Open Geospatial Consortium Inc.

Date: 2006-03-09

Reference number of this OGC® initiative document: **OGC 05-102r1**

Version: 0.0.5

Category: OpenGIS® Discussion Paper

Editors: David S. Burggraf, Stan Tillman

OWS 3 GML Topology Investigation

Document type: OpenGIS® Discussion Paper
Document stage: Draft
Document language: English

Contents

Figures Page

Tables

Preface

Suggested additions, changes, and comments on this draft report are welcome and encouraged. Such suggestions may be submitted by OGC portal message, email message, or by making suggested changes in an edited copy of this document.

The changes made in this document version, relative to the previous version, are tracked by Microsoft Word, and can be viewed if desired. If you choose to submit suggested changes by editing this document, please first accept all the current changes, and then make your suggested changes with change tracking on.

i. Submitting organizations

The following organizations submitted this document to the Open GIS Consortium Inc:

- Galdos Systems Inc.
- Intergraph Corporation

ii. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Contact	Company	Address
D. Burggraf, PhD	Galdos Systems Inc.	1300-409 Granville St. Vancouver, BC V6C 1T2 Canada
Darko Androsevic	Galdos Systems Inc.	1300-409 Granville St. Vancouver, BC V6C 1T2 Canada
Stan Tillman	Intergraph Corp.	

iii. Revision history

Date	Release	Editor	Primary clauses modified	Description
2005-31-08	0.0.1	S. Tillman	All	First draft
2005-25-10	0.0.2	S. Tillman	All	Incorporate comments from NGA and Galdos
2005-27-10	0.0.3	D. Burggraf	All	Incorporated content from Galdos investigation, harmonized DIPR documents.
2005-12-21	0.0.4	D. Burggraf	4, 6	Documentation of Galdos GML to Oracle topology development
2006-03-09	0.0.5	M. Kyle	7.5, 7.7	Update after GML WG comments: • Using xlink would make a large impact on the topology encoding size that is documented • An additional alternative to a Topology Service is extending the existing WFS specification

iv. Changes to the OpenGIS® Abstract Specification

The OpenGIS® Geography Markup Language Implementation Specification requires changes to accommodate the technical contents of this document. In particular clauses 6.1.5 *Lossless Topology Representations in GML* and 6.3.2.1 *Mapping GML elements to Oracle Topology* of this report provides suggestions for minimal changes to the GML specification to meet the requirements of lossless topology representations in the plane, which is supported by Oracle Spatial 10g. For more rigorous details on lossless topology representations in the plane see ([3], Kuijpers). The suggested changes are summarized as follows:

1. Introduce a mandatory requirement that a counter-clockwise cyclic order of directedEdges be followed on Node instances (documentation change only).

2. Add an optional Boolean-valued attribute called *universal* on both gml:Face and gml:TopoSolid. The contents of the Note of term 4.84 *universal face* ISO 19107 makes reference to universal faces and their use in applications:

   4.84
   universal face
   unbounded **face** in a 2-dimensional complex
   NOTE The **universal face** is normally not part of any feature, and is used to represent the unbounded portion of the data set. Its interior boundary (it has no exterior boundary) would normally be considered the exterior boundary of the map represented by the data set. This standard does not special case the **universal face**, but application schemas may find it convenient to do so.

Foreword

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

Introduction

The encoding and storage of topological relationships between geospatial features can offer several advantages for the storage and retrieval of geospatial data. A few such advantages are listed as follows:

1. Topological operations—the following questions can be answered using stored topology: (i) Are two polygons adjacent? (ii) Are two nodes connected by path? (iii) How many edges are incident at a node? (iv) What is the counter-clockwise sequence of edges around a node? (v) Are there any gaps in the data?

2. Data integrity—policy enforcement can be put in place that does not allow certain users to change the topology when updating feature data, e.g. roads can be resurveyed but disconnection or reconnection is not allowed.

3. Topological algorithms such as routing applications can be applied directly on the stored topology (using an OGC Web Processing Service or a database front-end application). A weighted network graph (where the weights on edges might be the curve length, expected travel time, scenic index, etc.) can be assembled from the topology model together with the other properties of the feature members in the collection.

Part 1 (Clause 6) of this investigation is conducted by Galdos Systems. In this part, the OWS3 MSD3 geometric description is extended to include a topology encoding as defined by the MSD3 schema. This MSD3 topology model makes use of the three GML topology primitives: Node, Edge, Face, and the GML topological "expressions": TopoPoint, TopoCurve, and TopoSurface. This topology enhanced GML dataset is to be supported by the Galdos Cartalinea WFS with an Oracle Spatial 10g database. The main purpose of this investigation is to determine if the GML topology representation in Oracle topology is lossless.

Part 2 (Clause 6.2) of this investigation is conducted by Intergraph Corp. and describes and discusses the impacts of encoding topology within the GML data. Although this study addresses a number of issues dealing with GML encoded topology, the basis for the study is in response to the paper written by John Vincent and David Danko titled, "Impacts of Topology on WFS Transactions: Observations from OWS2 Information Interoperability Thread".

OWS 3 GML Topology Investigation

1 Scope

This OpenGIS® document consists of two parts.

Part 1 (Clause 6) of this investigation is conducted by Galdos Systems. In this part, the OWS3 MSD3 geometric description is extended to include a topology encoding as defined by the MSD3 schema. The topology enhanced dataset is to be inserted into an Oracle Spatial 10g database using the Galdos Cartalinea WFS. The main purpose of this investigation is to determine if the GML topology representation in Oracle topology is lossless.

Part 2 (Clause 6.2) of this investigation is conducted by Intergraph Corp. and describes and discusses the impacts of encoding topology within the GML data. Although this study addresses a number of issues dealing with GML encoded topology, the basis for the study is in response to the paper written by John Vincent and David Danko titled, "Impacts of Topology on WFS Transactions: Observations from OWS2 Information Interoperability Thread".

2 Conformance

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

3 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

ISO DIS 19107, *Geographic Information – Spatial Schema*

OGC 03-105r1, OpenGIS® Geography Markup Language (GML) Implementation Specification, Version 3.1.1, April 2004.

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

4.1 boundary
A **set** that represents the limit of an entity [ISO 19107]

4.2 edge
1-dimensional topological primitive (topological element) [ISO 19107]

4.3 face
2-dimensional topological primitive (topological element) [ISO 19107]

4.4 node
0-dimensional topological primitive (topological element) [ISO 19107]

4.5 set
An unordered collection of related items (objects or values) with no repetition [ISO 19107]

4.6 topology geometry
An Oracle term for a spatial representation of a feature or real world object

5 Conventions

5.1 Symbols (and abbreviated terms)

API	Application Program Interface
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off The Shelf
DCE	Distributed Computing Environment
DCP	Distributed Computing Platform
DCOM	Distributed Component Object Model
GML	Geography Markup Language
ISO	International Organization for Standardization
OGC	Open GIS Consortium
UML	Unified Modeling Language
WFS	Web Feature Service
XML	eXtensible Markup Language

1D One Dimensional

2D Two Dimensional

3D Three Dimensional

5.2 UML notation

Most diagrams that appear in this specification are presented using the Unified Modeling Language (UML) static structure diagram, as described in Subclause 5.2 of the OGC Web Services Common Implementation Specification [OGC 04-016r2].

5.3 Document terms and definitions

This document uses the specification terms defined in Subclause 5.3 of [OGC 04-016r2].

6 Part 1: Galdos Topology Investigation

6.1 Introduction to GML Topology

Topology is a branch of mathematics devoted to the study of the properties of spatial objects that remain invariant under continuous deformation (i.e. “twisting” and “stretching”). In GML, topology is encoded using the topology primitives—nodes, edges, faces, and solids—together with a description of their connective relationships. Topology in its purest sense is not concerned with the position nor the shape of nodes, edges, faces and solids. Hence, GML topology is coordinate free, unlike GML geometry, which explicitly encodes elements such as <pos>, <posList>, <coordinates>. However, GML topology primitives, Node, Edge, Face, and TopoSolid, do have the option of being geometrically realized by the geometry primitives, Point, Curve, Surface, and Solid, respectively, using the GML Object/property encoding pattern: TopologyPrimitive/property/GeometricPrimitive. In this way, the position and shape of feature topology can be implicitly assigned. GML topology is primarily concerned with the encoding of spatial relationships between the primitives and enables the explicit encoding these associations, such as the isolation of a node in a face, coincidence of edges at a node, bounding edges of a face, and the adjacency of faces and solids.

6.1.1 GML Nodes, Edges and Faces

Figure 6-1 shows an example of a simple topology model for a road network. This example has three Nodes (A, B, C), four Edges (a, b, c, d) and two Faces (F1, F2)

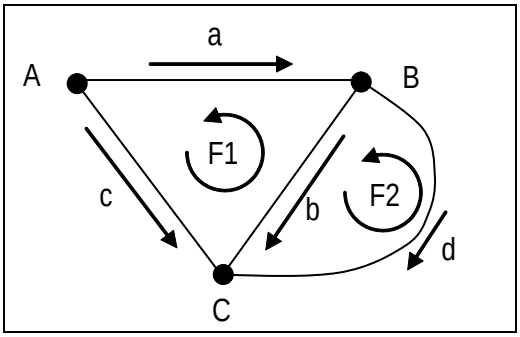


Figure 6-1: A Simple Two-dimensional Topology Network

The minimal instances of the nodes and edges are as follows:

```
<gml:Node gml:id="A"/>
<gml:Node gml:id="B"/>
<gml:Node gml:id="C"/>

<gml:Edge gml:id="a">
      <gml:directedNode orientation="-" xlink:href="#A"/>
      <gml:directedNode orientation="+" xlink:href="#B"/>
</gml:Edge>
```

```
<gml:Edge gml:id="b">
      <gml:directedNode orientation="-" xlink:href="#B"/>
      <gml:directedNode orientation="+" xlink:href="#C"/>
</gml:Edge>

<gml:Edge gml:id="c">
      <gml:directedNode orientation="-" xlink:href="#A"/>
      <gml:directedNode orientation="+" xlink:href="#C"/>
</gml:Edge>

<gml:Edge gml:id="d">
      <gml:directedNode orientation="-" xlink:href="#B"/>
      <gml:directedNode orientation="+" xlink:href="#C"/>
</gml:Edge>
```

In this example, the xlink:href attribute is used to reference Nodes that were previously defined. The orientation attribute is used to assign a negative orientation "-" to some of the nodes signifying that these nodes are at the start of the corresponding edge and positive orientation "+" to signify that these nodes are at the end.

