Open GIS Consortium Inc.

Date: 30-OCT-2003

Reference number of this OpenGIS® Project Document: **OGC 03-003r10**

Version: 0.0.10

Category: OpenGIS® OGC Interoperability Program Report-Engineering Specification

Editor: Panagiotis (Peter) A. Vretanos (CubeWerx Inc.)

Level 0 Profile of GML3 for WFS

Copyright notice

Warning

This document is not an OGC Standard or Specification. This document presents a discussion of technology issues considered in an Interoperability Initiative of the OGC Interoperability Program. The content of this document is presented to create discussion in the geo-spatial information industry on this topic; the content of this document is not to be considered an adopted specification of any kind. This document does not represent the official position of the OGC nor of the OGC Technical Committee. It is subject to change without notice and may not be referred to as an OGC Standard or Specification. However, the discussions in this document could very well lead to the definition of an OGC Implementation Specification.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type:	OpenGIS® Interoperability Program Report-Engineering Specification
Document subtype:	Critical Infrastructure Protection Initiative, Phase-1.2 (CIPI1.2)
Document stage:	Final
Document language:	English

Contents

i. Preface

The OpenGIS Consortium (OGC) is an international industry consortium of more than 220 companies, government agencies, and universities participating in a consensus process to develop publicly available geo-processing specifications. This Interoperability Program Report (IPR) is a product of the OGC Critical Infrastructure Protection Initiative, Phase-1 (CIPI1).

The OGC Critical Infrastructure Protection Initiative, Phase 1, is part of the OGC's Interoperability Program: a global, collaborative, hands-on engineering and testing program designed to deliver prototype technologies and proven candidate specifications into the OGC's Specification Development Program. In OGC Interoperability Initiatives, international teams of technology providers work together to solve specific geo-processing interoperability problems posed by Initiative sponsors.

ii. Submitting organizations

This draft Interoperability Program Report – Engineering Specification is being submitted to the OGC Interoperability Program by the following organizations:

CubeWerx Inc.
200 rue Montcalm, Suite R-13
Gatineau, QC J8Y 3B5
Canada

Galdos Inc.
Suite 200, 1155 West Pender St.
Vancouver, BC V6E 2P4
Canada

Document Contributor Contact Points

All questions regarding this submission should be directed to the editor or the submitters:

Panagiotis (Peter) A. Vretanos
CubeWerx Inc.
pvretano@cubewerx.com

Aleksander Milanovic
Galdos Inc.
amilanovic@galdosinc.com

iii. Revision history

Date	Release	Description
23-DEC-2003	0.0.11	• Incorporate Simon Cox comment about using lowerCamelCase for property names and upperCamelCase for feature names.
30-OCT-2003	0.0.10	• Integrate John Davidson final comments (very minor changes)
26-SEP-2003	0.0.9	• Integrate final Galdos comments. • Remove previous review comment ANNEX's.
25-SEP-2003	0.0.8	• Final scrub • Indicate that clients should be prepared to deal with properties in any order. • Include ROAD_BTS example in the examples clause • Address P.Daisy comments. • Indicate that only GML3 geometry elements are valid. That is that a compliant schema cannot use deprecated GML2 elements. • Describe the CubeWerx schema validator in ANNEX A – the conformance annex. • Add a future work annex.
28-JUL-2003	0.0.7	• Add conformance testing annex • Make minOccurs and maxOccurs attributes optional • Moved clause v. (changes to implementation specifications) to clause 6.1.2. • Editorial and content changes to section 6.1.2.
10-JUN-2003	0.0.6	• Integrate round 2 comments from Galdos • Integrate comments from John Davidson

22-May-2003	0.0.5	• Migrate document to support GML3 • Incorporate comments from Galdos
29-Mar-2003	0.0.4	• add requirements section based on JohnD email • put Galdos comments into Appendix • comment on Galdos comments • make changes based on Galdos comments
22-Mar-2003	0.0.4	Remove references to Option 1 and Option 3.
11-Feb-2003	0.0.3	Add normative references
02-Feb-2003	0.0.2	E.K. review comments
17-Jan-2003	0.0.1	Initial version

iv. Changes to the OpenGIS® Abstract Specification

None.

v. Changes to the OpenGIS® Implementation Specifications

Implementor should refer to clause 6.1.2 for a list of changes they will have make, relative to the Web Feature Service Implementation Specification (02-058) and the Filter Encoding Specification (02-059), in order to implement a Level 0 capability.

Foreword

Attention is drawn to the possibility that some of the elements of this part of OGC 03-003r10 may be the subject of patent rights. Open GIS Consortium Inc. shall not be held responsible for identifying any or all such patent rights.

This is the ninth revision of this document.

Introduction

The Web Feature Service Implementation Specification [2] was initially proposed nearly two years ago and defines a web service that supports query and transactional operations on features stored in web accessible data-stores. The specific operations defined in the WFS specification are: *GetCapabilities*, *DescribeFeatureType*, *GetFeature*, *GetFeatureWithLock*, *LockFeature* and *Transaction*.

The typical sequence of operations for interacting with a WFS is shown in figure 1.

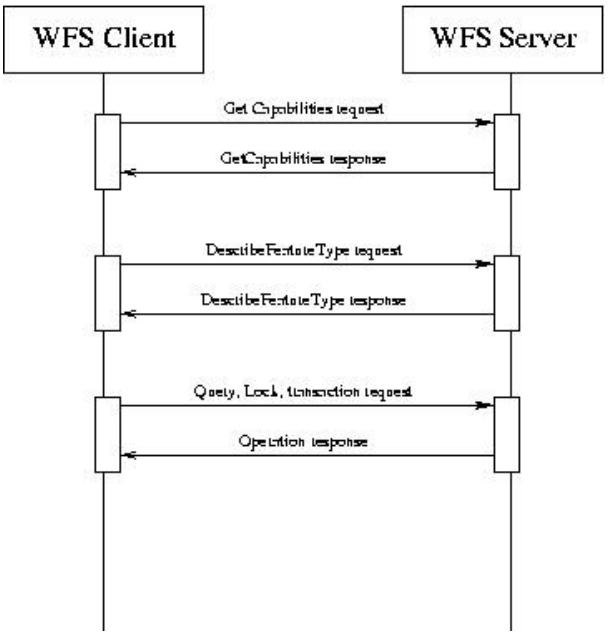

Figure 1 – Typical Interaction with a WFS

A client issues a *GetCapabilities* request to determine the list of feature types that a WFS offers. Then, the client issues a *DescribeFeatureType* request in order to obtain a description of one or more feature type(s). Finally, the client uses the description of the feature type(s) to formulate *GetFeature*, *GetFeatureWithLock*, *LockFeature* and/or *Transaction* requests in order to retrieve or modify feature instances.

According to the WFS specification [2], feature types must be described using GML [1] and XML-Schema [3]. This implies that the client application must be able to parse and interpret schemas expressed in XML-Schema.

XML-Schema is a large and complex specification designed to satisfy a large set of requirements. As a result, implementing a parser that is able to read and interpret an XML-Schema document represents a significant implementation hurdle.

When dealing with XML-Schema, a client application builder has two options:

1. Use an existing XML-Schema parser.

2. Build an XML-Schema parser from scratch.

Coding an XML-Schema parser from scratch represents a significant investment in time and resources before one even gets to the point of coding an actual WFS client. For many organizations, the effort required may be too expensive.

A reasonable number of open source and commercial tools that can manipulate XML Schema documents are currently available. However, the following issues still exist[1]:

1. Many of the existing XML-Schema parsers do not fully support the XML-Schema specification and thus lack the ability to correctly process GML application schemas.[2]

2. Most of the available XML-Schema parsers simply validate XML instance documents against XML-Schema documents. They lack the necessary API for interpreting XML-Schema document instances.[3]

3. Typically, the only language binding available is Java. While it is possible to invoke Java routines from other languages using the Java Native Interface (JNI), this approach introduces another set of technical issues to deal with.

   In addition, as a programming language Java is not suitable for all geo-processing requirements. Performance limitations and operations using array processing of geographic coordinates are a good examples of the limitations of Java.

4. The reliability, accuracy and stability of the available XML-Schema parsers reflect the maturity of the XML Schema specification, which is still evolving.

