GML 3.1.1 specification. The validity of the MGCP datasets is tested against the GML application schema (See

Figure 16: Validation of application schema and of data).

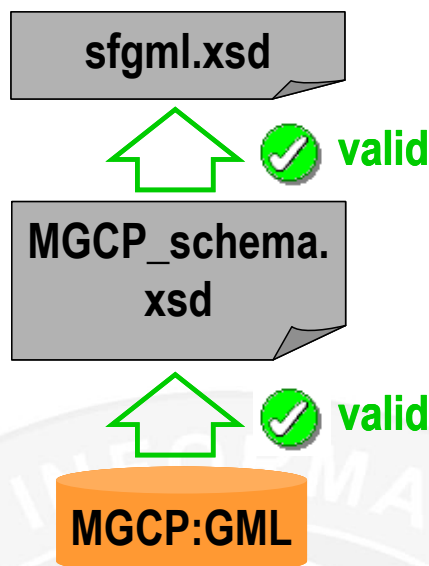


Figure 16: Validation of application schema and of data

The validation was performed using two integrated development environment tools for XML. The two products used are:

- Altova Professional XML Suite 2006 including XML Spy
- Stylus Studio 2006 Release 2 XML Enterprise Edition

The latter product was also used as translation engine for XSLT. It supports several XML validation engines. The validation engines to test the schemas and the data are listed in the next two sections.

10.1 MGCP GML application schema validation

The validity of the MGCP GML application schema was tested using a variety of conformance test tools. The validity was only tested against the simple feature profile since the GML 3.1.1 base specification itself has some problems with certain parsers. Due to these problems, the application schema test would result in error messages that are caused by the GML schemas. With respect to the profile – and this is the level of conformance that was the goal to achieve – the application schema lead to positive results using following XML parsers:

- xsv parser
- c++/parser
- msxml dom 4.0
- msxml sax 6.0
- .net xml parser
- XML Spy Professional 2006

10.2 Data validation against the MGCP GML application schema

The validity of the sample data was tested against the application schema before it was used for the statistical tests. In that respect, it was assured that the data is a valid MGCP GML dataset. The data passed tests performed by the following XML parsers:

- c++/parser
- msxml dom 4.0
- saxonica 8.6.1

- msxml sax 6.0
- .net xml parser
- Xerces-J 2.5.1
- XML Spy Professional 2006

11 Analysis of Test Data

11.1 Analysis results

The statistical analysis involved taking an average of the three test datasets received from DEU, USA and CAN. The analysis of the test data produced statistics on the relationships of the data volumes of four data types (AttributeValue, Geometry, FeatureID and XML-tag) for four types of features, MixedPointLineAreaFeature (Figure 17. Distribution of data volumes by type for MixedPointLineAreaFeature for the average of the MGCP test data cells.), AreaFeature (Figure 18. Distribution of data volumes by type for AreaFeature for the average of the MGCP test data cells.), LineFeatures (Figure 19. Distribution of data volumes by type for LineFeatures for the average of the MGCP test data cells.), and PointFeatures (Figure 20. Distribution of data volumes by type for PointFeatures for the average of the MGCP test data cells.). Further summary statistics on data volumes are shown in Table 3. Frequency of elements by geometric feature for one typical MGCP test data cell..

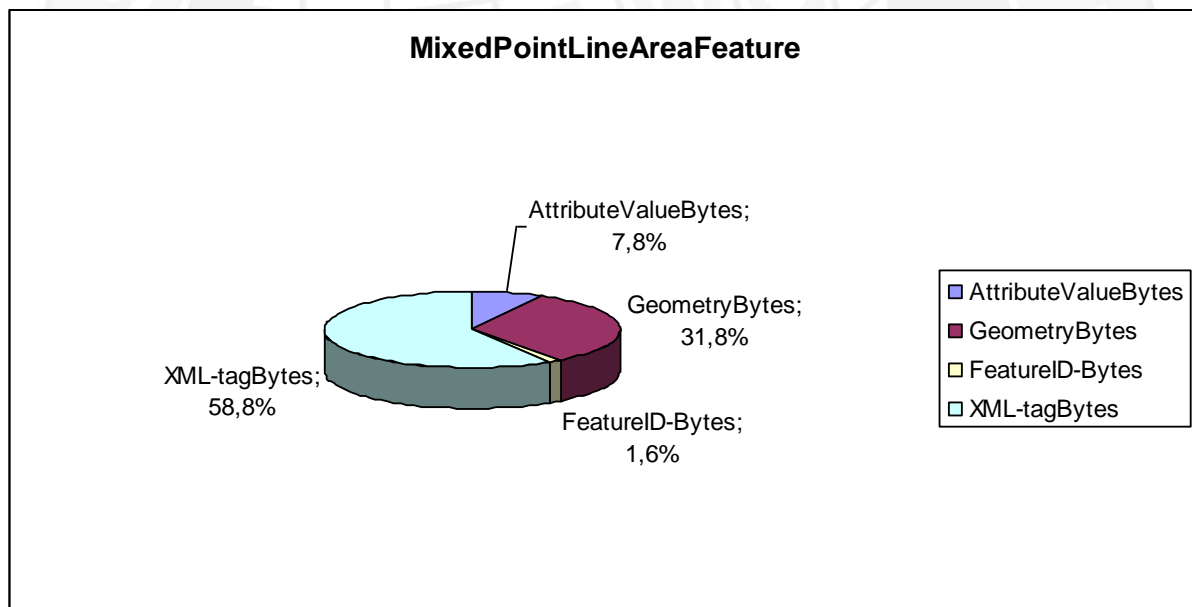


Figure 17. Distribution of data volumes by type for MixedPointLineAreaFeature for the average of the MGCP test data cells.

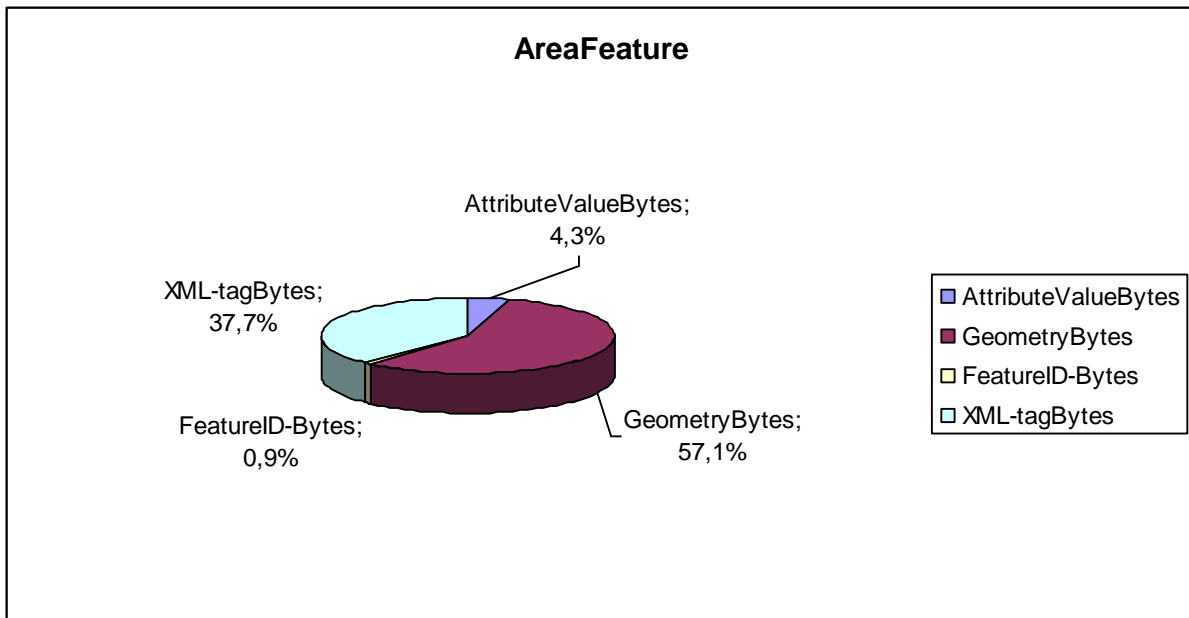


Figure 18. Distribution of data volumes by type for AreaFeature for the average of the MGCP test data cells.