Note that each Edge is inherently directed by its start and end node but can be traversed in two ways: positively or negatively. For example, the directed edge "+a" corresponds to traversing the path along "a" from A to B and "-a" traverses the path from B to A. The directed edge "-a" can be encoded in GML using the directedEdge property which uses the orientation attribute "-", as shown below:

```
<gml:directedEdge orientation="-" xlink:href="#a"/>
```

One of the possible routes from A to B can be expressed as a TopoCurve in GML, which contains a list of directed edges {+c, -b} forming a connected path. An example of this is given in the following instance:

```
<gml:TopoCurve>
      <gml:directedEdge orientation="+" xlink:href="#c"/>
      <gml:directedEdge orientation="-" xlink:href="#b"/>
</gml:TopoCurve>
```

In GML, each face is defined by its boundary, which consists of a list of directed edges. The directed edges in the boundary of each face are traversed in a counter-clockwise direction (following the convention in ISO TC 211/DIS 19107 *Spatial Schema*) as indicated by the arrow surrounding F1 and F2 in Figure 6-1. The orientation of each directed edge in the boundary of a face is either "+" or "-", depending on whether the inherent direction of the edge agrees or disagrees with the counter-clockwise orientation of the face. For example, the boundary of the Face labelled F1, which is traversed counter-clockwise, corresponds to the directed edges in the set {c,-b,-a}. The minimal instance of the faces are encoded as follows:

```
<gml:Face gml:id="F1">
      <gml:directedEdge orientation="+" xlink:href="#c"/>
      <gml:directedEdge orientation="-" xlink:href="#b"/>
      <gml:directedEdge orientation="-" xlink:href="#a"/>
</gml:Face>
```

```
<gml:Face gml:id="F2">
    <gml:directedEdge orientation="+" xlink:href="#b"/>
    <gml:directedEdge orientation="-" xlink:href="#d"/>
</gml:Face>
```

In Figure 6-1, b is the only Edge that has a face on either side of it, that is, both faces F1 and F2 contain the directedEdge b—with either a positive or negative orientation—in their boundary lists. In this case, the faces F1 and F2 are said to be in the co-boundary of b. In GML, the Edge primitive has an optional property called directedFace, whose value must be a Face that is in the co-boundary of the Edge. In the case of planar topology, the left and right co-bounding face of b is distinguished by assigning an orientation. A positive orientation corresponds to the left face and a negative orientation corresponds to the right face. Note that if the orientation of a directedFace in the co-boundary of an Edge is "+", then the Face must contain the directedEdge with the same orientation "+" in its boundary list of directed edges. The encoding of the Edge b that describes its co-boundary information in bold is as follows:

```
<gml:Edge gml:id="b">
    <gml:directedNode orientation="-" xlink:href="#B"/>
    <gml:directedNode orientation="+" xlink:href="#C"/>
    <gml:directedFace orientation="-" xlink:href="#F1"/>
    <gml:directedFace orientation="+" xlink:href="#F2"/>
</gml:Edge>
```

Similarly, each Node can be encoded with a co-boundary list of directedEdges to represent the edges that are incident upon the Node. A positive orientation on directedEdge corresponds to an edge that points towards the Node and a negative orientation corresponds to an edge emanating from the Node. For example, the co-bounding edges of Node B from Figure 6-1 are encoded as:

```
<gml:Node gml:id="B">
    <gml:directedEdge orientation="+" xlink:href="#a"/>
    <gml:directedEdge orientation="-" xlink:href="#b"/>
    <gml:directedEdge orientation="-" xlink:href="#d"/>
</gml:Node>
```

6.1.2 More Complex Topology Primitives

The co-boundary of a Face is a list of directed TopoSolids. For example, consider the Face F that represents the equatorial plane of the solid ball in Figure 6-2.

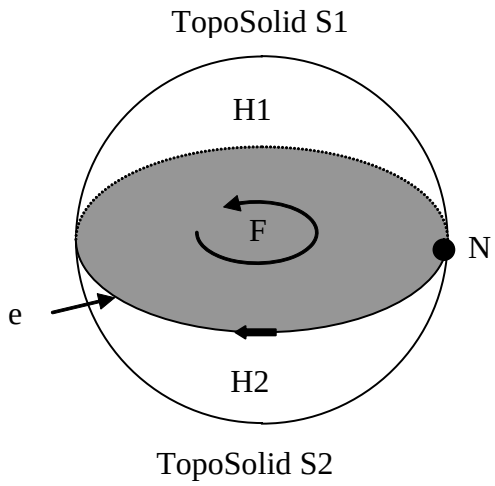


Figure 6-2: Two Solid half-balls in the Co-Boundary of a Face

The solid upper and lower half-balls are represented by the TopoSolids S1 and S1 respectively and taken together they form the solid ball shown in Figure 6-2. The boundary of S1 consists of two faces, representing the upper hemisphere H1 and the equatorial plane F. The TopoSolid S2 also has two faces in its boundary, F (with opposite orientation as used with S1) and the lower hemisphere H2. The Face F and the two TopoSolids S1 and S2 can be encoded as follows:

```
<gml:Face gml:id="F">
    <gml:directedEdge orientation="-">
        <gml:Edge gml:id="e">
            <gml:directedNode orientation="-">
                <gml:Node gml:id="N"/>
            </gml:directedNode>
            <gml:directedNode orientation="+" xlink:href="#N"/>
        </gml:Edge>
    </gml:directedEdge>
    <gml:directedTopoSolid orientation="+" xlink:href="#S1"/>
    <gml:directedTopoSolid orientation="-" xlink:href="#S2"/>
</gml:Face>

<gml:TopoSolid gml:id="S1">
    <gml:directedFace orientation="+">
        <gml:Face gml:id="H1">
            <gml:directedEdge orientation="+" xlink:href="#e"/>
        </gml:Face>
    </gml:directedFace>
    <gml:directedFace orientation="+" xlink:href="#F"/>
</gml:TopoSolid>

<gml:TopoSolid gml:id="S2">
    <gml:directedFace orientation="+">
        <gml:Face gml:id="H2">
            <gml:directedEdge orientation="+" xlink:href="#e"/>
        </gml:Face>
    </gml:directedFace>
```

```
        <gml:directedFace orientation="-" xlink:href="#F"/>
    </gml:TopoSolid>
```

The shaded face in Figure 6-3 shows a more complex use of planar topology. The face F3 has an exterior boundary Edge e1 and interior boundary edges e2, e3, e4, and e5. There is also an isolated Node N5 in the interior of F3 that is encoded using the gml:isolated property.

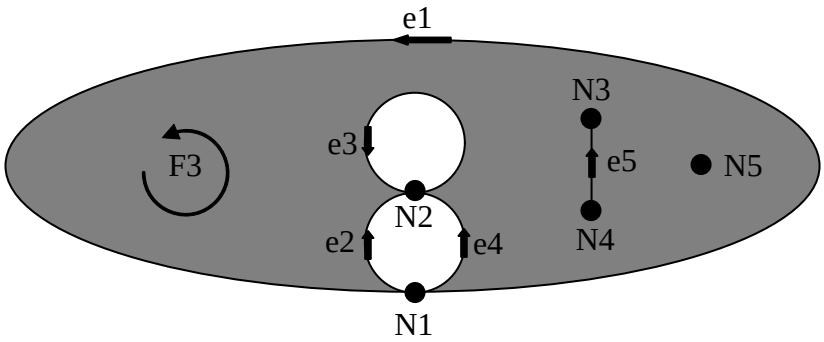


Figure 6-3: Complex Boundary Configuration of a Face

The directed edges "+e1" and "-e3" each form a boundary ring (a simple closed loop in the boundary) of the face F3 that agrees with the counter-clockwise orientation of F3. The directed edges "+e2" and "-e4" together form another boundary ring of F3 in agreement with the orientation of F3 and likewise, the directed edges "+e5" and "-e5" together form another boundary ring. Notice that the "dangling" edge e5 has F3 on both the left and the right and thus e5 occurs twice as a directedEdge in the following definition of F3, once with positive orientation and once with negative orientation. The boundary ring {+e1} in this case is referred to as the "exterior" boundary of F3 and the boundary rings formed by the sets {-e3}, {+e2, -e4}, {+e5, -e5} are all considered "interior" boundary rings following the convention used by ISO TC 211/IS 19107 *Spatial Schema*. Note that the notions of "interior" and "exterior" in the context of boundaries are not explicitly encoded in GML Topology unlike GML Geometry. Note also that any number of boundary rings can intersect at a common Node, but boundary rings cannot intersect along a common Edge. The Face F3 is encoded in GML as follows:

```
<gml:Face gml:id="F3">
    <gml:isolated>
        <gml:Node gml:id="N5"/>
    </gml:isolated>
    <gml:directedEdge orientation="+">
        <gml:Edge gml:id="e1">
            <gml:directedNode orientation="-">
                <gml:Node gml:id="N1"/>
            </gml:directedNode>
            <gml:directedNode orientation="+" xlink:href="#N1"/>
        </gml:Edge>
    </gml:directedEdge>
    <gml:directedEdge orientation="+">
```

```
            <gml:Edge gml:id="e2">
                <gml:directedNode orientation="-">
                    <gml:Node gml:id="N1"/>
                </gml:directedNode>
                <gml:directedNode orientation="+" xlink:href="#N2"/>
            </gml:Edge>
        </gml:directedEdge>
        <gml:directedEdge orientation="-">
            <gml:Edge gml:id="e3">
                <gml:directedNode orientation="-" xlink:href="#N2"/>
                <gml:directedNode orientation="+" xlink:href="#N2"/>
            </gml:Edge>
        </gml:directedEdge>
        <gml:directedEdge orientation="-">
            <gml:Edge gml:id="e4">
                <gml:directedNode orientation="-">
                    <gml:Node gml:id="N1"/>
                </gml:directedNode>
                <gml:directedNode orientation="+">
                    <gml:Node gml:id="N2"/>
                </gml:directedNode>
            </gml:Edge>
        </gml:directedEdge>
        <gml:directedEdge orientation="-">
            <gml:Edge gml:id="e5">
                <gml:directedNode orientation="-">
                    <gml:Node gml:id="N4"/>
                </gml:directedNode>
                <gml:directedNode orientation="+">
                    <gml:Node gml:id="N3"/>
                </gml:directedNode>
            </gml:Edge>
        </gml:directedEdge>
        <gml:directedEdge orientation="+" xlink:href="#e5"/>
    </gml:Face>
```

6.1.3 Geometric Realizations

In GML, there is symmetry between the geometric and topological primitives. The geometry primitives—Point, Curve, Surface, and Solid—can be encoded as geometric realizations of—Node, Edge, Face, and TopoSolid, respectively. These geometric realizations are expressed using the following properties: pointProperty, curveProperty, surfaceProperty and solidProperty.