Despite these issues, a number of WFS clients have been built and successfully tested in OGC testbeds. Although very little has been accomplished regarding interoperability across WFS's. However, the main issue presented here is not whether a WFS client can be built but what is the minimum effort required to build a simple[4] WFS client. At the moment, the effort is too high and this is hindering WFS interoperability, limiting the availability of WFS clients and limiting the adoption of GML3. This document proposes lowering the *"implementation bar"* for any organization that may want to commit time and resources for developing an interoperable and simple OGC WFS client application. This proposal does not in any way restrain usage of commercial XML-Schema parsers or eliminate the need for consensus building from *Information Communities* on XML-

---

[1] Castor, Xerces, Oracle XDK, MSParser, XmlSpy, XSV, Extensibility, JBind

[2] Specifically, the parsers do not support substitution groups that are mandatory for processing GML schemas.

[3] Xerces and Castor provide the necessary API for interpreting an XML-Schema document but neither is well documented and a using them represents a significant learning curve. A fair amount of trial and error is required to figure out how to use these APIs. There is a limited set of language bindings.

[4] A *simple* WFS client is one that can interact with a WFS to query features and then manipulate them (e.g. for display, transactions, etc…)

Schemas but promotes the idea that OGC should not only rely on those elements to promote and achieve interoperability with WFS services.

Data provider organizations, especially organizations supporting simple geographic data models with less resource capabilities, would rollout WFS-based geo-processing applications a lot faster by using, for example, GML application schemas generated using well-known and well-defined XML-Schema coding patterns.

Thus the goal of this document is one of making it easier to build clients that can interact, without modifying client code, with any conformant WFS. This document further asserts that this goal can be achieved by describing schema coding rules for GML3 application schemas that define a simplified representation of geographic features, albeit with limited expressiveness. The hypothesis is that defining a basic schema coding pattern that handles say 80% of the needs of WFS clients to access geographic features/geometry, significantly lowers the bar for implementing clients and fosters the "up-take" of GML, WFS and other OGC-adopted specifications in the marketplace.

Level 0 Profile of GML3 for WFS

1 Scope

The purpose of this document is to define a set of basic schema encoding rules that allow GML3 applications schemas to be created that define how to represent simple feature data in XML.

At a high level, this document describes:

1. Rigid rules for the use of a subset of XML Schema constructs (XML Schema profile)
2. Rigid rules for the use of a subset of GML constructs (GML profile)
3. Optional changes and enhancements to the WFS 1.0.0 interface required to support this specification.

The intent is to specify the encoding of application schemas sufficiently so that WFS client implementations do not need to deal with the entire scope of XML-Schema and GML but only need to understand a restricted subset of both specifications in order to be able interpret schema documents generated in response to a *DescribeFeatureType* request.

By doing so, it is expected that exchanging feature instances between web feature services[5] and building WFS clients become a less onerous task even if the proposed solutions should be viewed as a partial solution towards interoperability of web feature services. Issues regarding semantics of Geographic features are out of scope for this document

This document is ***NOT*** intended to supersede or impede the progress of GML in any way. The OGC GML Special Interest Group has demonstrated over the years the benefits and flexibility of GML for encoding geographic features including the ability to support very complex data models. In addition the OGC GML specification goes far beyond the encoding capability and the scope of this document by defining methods and apparatus for handling geo-processing needs from topology to dynamic features. In fact, it is hoped that by lowering the effort required to manipulate XML encoded feature data, organizations will be enticed to invest more time and effort with GML.

[5] Of course in exchanging feature instances between web feature services, the WFS sending the features, is in-fact a WFS client.

2 Conformance

Not required in an IP, DIPR, IPR or Discussion Paper.

3 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of this Interoperability Program Report. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this document (OGC 03-003r10) are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

[1] OGC, *OpenGIS® Geography Markup Language (GML) Implementation Specification, version 3.0*, http://www.opengis.org/techo/documents/02-023r4.pdf, 2002

[2] OGC Document 02-058, *OpenGIS Web Feature Service Implementation Specification version 1.0*, http://www.opengis.org/techno/specs/02-058.pdf, 2002

[3] W3C, *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation, 6 October 2000, http://www.w3.org/TR/REC-xml

[4] W3C, *XML Schema Part 1: Structures*, http://www.w3.org/TR/xmlschema-1

[5] W3C, *XML Schema Part 2: Datatypes*, http://www.w3.org/TR/xmlschema-2

[6] OGC Document 02-004, *Patterns in GML*, http://member.opengis.org/tc/archive/arch02/02-004.pdf, 2002

[10] OGC Document 01-025, *Simple GML: Geography Markup Language (GML) 2.0 Implementation Profile*, http://member.opengis.org/tc/archive/arch01/01-025.pdf, 2001

4 Terms and definitions

T.B.D.

5 Conventions

T.B.D.

5.1 Requirement levels

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 [1].

6 Requirements

6.1 Relationship to other OGC activities

6.1.1 Geography mark-up language (GML)

This document defines a profile of GML3. As such, it should have no direct effect on the definition of GML3 itself, but should track changes to GML3 in order to ensure that schemas that conform to this specification are also valid GML3 application schemas.

6.1.2 Web feature server (WFS)

Generating an application schema that conforms to this specification is an optional capability. A WFS instance may support the GML3L0 profile and other GML3 profiles and application schemas.

In the event that a WFS does support this specification, then the following changes will need to be implemented:

General Changes:

This specification recommends that format values in the WFS specification be changed from opaque strings to MIME types. Using MIME types will allow the WFS to be explicit about the format, version and profile being specified in a request.

The general form of such MIME types should be x-application/format:version[:profile]. Thus the currently defined format value of GML2 would, instead, be specified as x-application/gml:2.

GetCapabilities response:

The current WFS specification defines two elements, <**SchemaDescriptionLanguage**> and <**ResultFormat**>, that are used to indicate the supported output formats for the DescribeFeatureType and GetFeature operations.

These elements are defined to contain sub-elements that represent the supported output formats. For example, the element <**GML2**> is used to indicate an output format of GML2.

The problem with this approach is the each time you add a new output format, either the capabilities response schema must be updated to define the additional new elements or some awkward XML-Schema constructs, involving element substitution, must be used.

A better approach is to define the <**SchemaDescriptionLanguage**> and <**ResultFormat**> elements as lists of strings. Thus, new formats can be easily added without the need to redefine the capabilities response schema or use awkward XML-Schema constructs.

These revised definitions of the <**SchemaDescriptionLanguage**> and <**ResultFormat**> elements should be changed to:

```
<xsd:element name="SchemaDescriptionLanguage"
             type="wfs:OutputFormatListType"/>
<xsd:element name="ResultFormat"
             type="wfs:OutputFormatListType"/>
<xsd:simpleType name="OutputFormatListType">
  <xsd:list itemType="xsd:string"/>
</xsd:simpleType>
```

In addition, the previous practice of using arbitrary strings to represent the various output formats should be abandoned and MIME types should be defined (in accordance to MIME) for the supported output formats.

DescribeFeatureType request:

The values of **outputFormat** attribute of the *DescribeFeatureType* request must be changed to reflect the fact that a web feature service may be able to generate schemas that conform to more that one specification. Currently, the only defined value for the **outputFormat** attribute is **XMLSCHEMA**, indicating that a GML2 application schema must be generated in response to the request.

The accepted values for the **outputFormat** attribute for the *DescribeFeatureType* request should be changed to:

Table 1 – Values for outputFormat attribute of DescribeFeatureType request

Parameter Value	Description
XMLSCHEMA	This value is kept for backward compatibility and is used to indicate that a GML2 application schema must be generated.
x-application/gml:2	Same as XMLSCHEMA.
x-application/gml:3	This value indicates that a GML3 application schema must be generated. It may be a GML3 application schema that conforms to this specification but if that is the only schema that the WFS can generate.

x-application/gml:3:0	This value indicates that a GML3 application schema that conforms to the coding patterns laid out in this document must be generated.

GetFeature request

The values of **outputFormat** attribute of the *GetFeature* request must be changed to reflect the fact that a web feature service may be able to generate instance documents that conform to more that one version and/or profile of GML. Currently, the only defined value for the **outputFormat** attribute is **GML2** indicating that an instance document that validates against a GML2 application schema must be generated.

The accepted values for the **outputFormat** attribute for the GetFeature request should be changed to:

Table 2 – Values for outputFormat attribute of GetFeature request

Parameter Value	Description
GML2	This value is kept for backward compatibility and indicates that an XML instance document must be generated that validates against a GML2 application schema.
x-application/gml:2	Same as GML2.
x-application/gml:3	This value indicates that an XML instance document must be generated that validates against a GML3 application schema. The XML instance document may validate against a GML3 application schema that conforms to this specification is that is the only schema a WFS can handle.
x-application/gml:3:0	This value indicates that an XML instance document must be generated that validates against a GML3 application schema that conforms to the coding patterns laid out in this specification.