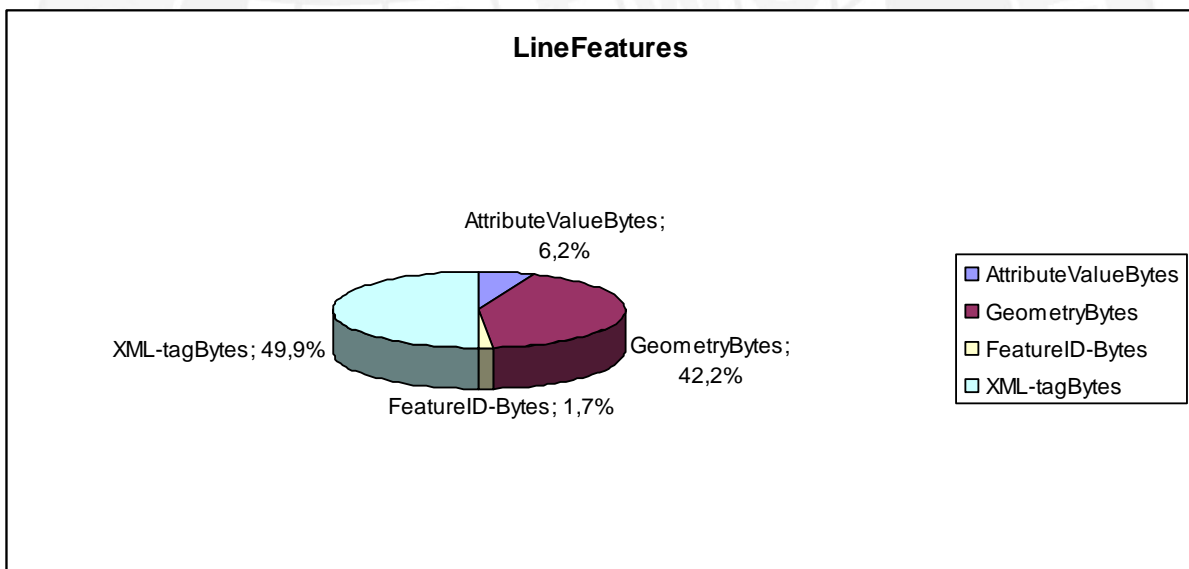


Figure 19. Distribution of data volumes by type for LineFeatures for the average of the MGCP test data cells.

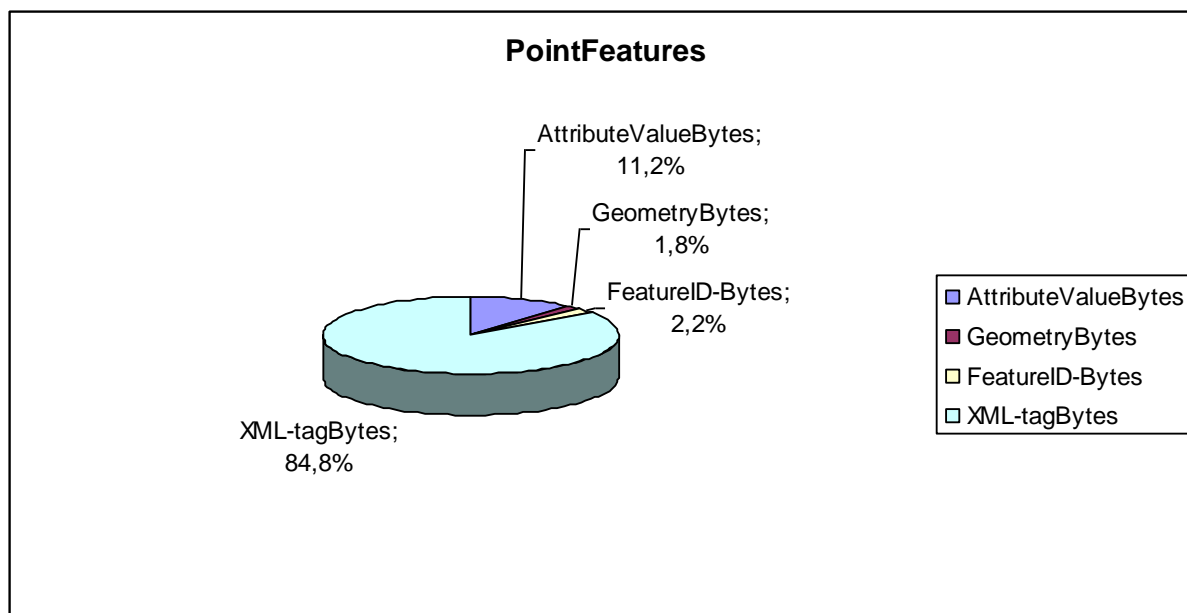


Figure 20. Distribution of data volumes by type for PointFeatures for the average of the MGCP test data cells.

Element	Features	Coordinates	Bytes
Points	27,166	27,166	42,061,284
Lines	13,219	262,869	23,918.796
Areas	10,181	433,018	30,763,221
Mixed Points, Lines and Areas	50,566	723,053	96,743,301

Table 3. Frequency of elements by geometric feature for one typical MGCP test data cell.

Normalizing numerical values as discussed in Section 5.4.2 can be only applied to values of type real. Real values occur only in a few attributes and in the coordinates. Table 4. Comparison of feature properties of type real gives the statistics for the number of occurrences in one typical MGCP test data cell. It shows that the vast majority of real values is on coordinate values. For attribute values, the real numbers are also only given with limited number of decimals.

	area	line	point
Number of real attribute values	20 555	8 795	54 332
Average bytes per attribute value	7.3	8.9	9.0
Number of coordinate tuples	425 566	253 805	27 166
Average bytes per coordinate tuple	35	35	35

Table 4. Comparison of feature properties of type real

The results of various types of compression on geometric features are shown in Table 5. Results of compression for each geometric feature for one typical MGCP test data cell. .

FileSize [Bytes]	uncompressed	LZW (*.zip)	%
------------------	--------------	-------------	---

Shape points	50 315 264	1 822 720	3.6%
Shape lines	28 028 928	3 862 528	13.8%
Shape areas	26 550 272	6 057 984	22.8%
GML points	42 061 284	1 527 808	3.6%
GML lines	23 918 796	3 534 848	14.8%
GML areas	30 763 221	5 775 360	18.8%

Table 5. Results of compression for each geometric feature for one typical MGCP test data cell.

The numbers in Table 5. Results of compression for each geometric feature for one typical MGCP test data cell. can more easily be interpreted when looking at Figure 21. Comparison of file sizes for the different feature types in GML and Shape file for the uncompressed and the LZW-compressed case

GML and Shape files have the same magnitude in terms of bytes and their compression behaviour is also similar. The data has a high proportion of coordinates, which does not lend itself well to stochastic compression. Whereas point feature data can be LZW-compressed by approximately factor 10, area feature data are only capable of a compression rate of approximately factor 5.