For example, suppose coordinates are assigned to the following simple topology network in Figure 6-4 as shown.

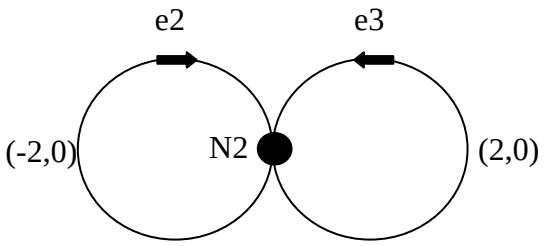


Figure 6-4: Geometric Realization of Topology

The simple topology network can be encoded with a geometric realization as follows:

```
<gml:Node gml:id="N2">
    <gml:pointProperty>
        <gml:Point>
            <gml:pos>0 0</gml:pos>
        </gml:Point>
    </gml:pointProperty>
</gml:Node>

<gml:Edge gml:id="e2">
    <gml:directedNode orientation="-" xlink:href="#N2"/>
    <gml:directedNode orientation="+" xlink:href="#N2"/>
    <gml:curveProperty>
        <gml:Curve>
            <gml:segments>
                <gml:Circle>
                    <gml:coordinates>0,0 –2,0 -1,1</gml:coordinates>
                </gml:Circle>
            </gml:segments>
        </gml:Curve>
    </gml:curveProperty>
</gml:Edge>

<gml:Edge gml:id="e3">
    <gml:directedNode orientation="-" xlink:href="#N2"/>
    <gml:directedNode orientation="+" xlink:href="#N2"/>
    <gml:curveProperty>
        <gml:Curve>
            <gml:segments>
                <gml:Circle>
                    <gml:coordinates>0,0 2,0 1,1</gml:coordinates>
                </gml:Circle>
            </gml:segments>
        </gml:Curve>
    </gml:curveProperty>
</gml:Edge>
```

6.1.4 Spatial Representations of a GML Feature

Suppose that the traffic model of a city is stored in a geographic information database and that you have access to this database via a Routing Service. You can send the following

question in the form of a query to the Routing Service: "Which route from point A to point B has the fewest intersections along the way?" The answer is naturally found by looking at the topology model of the road network rather than the geometry model. The topology model of the road network encodes intersections as nodes, road segments as edges and the connective relationships between the edges and nodes. One way to answer the query about the optimal route from point A to B is to analyse all the possible paths from A to B along edges in the topology model and count the number of nodes traversed along each path in order to find the optimal route. Of course other more sophisticated algorithms exist such as variants of Dijkstra's optimal routing algorithms.

A feature with spatial extent, such as the traffic model, usually has geometry-valued properties, topology valued properties, or both. Topology valued properties of a feature are often accompanied by geometry valued properties as shown in Figure 6-5, but this is not a requirement.

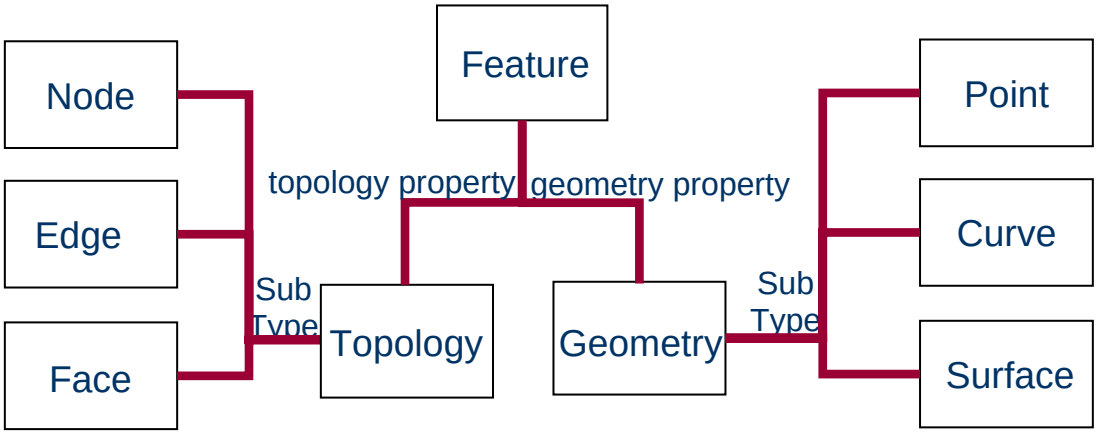


Figure 6-5: A Feature with Separate Geometry and Topology.

As GML Topology is separate from GML Geometry, features can have a stand-alone topology model without any geometric realizations. It is also possible for a feature to have a geometry model that is embedded in its topology model as shown in Figure 6-6.

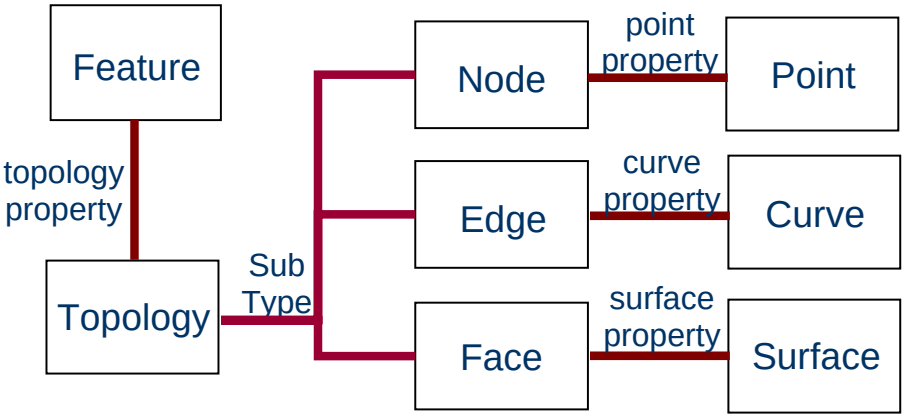


Figure 6-6: A Feature with Merged Geometry and Topology.

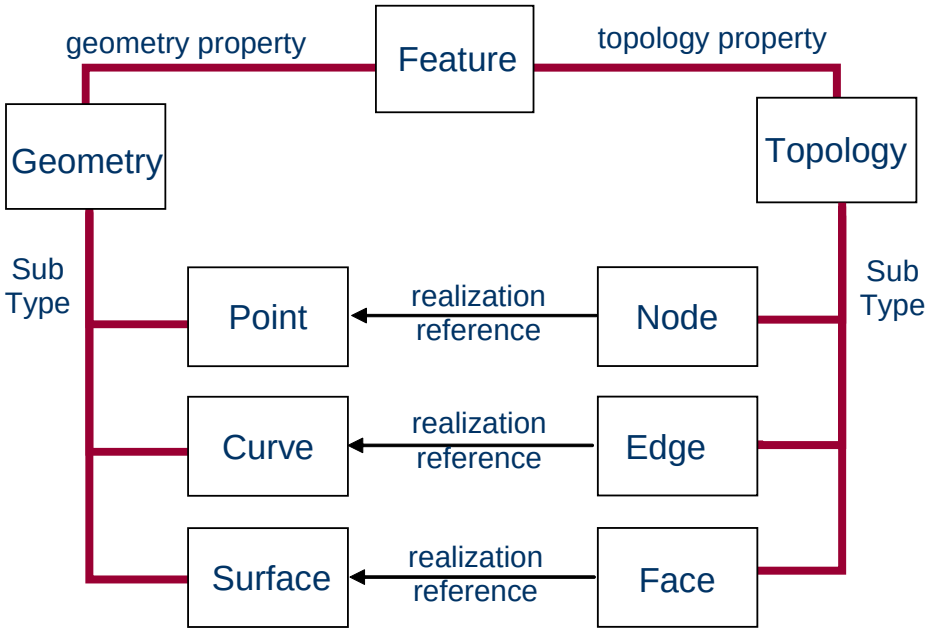


Figure 6-7: A Feature with Separate but linked Geometry and Topology

A feature with two-dimensional spatial extent that has both a topology and geometry model can be organized as shown in Figure 6-5, Figure 6-6, or Figure 6-7. The best organizational strategy depends on the use cases. For example, most, if not all, OGC Web Map Servers will be able to render a map from the geometric data using the organization of Figure 6-5 or Figure 6-7, however fewer will handle the organization of Figure 6-6 because the geometry is more deeply nested in the topology model. An optimal route query as stated in the introduction of this section will likely be easier using the organization of Figure 6-6 or Figure 6-7, since these models enable an application to relate the topological data to the geometric data. Note that the organization of Figure 6-7 assumes an application has support for xlink:href resolution.

If several members of a feature collection have spatial extent, it may be advantageous to collect the topology primitives together with their geometry realizations in a TopoComplex and encode it as a property of the feature collection. The individual topology and geometry property values of the feature members can then reference the corresponding topology and geometry primitives in the TopoComplex. Subclause 6.2.1 provides examples of this.

6.1.4.1 Topology Modelling of a Feature Collection

Suppose a city is modelled as a feature collection and includes the street segments and intersections as shown in Figure 6-8.

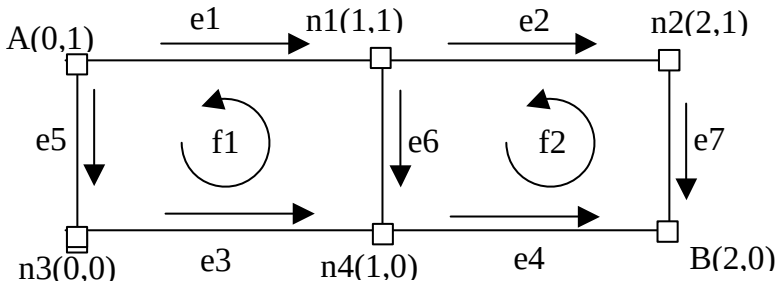


Figure 6-8: Sample Feature Collection.