Transaction request:

Currently, the WFS specification assumes that the structure of features on input (i.e. *Insert* and *Update* operations) is defined by a schema document generated using the *DescribeFeatureType* request.

Since a WFS may potentially support features encoded using a number of different vocabularies (GML2, GML3, GML3-LEVEL0), there is a need to define an **inputFormat** attribute to allow a WFS to unambiguously determine how feature instances are encoded on input. The **inputFormat** attribute will need to be defined on the <**Insert**> and <**Update**> elements.

The following schema fragments redefine the <**Insert**> and <**Update**> elements accordingly:

```
<xsd:element name="Insert" type="wfs:InsertElementType" />
<xsd:complexType name="InsertElementType">
   <xsd:sequence>
      <xsd:element ref="gml:_Feature" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="handle" type="xsd:string" use="optional"/>
   <xsd:attribute name="inputFormat" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:element name="Update" type="wfs:UpdateElementType" />
<xsd:complexType name="UpdateElementType">
   <xsd:sequence>
      <xsd:element ref="wfs:Property" maxOccurs="unbounded" />
      <xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1" />
   </xsd:sequence>
   <xsd:attribute name="handle" type="xsd:string" use="optional"/>
   <xsd:attribute name="typeName" type="xsd:QName" use="required"/>
   <xsd:attribute name="inputFormat" type="xsd:string" use="required"/>
</xsd:complexType>
```

The following table defines possible default values for the inputFormat attribute:

Table 3 – Values for inputFormat attribute for Insert and Update operations

Parameter Value	Description
x-application/gml:2	This value indicates that the input feature must validate against a GML2 application schema.
x-application/gml:3	This value indicates that the input feature must validate against a GML3 application schema. The input feature may validate against a GML3 application schema that conforms to this specification.
x-application/gml:3:0	This value indicates that the input feature must validate against a GML3 application schema that conforms to the coding patterns laid out in this document

Changes to the Filter Encoding Specification (02-059):

The **BBOX** operator defined in the filter encoding specification should be changed so that the <**propertyName>** element is optional. The following XML-Schema fragment shows how the **BBOX** element should be re-defined:

```
<xsd:element name="BBOX"
             type="ogc:BBOXType"
             substitutionGroup="ogc:spatialOps"/>

<xsd:complexType name="BBOXType">
   <xsd:complexContent>
      <xsd:extension base="ogc:SpatialOpsType">
         <xsd:sequence>
            <xsd:element ref="ogc:propertyName" minOccurs="0"/>
            <xsd:element ref="gml:Envelope"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

In the event that the <**propertyName**> element is not specified, a WFS must determine which property of a feature is the spatial key and apply the **BBOX** operator accordingly. For a feature type that has a single spatial property, this is a trivial matter. In the case where a feature has multiple spatial properties, the WFS either knows which property is the spatial key or it generates an exception indicating that the feature contains multiple spatial properties and the <**propertyName**> element must be specified. Of course, a client application always has the option of avoiding the exception by using the *DescribeFeatureType* request to get a description of the feature type being queried.

6.2 Usage Scenarios

6.2.1 Simple desktop or browser based map viewer

Allow users to select a collection of features for access and subsequent visualization by client application.

6.2.2 Get a collection of features using spatial and non-spatial constraints

In this scenario, a client application is able to process a schema description of the features that a WFS serves sufficiently to be able to generate a valid and meaningful query, that uses spatial and non-spatial constraints, to obtain one or more feature instances.

6.2.3 "Value-add" editor (edit geometry and other attribute values)

Allow users to augment geo-spatial information supplied by a data producer with data of their own, creating new features or modifying existing features.

6.2.3.1 Create feature instances of specified type or id

In this scenario, a client application is able to process a schema description of the features that a WFS serves sufficiently to be able to generate a valid, new feature instances that may then be inserted into the feature store using a WFS Transaction request.

6.2.3.2 Update features with modified geometry and/or attribute values

In this scenario, a client application is able to process a schema description of the features that a WFS serves sufficiently to understand the type of each attribute of a feature and be able to generate a valid Transaction request that change the values of one or more properties on a feature. The properties may be spatial or non-spatial.

6.3 General requirements

6.3.1 No changes to software

Application client software must be able to access, without software modification, more than one WFS (T) to retrieve geographic feature data regardless of their intended use, semantic, native structure, source or quality.

6.3.2 Valid XML output

Geographic feature data must be returned from WFS encoded as correct XML as defined in the *Extensible Markup Language (XML) 1.0 (Second Edition)* [3] specification.

6.3.3 Valid GML3 output

Geographic feature data must be returned from WFS encoded as GML3 that validates against a GML3 application schema that conforms to this specification.

6.3.4 Simple clients

Relatively unsophisticated client applications (e.g., a simple desktop application or browser applet/plug-in for map display) should be able to:

1. request a feature description and be able to parse that description so that the application can determine the names and types of all the properties of the feature

2. inspect the value of feature properties of a particular feature (or set of features)

3. manipulate feature instances locally for whatever purpose including display geographic features retrieved from WFS as elements of a portrayed map

6.3.5 Well known structural view

Provide a common ("well known") structural view of geo-spatial vector data, encoded as GML3, regardless of their underlying semantic, information or storage models. The approach that this document takes is to define a GML3 profile[6] that:

- handles a wide variety of spatial data models but not all possible models
- defines a profile of the base GML3 spec, including:
    1. restrictions and requirements for use of specific spatial types (e.g., geometry)
    2. restrictions and requirements for use of GML3 modules, constructs and types
    3. naming conventions for modules and types
    4. use of XML and XML Schema encoding specs, including allowable facets

6.3.6 Implementations must be testable for conformance

It must be possible to define methods and/or apparatus to test the validity of a GML3 application schema that claims to conform to this specification[7].

6.3.7 Language bindings

GML3 application schemas that conform to this specification should be implementable and supported by multiple vendors for multiple platforms, tools and programming languages (e.g., Web Service, browser, desktop, Visual Basic/C/C++/C#/Java)

6.3.8 Simple and sufficient

A level 0 compliant schema must be able to adequately represent geographic objects for basic display and attribute inspection with minimum of encoding/decoding overhead and complexity.

An addition minimum sufficiency requirement is that a level 0 schema must be able to represent simple features as described in the OGC simple feature implementation specifications (99-049, 99-050).

---

[6] see: SDTS spec (http://mcmcweb.er.usgs.gov/sdts/profile.html), ISO TC/211 19106 (Geographic information - Profiles) and OGC 02-023r3.

[7] See ANNEX A for conformance clause.

6.3.9 XML Schema Interpretation

As stated in clause 6.3.2 and 6.3.3, a schema that conforms to this specification must be valid XML and a valid GML3 application schema and as such may be processed using available XML-Schema parsers and interpreters.

However, the goal of this specification is to describe methods and apparatus so that there is no need to process **any** of the more advanced features of XML-Schema. One simply needs to locate the definition for a feature type(s) of interest, and scan the property-name/simple-data-type pairs in order to interpret the schema. A rigid profile makes processing XML Schema as easy as scanning a simple list of these data pairs

6.4 Detailed requirements

6.4.1 Validity of output

A schema document that conforms to this specification must be valid with respect to XML [3], and GML3 [1].

6.4.2 Supported geometries

The following GML3 geometry property types must be supported:

Table 4 – Supported GML Geometric Property Types

GML Geometric Property Type	Defined in GML Schema File	Restrictions
gml:PointPropertyType	geometryBasic0d1d.xsd	none
gml:CurvePropertyType	geometryBasic0d1d.xsd	only LineString allowed as value
gml:SurfacePropertyType	geometryBasic2d.xsd	only Polygon allowed as value
gml:MultiCurvePropertyType	geometryAggregates.xsd	only MultiLineString allowed as value
gml:MultiSurfacePropertyType	geometryAggregates.xsd	MultiPolygon and MultiSurface allowed as value; MultiSurface can use only linear (sub)geometries
gml:LinearRingPropertyType	geometryBasic2d.xsd	this was missing in GML-2 and has justifiably been added to GML-3
gml:RingPropertyType	geometryPrimitives.xsd	only LinearRing or Ring with LineStrings can appear as value

Additional requirements for Geometry include:

- Handle simple geometric shapes 0-2.5 D. Don't need 3D solids
- Only LineString type curves (CurveSegments) must be supported. All others must not be supported
- Allow a feature to have any number of geometric properties
- In instance documents, all geometries are to have **srsName** attributes
- The use of deprecated, GML2 geometry constructs (e.g. innerBoundaryIs), is not supported.

6.4.3 Simple feature structure

Features must be defined using a "well-known" and simplified feature model that doesn't require multiple nesting (inheritance) levels, substitution groups, choices, unions, etc… except as prescribed in this specification.

This specification is concerned with the 'basic' schemas for use with relatively simple systems, such as those that use features that are represented (at least conceptually) by a single database table. That is, they have a flat list of feature properties and the properties are of simple types like Number, String, and Boolean and there may be more than one property of a concrete Geometry type.

Systems that are more complex than this can use full-blown, unrestricted GML.

6.4.4 Homogeneous feature collections

Feature collections must be homogeneous with respect to the coordinate reference system used.

6.4.5 Feature references

References to local and remote resources may be made using Xlink.

6.4.6 Simple relationships

Must support simple relationships in which relationships do not carry properties.

6.4.7 2.5D geometry support

This specification must support "2.5D" geometry

6.4.8 GML3 restrictions

Compliant schemas must not use constructs for handling/encoding topology, metadata, dictionaries, temporal features, dynamic features, styles, coverages, and observations.

6.4.9 WFS restrictions

6.4.9.1.1 Introduction

This section outlines restrictions that arise because of this specification on WFS operations.

6.4.9.1.2 GetCapabilities request

See clause 6.1.2.

6.4.9.1.3 DescribeFeatureType request

See clause 6.1.2.

6.4.9.1.4 GetFeature / GetFeatureWithLock requests

See clause 6.1.2.

Also, to comply with this specification, a WFS must generate a single feature collection as defined in clause 7.4.

The feature collection is, itself, not considered a feature but is simply a container for the response to a *GetFeature/GetFeatureWithLock* request. This implies that the feature collection cannot be queried as a feature.

6.4.9.1.5 LockFeature request

None.

6.4.9.1.6 Transaction request

6.4.9.1.6.1 Insert operation

See clause v.

6.4.9.1.6.2 Update operation

See clause v.

6.4.9.1.6.3 Delete operation

See clause v.

6.4.10 CRS Support

A WFS that implements this specification must support the coordinate reference system for *Geographic/WGS84* using decimal degrees to encode latitude and longitude[8]. All other CRS definitions are optional.

7 A rigid coding pattern for GML application schemas

7.1 Introduction

Clause 7 describes a rigid coding pattern for GML application schemas. The main motivation behind this pattern is to limit the set of XML-Schema and GML3 features that may be used to code a GML application schema. This in turn should simplify the task of building WFS clients that can ingest schema documents that conform to this coding pattern and understand the structure of the feature types defined within.

In the following clauses, schema fragments defined may be combined to create a complete GML3 application schema.

The schema fragments must be ***structurally*** encoded exactly as presented in the document. This means that all mandatory elements and attributes presented in the fragment must be included as shown even if they are optional in XML-Schema. Furthermore, no other optional elements or attributes that might be defined in XML-Schema or GML3 may be used unless specified in this document.

Please note, that these requirements have absolutely nothing to do with the formatting of the XML fragments. They are structural and syntactic requirements, not formatting requirements. White spaces can be used freely to format the generated schema documents in any way.

7.2 Root element

The following XML fragment shows how to encode the GML application schema document's root element:

```
1 <xs:schema
2   targetNamespace="target_name_space"
3   xmlns:prefix="target_name_space"
4   xmlns:xs="http://www.w3.org/2001/XMLSchema"
5   xmlns:gml="http://www.opengis.net/gml"
6   elementFormDefault="qualified"
7   version="0.0.7">
```

The attributes shown for the <**schema**> element can be specified in any order.

---

[8] Within OGC, EPSG:4326 has incorrectly been used to represent this coordinate system definition. The problem is that EPSG:4326 is defined in degrees, minutes and seconds rather than decimal degrees. When this issue is resolved within the OGC, the correct EPSG reference will be supplied.

Line 2 declares the target namespace for the elements defined in the schema document. The value *target_name_space* is a placeholder for an actual namespace value.

Line 3 defines a prefix for the target namespace. The value *prefix* is a placeholder for an actual prefix value.

Line 4 declares the prefix for the XML-Schema namespace, which contains all the elements used to define a schema. In this document, the prefix **xs** is used to represent the namespace for the XML-Schema elements. However, a schema document that conforms to this specification may set the prefix to have any desired value as long as it is correctly bound to the XML-Schema namespace (http://www.w3.org/2001/XMLSchema). A conformant schema may also declare the XML-Schema namespace as the default namespace in which case not prefix is defined at all.

Line 5 declares the prefix **gml** for the GML namespace.

Line 6 sets the default value for the **form** attribute for elements to **qualified**. This indicates that locally defined elements are added to the target namespace.

Finally, the **version** attribute should be set to reflect the version of the schema document being generated. The fragment uses version **0.0.7,** which is the version of this document, but it can be any value that has meaning to the entity or organization creating the schema document.

7.3 Importing the GML3 schemas

The following XML fragment imports the GML feature schema:

```
<xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://<…schema repository…>/schemas/gml/3.0.0/feature.xsd"/>
```

The value of the **namespace** attribute must match the GML namespace declaration in the root element.

The value of the **schemaLocation** attribute must be a URI that resolves to the contents of the GML3 schema file *feature.xsd* [1]. The schema fragment above shows a URL that points to the physical file, *feature.xsd*, but any URI may be used (e.g. an HTTP GET request that resolved to a schema).

7.4 Response container

The <**wfs:FeatureCollection**> element, as defined in the WFS specification[2], must be used as the container for the response to *GetFeature/GetFeatureWithLock* requests. This feature collection shall not be queryable, but simply act as a container for the query response.

7.5 Coding pattern for feature types

7.5.1 Introduction

In GML, a feature is encoded as an XML element whose name is the feature type according to some classification. The feature instance contains feature properties, each encoded as an XML element whose name is the property name. Each of these contains another element whose name is the type of the property value or instance; this produces a "layered" syntax in which properties and instances are interleaved.

Following the GML lexical convention, this specification uses *UpperCamelCase* for the XML element names representing instances of Feature Types, and *lowerCamelCase* for the XML element names representing properties[9]. This provides a visual clue distinguishing the two "layers" within long instance documents, and thus assists inspection of instances by developers.

7.5.2 Basic data types

XML-Schema defines a rich set of basic data types that can be used to define XML documents. However, since data served by a WFS can originate from any number of data sources (all of them having a different set of supported basic types), this document limits the set of available basic types to a smaller subset. The rational being that a smaller common set of supported basic data types is likely to be more interoperable.

The list of supported basic data types is:

1. Integers of a specified precision
2. Reals of a specified precision and scale
3. Character strings of a specified maximum length
4. Date
5. Boolean
6. Binary data
7. Geometric property types supported by GML and specified in clause 6.4.2.

The following clauses present XML fragments that act as templates for defining elements whose content type corresponds to one of the seven basic types supported by this specification. Unless otherwise specified, all elements and attributed presented in the templates, and only those element and attribute, must be specified.

---

[9] Although the use of the GML lexical convention is strongly recommended by this specification, it is not mandated in order to be compliant GML3 L0 schema.

7.5.2.1 Null values

In this specification, if ***minOccurs=0*** is specified for an element, this shall be interpreted as meaning that the corresponding property may have a NULL value.

In instance documents, an element that is not specified shall be interpreted as meaning that the corresponding property has a NULL value.

7.5.2.2 Defining elements with integer content

The following XML-Schema fragment shows how to define an element with integer content:

```
<xs:element name="propertyName" minOccurs="0|N" maxOccurs="0|N|unbounded">
   <xs:simpleType>
      <xs:restriction base="xs:integer">
         <xs:totalDigits value="nDigits"/>
      </xs:restriction>
   </xs:simpleType>`
</xs:element>
```

The value of the mandatory **name** attribute, *propertyName*, is a placeholder for the name of the element being defined which should match the name of feature property being encoded.

The attributes **minOccurs** and **maxOccurs** may be specified and indicate the minimum and maximum number of times that the element must appear in an instance document. If these attributes are omitted, then they are assumed to have the default values defined in XML-Schema for these facets – one (1).

A value of zero for the **minOccurs** attribute shall be used to indicate that a property may have a null value in an instance document.

An element with integer content (or an integer-valued property) must be derived from the base XML-Schema type *xs:integer*.

The maximum number of digits that the integer may have must be specified using the **value** attribute on the <**totalDigits**> element. The value *nDigits* in the XML fragment is a placeholder for the value representing the maximum number digits in the integer.

7.5.2.3 Defining elements with real content

The following XML-Schema fragments show how to define elements with real content in a GML application schema that conforms to this specification[10]:

```
<xs:element name="propertyName" minOccurs="0|N" maxOccurs="0|N|unbounded">
   <xs:simpleType>
      <xs:restriction base="xs:decimal">
         <xs:totalDigits value="nDigits"/>
```

[10] Only one of the three patterns presented should be used to define any given property. However, a conformant schema document may have multiple real valued properties defined in multiple feature types with each definition using a different pattern.