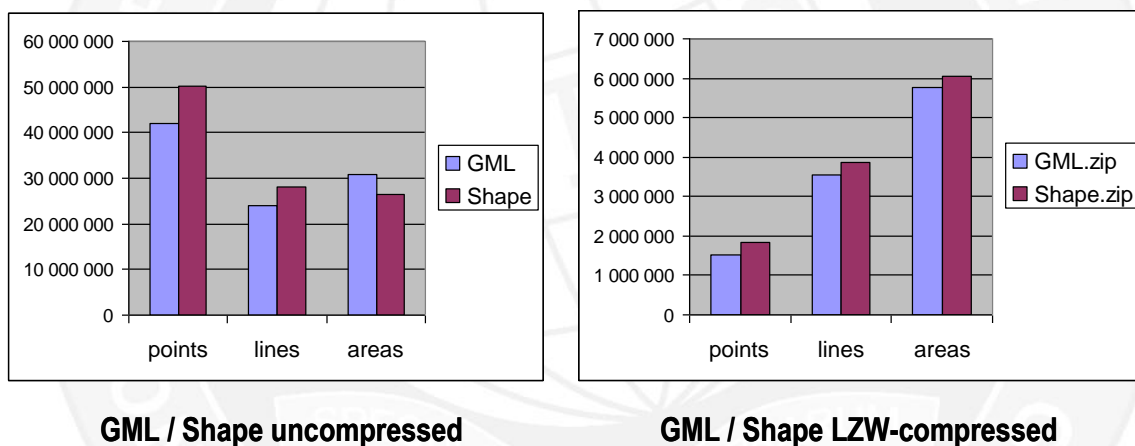


Figure 21. Comparison of file sizes for the different feature types in GML and Shape file for the uncompressed and the LZW-compressed case

From the charts and tables above, it can be concluded that the use of normalized binary fractions is appropriate; however, there are not enough binary coordinate strings to make the introduction of differential coordinates a worthwhile effort, as stated already in Section 5.5.

11.2 Expected results in other solutions

Different independent datasets from different production agencies were analyzed. It shows that MGCP data are dominated by XML tags. Most of the XML tags define feature attributes. Since many of the attribute values are predefined in the MGCP GML application schema with code lists, only the codes have to be stored with the data. That leads to short expressions for the actual attribute code with a considerable huge amount of metadata declaring the attribute property with a starting tag and an end tag. This kind of data can be compressed very effectively using techniques such as stochastic compression. A much larger portion of coordinate values in the data leads to a less efficient stochastic compression. The results are homogenous for the different data sets investigated. Variations occur with the area covered. The results may not be transferred to datasets with a considerably different application schema.

Database-oriented solutions such as SQL MM have not been investigated in this study.

11.3 Inefficiencies of GML

The GML application schema developed for MGCP is based on the simple feature profile. That means it supports only features with simple geometry. It also consists of a flat data structure and does not support complex nesting of features. With that respect, the application schema does not demand highly sophisticated methods for producing or interpreting GML data. Nonetheless, tools for handling XML data are demanding a huge amount of resources in terms of computer memory and processing capacities. The resource consumption increases with the file size of the GML data. Since computer memory is limited, it is not possible to process arbitrarily large data files. For example, in the approach that was chosen to transform the data from a row XML file to a GML data file that verifies with the MGCP GML application schema, it was not possible to process a whole 100MByte MGCP cell encoded as one GML file on a 1GByte memory machine using a process based on XSLT. While XSLT is a common way of processing XML data, there are other methods that are capable of handling much larger data files.

From this experience, it can be recommended that GML datasets should be limited to between 5 and 10 MByte due to the limitation of tools. Even then, depending on the type of processing a manipulation of the GML file may take in the order of several minutes meaning that conversion on-the-fly during querying will be unacceptable. The access of large volumes of data (Use cases 5, 6, 10 and in some cases also 11) is not suitable with GML for near real-time requirements. The real issue is thereby on the client-side of reading GML, not on the server-side. Measures need to be taken on server side to ease the burden for the client, such as by delivering the data only in small portions.

12 Open Issues

12.1 GML versions

The MGCP GML application schema developed in this project will be useful for nations in their further development of the MGCP distribution processes and it will improve harmonization in data production processes, since GML data can be validated with respect to the official MGCP GML application schema. Passing this validity test is a necessary condition for MGCP data to be in alignment with the product specification, but it is not a sufficient test, since different errors may still occur in the data that can not be detected by this test. This schema is based on the current officially available OGC implementation specification for GML in version 3.1.1, which has been provided to ISO/TC 211 to become an International Standard. In the specification process of ISO/TC 211, there had been comments from nations that do not participate in OGC. In the process of resolving these comments, the base specification had to be changed. This change improved the specification, but it also resulted in changes to the XML schemas that define GML. In other words, when GML will be published by OGC and ISO simultaneously as GML 3.2 and ISO 19136, the MGCP GML application schema may have to be adapted to the new release.

12.2 DGIWG spatial schema profile

The MGCP GML application schema is based on a subset of the GML 3.1.1 schemas. This subset is known as the "OGC simple feature profile of GML". DGIWG is currently also working on profiles of GML. These profiles shall reflect the DGIWG profiling efforts to subset the spatial schema standard into smaller units suitable for different kinds of applications. One of these profiles will be suitable for MGCP. Once these profiles are available, the MGCP GML application schema shall be adapted to the DGIWG profile.

12.3 NATO GML schema profile

During the development of the MGCP GML application schema, NATO has also developed a profile of the GML base schemas. It has not been investigated how this profile is related to the OGC simple feature profile and the other profiles on DGIWG's agenda. An investigation on the compatibility of the MGCP application schema with the NATO core profile is an open issue.

12.4 Multi-geometries

Shape file, the specification currently used for distribution of MGCP data, supports multi-geometries, i.e. features with more than one geometry property of the same type. MGCP will need to determine whether features with multipoint, multicurve or multisurface type geometry properties should be part of the scope. If multi-geometries should be supported, the MGCP application schema has to be extended to support this spatial data types.

12.5 Metadata

The MGCP application schema is currently not referencing any implementation specification for metadata. The only metadata it supports is the feature level metadata that is modelled as feature properties in the feature catalogue. These feature level metadata are mapped to the MGCP GML application schema as properties of the feature. The MGCP GML application schema shall reference the ISO 19115/19139 MGCP metadata profile for dataset level metadata.

12.6 Conformance testing

The MGCP GML application schema was developed under the guidance of experiences from other GML projects in the geospatial intelligence community, such as the NGA GML application schema for legacy geodata or the S-57 GML application schema for hydrographic data. The development has concentrated on conformance testing to be valid with respect to the relevant base specifications; however, interoperability of the schema and data with other software packages has not been tested.

13 Recommendations

13.1 Recommendations regarding GML encoding of MGCP Data

GML is well suited for transactions in a web environment where relatively small amounts of data are exchanged, such as data transmission that occurs with WFS. In this case, it is best to make use of GML in its conventional XML encoding with tag delimiting.

For bulk-exchange, GML data should be fragmented into manageable units. The maximum size of GML files should be limited to 5-10 MByte so they can be managed in computers with less memory. Dividing MGCP cells into smaller portions can be done by different criteria, either spatially by tiling the extent of features into smaller areas or thematically by splitting the content with respect to feature types or groups of feature types. For any dividing method, it has to be ensured that all features have unique feature identifiers and that missing features or duplicate features are avoided. A tiling by cutting along an artificial tile border and by clipping parts of the feature outside the tile shall be avoided.

NOTE: The upload and download to and from IGW as currently defined implies the transmission of large data sets. Cells can be similar to the shape files distributed into smaller units of GML data files.

For distribution of GML encoded MGCP data in high volume over the web, apply a systemic compression through binary encoding using BXML, and a tag length delimiting mechanism and then apply a stochastic compression using either GZIP or BZIP. The compression tools will dictate the format of choice.