The topology primitives can be collected together with their corresponding embedded geometry realizations and encoded as a TopoComplex as follows:

```
<gml:TopoComplex gml:id=”TC1”>
    <gml:maximalComplex xlink:href=”#TC1”/>
    <gml:topoPrimitiveMembers>
        <gml:Node gml:id="A">
            <gml:pointProperty>
                <gml:Point gml:id="p0" srsName=“urn:epsg:v6.1:coordinateReferenceSystem:4267”>
                    <gml:coordinates>0,1</gml:coordinates>
                </gml:Point>
            </gml:pointProperty>
        </gml:Node>
            ...
        <!-- additional Nodes -->
            ...
        <gml:Edge gml:id="e1">
            <gml:directedNode orientation=”-” xlink:href=”#A”/>
            <gml:directedNode orientation=”+” xlink:href=”#n1”/>
            <gml:curveProperty>
                <gml:LineString gml:id="c1"
srsName=“urn:epsg:v6.1:coordinateReferenceSystem:4267”>
                    <gml:coordinates>0,1 1,1</gml:coordinates>
                </gml:LineString>
            </gml:curveProperty>
        </gml:Edge>
            ...
        <!-- additional Edges -->
            ...
        <gml:Face gml:id="f1">
            <gml:directedEdge orientation=”-” xlink:href=”#e1”/>
            <gml:directedEdge orientation=”+” xlink:href=”#e5”/>
            <gml:directedEdge orientation=”+” xlink:href=”#e3”/>
            <gml:directedEdge orientation=”-” xlink:href=”#e6”/>
            <gml:surfaceProperty>
                <gml:Polygon gml:id="P1" srsName=“urn:epsg:v6.1:coordinateReferenceSystem:4267”>
                    <gml:exterior>
                        <gml:LinearRing>
                            <gml:coordinates>
                                0,0 1,0 1,1 0,1 0,0
                            </gml:coordinates>
                        </gml:LinearRing>
```

```
                    </gml:exterior>
                </gml:Polygon>
            </gml:surfaceProperty>
        </gml:Face>
    </gml:topoPrimitiveMembers>
</gml:TopoComplex>
```

The City feature collection can now be encoded as follows:

```
<City gml:id="C1">
    <gml:boundedBy>
        <gml:Envelope srsName="urn:epsg:v6.1:coordinateReferenceSystem:4267">
            <gml:pos>0 0</gml:pos>
            <gml:pos>100 100</gml:pos>
        </gml:Envelope>
    </gml:boundedBy>
    <gml:featureMember>
        <Intersection gml:id="I11">
            <description>Howe St 100 block</gml:description>
            <position xlink:href="#A"/>
        <Intersection>
    </gml:featureMember>
        ...
    <gml:featureMember>
        <StreetSegment gml:id="Hwe100">
            <name>Howe St 100 block</gml:name>
            <spatialExtent xlink:href="#e1"/>
        <StreetSegment>
    </gml:featureMember>
        ...
    <gml:featureMember>
        <CityBlock gml:id="B11">
            <spatialExtent xlink:href="#f1"/>
        <CityBlock>
    </gml:featureMember>
        ...
</City>
```

In GML, the following built-in properties can be used to describe topology aggregates and composites in the topology model:

- topoPointProperty
- topoCurveProperty
- topoSurfaceProperty
- topoVolumeProperty

The values of these properties are, respectively:

- TopoPoint
- TopoCurve
- TopoSurface
- TopoVolume

These GML topology types contain lists of directed topology primitives, such as directed nodes, directed edges, directed faces, and directed TopoSolids, respectively.

For example, the bus route feature starting at node A and ending at node B shown in Figure 6-9 can be modelled by the following TopoCurve:

```
<BusRoute gml:id="BRt66">
    <gml:topoCurveProperty>
        <TopoCurve>
            <gml:directedEdge orientation="+" xlink:href="#e5"/>
            <gml:directedEdge orientation="+" xlink:href="#e3"/>
            <gml:directedEdge orientation="-" xlink:href="#e6"/>
            <gml:directedEdge orientation="+" xlink:href="#e2"/>
            <gml:directedEdge orientation="+" xlink:href="#e7"/>
        </TopoCurve>
    </gml:topoCurveProperty>
</BusRoute>
```

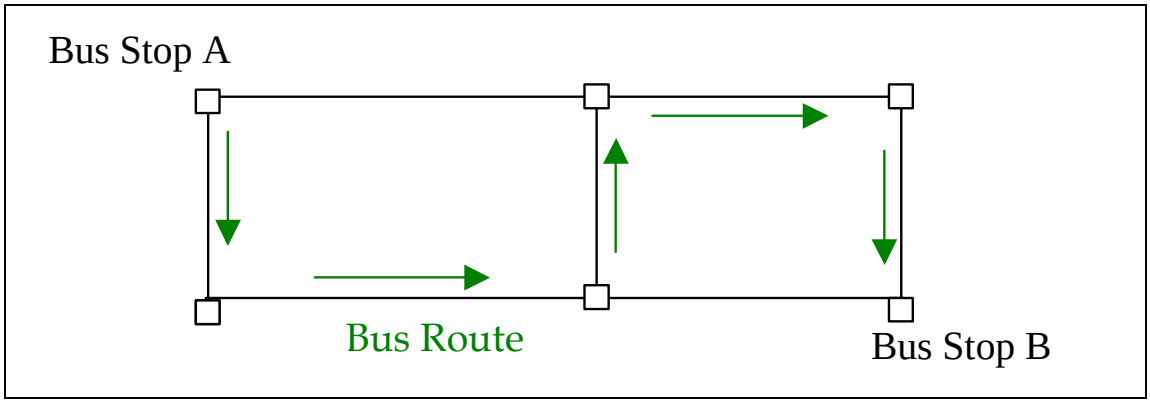


Figure 6-9: Bus Route Member of the City Feature Collection

6.1.5 Lossless Topology Representations in GML

Oracle Spatial 10g supports the topological constructs required to provide a lossless topology representation in the plane as defined by ([3], Kuijpers). To motivate the discussion of lossless topology representations, first consider the two different road networks in Figure 6-10.

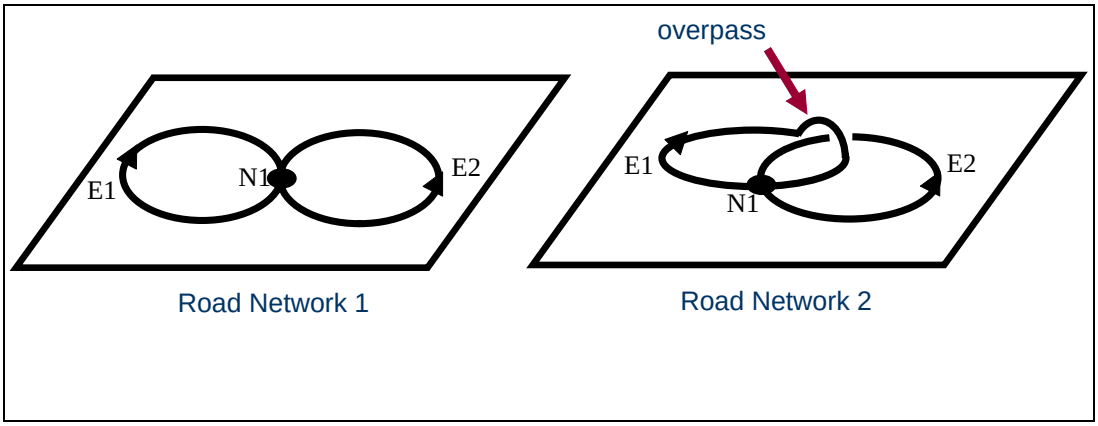


Figure 6-10: A planar and non-planar road network with identical edges and nodes

The topology model of Road Network 1 can be encoded as the following minimal TopoComplex instance:

```
<gml:TopoComplex gml:id="TC2">
    <gml:maximalComplex xlink:href="#TC2"/>
    <gml:topoPrimitiveMembers>
        <gml:Node gml:id="N1"/>
        <gml:Edge gml:id="E1">
            <gml:directedNode orientation="-" xlink:href="#N1"/>
            <gml:directedNode orientation="+" xlink:href="#N1"/>
        </gml:Edge>
        <gml:Edge gml:id="E2">
            <gml:directedNode orientation="-" xlink:href="#N1"/>
            <gml:directedNode orientation="+" xlink:href="#N1"/>
        </gml:Edge>
    </gml:topoPrimitiveMembers>
</gml:TopoComplex>
```

Notice that this encoding applies equally well to Road Network 2 and thus fails to distinguish between the two road networks. An immediately recognizable difference between the two road networks that can be seen in Figure 6-10 is the cyclic order of the incident edges around the central node N1. This difference is exposed by the differing cyclic orders of the directed edges (colour-coded below) in the following co-boundary encoding of N1 for each road network:

Road Network 1:

```
<Node gml:id="N1">
  <directedEdge orientation="+" xlink:href="#E1"/>
  <directedEdge orientation="-" xlink:href="#E1"/>
  <directedEdge orientation="-" xlink:href="#E2"/>
  <directedEdge orientation="+" xlink:href="#E2"/>
</Node>
```

Road Network 2:

```
<Node gml:id="N1">
  <directedEdge orientation="+" xlink:href="#E1"/>
  <directedEdge orientation="+" xlink:href="#E2"/>
  <directedEdge orientation="-" xlink:href="#E1"/>
  <directedEdge orientation="-" xlink:href="#E2"/>
</Node>
```

Note: a cyclic order (e.g. counter-clockwise) of directed edges on the node is not currently required in GML 3.2.0 (or previous versions). A minor change to the documentation in GML that states the requirement of a counter-clockwise cyclic order of directedEdges on the node would enable a formal mapping to the lossless topology representation in Oracle.

Although the co-boundary information of N1 is sufficient to distinguish between the two different topology networks in Figure 6-10, this is not the case for Topology Networks 1 and 2 in Figure 6-11 and Figure 6-12, respectively. Each of the planar topology networks shown in Figure 6-11 and Figure 6-12 have the same node, edge and face descriptions but are not equivalent. The nodes are labelled N1, N2, the edges are labelled E1, E2, E3, and the faces are labelled F1, F2, F3 in each topology network.