```
            <xs:fractionDigits value="nDigits"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:element>
```

- or -

```
   <xs:element name="propertyName"
               type="xs:float|xs:double"
               minOccurs="0|N" maxOccurs="0|N|unbounded"/>
```

The value of the mandatory **name** attribute, *propertyName*, is a placeholder for the name of the element being defined which should match the name of feature property being encoded.

The use of the attributes **minOccurs** and **maxOccurs** is described in section 7.5.2.2.

A real-valued property (or an element with real content) may be of type *xs:float* or *xs:double* or may be derived, by restriction, from the base XML-Schema type, *xs:decimal*.

In the case where the element definition is derived from the type *xs:decimal*, the maximum number of digits that the real may have must be specified using the **value** attribute on the <**totalDigits**> element. The value *nDigits*, in the XML fragment, is a placeholder for the value representing the maximum number digits in the real.

The maximum number of digits to the right of the decimal point must be specified using the **value** attribute on the <**fractionDigits**> element. The value *nDigits*, in the XML fragment, is a placeholder for the value representing the maximum number of digits to the right of the decimal point.

7.5.2.4 Defining elements with character content

The following XML-Schema fragment shows how to define an element with character content:

```
   <xs:element name="propertyName" minOccurs="0|N" maxOccurs="0|N|unbounded">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:maxLength value="nCharacters"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:element>
```

The value of the mandatory **name** attribute, *propertyName*, is a placeholder for the name of the element being defined which should match the name of feature property being encoded.

The use of the attributes **minOccurs** and **maxOccurs** is described in section 7.5.2.2.

A character string-valued property (or an element with character content) must be derived, by restriction, from the base XML-Schema type *xs:string*.

The *nCharacters* value of the **value** attribute of the <**maxLength**> element must be used to indicate the maximum number of characters that the string may contain. Note that this value may **not** be the same as the maximum length in bytes of the character string. The length of

the string in bytes may be longer than the *nDigits* value if a multi-byte character set is being used.

7.5.2.5 Defining elements with date content

The following XML-Schema fragment shows how to encode an element with date content:

```
<xs:element name="propertyName"
            type="xs:date|xs:dateTime"
            minOccurs="0|N" maxOccurs="0|N|unbounded"/>
```

The value of the mandatory **name** attribute, *propertyName*, is a placeholder for the name of the element being defined which should match the name of feature property being encoded.

The use of the attributes **minOccurs** and **maxOccurs** is described in section 7.5.2.2.

An element that contains date content can be of type *xs:date* or *xs:dateTime* depending on whether time is important or not. The actual instances of date values must be encoded according to the ISO8601 standard.

7.5.2.6 Defining elements with Boolean content

The following XML-Schema fragment shows how to define an element with Boolean content in a GML application schema that conforms to this specification:

```
<xs:element name="propertyName"
            type="xs:boolean"
            minOccurs="0|N" maxOccurs="0|N|unbounded"/>
```

The value of the mandatory **name** attribute, *propertyName*, is a placeholder for the name of the element being defined which should match the name of feature property being encoded.

The use of the attributes **minOccurs** and **maxOccurs** is described in section 7.5.2.2.

The value space of boolean content is {true, 1, false, 0}.

7.5.2.7 Defining elements with binary content

The following XML-Schema fragment shows how to define an element with binary content in a GML application schema that conforms to this specification:

```
<xs:element name="propertyName" minOccurs="0|N" maxOccurs="0|N|unbounded">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:base64Binary|xs:hexBinary">
                <xs:attribute name="url"      type="xs:anyURI" use="optional"/>
                <xs:attribute name="mimeType" type="xs:string" use="required"/>
                <xs:attribute name="role"     type="xs:string" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

The value of the mandatory **name** attribute, *propertyName*, is a placeholder for the name of the element being defined which should match the name of feature property being encoded.

The use of the attributes **minOccurs** and **maxOccurs** is described in section 7.5.2.2.

Binary content can either be referenced from an external URI or encoded inline in base64 or hex format.

When binary data is referenced from an external URI, the **url** attribute must be used to point to the location of the data. The **mimeType** attribute must also be specified to indicate the type or format of binary data that is being referenced. Finally, the optional **role** attribute can be used to assign a user-define role to the data. The role attribute allows complex binary formats like HDF/EOS, which contains multiple independent binary components, to be supported.

When binary data in encoded inline, the **mimeType** attribute must be specified to indicate the type or format of the binary data. The optional **role** attribute can also be specified to assign a user-defined role to the data. In lined binary data is either encoded in base64 format (indicated by defining the type of the element as *xs:base64Binary*) or hex (indicated by defining the type of the element as *xs:hexBinary*).

7.5.2.8 Defining elements with geometric content

The following XML-Schema fragment shows how to define an element with geometric content:

```
<xs:element name="propertyName"
            type="gml_geometric_property_type"
            minOccurs="0|N" maxOccurs="0|N|unbounded">
```

The value of the mandatory **name** attribute, *propertyName*, is a placeholder for the name of the element being defined which should match the name of feature property being encoded.

The use of the attributes **minOccurs** and **maxOccurs** is described in section 7.5.2.2.

The value *gml_geometric_property_type* for the **type** attribute is a placeholder for one of the geometric property types defined in Table 1, clause 6.4.2.

7.5.3 Defining feature types

In clause 7.5.2, XML fragments are presenting for encoding the property values of a feature type in XML. This clause now shows how to combine these fragments to encode feature types in a valid GML application schema.

The following XML-Schema fragment shows how to define a feature type in a GML application schema that conforms to this specification:

```
1  <xs:element name="FeatureTypeName"
2              type="prefix:FeatureTypeName_Type"
3              substitutionGroup="gml:_Feature"/>
```

```
4
5  <xs:complexType name="FeatureTypeName_Type">
6     <xs:complexContent>
7        <xs:extension base="gml:AbstractFeatureType">
8           <xs:sequence>
9
10          … one or more element definitions as described in sec. 7.5.2 …
11
12          </xs:sequence>
13       </xs:extension>
14    </xs:complexContent>
15 </xs:complexType>
```

The root element of the feature type is defined in line 1. The value *FeatureTypeName* is a placeholder for the actual name of the feature type. The root element must be defined to be of type *FeatureTypeName_Type*[11] and must be a substitution element for a GML feature.

Line 5 begins the definition of the XML type that defines the feature type, type. The complex type must be an extension of the GML abstract feature type and must contain one or more element definitions that represent the properties of the feature type. Clauses 7.5.2 describes how to code elements of the various supported content types.

7.6 Comments and annotations

XML comments and annotations, using the <**annotation**> element, may be used freely in an application schema that conforms to this specification wherever they are legally allowed by XML-Schema and GML.

7.7 Property Order

Schemas that comply with this specification may define properties in any order. Thus, generic client applications should be prepared to process schema (and by extension instance) documents that define properties in any order.

However, attention is drawn to the fact that in XML, and thus GML, property order does make a difference. Identical property lists defined in differing order result in different feature type definitions. If property order is significant for a particular application, then this ordering may need to be defined ahead of time.

7.7.1 Example

In the CIPI1.2 tested, the source data was converted from SHAPE files. SHAPE is a multi-file format with separate files used to store the spatial and non-spatial properties. Non-spatial properties are stored in DBF format files and their order is fixed. The order of the non-spatial property, however, is not defined.