For data archival applications on fixed storage media, a self-contained directory structure for MGCP data storage is the best solution.

There is no need to include topological primitives in the GML schema. Geometric primitives are sufficient. The GML expression of the data will lose the segmentation of the data in the separate layers resultant from the Shape files; therefore, the data must be pre-tested at data collection time to ensure that it is topologically consistent (i.e. geometrically clean).

The complexity of converting to structures such as differential coordinates and the sparseness of binary coordinate strings in MGCP data suggests that the employment of normalized binary fractions for MGCP is not worthwhile.

When GML 3.2 and its equivalent ISO 19136 standard is released, the MGCP GML application schema may have to be adapted to this new version. Similarly, once the DGIWG GML profiles are available, the MGCP GML application schema should adapt to this profile. DGIWG is currently working on GML profiles that reflect the DGIWG / TSMAD profiles of the ISO 19107 spatial schema. For the requirements of different applications, these profiles provide different representation of geometry in 2D, 2.5D and 3D, with and without topology. The DGIWG GML profiles will apply the same restrictions to the GML schemas. The MGCP GML application schema that is currently based on the simple feature profile developed by OGC, may then be replaced by a suitable DGIWG GML profile. In the DGIWG GML profile development process, the NATO profile is taken into account. It is anticipated that the majority of the DGIWG profiles will subset the NATO profile. The profile that replaces the simple feature profile currently used in the MGCP GML application schema will belong to the DGIWG profiles that are compatible with the NATO profile.

Once the implementation schema for metadata in ISO 19139 is finalized, the MGCP GML application schema should reference this metadata XML schema for data level metadata.

If multi-geometries should be supported, the MGCP application schema has to be extended to support this spatial data types.

Regarding quality management, the ability to validate GML data against the application schema can be used to improve the quality of the data and to reduce the effort on conformance testing.

13.2 Recommendations specific to MGCP and IGW operations

GML is deemed to be appropriate for IGW.

The IGW should evolve by adding standardized services, specifically WFS.

The access of large volumes of data is not suitable with GML for near real-time requirements. Measures need to be taken on server side to ease the burden for the client, such as by delivering the data only in small portions.

Access control services that comply with the OGC OWS security services should be added to the IGW, once the specifications exist and are proven to be mature.

Digital rights management should be added to support the MGCP data sharing model.

13.2.1 Recommended plan for MGCP implementation of GML

MGCP should set forth a plan for implementation of the GML applications schema. This plan should include the following tasks:

- Encourage the nations that are involved in the MGCP capturing process to participate in the testing procedure and comment officially, so that a final version is constructed through a consensus process. The adopted MGCP GML application schema shall be part of the MGCP product specification and shall be used as a reference for conformance testing.
- Establish maintenance procedures for the MGCP feature catalogue, MGCP GML application schema and GML profiles in order to adapt for changed requirements and to new releases of the base standards. There will be more than one DGIWG GML profile, so the appropriate profile for the application schema has to be chosen.
- Develop a standard operating procedure for GML-encoding of MGCP data as a guide for all member nations. This will include a comprehensive set of GML encoding rules.
- Coordinate closely with DGIWG for the development of standards (military profiles of international standards) for the data structures and service interfaces for MGCP data and services.
- Coordinate closely with NATO Core GIS for schema conformance, software development and testing by providing any results coming out of the DGIWG standardization process at an early stage to NATO and by taking NATO deliverables during the development into account.
- Develop and adopt common tools for GML encoding and compression
- Adopt standard interface specifications for services for sharing geospatial data to realize the use cases in this document.
- Implement software for these geospatial services in the IGW and encourage the implementation of software in MGCP member nations.
- Establish a repository in IGW for GML-encoded MGCP data.
- Test out the transmission of the GML-encoded MGCP data against the various use cases in this document.

13.3 Recommendations specific to DGIWG

The DFDD maintenance team should change the type for attribute track and lane count with the attribute code LTN to integer in DFDD and MGCP feature catalogue.

It is recommended for DFDD to change all feature attribute codes from upper to lower case letters in order to make a direct mapping from the catalogue to the application schema.

In the process of revision of the DGIWG web site, the directory schemas with a sub-directory MGCP and a second level sub-directory for the version numbering system should be established as schema location for application schemas. New versions of application schemas for other datasets can be located there under the same structure. The MGCP GML application schema should be posted onto the web site.

DGIWG and NATO Core GIS should develop a plan for joint development of a common GML application schema.

13.4 Recommendations for further studies

The following studies are recommended:

- Investigate the compatibility of the MGCP application schema with the NATO core profile.
- Test the interoperability of the MGCP GML schema and data with various software packages.

- Study the DER and PER encoding rules of ISO 8825-4 for their ability to offer higher levels of compaction

14 References

NOTE: At the time the project was conducted, the MGCP TRD documents in version 2 were not yet developed. The study results are based on the MGCP documentation available at that time. The references to the used documents is listed below.

- R 1 Bruce, C.S. Position Paper for Binary Interchange of XML, Cubewerx, Gatineau QC Canada [Ref: <http://www.w3.org/2003/08/binary-interchange-workshop/05-cubewerx-position-w3c-bxml.pdf>] 2003
- R 2 Burggraf, D. S. S-57 Schema and Related Tools Manual. S-57/GML Project. 2004
- R 3 TR2004-250-02 – V0.1 prepared for UKHO, 14 July 2004
- R 4 DGIWG. DIGEST edition 2.1 September 2000
- R 5 Lake, R., Burggraf, D.S, Trninic, M. and Rae, L.. Geography Mark-up Language (GML). John Wiley & Sons Ltd. 2004.
- R 6 OGC. GML Performance Investigation by CubeWerx. OGC 05-050 Version 1.0.0. Open Geospatial Consortium, 31 October 2005.
- R 7 OGC. OWS 3 GML Investigations—Performance Experiment. OGC 05-101. Open Geospatial Consortium
- R 8 IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754-1985)
- R 9 ISO 8825 Information Technology - ASN.1 Encoding Rules (a multi-part standard including several different encoding rules), 2002
- R 10 ISO 19107 Geographic information – Spatial Schema. 09 September 2002.
- R 11 ISO 19103 Geographic information – Conceptual Schema Language. Draft 01 September 2003.
- R 12 ISO 19108:2002 Geographic Information - Temporal Schema
- R 13 ISO 19109 Geographic information – Rules for Application Schema. 26 November 2003.
- R 14 ISO 19110 Geographic information – Methodology for Feature Cataloguing. 23 January 2004.
- R 15 ISO 19139 Geographic information – Metadata XML Schema Implementation. 30 June 2004.
- R 16 ISO standard ISO/IEC 8825-5 ITU-T Rec X.694 "Mapping W3C XML Schema definitions into ASN.1. [<http://www.itu.int/ITU-T/studygroups/com17/languages/X694.pdf>]. January 2004.

- R 17 ISO 19135 Geographic information – Procedures for Item Registration. 21 April 2004.
- R 18 ISO 19136 Geographic information – Geography Markup Language (GML). 15 April 2005.

15 Bibliography

- B 1 Boynings Consulting. Errors Found in the GML 3.1.0 XML Schemas. Boynings Consulting Inc. for UK Defence Geospatial Metadata Project. 14 February 2004.
- B 2 DGIWG. DGIWG/TSMAD spatial schema profile of the ISO 19107 spatial schema standard. Revised Draft Standard. 31 January 2005.
- B 3 Geo SE. MGCP Semantic Conceptual Model. GIS-Geo, Swedish Armed Forces, 15 June 2005.
- B 4 OGC. Binary-XML Encoding Specification. OGC 03-002r8 Version 0.0.8. Open Geospatial Consortium, 07 April 2003.
- B 5 OGC. OpenGIS® Geography Markup Language (GML) Implementation Specification, version 3.1.0. OGC 03105r1. Open Geospatial Consortium, February 2004.