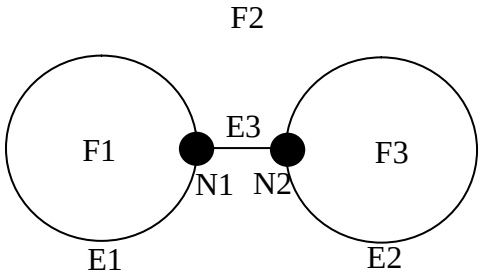


Figure 6-11: Topology Network 1

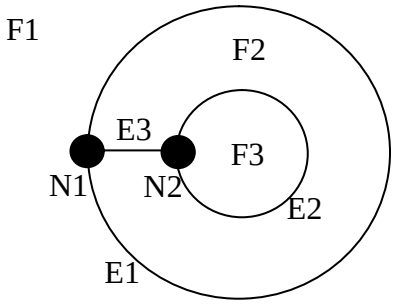


Figure 6-12: Topology Network 2

The GML 3.0 TopoComplex encoding of the topology models in Figure 6-11 and Figure 6-12 are indistinguishable even if the co-boundary information of the two nodes are included. One way to discriminate between them is to use the lossless representation of planar topological data introduced in the paper ([3], Kuijpers). If two non-equivalent planar embeddings of topology networks are represented the same way, then some information is lost in the planar topology representation. The representation introduced ([3], Kuijpers) can distinguish between any two non-equivalent planar topology networks; hence the representation is referred to as *lossless* on planar topological data (on the other hand it is not lossless on topological data embedded in more general surfaces as shown in Figure 6-13). The topological representation introduced in ([3], Kuijpers) is called a *PLA Structure* and generalizes the common Point-Line-Area (PLA) representation by encoding the universal face (the unique face with infinite area in the planar topology network, for example F2 and F1 are the universal faces in Topology Networks 1 and 2, respectively) in addition to the list of *node observations*. A *node observation* ([3], Kuijpers) at a node n0 is a sequence of incident edges and adjacent faces arranged in counter-clockwise cyclic order around n0 (following a common convention used for co-boundary edges, which agrees with ISO 19107 and GML). For example, the

node observation at the node N1 in topology network in both Figure 6-11 or Figure 6-12 consists of the cyclic list [E1, F1, E1, F2, E3, F2]. Suppose that the Topology Networks 1 and 2 are the topology models of Network1 and Network2 feature collections, respectively. The PLA Structure of the topology model of each feature collection can be encoded in GML by adding two properties universalFace and nodeObservations to the feature collection as shown in the following instance:

```
<Network1 gml:id="N1">
      ...
   <universalFace xlink:href="#F2"/>
   <nodeObservations>
      <NodeObservation>
         <node xlink:href="#N1"/>
         <edge xlink:href="#E1"/>
         <face xlink:href="#F1"/>
         <edge xlink:href="#E1"/>
         <face xlink:href="#F2"/>
         <edge xlink:href="#E3"/>
         <face xlink:href="#F2"/>
      </NodeObservation>
      <NodeObservation>
         <node xlink:href="#N2"/>
         <edge xlink:href="#E2"/>
         <face xlink:href="#F3"/>
         <edge xlink:href="#E2"/>
         <face xlink:href="#F2"/>
         <edge xlink:href="#E3"/>
         <face xlink:href="#F2"/>
      </NodeObservation>
   </nodeObservations>
      ...
</Network1>
```

The value of universalFace is the universal face F2 and the value of nodeObservations is an array of NodeObservations. The value of each property of NodeObservation is a topology primitive, where the first topology primitive is the Node at which the observation is made. The remaining values are the incident Edges and adjacent Faces listed in counter-clockwise order surrounding the Node. The PLA Structure of the topology model of Network2 differs only in the value of universalFace as the following instance shows:

```
<Network2 gml:id="N2">
      ...
   <universalFace xlink:href="#F1"/>
   <nodeObservations>
      <NodeObservation>
         <node xlink:href="#N1"/>
         <edge xlink:href="#E1"/>
         <face xlink:href="#F1"/>
         <edge xlink:href="#E1"/>
         <face xlink:href="#F2"/>
         <edge xlink:href="#E3"/>
         <face xlink:href="#F2"/>
```

```
        </NodeObservation>
        <NodeObservation>
            <node xlink:href="#N2"/>
            <edge xlink:href="#E2"/>
            <face xlink:href="#F3"/>
            <edge xlink:href="#E2"/>
            <face xlink:href="#F2"/>
            <edge xlink:href="#E3"/>
            <face xlink:href="#F2"/>
        </NodeObservation>
    </nodeObservations>
        ...
</Network2>
```

The schema fragments corresponding to these instances of the PLA Structure can be modelled as follows:

```
    <element name="Network1" type="app:NetworkType" substitutionGroup="gml:_FeatureCollection"/>
    <element name="Network2" type="app:NetworkType" substitutionGroup="gml:_FeatureCollection"/>
    <!-- ========================================== -->
    <complexType name="NetworkType">
        <complexContent>
            <extension base="gml:AbstractFeatureCollectionType">
                <sequence>
                    <element name="networkTopology" type="gml:TopoComplexMemberType"/>
                    <element name="universalFace" type="app:FacePropertyType"/>
                    <element name="nodeObservations" type="app:NodeObservationArrayPropertyType"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <!-- ========================================== -->
    <complexType name="NodeObservationArrayPropertyType">
        <sequence>
            <element name="NodeObservation" type="app:NodeObservationType"
maxOccurs="unbounded"/>
        </sequence>
    </complexType>
    <!-- ========================================== -->
    <complexType name="NodeObservationType">
        <sequence>
            <element name="node" type="app:NodePropertyType"/>
            <sequence maxOccurs="unbounded">
                <element name="edge" type="app:EdgePropertyType" minOccurs="0"/>
                <element name="face" type="app:FacePropertyType" minOccurs="0"/>
            </sequence>
        </sequence>
    </complexType>
    <!-- ========================================== -->
    <complexType name="NodePropertyType">
        <sequence>
            <element ref="gml:Node" minOccurs="0"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </complexType>
    <!-- ========================================== -->
    <complexType name="EdgePropertyType">
```

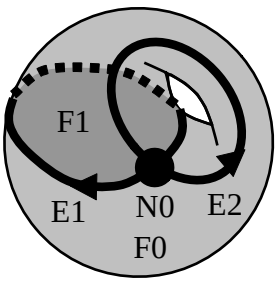
```
        <sequence>
            <element ref="gml:Edge" minOccurs="0"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </complexType>
    <!-- ========================================== -->
    <complexType name="FacePropertyType">
        <sequence>
            <element ref="gml:Face" minOccurs="0"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </complexType>
```

The PLA Structures do not generalize to a lossless representation on more general surfaces such as the torus (sphere with one handle shown below). The two topology networks embedded as shown in Figure 6-13 are not equivalent since the first cannot be continuously deformed into the second (the edge E2 in the first topology model would have to "break through the handle"). There is no universal face, just a single bounded face F0 and only one node observation in each topology model. Furthermore, the node observation at the single node N0 is identical for each topology model.

Topology Model 1 Topology Model 2

Figure 6-13: PLA Structures are not Lossless on General Surfaces.

One method that can be used to distinguish between the topology models in Figure 6-13 is to encode them as 3-dimensional TopoComplexes. For example, in the case of Topology Model 1, the 3-dimensional picture is as shown in Figure 6-14.



TopoSolid S0
inside torus

Figure 6-14: A Lossless Representation of Topology Network 1

The encoding of the TopoComplex TM1 corresponding to Topology Model 1 is as follows:

```
<gml:TopoComplex gml:id="TM1">
    <gml:maximalComplex xlink:href="#TM1"/>
    <gml:topoPrimitiveMembers>
        <gml:Node gml:id="N0"/>
        <gml:Edge gml:id="E1">
            <gml:directedNode orientation="-" xlink:href="#N0"/>
            <gml:directedNode orientation="+" xlink:href="#N0"/>
        </gml:Edge>
        <gml:Edge gml:id="E2">
            <gml:directedNode orientation="-" xlink:href="#N0"/>
            <gml:directedNode orientation="+" xlink:href="#N0"/>
        </gml:Edge>
        <gml:Face gml:id="F0">
            <gml:directedEdge orientation="+" xlink:href="#E2"/>
            <gml:directedEdge orientation="+" xlink:href="#E1"/>
            <gml:directedEdge orientation="-" xlink:href="#E2"/>
            <gml:directedEdge orientation="-" xlink:href="#E1"/>
        </gml:Face>
        <gml:Face gml:id="F1">
            <gml:directedEdge orientation="-" xlink:href="#E1"/>
            <gml:directedTopoSolid orientation="-" xlink:href="#S0"/>
            <gml:directedTopoSolid orientation="+" xlink:href="#S0"/>
        </gml:Face>
        <gml:TopoSolid gml:id="S0">
            <gml:directedFace orientation="+" xlink:href="#F0"/>
            <gml:directedFace orientation="-" xlink:href="#F1"/>
            <gml:directedFace orientation="+" xlink:href="#F1"/>
        </gml:TopoSolid>
    </gml:topoPrimitiveMembers>
</gml:TopoComplex>
```

Note that the Face F1 is bounded by the directedEdge E1 and is cobounded by the TopoSolid S0. The TopoSolid S0 is on either side of the Face F1 and thus occurs twice (with opposite orientations) in the co-boundary of F1. Even more can be gleaned from this encoding. The double occurrence of TopoSolid S0 in the co-boundary of F1 indicates that F1 and its boundary Edge E1 are both contractible to a point in the 3-dimensional topology. This means that the Edge E1 in Topology Model 1 corresponds to the edge that "goes through the tunnel", rather than "forming the bridge" (the Edge E2 that forms the bridge in Topology Model 1 is NOT contractible to a point in the 3-dimensional topology model).