As a result, the two servers involved in the CIPI1.2 test bed defined *almost* identical schemas based on this specification differing only in the order of the spatial property. One server

---

[11] The name of the XML type for a feature type must be composed of the name of the feature type and the suffix *_Type*.

defined the spatial property first, the other last. This had no effect on the test bed since the client was able to handle properties in any order. However, the point of this example is to illustrate the fact that the input file format may not completely define the order of properties and if property order is important, it may have to be defined ahead of time.

7.8 Examples

7.8.1 News item example

The following is a GML application schema that defines two features types, *Reporter* and *NewsItem* and complies with the coding patterns described in this document:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema
   targetNamespace="http://www.cubewerx.com/cw"
   xmlns:cw="http://www.cubewerx.com/cw"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:gml="http://www.opengis.net/gml"
   elementFormDefault="qualified"
   version="1.0">

   <xs:import namespace="http://www.opengis.net/gml"
   schemaLocation="http://schemas.cubewerx.com/schemas/gml/3.0.0/feature.xsd"/>

   <!-- ========================================================
        define feature types
        ======================================================== -->
   <xs:element name="Reporter"
               type="cw:Reporter_Type"
               substitutionGroup="gml:_Feature"/>

   <xs:complexType name="Reporter_Type">
      <xs:complexContent>
         <xs:extension base="gml:AbstractFeatureType">
            <xs:sequence>
               <xs:element name="reporterId">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:maxLength value="9"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element name="firstName"
                           minOccurs="0" maxOccurs="1">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:maxLength value="20"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element name="lastName"
                           minOccurs="0" maxOccurs="1">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:maxLength value="20"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element name="organization"
                           minOccurs="0" maxOccurs="1">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:maxLength value="50"/>
                     </xs:restriction>
```

```
                  </xs:simpleType>
               </xs:element>
               <xs:element name="email"
                           minOccurs="0" maxOccurs="1">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:maxLength value="50"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element name="age"
                           minOccurs="0" maxOccurs="1">
                  <xs:simpleType>
                     <xs:restriction base="xs:integer">
                        <xs:totalDigits value="10"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element name="photo"
                           minOccurs="0" maxOccurs="1">
                  <xs:complexType>
                     <xs:simpleContent>
                        <xs:extension base="xs:base64Binary">
                           <xs:attribute name="url"
                                         type="xs:anyURI" use="optional"/>
                           <xs:attribute name="mimeType"
                                         type="xs:string" use="required"/>
                           <xs:attribute name="role"
                                         type="xs:string" use="optional"/>
                        </xs:extension>
                     </xs:simpleContent>
                  </xs:complexType>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>

   <xs:element name="NewsItem"
               type="cw:NewsItem_Type"
               substitutionGroup="gml:_Feature"/>

   <xs:complexType name="NewsItem_Type">
      <xs:complexContent>
         <xs:extension base="gml:AbstractFeatureType">
            <xs:sequence>
               <xs:element name="location"
                           type="gml:PointPropertyType"
                           minOccurs="1" maxOccurs="1"/>
               <xs:element name="reporterId"
                           minOccurs="1" maxOccurs="1">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:maxLength value="9"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element name="eventDate"
                           type="xs:dateTime"
                           minOccurs="1" maxOccurs="1"/>
               <xs:element name="byLine"
                           minOccurs="1" maxOccurs="1">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:maxLength value="30"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element name="details"
                           minOccurs="1" maxOccurs="1">
```

```
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:maxLength value="20000"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element name="images"
                           minOccurs="0" maxOccurs="5">
                  <xs:complexType>
                     <xs:simpleContent>
                        <xs:extension base="xs:base64Binary">
                           <xs:attribute name="url"
                                         type="xs:anyURI" use="optional"/>
                           <xs:attribute name="mimeType"
                                         type="xs:string" use="required"/>
                           <xs:attribute name="role"
                                         type="xs:string" use="optional"/>
                        </xs:extension>
                     </xs:simpleContent>
                  </xs:complexType>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```

7.8.2 Roads_bts example from the CIPI1.2 testbed

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified"
   targetNamespace="http://www.opengis.org/cipi1.2/level0/bts"
   xmlns:bts="http://www.opengis.org/cipi1.2/level0/bts"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:annotation>
      <xs:appinfo source="urn:x-bts-roads_bts.xsd:schema:roads-bts:v0.96">roads_bts.xsd v0.96
2003-09</xs:appinfo>
      <xs:documentation>roads_bts schema.</xs:documentation>
   </xs:annotation>
   <xs:import namespace="http://www.opengis.net/gml"

schemaLocation="http://wfs.galdosinc.com:8045/wfs/http?request=GetSchema&amp;urn=urn%3Aopengi
s%3Aspecification%3Agml%3Aschema-xsd%3Agml%3Av3.00"/>
   <xs:element name="FeatureCollection"
      substitutionGroup="gml:_FeatureCollection"
      type="bts:FeatureCollectionType"/>
   <xs:complexType name="FeatureCollectionType">
      <xs:complexContent>
         <xs:extension base="gml:AbstractFeatureCollectionType"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="Roads_bts" substitutionGroup="gml:_Feature"
      type="bts:Roads_btsType"/>
   <xs:complexType name="Roads_btsType">
      <xs:complexContent>
         <xs:extension base="gml:AbstractFeatureType">
            <xs:sequence>
               <xs:element maxOccurs="1" minOccurs="1" name="Objectid_1">
                  <xs:simpleType>
                     <xs:restriction base="xs:integer">
                        <xs:totalDigits value="10"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element maxOccurs="1" minOccurs="1" name="Objectid">
                  <xs:simpleType>
```

```
      <xs:restriction base="xs:integer">
         <xs:totalDigits value="10"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
<xs:element maxOccurs="1" minOccurs="1" name="FNode_">
   <xs:simpleType>
      <xs:restriction base="xs:integer">
         <xs:totalDigits value="10"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
<xs:element maxOccurs="1" minOccurs="1" name="TNode_">
   <xs:simpleType>
      <xs:restriction base="xs:integer">
         <xs:totalDigits value="10"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
<xs:element maxOccurs="1" minOccurs="0" name="LPoly_">
   <xs:simpleType>
      <xs:restriction base="xs:integer">
         <xs:totalDigits value="10"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
<xs:element maxOccurs="1" minOccurs="0" name="RPoly_">
   <xs:simpleType>
      <xs:restriction base="xs:integer">
         <xs:totalDigits value="10"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
<xs:element maxOccurs="1" minOccurs="0" name="Length">
   <xs:simpleType>
      <xs:restriction base="xs:decimal">
         <xs:totalDigits value="27"/>
         <xs:fractionDigits value="8"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
<xs:element maxOccurs="1" minOccurs="1" name="Bdt_roads_">
   <xs:simpleType>
      <xs:restriction base="xs:integer">
         <xs:totalDigits value="10"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
<xs:element maxOccurs="1" minOccurs="1" name="Bdt_roads1">
   <xs:simpleType>
      <xs:restriction base="xs:integer">
         <xs:totalDigits value="10"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
<xs:element maxOccurs="1" minOccurs="0" name="Prefix">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:maxLength value="2"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
<xs:element maxOccurs="1" minOccurs="0" name="Name">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:maxLength value="30"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
<xs:element maxOccurs="1" minOccurs="0" name="Type">
```

```
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:maxLength value="4"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element maxOccurs="1" minOccurs="0" name="Suffix">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:maxLength value="2"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element maxOccurs="1" minOccurs="1" name="Fcc">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:maxLength value="3"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element maxOccurs="1" minOccurs="1" name="Fips">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:maxLength value="11"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element maxOccurs="1" minOccurs="0" name="Shape_len">
                  <xs:simpleType>
                     <xs:restriction base="xs:decimal">
                        <xs:totalDigits value="30"/>
                        <xs:fractionDigits value="15"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element maxOccurs="1" minOccurs="0" name="Geometry"
                  type="gml:CurvePropertyType"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```

ANNEX A – Conformance Testing

A.0 Introduction

This annex outlines how a GML3 application schema can be tested for compliance to this specification.

It is assumed that a method exists for lexically scanning the application schema being tested in order to access all the elements and attributes contained therein.

The following description is only concerned with interpreting the schema definition in order to ascertain if the schema is compliant with this specification.

The best way to test a GML3 application schema for compliance to this specification is to build an XML-Schema interpreter that checks all the elements and attributes defined in the application schema for compliance.

The following clauses present each of the XML-Schema fragments/templates defined in this specification and describe how to test them for conformance.

A.1 Elements to ignore

All XML comments may be ignored. All XML-Schema annotation elements and sub-elements may be ignored.