16 Annex A. Terms and Definitions

Term	Definition
ASCII	American Standard Code for Information Interchange
Binary XML	Refers to any specification which attempts to encode an XML document in a binary data format, rather than plain text.
Codec	Encoder-decoder pair
DER Encoding	Distinguishing Encoding Rules, a canonical encoding that allows digital signatures and encryption
DFDD	DGIWG Feature Data Dictionary
DGIWG	Digital Geospatial Information Working Group
DIGEST	Digital Geographic Information Exchange Standard (legacy) [DGIWG 2000]
GML	Geography Markup Language
IGW	International Geospatial Warehouse
ISO	Organisation for International Standards
ITU	International Telegraphic Union
MGCP	Multinational Geospatial Co-production Program
OASIS	Organization for the Advancement of Structured Information Standards
OGC	Open Geospatial Consortium
PER Encoding	Packed Encoding Rules
Stochastic Compression	Compression using statistical techniques to reduce data.
Systemic Compression	Compression through elimination of unnecessary data.
URI	Universal Resource Identifier
W3C	World Wide Web Consortium
WFS	Web Feature Service
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

17 Annex B. GML

The Geography Markup Language (GML) is an OGC specification that defines data types using XML schema for geospatial information. GML can be used for encoding of geospatial information if accompanied by a GML application schema. A valid application schema has to be defined as XML schema utilizing the geospatial data types provided by GML and adhering to the rules defined in the GML specification.

The application schema defines the content and the structure of the geospatial data that has to be encoded. It contains a formal definition of the feature classes, their possible geometric primitives, their possible relationships and their attributes together with the attribute types and the possible attribute values including how often a certain property of a feature is allowed to occur in an instance file (cardinality). The instance files are the actual encoded geospatial data.

The development of the Geography Markup Language has started in 2000 with version 1 and has meanwhile reached version 3. Version 3 was published in 2002 and has undergone several releases with improvements and bug fixes. OGC is currently working in a joint effort with ISO/TC 211 at release 3.2, which will simultaneously be the International Standard 19136.

GML version 3 is a very comprehensive language that supports all kinds of encoding of geospatial phenomena. The following list gives an impression of the topics that are covered in detail on an implementation level for XML.

- Features and Feature Collections
- Spatial Geometries
- Spatial Topologies
- Time
- Coordinate Reference Systems
- Coverages
- Observations
- Dictionaries and Definitions
- Values

The spatial part is supporting a variety of different geometric elements in 2 and 3 dimensions. It also supports a long list of interpolation functions between vertices.

Due to the flexibility and to the variety of the GML specification, it is hardly possible to implement the specification in its entirety. Different applications have different requirements of the topics or elements needed. This will lead to implementations of different parts of the specification. Only the common parts can interact in an interoperable way. For that reason, it is required to state clearly which part of the specification is supported. The specified part of the GML specification is also called GML profile.

DGIWG has established a project team that defines GML profiles that support different military requirements. The project team on DGIWG GML profiles has the official project number A05 and is assigned to the Data Access Technical Panel (DATP).

Until A05 releases a profile that fulfills the requirements of MGCP, the officially released OGC profile for basic geometry, also called "simple feature profile", will be used.

The MGCP GML application schemas are held in separate files associated with this document. These schemas include the GML base schemas.



18 Annex C. Conversion scripts from MGCP feature catalogue to GML application schema

The code of the MGCP GML application schema is too bulky to be printed as an annex. It is provided in digital form. Instead the scripts that derive the schema from the feature catalogue are printed here. They are provided in digital form as well.

18.1 Script 1: transform_catalog v0.1.0.xsl

```

<?xml version="1.0" ?>
= <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fc="http://mgcp.ign.fr/xsd/fcXML"
  xmlns:sc="http://eden.ign.fr/xsd/metafor/20040507/scXML"
  xmlns:dgiwg="http://mgcp.ign.fr/xsd/dgiwg" exclude-result-prefixes="dgiwg sc fc">
<xsl:output method="xml" encoding="UTF-8" />
- <!--
metadata for stylesheet transform_catalog.xsl
version 0.1.0
produced by dotGIS, Dr. Gerhard Joos
2006-04-20
input: MGCP_FeatureCatalogue_050721.xml
-->
- <!-- Root Template -->
= <xsl:template match="/">
= <xsl:element name="xs:schema" namespace="http://www.w3.org/2001/XMLSchema" use-
  attribute-sets="schema_element">
- <!-- element xs:annotation -->
= <xsl:element name="xs:annotation">
= <xsl:element name="xs:appinfo" use-attribute-sets="appinfo_element">
<xsl:element name="gmlsf:ConformanceLevel">1</xsl:element>
<xsl:element
  name="gmlsf:GMLProfileSchema">http://schemas.opengis.net/gml/3.1.1/profiles/gmls
  fProfile/1.0.0/gmlsf.xsd</xsl:element>
</xsl:element>
<xsl:element name="xs:documentation">MGCP GML Application Schema Version
  0.0.1</xsl:element>
</xsl:element>
- <!-- element xs:import -->
<xsl:element name="xs:import" use-attribute-sets="import_element" />
- <!-- FeatureCollection declaration -->
- <!-- element xs:element name=CELL type=FeatureCollectionNameType -->
<xsl:element name="xs:element" use-attribute-sets="featureCollection_element" />
- <!-- xs:complexType FeatureCollectionNameType -->
= <xsl:element name="xs:complexType" use-attribute-sets="featureCollection_complexType">
= <xsl:element name="xs:complexContent">
= <xsl:element name="xs:extension" use-attribute-sets="featureCollection_extension">
= <xsl:element name="xs:sequence" use-attribute-sets="featureCollection_sequence">
= <xsl:element name="xs:element" use-attribute-sets="featureCollection_featureMember">
= <xsl:element name="xs:complexType">
= <xsl:element name="xs:sequence">
<xsl:element name="xs:element" use-attribute-sets="featureCollection_gmlFeature" />
  </xsl:element>
  </xsl:element>
  </xsl:element>
  </xsl:element>
  </xsl:element>
  </xsl:element>