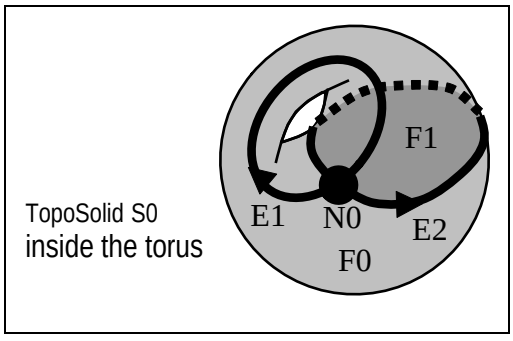


Figure 6-15: A Lossless Representation of Topology Network 2

The following instance of TopoComplex TM2 corresponding to the 3-dimensional Topology Model 2 is noticeably different to that of TM1 in the encoding of the Face F1:

```
<gml:TopoComplex gml:id="TM2">
    <gml:maximalComplex xlink:href="#TM2"/>
    <gml:topoPrimitiveMembers>
        <gml:Node gml:id="N0"/>
        <gml:Edge gml:id="E1">
            <gml:directedNode orientation="-" xlink:href="#N0"/>
            <gml:directedNode orientation="+" xlink:href="#N0"/>
        </gml:Edge>
        <gml:Edge gml:id="E2">
            <gml:directedNode orientation="-" xlink:href="#N0"/>
            <gml:directedNode orientation="+" xlink:href="#N0"/>
        </gml:Edge>
        <gml:Face gml:id="F0">
            <gml:directedEdge orientation="+" xlink:href="#E2"/>
            <gml:directedEdge orientation="+" xlink:href="#E1"/>
            <gml:directedEdge orientation="-" xlink:href="#E2"/>
            <gml:directedEdge orientation="-" xlink:href="#E1"/>
        </gml:Face>
        <gml:Face gml:id="F1">
            <gml:directedEdge orientation="+" xlink:href="#E2"/>
            <gml:directedTopoSolid orientation="-" xlink:href="#S0"/>
            <gml:directedTopoSolid orientation="+" xlink:href="#S0"/>
        </gml:Face>
        <gml:TopoSolid gml:id="S0">
            <gml:directedFace orientation="+" xlink:href="#F0"/>
            <gml:directedFace orientation="-" xlink:href="#F1"/>
            <gml:directedFace orientation="+" xlink:href="#F1"/>
        </gml:TopoSolid>
    </gml:topoPrimitiveMembers>
</gml:TopoComplex>
```

Note that the directedEdge E2 bounds the Face F1 in this case and the double occurrence of the TopoSolid S0 in the co-boundary of F1 indicates that E1 is contractible to a point in the 3-dimensional topology and hence is the "tunnelling" edge in this case.

6.2 Introduction to Oracle Spatial 10g Topology

This clause describes the main elements of the Oracle topology model, its mapping to the GML topology model and a discussion of Oracle topology support within the Cartalinea web feature server. The Oracle topology model is similar to the GML topology model in some ways and different in others. A brief comparison between the Oracle and GML topology models is presented and the challenges in bridging the differences using the Galdos Cartalinea WFS are discussed.

6.2.1 Oracle topology model

The Oracle topology model contains the topological primitives: nodes, edges and faces. In Oracle terminology, a *topology geometry* is a spatial representation of a feature or real world object and is stored as a set of topological primitives. Each *topology geometry* has a unique ID (assigned by Oracle when records are imported or loaded). A *topology geometry layer* consists of *topology geometries*, usually of a specific *topology geometry* type, although it can be a collection of multiple types. A *feature table* contains data for each *topology geometry* layer.

Each *topology geometry* is defined as an object of type SDO_TOPO_GEOMETRY which identifies the *topology geometry* type, *topology geometry* ID, *topology geometry* layer ID, and topology ID for the topology. Topology metadata is automatically maintained by Oracle in the USER_SDO_TOPO_METADATA and ALL_SDO_TOPO_METADATA views. The USER_SDO_TOPO_INFO and ALL_SDO_TOPO_INFO views contain a subset of this topology metadata.

6.2.2 Building the topology

Currently there are two approaches to creating Oracle topology:

- Create topology by explicitly describing the nodes, edges and faces.
- Create topology implicitly, by generating it from existing spatial geometries.

Both approaches apply to the initial topology creation, but for any subsequent modifications, use of PL/SQL or the Java API is strongly recommended by Oracle to ensure spatial integrity, since direct modifications of Oracle topology tables (node, edge, face) will not be synchronized with the rest of the topology model by Oracle.

6.2.2.1 Procedure for creating topology that includes geometry

The procedure for creating Oracle topology using nodes, edges and faces that include geometry are itemized in the following steps. Note that all examples in this clause are informative and only used to illustrate syntax, they are not meant to produce valid topologies.

1. Create the topology, using the SDO_TOPO.CREATE_TOPOLOGY procedure. This causes the <topology-name>_EDGE$, <topology-name>_NODE$, <topology-name>_FACE$, and <topology-name>_HISTORY$ tables to be created.

   Example that creates the topology named 'OWS3_DATA' with Null SRID:

   ```
   EXECUTE SDO_TOPO.CREATE_TOPOLOGY('OWS3_DATA', 0.00005);
   ```

2. Load topology data into the node, edge, and face tables created in Step 1. This is typically done using a bulk-load utility, but it can be done using SQL INSERT statements. Note: The face table needs to have a universal face (faco0 in the example below).

Examples:

```
-- edge
    INSERT INTO OWS3_DATA_edge$ VALUES(1, 1, 1, 1, 1, -1, -1, 1, -1,
      SDO_GEOMETRY(2002, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 2, 1),
  SDO_ORDINATE_ARRAY(8,30, 16,30, 16,38, 3,38, 3,30, 8,30)));

-- node
    INSERT INTO OWS3_DATA_node$ VALUES(1, 1, NULL,
      SDO_GEOMETRY(2001, NULL, SDO_POINT_TYPE(8,30,NULL), NULL, NULL));

-- universe face (id = -1, not 0) This face with id=-1 is always required
    INSERT INTO OWS3_DATA_face$ VALUES(-1, NULL, SDO_LIST_TYPE(-1, -2, 4, 6),
    SDO_LIST_TYPE(), NULL);
-- face
    INSERT INTO OWS3_DATA _face$ VALUES(1, 1, SDO_LIST_TYPE(25, -26), SDO_LIST_TYPE(),
      SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1,1003,3),
        SDO_ORDINATE_ARRAY(3,30, 15,38)));
```

3. Use the SDO_TOPO_GEOMETRY type to represent the *topology geometry* in feature table.

```
CREATE TABLE road (
feature_name VARCHAR2(30) PRIMARY KEY,
topology SDO_TOPO_GEOMETRY);
```

4. Associate the feature tables with the topology, using the SDO_TOPO.ADD_TOPO_GEOMETRY_LAYER procedure for each feature table. This causes the <topology-name>_RELATION$ table to be created

```
EXECUTE SDO_TOPO.ADD_TOPO_GEOMETRY_LAYER('OWS3_DATA', 'road','topology', 'LINE');
```

   As a result, Spatial generates a unique TG_LAYER_ID for each layer in the topology metadata (USER/ALL_SDO_TOPO_METADATA).

5. Initialize topology metadata, using the SDO_TOPO.INITIALIZE_METADATA procedure. (This procedure also creates spatial indexes on the OWS3_DATA_EDGE$, OWS3_DATA_NODE$, and OWS3_DATA_FACE$ tables, and additional B-tree indexes on the OWS3_DATA_EDGE$ and OWS3_DATA_NODE$ tables.

```
EXECUTE SDO_TOPO.INITIALIZE_METADATA('OWS3_DATA');
```

6. Load the feature tables using the SDO_TOPO_GEOMETRY constructor.

```
INSERT INTO road VALUES ('R1', -- Feature name
  SDO_TOPO_GEOMETRY(
    'OWS3_DATA', -- Topology name
    2, -- Topology geometry type (line string)
    3, -- TG_LAYER_ID for this topology (from ALL_SDO_TOPO_METADATA)
    SDO_TOPO_OBJECT_ARRAY (
      SDO_TOPO_OBJECT (9, 2),
      SDO_TOPO_OBJECT (-10, 2),
```

```
    SDO_TOPO_OBJECT (11, 2))) -- edge_ids = 9, -10, 11
);
```

In order to execute this statement the user should know the TG_LAYER_ID generated in the step 4 and all topology elements IDs that belong to this layer.

6.2.2.2 Creating the topology using only spatial geometries (Oracle 10.2 version only)

It is assumed here that the spatial tables already exist.

1. The same as item 1 of 6.2.2.1.

2. Create the universe face

```
-- universe face (id = -1, not 0) This face with id=-1 is always required
INSERT INTO OWS3_DATA_face$ VALUES(-1, NULL, SDO_LIST_TYPE(-1, -2, 4, 6),
SDO_LIST_TYPE(), NULL);
```

3. The same as in item 3 of 6.2.2.1

4. The same as in item 4 of 6.2.2.1

5. Create a TopoMap object and load the whole topology into cache.

```
EXECUTE SDO_TOPO_MAP.CREATE_TOPO_MAP('OWS3_DATA', 'OWS3_DATA_TOPOMAP');
EXECUTE SDO_TOPO_MAP.LOAD_TOPO_MAP('OWS3_DATA_TOPOMAP', 'true');
```

6. Load the feature tables, inserting data from the spatial tables and using the SDO_TOPO_MAP.CREATE_FEATURE function.

```
BEGIN
  FOR street_rec IN (SELECT name, geometry FROM existing_spatal_table) LOOP
   INSERT INTO city_streets VALUES(street_rec.name,
     SDO_TOPO_MAP.CREATE_FEATURE('OWS3_DATA', 'road', 'topology',
        street_rec.geometry));
  END LOOP;
END;
/

CALL SDO_TOPO_MAP.COMMIT_TOPO_MAP();
CALL SDO_TOPO_MAP.DROP_TOPO_MAP('OWS3_DATA_TOPOMAP');
```

7. Initialize topology metadata, using the SDO_TOPO.INITIALIZE_METADATA procedure.

6.2.3 Querying the topology

With the topology data model PL/SQL API, you can use the Oracle Spatial operators, except for the following:

- SDO_RELATE (but you can use the SDO_RELATE convenience operators that do not use the mask parameter)
- SDO_NN
- SDO_NN_DISTANCE
- SDO_WITHIN_DISTANCE

Example:

```
SELECT c.feature_name FROM city_streets c, land_parcels l
  WHERE l.feature_name = 'P3' AND
    SDO_ANYINTERACT (c.feature, l.feature) = 'TRUE';
```

6.2.4 Oracle Feature and Topology Tables

The relationship between feature tables and topology tables in Oracle is illustrated in Figure 6-16.