Attention is drawn to the fact that in this annex, as in the specification, the namespace prefix ***xs*** is used for the XML-Schema namespace. This is purely for illustrative purposes and the prefix ***xs*** should be considered a placeholder for the actual prefix defined in any particular schema instance.

A.2 General Conformance Rules

Only the elements and attributes defined or discussed in this specification may appear in a compliant schema regardless of the implicit elements and attributes that may be defined in XML-Schema or GML.

The order in which element are defined in not important except where the order matters with regard to XML-Schema or GML.

Only opening tags or elements are described in this annex. It is assumed that the schema is well formed and the corresponding closing tags exist.

The order in which attributes appear in opening elements is not important. The order chosen in this document is purely for clarities sake.

A.3 Root element

A compliant application schema must be a valid GML3 application schema, which means that is must be a valid XML-Schema document. The following fragment defines the root element of a compliant XML-Schema document:

```
1 <xs:schema
2    targetNamespace="target_name_space"
3    xmlns:prefix="target_name_space"
4    xmlns:xs="http://www.w3.org/2001/XMLSchema"
5    xmlns:gml="http://www.opengis.net/gml"
6    elementFormDefault="qualified"
7    version="0.0.7">
```

Conformance Rules:

a) the root element must be **xs:schema** (line 1)
b) the attribute **targetNamespace** must be present (line 2)
   - its value is user defined
c) the attribute **xmlns:*prefix*** must be present (line 3)
   - the value of ***prefix*** is user defined
   - the namespace value must be same as target namespace(line 2)
d) the attribute **xmlns:xs** must be present (line 4)
   - its value must be 'http://www.w3.org/2001/XMLSchema'
e) the attribute **xmlns:gml** must be present (line 5)
   - its value must be 'http://www.opengis.net/gml'
f) the attribute **elementFormDefault** must be present
   - its value must be 'qualified'
g) the attribute **version** must be present
   - its value is user defined

A.4 Importing the GML3 feature schema

All features in a compliant application schema must be substitutable for gml features and their definition must be derived from the GML abstract feature definition.

This implies that a compliant application schema must import the GML3 feature schema. The following element must appear in a compliant application schema document:

```
1 <xs:import
2    namespace="http://www.opengis.net/gml"
3    schemaLocation="http://…/schemas/gml/3.0.0/feature.xsd"/>
```

Conformance Rules:

a) an **xs:import** element must be present
b) the attribute **namespace** must be present
   - its value must he 'http://www.opengis.net/gml'
c) The attribute **schemaLocation** must be present
   - its value must be a valid URI reference to feature.xsd

A.5 Feature types

A.5.1 Root element

A compliant application schema must define one or more feature types by defining one or more root elements for those feature types.

The root element for a feature type is defined by the following XML-Schema fragment:

```
1  <xs:element name="FeatureTypeName"
2              type="prefix:FeatureTypeName_Type"
3              substitutionGroup="gml:_Feature"/>
```

Conformance Rules:

a) an **xs:element** element must be present for each feature type defined in the application schema
b) the attribute **name** must be present
   - its value is user defined and represents the name of the feature type
c) the attribute **type** must be present
   - its value must be the name of a complex type defined elsewhere in the document
     - the value must follow the following pattern: 'prefix:FeatureTypeName_Type'
       - the *prefix* must match the target namespace prefix defined in the root element of the schema document
       - the *FeatureTypeName* is the same as the value of the name attribute
       - the suffix must be the literal '_Type'
d) the attribute **substitutionGroup** must be present
   - its value must be 'gml:_Feature'

A.5.2 Complex type

A complex type must be defined that correspond to the value of the **type** attribute in the definition of the root element of each feature type.

```
1  <xs:complexType name="FeatureTypeName_Type">
2     <xs:complexContent>
3        <xs:extension base="gml:AbstractFeatureType">
4           <xs:sequence>
5
6           …one or more element definitions as described in sec. 7.5.2 …
7
8           </xs:sequence>
9        </xs:extension>
10    </xs:complexContent>
11 </xs:complexType>
```

Conformance Rules:

a) a **complexType** element must be present to define the XML type of each feature type
b) the attribute **name** must be present
   - the value must follow the pattern in validation rule A.5.1(c)
c) the element **xs:complexContent** must be present
d) the element **xs:extension** must be present
e) the attribute **base** must be present
   - its value must be 'gml:AbstractFeatureType'
f) the element **xs:sequence** must be present
g) one or more property definitions must follow

A.5.3 Properties

A.5.3.1 Integer valued properties

Each integer valued property must be defined using the following XML-Schema fragment:

```
1   <xs:element name="propertyName"
2               minOccurs="0|N" maxOccurs="0|N|unbounded">
3      <xs:simpleType>
4         <xs:restriction base="xs:integer">
5            <xs:totalDigits value="nDigits"/>
6         </xs:restriction>
7      </xs:simpleType>`
8   </xs:element>
```

Conformance Rules:

a) the element **xs:element** must be present (line 1)
b) the attribute **name** must be present (line 1)
   - it value is user defined and represent the name of property
c) the attribute **minOccurs** may be present (line 2)
   - if it is present, its value must be 0 or some integer N
   - if it is not present, the default value is 1
d) the attribute **maxOccurs** may be present (line 2)
   - if it is present, its value must 0 or some integer N or 'unbounded'
   - if it is not present, the default value is 1
e) the element **xs:simpleType** must be present (line 3)
f) the element **xs:restriction** must be present (line 4)
g) the attribute **base** must be present (line 4)
   - its value must be 'xs:integer'
h) the element **xs:totalDigits** must be present (line 5)
i) the attribute **value** must be present (line 5)
   - its value is user defined and represent the number of digits in the int

A.5.3.2 Real valued properties

```
1 <xs:element name="propertyName"
2               minOccurs="0|N" maxOccurs="0|N|unbounded">
3    <xs:simpleType>
4      <xs:restriction base="xs:decimal">
5          <xs:totalDigits value="nDigits"/>
```

```
6          <xs:fractionDigits value="nDigits"/>
7       </xs:restriction>
8     </xs:simpleType>
9 </xs:element>
```

Conformance Rules:

a) the element **xs:element** must be present (line 1)
b) the attribute **name** must be present (line 1)
   - it value is user defined and represent the name of property

c) the attribute **minOccurs** may be present (line 2)
   - if it is present, its value must be 0 or some integer N
   - if it is not present, the default value is 1

d) the attribute **maxOccurs** may be present (line 2)
   - if it is present, its value must 0 or some integer N or 'unbounded'
   - if it is not present, the default value is 1

e) the element **xs:simpleType** must be present (line 3)
f) the element **xs:restriction** must be present (line 4)
g) the attribute **base** must be present (line 4)
   - its value must be 'xs:decimal'

h) the element **xs:totalDigits** must be present (line 5)
i) the attribute **value** must be present (line 5)
   - its value is user defined and represent the number of digits in the real

j) the element **xs:fractionDigits** must be present (line 6)
   - its value is user defined and represent the number of digits to the right of the decimal point

A.5.3.3 Character valued properties

```
1 <xs:element name="propertyName"
2             minOccurs="0|N" maxOccurs="0|N|unbounded">
2    <xs:simpleType>
3       <xs:restriction base="xs:string">
4          <xs:maxLength value="nCharacters"/>
5       </xs:restriction>
6    </xs:simpleType>
7 </xs:element>
```

a) the element **xs:element** must be present (line 1)
b) the attribute **name** must be present (line 1)
   - it value is user defined and represent the name of property

c) the attribute **minOccurs** may be present (line 2)
   - if it is present, its value must be 0 or some integer N
   - if it is not present, the default value is 1

d) the attribute **maxOccurs** may be present (line 2)
   - if it is present its value must 0 or some integer N or 'unbounded'
   - if it is not present the default value is 1

e) the element **xs:simpleType** must be present (line 3)
f) the element **xs:restriction** must be present (line 4)
g) the attribute **base** must be present (line 4)
   - its value must be 'xs:string'

h) the element **xs:maxLength** must be present (line 5)

i) the attribute **value** must be present (line 5)
   - its value is user defined and represent the max number of chars

A.5.3.4 Date valued properties

```
1 <xs:element name="propertyName"
2             type="xs:date|xs:dateTime"
3             minOccurs="0|N" maxOccurs="0|N|unbounded"/>
```