```

```

    </xsl:element>
  </xsl:element>
<xsl:for-each select="fc:FC_FeatureCatalogue/fc:featureType/fc:FC_FeatureType">
<xsl:choose>
  - <!-- AbstractFeature declaration -->
<xsl:when test="fc:isAbstract/sc:Boolean = 'true'">
  <xsl:element name="xs:element" use-attribute-sets="abstractFeature_element" />
  <xsl:element name="xs:complexType" use-attribute-sets="abstractFeature_complexType">
  <xsl:element name="xs:complexContent">
  <xsl:element name="xs:extension" use-attribute-sets="abstractFeature_restriction">
  <xsl:call-template name="getAttributes" />
    </xsl:element>
    </xsl:element>
    </xsl:element>
  </xsl:when>
  - <!-- Instantiable Feature Declaration -->
<xsl:otherwise>
  <xsl:element name="xs:element" use-attribute-sets="feature_element" />
  <xsl:element name="xs:complexType" use-attribute-sets="feature_complexType">
  <xsl:element name="xs:annotation">
  <xsl:element name="xs:documentation">
  <xsl:value-of select="fc:typeName/sc:LocalName" />
    </xsl:element>
    </xsl:element>
  <xsl:element name="xs:complexContent">
  <xsl:element name="xs:extension" use-attribute-sets="feature_extension">
  <xsl:call-template name="getAttributes" />
    </xsl:element>
    </xsl:element>
    </xsl:element>
  </xsl:otherwise>
  </xsl:choose>
  </xsl:for-each>
</xsl:element>
</xsl:template>

- <!-- ===== -->
- <!-- template feature attributes -->
- <!-- ===== -->
<xsl:template name="getAttributes">
<xsl:element name="xs:sequence">
<xsl:for-each select="fc:carrierOfCharacteristics/fc:FC_FeatureAttribute">
<xsl:choose>
  - <!-- optional attributes -->
<xsl:when
  test="fc:constrainedBy/dgiwg:DGIWG_Requirement/fc:description/dgiwg:DGIWG_ReqL
  evel = 'O'">
<xsl:choose>
  - <!-- optional attributes type code_list -->
<xsl:when test="fc:valueType/sc:TypeName/sc:aName/sc:CharacterString = 'CodeList'">
<xsl:element name="xs:element" use-attribute-sets="opt_codeList">
<xsl:element name="xs:simpleType">
<xsl:element name="xs:restriction" use-attribute-sets="restrict_element_attributes">
<xsl:for-each select="fc:listedValue/fc:FC_ListedValue">
<xsl:element name="xs:enumeration" use-attribute-sets="enumeration_listedValue">
<xsl:element name="xs:annotation">
<xsl:element name="xs:documentation">
<xsl:value-of select="fc:label/sc:CharacterString" />

```



```

</xsl:element>
</xsl:element>
</xsl:element>
</xsl:for-each>
</xsl:element>
</xsl:element>
</xsl:element>
</xsl:when>
- <!-- other optional attributes -->
<xsl:otherwise>
<xsl:choose>
- <!-- optional attributes type measureType -->
<xsl:when test="fc:valueMeasurementUnit">
<xsl:choose>
- <!-- optional attributes type measureType having value type real -->
<xsl:when test="fc:valueMeasurementUnit/sc:UomLength = 'Metre'">
<xsl:element name="xs:element" use-attribute-sets="opt_measureType" />
</xsl:when>
<xsl:when test="fc:valueMeasurementUnit/sc:UomLength = 'Square Metre'">
<xsl:element name="xs:element" use-attribute-sets="opt_measureType" />
</xsl:when>
<xsl:when test="fc:valueMeasurementUnit/sc:UomLength = 'Arc Degree'">
<xsl:element name="xs:element" use-attribute-sets="opt_measureType" />
</xsl:when>
<xsl:when test="fc:valueMeasurementUnit/sc:UomLength = 'Kilovolt'">
<xsl:element name="xs:element" use-attribute-sets="opt_measureType" />
</xsl:when>
<xsl:when test="fc:valueMeasurementUnit/sc:UomLength = 'Percent'">
<xsl:element name="xs:element" use-attribute-sets="opt_measureType" />
</xsl:when>
<xsl:when test="fc:valueMeasurementUnit/sc:UomLength = 'Hertz'">
<xsl:element name="xs:element" use-attribute-sets="opt_measureType" />
</xsl:when>
- <!-- optional attributes type measureType having value type not real are changed to optional
attributes type integer -->
<xsl:otherwise>
<xsl:element name="xs:element" use-attribute-sets="opt_integer" />
</xsl:otherwise>
</xsl:choose>
</xsl:when>
- <!-- other optional attributes -->
<xsl:otherwise>
<xsl:choose>
- <!-- optional attributes type characterString -->
<xsl:when test="fc:valueType/sc:TypeName/sc:aName/sc:CharacterString =
'CharacterString'">
<xsl:element name="xs:element" use-attribute-sets="opt_characterString" />
</xsl:when>
- <!-- optional attributes type real -->
<xsl:when test="fc:valueType/sc:TypeName/sc:aName/sc:CharacterString = 'Real'">
<xsl:element name="xs:element" use-attribute-sets="opt_real" />
</xsl:when>
- <!-- optional attributes type integer -->
<xsl:when test="fc:valueType/sc:TypeName/sc:aName/sc:CharacterString = 'Integer'">
<xsl:element name="xs:element" use-attribute-sets="opt_integer" />
</xsl:when>
</xsl:choose>

```

```

</xsl:otherwise>
</xsl:choose>
</xsl:otherwise>
</xsl:choose>
</xsl:when>
- <!-- mandatory attributes -->
- <xsl:otherwise>
- <xsl:choose>
- - <!-- mandatory attributes type Code_List -->
- <xsl:when test="fc:valueType/sc:TypeName/sc:aName/sc:CharacterString = 'CodeList'">
- <xsl:element name="xs:element" use-attribute-sets="man_codeList">
- <xsl:element name="xs:simpleType">
- <xsl:element name="xs:restriction" use-attribute-sets="restrict_element_attributes">
- <xsl:for-each select="fc:listedValue/fc:FC_ListedValue">
- <xsl:element name="xs:enumeration" use-attribute-sets="enumeration_listedValue">
- <xsl:element name="xs:annotation">
- <xsl:element name="xs:documentation">
- <xsl:value-of select="fc:label/sc:CharacterString" />
- </xsl:element>
- </xsl:element>
- </xsl:element>
- </xsl:for-each>
- </xsl:element>
- </xsl:element>
- </xsl:when>
- <xsl:otherwise>
- - <!-- other mandatory attributes -->
- <xsl:choose>
- - <!-- mandatory attributes type measureType -->
- <xsl:when test="fc:valueMeasurementUnit">
- <xsl:choose>
- - <!-- mandatory attributes type measureType having value type real -->
- <xsl:when test="fc:valueMeasurementUnit/sc:UomLength = 'Metre'">
- <xsl:element name="xs:element" use-attribute-sets="man_measureType" />
- </xsl:when>
- <xsl:when test="fc:valueMeasurementUnit/sc:UomLength = 'Square Metre'">
- <xsl:element name="xs:element" use-attribute-sets="man_measureType" />
- </xsl:when>
- <xsl:when test="fc:valueMeasurementUnit/sc:UomLength = 'Arc Degree'">
- <xsl:element name="xs:element" use-attribute-sets="man_measureType" />
- </xsl:when>
- <xsl:when test="fc:valueMeasurementUnit/sc:UomLength = 'Kilovolt'">
- <xsl:element name="xs:element" use-attribute-sets="man_measureType" />
- </xsl:when>
- <xsl:when test="fc:valueMeasurementUnit/sc:UomLength = 'Percent'">
- <xsl:element name="xs:element" use-attribute-sets="man_measureType" />
- </xsl:when>
- <xsl:when test="fc:valueMeasurementUnit/sc:UomLength = 'Hertz'">
- <xsl:element name="xs:element" use-attribute-sets="man_measureType" />
- </xsl:when>
- - <!-- mandatory attributes type measureType having value type not real are changed to
optional attributes type integer -->
- <xsl:otherwise>
- <xsl:element name="xs:element" use-attribute-sets="man_integer" />
- </xsl:otherwise>
- </xsl:choose>