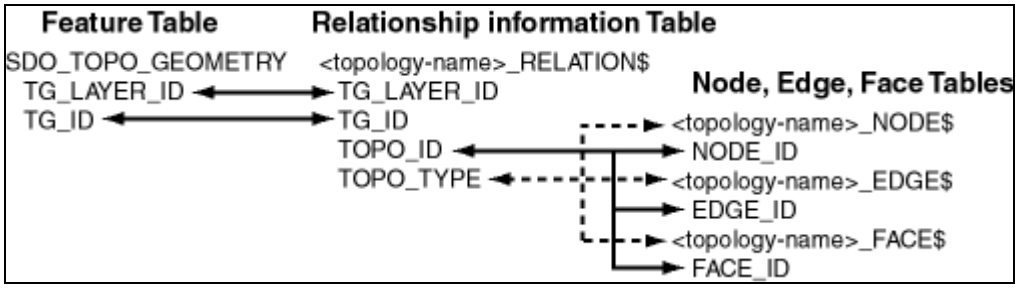


Figure 6-16: Mapping Feature Tables to Topology Tables.

Tables 6-1 to 6-3 describe the Oracle column headings for each topology primitive.

Table 6-1 Column Headings in the <topology-name>_NODE$ Table

Column Name	Data Type	Description
NODE_ID	NUMBER	Unique ID number for this node
EDGE_ID	NUMBER	ID number (signed) of the edge (if any) associated with this node
FACE_ID	NUMBER	ID number of the face (if any) associated with this node

Column Name	Data Type	Description
GEOMETRY	SDO_GEOMETRY	Geometry object (point) representing this node

Table 6-2 Column Headings in the <topology-name>_EDGE$ Table

Column Name	Data Type	Description
EDGE_ID	NUMBER	Unique ID number for this edge
START_NODE_ID	NUMBER	ID number of the start node for this edge
END_NODE_ID	NUMBER	ID number of the end node for this edge
NEXT_LEFT_EDGE_ID	NUMBER	ID number (signed) of the next left edge for this edge
PREV_LEFT_EDGE_ID	NUMBER	ID number (signed) of the previous left edge for this edge
NEXT_RIGHT_EDGE_ID	NUMBER	ID number (signed) of the next right edge for this edge
PREV_RIGHT_EDGE_ID	NUMBER	ID number (signed) of the previous right edge for this edge
LEFT_FACE_ID	NUMBER	ID number of the left face for this edge
RIGHT_FACE_ID	NUMBER	ID number of the right face for this edge
GEOMETRY	SDO_GEOMETRY	Geometry object (line string) representing this edge, listing the coordinates in the natural order for the positive directed edge

Table 6-3 Column Headings in the <topology-name>_FACE$ Table

Column Name	Data Type	Description
FACE_ID	NUMBER	Unique ID number for this face
BOUNDARY_EDGE_ID	NUMBER	ID number of the boundary edge for this face. The sign of this number (which is ignored for use as a

Column Name	Data Type	Description
		key) indicates which orientation is being used for this boundary component (positive numbers indicate the left of the edge, and negative numbers indicate the right of the edge).
ISLAND_EDGE_ID_LIST	SDO_LIST_TYPE	Island edges (if any) in this face. (The SDO_LIST_TYPE type is described in Section 1.6.6.)
ISLAND_NODE_ID_LIST	SDO_LIST_TYPE	Island nodes (if any) in this face. (The SDO_LIST_TYPE type is described in Section 1.6.6.)
MBR_GEOMETRY	SDO_GEOMETRY	Minimum bounding rectangle (MBR) that encloses this face. (This is required, except for the universe face.) The MBR must be stored as an optimized rectangle (a rectangle in which only the lower-left and the upper-right corners are specified). The SDO_TOPO.INITIALIZE_METADATA procedure creates a spatial index on this column.

The object type SDO_TOPO_GEOMETRY is defined as:

```
CREATE TYPE sdo_topo_geometry AS OBJECT
  (tg_type      NUMBER,
   tg_id        NUMBER,
   tg_layer_id  NUMBER,
   topology_id  NUMBER);
```

The Oracle column headings for SDO_TOPO_GEOMETRY are described in Table 6-4.

Table 6-4 SDO_TOPO_GEOMETRY **Type Attributes**

Attribute	Explanation
TG_TYPE	Type of topology geometry: 1 = point or multipoint, 2 = line string or multiline string, 3 = polygon or multipolygon, 4 = heterogeneous collection
TG_ID	Unique ID number (generated by Spatial) for the topology geometry
TG_LAYER_ID	ID number for the *topology geometry* layer to which the *topology geometry*

Attribute	Explanation
	belongs. (This number is generated by Spatial, and it is unique within the *topology geometry* layer.)
TOPOLOGY_ID	Unique ID number (generated by Spatial) for the topology

6.2.5 Editing the topology

The user should always use PL/SQL or Java API to edit topologies. Oracle strongly recommends this to ensure spatial integrity, since direct modifications of Oracle topology tables (node, edge, face) will not be synchronized with the rest of the topology model by Oracle. The editing operations require the oracle topology in-memory cache to be constructed. The "TopoMap" object represents the object that contains the Oracle cache and the "TopoMap" API provides topology-editing operations. The following subset of operations can be executed:

- Adding a Node
- Moving a Node
- Removing a Node
- Removing Obsolete Nodes
- Adding an Edge
- Moving an Edge
- Removing an Edge
- Updating an Edge

Oracle 10.2 also adds additional operations to manipulate topology using geometries:

- ADD_LINEAR_GEOMETRY
- ADD_POINT_GEOMETRY
- ADD_POLYGON_GEOMETRY

6.3 GML to Oracle Topology Development Approaches

There are two possible approaches to support the Oracle topology model in the Cartalinea WFS. These two approaches are detailed in sub-clauses 6.3.1 and **Error! Reference source not found.**.

6.3.1 Build Spatial Topology Tables from Geometry (Oracle 10.2 only)

The first approach is only possible in Oracle 10.2 or higher and entails mapping GML topology elements to Oracle non-topological tables and using the corresponding geometries to build the Oracle topology model.

The feature table would contain a topology reference and will have a column with the object type SDO_TOPO_GEOMETRY.

Example:

```
TOPOSURFACE {
TOPOSURFACE_PROPERTY VARCHAR(255),
TOPOSURFACE_TOPOLOGY  SDO_TOPO_GEOMETRY
}
```

The feature class that contains the topological property would get a new property that will be mapped to the TOPOSURFACE_TOPOLOGY column using a custom user type.

6.3.1.1 Initializing topology

The procedure for initializing the topology is itemized as follows:

1. The topology should be created by calling the SDO_TOPO.CREATE_TOPOLOGY(topology_name, tolerance, srid...) procedure. This will create the topology_name.EDGE$, topology_name.NODE$ and topology_name.FACE$ tables. Note that the "topology_name" parameter specifies the name of the topology.

   Example: SDO_TOPO.CREATE_TOPOLOGY('topology_name', 0.00005);

2. The face table must have a universal face:

   ```
   INSERT INTO topology_name _face$
   VALUES (-1, NULL, SDO_LIST_TYPE(), SDO_LIST_TYPE(), NULL);
   ```

3. Every feature table that can contain a topological element would have a column of type SDO_TOPO_GEOMETRY. The feature table must be associated with the topology by calling

   ```
   SDO_TOPO.ADD_TOPO_GEOMETRY_LAYER('topology_name', 'feature_table_name',
   'column_name',  'type_of_geometry');
   ```

   Note that the the "type_of_geometry" would be POINT, LINE, CURVE, POLYGON, or COLLECTION.

   Example:

   ```
   SDO_TOPO.ADD_TOPO_GEOMETRY_LAYER('topology_name', 'TOPOSURFACE ', '
   TOPOSURFACE _TOPOLOGY',  'POLYGON');
   ```

4. Initialize the topology metadata:

```
SDO_TOPO.INITIALIZE_METADATA('topology_name');
```

The topology initialization can be specified using the DDL scripts and be executed as a precondition for topology support.

6.3.1.2 Inserting topology elements

The procedure for initializing the topology is itemized in the following list. There is no way to insert topology elements after the topology is initialized, so the insert operation here is treated as a special case of the edit operation.

1. Creating and loading "TOPOMAP"

   The "TOPOMAP" object has to be created with the topology loaded into it before any editing operation happens.

   - If you use the PL/SQL API, you can either explicitly create and use the cache or allow Oracle to create and use the cache automatically.
   - If you use the Java API, you must explicitly create and use the cache.

   The "TOPOMAP" object is created by calling SDO_TOPO_MAP.CREATE_TOPO_MAP('topology_name', 'topo_map_name');

   The topology is loaded by calling SDO_TOPO_MAP.LOAD_TOPO_MAP('topo_map_name', 'true');

   Periodically, one can call SDO_TOPO_MAP.UPDATE_TOPO_MAP to validate data and update tables (without a commit). One should collect all geometries that belong to particular topological properties of the feature. For example: a face may have a polygon, an edge may have a line string and a node may have a point.

2. Inserting geometries

   The appropriate SDO_TOPO_GEOMETRY constructor needs to be invoked. The corresponding INSERT SQL statement would be as follows:

   ```
   INSERT INTO TOPOSURFACE VALUES('Vancouver roads',
   SDO_TOPO_MAP.CREATE_FEATURE('topology_name', 'TOPOSURFACE',
   'TOPOSURFACE_TOPOLOGY', geometry_collection));
   ```

   - Note that the geometry_collection parameter is a constructor for the SDO_GEOMETRY object.
   - This CREATE_FEATURE procedure would in turn create all Oracle topological elements but the user has to specify all geometries as part of the SDO_GEOMETRY, which can be a convenience.

Alternatively, one can insert geometries using the following methods:

SDO_TOPO_MAP.ADD_LINEAR_GEOMETRY (curve or line string)

SDO_TOPO_MAP.ADD_POINT_GEOMETRY (point)

SDO_TOPO_MAP.ADD_POLYGON_GEOMETRY (polygon or multipolygon geometry)

Each of these methods returns the id of newly inserted elements. After this, the feature table should be updated using the syntax:

```
INSERT INTO TOPOSURFACE VALUES ('Vancouver roads', -- Feature name
SDO_TOPO_GEOMETRY('topology_name', -- Topology name


    ' TOPOSURFACE ', -- Table name
    'TOPOSURFACE_TOPOLOGY', -- Column name
    4, -- Topology geometry type (collection)
    SDO_TOPO_OBJECT_ARRAY (
    SDO_TOPO_OBJECT (3, 3), -- face_id = 3
    SDO_TOPO_OBJECT (6, 3))) -- face_id = 6
    );
```

This method allows simpler geometry manipulation but on the other hand introduces object IDs into the game.