Conformance Rules:

a) the element **xs:element** must be present (line 1)
b) the attribute **name** must be present (line 1)
   - it value is user defined and represents the name of property
c) the attribute **type** must be present (line 2)
   - it value must be xs:date OR xs:dateType
d) the attribute **minOccurs** may be present (line 3)
   - if it is present, its value must be 0 or some integer N
   - if it is not present, the default value is 1
e) the attribute **maxOccurs** may be present (line 3)
   - if it is present, its value must 0 or some integer N or 'unbounded'
   - if it is not present, the default value is 1

A.5.3.5 Boolean valued properties

```
1 <xs:element name="propertyName"
2             type="xs:boolean"
3             minOccurs="0|N" maxOccurs="0|N|unbounded"/>
```

Conformance Rules:

a) the element **xs:element** must be present (line 1)
b) the attribute **name** must be present (line 1)
   - it value is user defined and represent the name of property
c) the attribute **type** must be present (line 2)
   - it value must be xs:boolean
d) the attribute **minOccurs** may be present (line 3)
   - if it is present, its value must be 0 or some integer N
   - if it is not present, the default value is 1
e) the attribute **maxOccurs** may be present (line 3)
   - if it is present, its value must 0 or some integer N or 'unbounded'
   - if it is not present, the default value is 1

A.5.3.6 Binary valued properties

```
1  <xs:element name="propertyName" minOccurs="0|N" maxOccurs="0|N|unbounded">
2     <xs:complexType>
3        <xs:simpleContent>
4           <xs:extension base="xs:base64Binary|xs:hexBinary">
5              <xs:attribute name="url"      type="xs:anyURI" use="optional"/>
6              <xs:attribute name="mimeType" type="xs:string" use="required"/>
7              <xs:attribute name="role"     type="xs:string" use="optional"/>
8           </xs:extension>
9        </xs:simpleContent>
```

```
10    </xs:complexType>
11 </xs:element>
```

Conformance Rules:

a) the element **xs:element** must be present (line 1)
b) the attribute **name** must be present (line 1)
   - it value is user defined and represent the name of property
c) the attribute **minOccurs** may be present (line 1)
   - if it is present, its value must be 0 or some integer N
   - if it is not present, the default value is 1
d) the attribute **maxOccurs** may be present (line 1)
   - if it is present, its value must 0 or some integer N or 'unbounded'
   - if it is not present, the default value is 1
e) the element **xs:complexType** must be present (line 2)
f) the element **xs:simpleContent** must be present (line 3)
g) the element **xs:extension** must be present (line 4)
h) the attribute **base** must be present (line 4)
   - its value must one or xs:base64Binary or xs:hexBinary
i) the element **xs:attribute** must be present (line 5)
j) the attribute **name** must be present (line 5)
   - its value must be 'url'
k) the attribute **type** must be present (line 5)
   - its value must be 'xs:anyURI'
l) the attribute **use** must be present (line 5)
   - its value must be 'optional'
m) the element **xs:attribute** must be present (line 6)
n) the attribute **name** must be present (line 6)
   - its value must be 'mimeType'
o) the attribute **type** must be present (line 6)
   - its value must be 'xs:string'
p) the attribute **use** must be present (line 6)
   - its value must be 'required'
q) the element **xs:attribute** must be present (line 7)
r) the attribute **name** must be present (line 7)
   - its value must be 'role'
s) the attribute **type** must be present (line 7)
   - its value must be 'xs:string'
t) the attribute **use** must be present (line 7)
   - its value must be 'optional'

A.5.3.7 Geometry value properties

```
1 <xs:element name="propertyName"
2             type="gml_geometric_property_type"
3             minOccurs="0|N" maxOccurs="0|N|unbounded">
```

Conformance Rules:

a) the element **xs:element** must be present (line 1)
b) the attribute **name** must be present (line 1)
   - it value is user defined and represent the name of property
c) the attribute **type** must be present (line 2)

- it value must be one of: gml:PointPropertyType, gml:CurvePropertyType, gml:SurfacePropertyType, gml:MultiCurvePropertyType, gml:MultiSurfacePropertyType, gml:LinearRingPropertyType, gml:RingPropertyType

d) the attribute **minOccurs** may be present (line 3)

- if it is present, its value must be 0 or some integer N
- if it is not present, the default value is 1

e) the attribute **maxOccurs** may be present (line 3)

- if it is present, its value must 0 or some integer N or 'unbounded'
- if it is not present, the default value is 1

A.6 Schema Validator

A schema validation tool has been built by CubeWerx Inc., which checks schemas for conformance to this specification. The URL of the tool is:

http://demo.cubewerx.com/gml3l0/cwgml3l0.cgi

The schema validator accepts the following parameters:

Table 5 – Parameters for CWGML3L0 Schema Validator

PARAMETER	M/O	DESCRIPTION
SERVER **(mutually exclusive with URI)**	M	This is the URL to a compliant WFS. The validator will automatically generate a *DescribeFeatureType* request using this URL to obtain a description of all feature types using *x-application/gml:3:0* as the **outputFormat**.
URI **(mutually exclusive with SERVER)**	M	This is the URI to a resource that is a compliant GML3 level 0 schema. This may be a reference to a flat file or a service that generates the schema, etc… In this instance, the service will use the URI as is and does not modify it in any way.

ANNEX B – Future Work

1. Need to update NULL handling to reflect what is being done in GML3.

ANNEX C – Usage Narrative

Fred T. Programmer has 5 years of for-hire programming experience and is a wizard at writing easy-to-use, easy-to-maintain graphical applications in Visual Basic. He brings to the project an extensive set of VB widgets for graphic object display and manipulation. He is familiar with maps and has moderate experience writing applications that deal with digital vector data (e.g., SDTS, Shape, DGN, MapObjects, etc). He has read a lot about XML and XML tools but has never written an application that manipulates XML documents. He has experience writing code to access services on the Web via HTTP GET/POST. He has read the WFS 1.0 Implementation Specification. He has not read the GML2 or GML3 specifications.

Fred is contracted to write, in 3 days(!), a general-purpose map viewer client application called "Planet3" that is designed to allow users to:

- display geometry for any GML-encoded geographic features it gets its hands on,
- label the graphic representations of features with the name of the feature types and/or the value of any of their attributes
- interactively assign symbols (marker, line, fill) to individual features or sets of features for local display,
- point to a graphic representation of a feature instance with an input device and view a report of its attribute names, types and values.
- modify the attribute value (including geometry) of one or more feature instances and commit the changes back to the source

Fred must not make assumptions about the potential data sources, data models, feature types, semantics, geographic extent, CRS, quality or fitness-for-use of the data that may ultimately be manipulated by users of his program. Initially, the application program must be able to access (for use as described above and without modification to code or client configuration) data from three different WFSTs. Each web feature service is serving transportation data` (i.e., road, airfield, railroad, transit features) encoded as GML that, in their native stores, conform to the data product specifications for USGS DLG "7.5 minute", US Census TIGER/Line and NIMA/DIGEST VPF0, respectively. Planet3, must also support other unknown themes and data “products”, again without modification to code, as they come online (e.g., political boundaries, hydrography, cadastral themes from local, state, national, international and commercial providers).

Assumption: for purposes of "entry-level" data transfer and simple map display by their customers, all data providers will agree to serve, via WFSTs, their data products using the proposed "Level 0 GML Profile". They may of course also choose to support serving the

same data, represented using "high-value" community- or use-specific GML application schema, for use by applications that are more sophisticated and customers.

Our premise: Fred the programmer is totally focused on writing a useful and user-friendly application that, regardless of the source of the data, will satisfy all the functional behaviours described above. As new data sources/products come online, Fred must not have to modify his code. Rather than writing code that can flawlessly handle all possible ways in which these data can be structured and encoded in GML for specialized purposes, the objective of this CIPI1.2 work item is to define a common "Level 0" profile of GML3 that specifies a simple and sufficient data model, structure and/or set of encoding rules that the programmer can assume will be supported by all WFSTs his application will access. In this way, Fred can focus on the important functional parts of the "Planet3" application rather than on interoperability issues with data encodings and service access.

Bibliography

[7] OGC Document 99-049, *OpenGIS® Simple Features Specification for SQL*, Revision 1.1, http://www.opengis.org/techno/specs/99-049.pdf, May 1999.

[8] IETF RFC 1521, N. Borenstein, et al., *MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*, http://www.ietf.org/rfc/rfc1952.txt, September 1993.

[9] W3C, David C. Fallside (ed.), *XML Schema Part 0: Primer*, http://www.w3.org/TR/xmlschema-0/, May 2001.