```

```

    </xsl:when>
  - <!-- other mandatory attributes -->
- <xsl:otherwise>
- <xsl:choose>
  - <!-- mandatory attributes type characterString -->
- <xsl:when test="fc:valueType/sc:TypeName/sc:aName/sc:CharacterString =
  'CharacterString'">
  <xsl:element name="xs:element" use-attribute-sets="man_characterString" />
  </xsl:when>
  - <!-- geometry attribute type point -->
- <xsl:when test="fc:valueType/sc:TypeName/sc:aName/sc:CharacterString = 'GM_Point'">
  <xsl:element name="xs:element" use-attribute-sets="point" />
  </xsl:when>
  - <!-- geometry attribute type curve -->
- <xsl:when test="fc:valueType/sc:TypeName/sc:aName/sc:CharacterString = 'GM_Curve'">
  <xsl:element name="xs:element" use-attribute-sets="line" />
  </xsl:when>
  - <!-- geometry attribute type area -->
- <xsl:when test="fc:valueType/sc:TypeName/sc:aName/sc:CharacterString = 'GM_Surface'">
  <xsl:element name="xs:element" use-attribute-sets="area" />
  </xsl:when>
  - <!-- mandatory attributes type real -->
- <xsl:when test="fc:valueType/sc:TypeName/sc:aName/sc:CharacterString = 'Real'">
  <xsl:element name="xs:element" use-attribute-sets="man_real" />
  </xsl:when>
  - <!-- mandatory attributes type integer -->
- <xsl:when test="fc:valueType/sc:TypeName/sc:aName/sc:CharacterString = 'Integer'">
  <xsl:element name="xs:element" use-attribute-sets="man_integer" />
  </xsl:when>
  </xsl:choose>
</xsl:otherwise>
</xsl:choose>
</xsl:otherwise>
</xsl:choose>
</xsl:otherwise>
</xsl:choose>
</xsl:otherwise>
</xsl:choose>
</xsl:for-each>
</xsl:element>
</xsl:template>

- <!-- ===== -->
- <!-- template change upper to lower case -->
- <!-- ===== -->
- <xsl:template name="to_lower_case">
  <xsl:param name="expr" />
  <xsl:variable name="uc" select="'ABCDEFGHIJKLMNOPQRSTUVWXYZ_0123456789'" />
  <xsl:variable name="lc" select="'abcdefghijklmnopqrstuvwxyz_0123456789'" />
  <xsl:value-of select="translate($expr,$uc,$lc)" />
  </xsl:template>

- <!-- ===== -->
- <!-- attribute sets for feature attributes -->
- <!-- ===== -->
- <xsl:attribute-set name="man_characterString">
- <xsl:attribute name="name">
- <xsl:call-template name="to_lower_case">

```

```

<xsl:with-param name="expr"
  select="fc:definitionReference/fc:FC_DefinitionReference/fc:sourceIdentifier/sc:CharacterString" />
</xsl:call-template>
</xsl:attribute>
<xsl:attribute name="type">xs:string</xsl:attribute>
</xsl:attribute-set>
- <xsl:attribute-set name="point">
  <xsl:attribute name="name">pointGeometry</xsl:attribute>
  <xsl:attribute name="type">gml:PointPropertyType</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="line">
  <xsl:attribute name="name">lineGeometry</xsl:attribute>
  <xsl:attribute name="type">gml:CurvePropertyType</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="area">
  <xsl:attribute name="name">areaGeometry</xsl:attribute>
  <xsl:attribute name="type">gml:SurfacePropertyType</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="man_real">
- <xsl:attribute name="name">
- <xsl:call-template name="to_lower_case">
- <xsl:with-param name="expr"
  select="fc:definitionReference/fc:FC_DefinitionReference/fc:sourceIdentifier/sc:CharacterString" />
  </xsl:call-template>
  </xsl:attribute>
  <xsl:attribute name="type">xs:double</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="man_integer">
- <xsl:attribute name="name">
- <xsl:call-template name="to_lower_case">
- <xsl:with-param name="expr"
  select="fc:definitionReference/fc:FC_DefinitionReference/fc:sourceIdentifier/sc:CharacterString" />
  </xsl:call-template>
  </xsl:attribute>
  <xsl:attribute name="type">xs:int</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="man_measureType">
- <xsl:attribute name="name">
- <xsl:call-template name="to_lower_case">
- <xsl:with-param name="expr"
  select="fc:definitionReference/fc:FC_DefinitionReference/fc:sourceIdentifier/sc:CharacterString" />
  </xsl:call-template>
  </xsl:attribute>
  <xsl:attribute name="type">gml:MeasureType</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="man_codeList">
- <xsl:attribute name="name">
- <xsl:call-template name="to_lower_case">
- <xsl:with-param name="expr"
  select="fc:definitionReference/fc:FC_DefinitionReference/fc:sourceIdentifier/sc:CharacterString" />
  </xsl:call-template>
  </xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="enumeration_listedValue">
- <xsl:attribute name="value">

```

```

<xsl:value-of select="fc:code/sc:CharacterString" />
  </xsl:attribute>
</xsl:attribute-set>
- <xsl:attribute-set name="opt_characterString">
- <xsl:attribute name="name">
- <xsl:call-template name="to_lower_case">
- <xsl:with-param name="expr"
  select="fc:definitionReference/fc:FC_DefinitionReference/fc:sourceIdentifier/sc:Charact
erString" />
  </xsl:call-template>
  </xsl:attribute>
<xsl:attribute name="type">xs:string</xsl:attribute>
<xsl:attribute name="minOccurs">0</xsl:attribute>
<xsl:attribute name="nillable">true</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="opt_real">
- <xsl:attribute name="name">
- <xsl:call-template name="to_lower_case">
- <xsl:with-param name="expr"
  select="fc:definitionReference/fc:FC_DefinitionReference/fc:sourceIdentifier/sc:Charact
erString" />
  </xsl:call-template>
  </xsl:attribute>
<xsl:attribute name="type">xs:double</xsl:attribute>
<xsl:attribute name="minOccurs">0</xsl:attribute>
<xsl:attribute name="nillable">true</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="opt_integer">
- <xsl:attribute name="name">
- <xsl:call-template name="to_lower_case">
- <xsl:with-param name="expr"
  select="fc:definitionReference/fc:FC_DefinitionReference/fc:sourceIdentifier/sc:Charact
erString" />
  </xsl:call-template>
  </xsl:attribute>
<xsl:attribute name="type">xs:int</xsl:attribute>
<xsl:attribute name="minOccurs">0</xsl:attribute>
<xsl:attribute name="nillable">true</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="opt_measureType">
- <xsl:attribute name="name">
- <xsl:call-template name="to_lower_case">
- <xsl:with-param name="expr"
  select="fc:definitionReference/fc:FC_DefinitionReference/fc:sourceIdentifier/sc:Charact
erString" />
  </xsl:call-template>
  </xsl:attribute>
<xsl:attribute name="type">gml:MeasureType</xsl:attribute>
<xsl:attribute name="minOccurs">0</xsl:attribute>
<xsl:attribute name="nillable">true</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="opt_codeList">
- <xsl:attribute name="name">
- <xsl:call-template name="to_lower_case">
- <xsl:with-param name="expr"
  select="fc:definitionReference/fc:FC_DefinitionReference/fc:sourceIdentifier/sc:Charact
erString" />
  </xsl:call-template>
  </xsl:attribute>
<xsl:attribute name="minOccurs">0</xsl:attribute>

```

```

<xsl:attribute name="nillable">true</xsl:attribute>
  </xsl:attribute-set>
- <!-- ===== -->
- <!-- other attribute sets -->
- <!-- ===== -->
- <xsl:attribute-set name="schema_element">
  <xsl:attribute
    name="xmlns:MGCP">http://www.dgiwg.org/schemas/MGCP/0.1.0</xsl:attribute>
  <xsl:attribute name="xmlns:gml">http://www.opengis.net/gml</xsl:attribute>
  <xsl:attribute name="xmlns:gmlsf">http://www.opengis.net/gmlsf</xsl:attribute>
  <xsl:attribute
    name="targetNamespace">http://www.dgiwg.org/schemas/MGCP/0.1.0</xsl:attribute>
  <xsl:attribute name="elementFormDefault">qualified</xsl:attribute>
  <xsl:attribute name="version">0.0.1</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="appinfo_element">
  <xsl:attribute
    name="source">file:///C:/projekte/mgcp/input_gerhard/gml/3.1.1/Profiles/SimpleFeatures/1.0.0withDocumentation/gmlsfLevels.xsd</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="import_element">
  <xsl:attribute name="namespace">http://www.opengis.net/gml</xsl:attribute>
  <xsl:attribute
    name="schemaLocation">C:\projekte\mgcp\input_gerhard\gml\3.1.1\Profiles\SimpleFeatures\1.0.0withDocumentation\gmlsf.xsd</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="featureCollection_element">
  <xsl:attribute name="name">CELL</xsl:attribute>
  <xsl:attribute name="type">MGCP:FeatureCollectionNameType</xsl:attribute>
  <xsl:attribute name="substitutionGroup">gml:_GML</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="featureCollection_complexType">
  <xsl:attribute name="name">FeatureCollectionNameType</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="featureCollection_extension">
  <xsl:attribute name="base">gml:AbstractFeatureType</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="featureCollection_sequence">
  <xsl:attribute name="minOccurs">0</xsl:attribute>
  <xsl:attribute name="maxOccurs">unbounded</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="featureCollection_featureMember">
  <xsl:attribute name="name">featureMember</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="featureCollection_gmlFeature">
  <xsl:attribute name="ref">gml:_Feature</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="abstractFeature_element">
  <xsl:attribute name="name">MGCP_Feature</xsl:attribute>
  <xsl:attribute name="type">MGCP:MGCP_FeatureType</xsl:attribute>
  <xsl:attribute name="substitutionGroup">gml:_Feature</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="abstractFeature_complexType">
  <xsl:attribute name="name">MGCP_FeatureType</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="abstractFeature_restriction">
  <xsl:attribute name="base">gml:AbstractFeatureType</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="restrict_element_attributes">

```

```

<xsl:attribute name="base">xs:string</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="feature_element">
- <xsl:attribute name="name">
  <xsl:value-of select="@id" />
  </xsl:attribute>
- <xsl:attribute name="type">
  MGCP:
  <xsl:value-of select="@id" />
  Type
  </xsl:attribute>
  <xsl:attribute name="substitutionGroup">gml:_Feature</xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="feature_complexType">
- <xsl:attribute name="name">
  <xsl:value-of select="@id" />
  Type
  </xsl:attribute>
  </xsl:attribute-set>
- <xsl:attribute-set name="feature_extension">
  <xsl:attribute name="base">MGCP:MGCP_FeatureType</xsl:attribute>
  </xsl:attribute-set>
</xsl:stylesheet>

```

18.2 Script 2: sort_transformed_catalog v0.1.0.xsl

```

<?xml version="1.0" ?>
- <xsl:stylesheet version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
- <!--
  metadata for stylesheet sort_transform_catalog.xsl
  version 0.1.0
  produced by dotGIS, Dr. Gerhard Joos
  2006-04-20
  input: output of transform_catalog v0.1.0.xsl
  -->
- <xsl:template match="/">
- <xsl:element name="xs:schema" namespace="http://www.w3.org/2001/XMLSchema" use-
  attribute-sets="schema">
  <xsl:copy-of select="xs:schema/xs:annotation" />
  <xsl:copy-of select="xs:schema/xs:import" />
- <xsl:for-each select="xs:schema/xs:element">
  <xsl:copy-of select="." />
  </xsl:for-each>
- <xsl:for-each select="xs:schema/xs:complexType">
- <xsl:choose>
- <xsl:when test="@name = 'FeatureCollectionNameType'">
  <xsl:copy-of select="." />
  </xsl:when>
- <xsl:when test="@name = 'MGCP_FeatureType'">
  <xsl:copy-of select="." />
  </xsl:when>
- <xsl:otherwise>
- <xsl:copy use-attribute-sets="complexType">
  <xsl:copy-of select="xs:annotation" />
- <xsl:element name="xs:complexContent">
- <xsl:element name="xs:extension" use-attribute-sets="extension">
- <xsl:element name="xs:sequence">

```

```

- <xsl:for-each select="descendant::xs:element">
- <xsl:choose>
- <xsl:when test="@name = 'areaGeometry'" />
- <xsl:when test="@name = 'lineGeometry'" />
- <xsl:when test="@name = 'pointGeometry'" />
- <xsl:otherwise>
- <xsl:copy-of select="." />
- </xsl:otherwise>
- </xsl:choose>
- </xsl:for-each>
- <xsl:call-template name="geometry" />
- </xsl:element>
- </xsl:element>
- </xsl:element>
- </xsl:copy>
- </xsl:otherwise>
- </xsl:choose>
- </xsl:for-each>
- </xsl:element>
- </xsl:template>
- <!-- ===== -->
- <!-- Templates -->
- <!-- ===== -->
- <xsl:template name="geometry">
- <xsl:choose>
- <xsl:when test="descendant::xs:element/@name = 'areaGeometry'">
- <xsl:element name="xs:element" use-attribute-sets="element_area" />
- </xsl:when>
- <xsl:when test="descendant::xs:element/@name = 'lineGeometry'">
- <xsl:element name="xs:element" use-attribute-sets="element_line" />
- </xsl:when>
- <xsl:when test="xs:complexContent/xs:extension/xs:sequence/xs:element/@name =
- 'pointGeometry'">
- <xsl:element name="xs:element" use-attribute-sets="element_point" />
- </xsl:when>
- </xsl:choose>
- </xsl:template>
- <!-- ===== -->
- <!-- Attribute_Sets -->
- <!-- ===== -->
- <!-- attribute_set of element xs:element representing a pointGeometry -->
- <xsl:attribute-set name="element_point">
- <xsl:attribute name="name">pointGeometry</xsl:attribute>
- <xsl:attribute name="type">gml:PointPropertyType</xsl:attribute>
- </xsl:attribute-set>
- <!-- attribute_set of element xs:element representing a lineGeometry -->
- <xsl:attribute-set name="element_line">
- <xsl:attribute name="name">lineGeometry</xsl:attribute>
- <xsl:attribute name="type">gml:CurvePropertyType</xsl:attribute>
- </xsl:attribute-set>
- <!-- attribute_set of element xs:element representing a areaGeometry -->
- <xsl:attribute-set name="element_area">
- <xsl:attribute name="name">areaGeometry</xsl:attribute>
- <xsl:attribute name="type">gml:SurfacePropertyType</xsl:attribute>
- </xsl:attribute-set>
- <!-- attribute_set of element xs:complexType -->

```



```
- <xsl:attribute-set name="complexType">
- <xsl:attribute name="name">
  <xsl:value-of select="@name" />
  </xsl:attribute>
  </xsl:attribute-set>
- <!-- attribute_set of element xs:extension -->
- <xsl:attribute-set name="extension">
- <xsl:attribute name="base">
  <xsl:value-of select="descendant::xs:extension/@base" />
  </xsl:attribute>
  </xsl:attribute-set>
- <!-- attribute_set of element xs:schema -->
- <xsl:attribute-set name="schema">
  <xsl:attribute
    name="xmlns:MGCP">http://www.dgiwg.org/schemas/MGCP/0.1.0</xsl:attribute>
  <xsl:attribute name="xmlns:gml">http://www.opengis.net/gml</xsl:attribute>
  <xsl:attribute name="xmlns:gmlsf">http://www.opengis.net/gmlsf</xsl:attribute>
  <xsl:attribute
    name="targetNamespace">http://www.dgiwg.org/schemas/MGCP/0.1.0</xsl:attribute>
  <xsl:attribute name="elementFormDefault">qualified</xsl:attribute>
  <xsl:attribute name="version">0.0.1</xsl:attribute>
  </xsl:attribute-set>
  </xsl:stylesheet>
```