3. Committing and releasing "topomap"

   Finally, the user should commit the topomap:

```
SDO_TOPO_MAP.COMMIT_TOPO_MAP and drop the topomap
SDO_TOPO_MAP.DROP_TOPO_MAP('topo_map_name').
```

6.3.1.3 Retrieving and querying topology elements

Retrieval of the topology elements would not be changed in Cartalinea since the original GML data storage model is preserved.

Example:

```
SELECT * FROM road a
 WHERE SDO_ANYINTERACT (a.road_topology, SDO_GEOMETRY(...) = 'TRUE');
```

6.3.1.4 Editing topology elements

Same procedure as described in 6.2.5.

6.3.1.5 General observations of this method

The pros and cons of the method presented in 6.3.1 are itemized in the lists below.

Pros:

- Preserves the GML model—preserves the standard GML properties like metadata, name, and description.
- Preserves current Oracle implementations—this method would preserve many currently implemented mappings.
- The Oracle topology model is transparent—since this method uses only geometries to construct topologies one does not have to deal with the complexity mappings from GML to Oracle topological tables.

Cons:

- Updating geometries—currently, there are no methods to update geometries.
- Duplicated data—there will be some duplicated data as a consequence of this method.
- Creating topology without geometries is not possible—this method does not work in the case where a topology is created without geometries.
- Only supported by the Oracle 10.2+ version
- This is not the full round trip—the oracle topology is used only for queries but retrieval is based on the current implementation. **This means that users with already defined topologies in their Oracle data model will not be able to use this approach.**

6.3.2 Direct Representation of GML in Oracle Topology

This approach will require a direct manipulation of the tables in the case of faces, since there is currently no stored procedure or function in Oracle Spatial 10.2g to add a face. However, for any subsequent editing, the stored procedures are used.

6.3.2.1 Mapping GML elements to Oracle Topology

In this mapping the standard GML properties and attributes are ignored, since Oracle topology does support these properties.

6.3.2.1.1 Node type

The GML `NodeType` is defined in XML Schema as follows:

```
<complexType name="NodeType">
  <annotation>
    <documentation> Its optional co-boundary is a set of connected directedEdges.  The orientation of one
of these dirEdges is "+" if the Node is the "to" node of the Edge, and "-" if it is the "from" node.
</documentation>
  </annotation>
  <complexContent>
    <extension base="gml:AbstractTopoPrimitiveType">
      <sequence>
        <element ref="gml:directedEdge" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="gml:pointProperty" minOccurs="0"/>
        <!-- <element name="geometry" type="gml:PointPropertyType" minOccurs="0"/> -->
```

```
    </sequence>
   </extension>
  </complexContent>
 </complexType>
```

Table 6-5 GML/Oracle Node Comparison

GML NODE	ORACLE
pointProperty/Point	GEOMETRY
directedEdge/@orientation	The orientation attribute of type gml:SignType on the directedEdge element expresses the sense in which the referenced edge is used: outward (“-“) or inward (“+”) from the node.
directedEdge[@orientation=’+’]/Edge The orientation attribute of type gml:SignType on the directedEdge element expresses the sense in which the referenced Edge is used. In this case: outward (“-“) or inward (“+”) from the node.	EDGE$. START_NODE_ID
directedEdge[@orientation=’-’]/Edge	EDGE$.END_NODE_ID
directedEdge/Edge/@gml:id	EDGE_ID If EDGE_ID=null (The node is an isolated node)
container/Face/@gml:id	FACE_ID If FACE_ID=null (The node is not an isolated node)
The cyclic order (e.g. counter-clockwise) of cobounding edges on the node is not currently required in GML 3.2.0 (or previous versions). A minor change to the documentation in GML that makes this order mandatory on the node would enable a formal mapping (although indirect) to NEXT_LEFT_EDGE, PREV_LEFT_EDGE, etc. in Oracle.	EDGE$. NEXT_LEFT_EDGE_ID, EDGE$. PREV_LEFT_EDGE_ID, EDGE$.NEXT_RIGHT_EDGE_ID, EDGE$. PREV_RIGHT_EDGE_ID

6.3.2.1.2 Edge type

The GML EdgeType is defined in XML Schema as follows:

```
 <complexType name="EdgeType">
  <annotation>
   <documentation>There is precisely one positively directed and one negatively directed node in the
boundary of every edge. The negatively and positively directed nodes correspond to the start and end
nodes respectively. The optional coboundary of an edge is a circular sequence of directed faces which are
incident on this edge in document order. Faces which use a particular boundary edge in its positive
orientation appear with positive orientation on the coboundary of the same edge. In the 2D case, the
orientation of the face on the left of the edge is "+"; the orientation of the face on the right on its right is "-".
An edge may optionally be realised by a 1-dimensional (curve) geometric primitive.</documentation>
  </annotation>
  <complexContent>
   <extension base="gml:AbstractTopoPrimitiveType">
```

```
      <sequence>
        <element ref="gml:directedNode" minOccurs="2" maxOccurs="2"/>
        <element ref="gml:directedFace" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="gml:curveProperty" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Table 6-6 GML/Oracle Edge Comparison

GML EDGE	ORACLE
directedNode[@orientation='+']/Node/gml:id The orientation attribute of type gml:SignType on the directedNode element expresses the sense in which the referenced Node is used. In this case: end ("-") or start ("+") of the edge.	START_NODE_ID
directedNode[@orientation='-']/Node/gml:id	END_NODE_ID
directedNode[@orientation='+']/Face/gml:id The orientation attribute of type gml:SignType on the directedFace element expresses the sense in which the referenced Face is used. In the planar case: disagrees ("-") or agrees ("+") with Face orientation.	LEFT_FACE_ID
directedNode[@orientation='-']/Face/gml:id	RIGHT_FACE_ID
curveProperty/Polygon	GEOMETRY
directedFace[n], n>2	Not supported in Oracle 10.2 (planar topology only)

6.3.2.1.3 Face type

The GML FaceType is defined in XML Schema as follows:

```
<complexType name="FaceType">
  <annotation>
    <documentation>. The topological boundary of a face consists of a set of directed edges. Note that all
edges associated with a Face, including dangling and interior edges, appear in the boundary. Dangling
and interior edges are each referenced by pairs of directedEdges with opposing orientations. The optional
coboundary of a face is a pair of directed solids which are bounded by this face. If present, there is
precisely one positively directed and one negatively directed solid in the coboundary of every face. The
positively directed solid corresponds to the solid which lies in the direction of the positively directed normal
to the face in any geometric realisation. A face may optionally be realised by a 2-dimensional (surface)
geometric primitive.</documentation>
  </annotation>
  <complexContent>
    <extension base="gml:AbstractTopoPrimitiveType">
      <sequence>
        <element ref="gml:directedEdge" maxOccurs="unbounded"/>
```

```
        <element ref="gml:directedTopoSolid" minOccurs="0" maxOccurs="2"/>
        <element ref="gml:surfaceProperty" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Table 6-7 GML/Oracle Face Comparison

GML FACE	ORACLE
directedEdge[@orientation='+']/Edge/@gml:id The orientation attribute of type gml:SignType on the directedEdge element expresses the sense in which the referenced Edge is used. In this case: disagrees ("-") or agrees ("+") with Face orientation.	EDGE$.LEFT_FACE_ID
directedEdge[@orientation='-']/Edge/@gml:id	EDGE$.RIGHT_FACE_ID
surfaceProperty/Polygon	GEOMETRY
directedTopoSolid	Note supported in Oracle 10.2 (planar topology only)
Not Supported in GML although ISO 19107, term 4.84 states that it may be convenient to support in applications.	FACE_ID=-1 The universe face (id = -1) is required by Oracle

6.3.2.1.4 Supporting GML Features, Topology Expressions and TopoSolid

A feature table would contain the SDO_TOPO_GEOMETRY data type (assuming it is a spatial feature), which is the object type that establishes a relationship between a feature and one or more topological elements. This type may contain a collection of heterogeneous topological elements defined using the constructor and the SDO_TOPO_OBJECT_ARRAY object as a parameter. The SDO_TOPO_OBJECT_ARRAY type is defined as a VARRAY of SDO_TOPO_OBJECT objects. The SDO_TOPO_OBJECT type has the following two attributes: topo_id NUMBER, topo_type NUMBER. The topo_id represents the ID of the element and topo_type represents its type (i.e. 1=node, 2=edge, 3=face). This implies that the user should know the IDs of the topological elements before inserting the SDO_TOPO_GEOMETRY, which can be obtained from a metadata table.

There are no Oracle Topologies that correspond to gml:TopoSolid and the topological "expressions" gml:TopoPoint, gml:TopoCurve, gml:TopoSurface, and gml:TopoVolume. However a feature table and SDO_TOPO_GEOMETRY object type can be used to represent these.

Table 6-8 Oracle Support for Features, Topology Expressions and TopoSolid

GML Element	Oracle Feature Table

TopoCurve	INSERT INTO feature_table VALUES ('route 22', -- Feature name SDO_TOPO_GEOMETRY('BUS_ROUTE', -- Topology name 2, -- Topology geometry type (line string) 3, -- TG_LAYER_ID for this topology (from ALL_SDO_TOPO_METADATA) SDO_TOPO_OBJECT_ARRAY (SDO_TOPO_OBJECT (9, 2), SDO_TOPO_OBJECT (-10, 2), SDO_TOPO_OBJECT (11, 2))) -- edge_ids = 9, -10, 11);
TopoPoint	Similar to Oracle Feature Table for TopoCurve above
TopoSurface	Similar to Oracle Feature Table for TopoCurve above
TopoVolume	Similar to Oracle Feature Table for TopoCurve above
TopoSolid	Similar to Oracle Feature Table for TopoCurve above

6.3.2.1.5 Hierarchical topologies

Hierarchical topologies in Oracle can be used to represent GML TopoComplexes, but these are not used in the MSD3 data model and are considered out of scope in this report.

6.3.2.2 General observations of this method

The pros and cons for the method presented in 6.3.2 are itemized in the lists below.

Pros:

- Directly uses the underlying oracle topology elements for both querying and retrieval for each of the primitives: node, edge, face, so there is no duplicated data. This represents the full round-trip. This means that users with already defined topologies in their Oracle data model can use this approach.

Cons:

- Slightly more calculation overhead is required because of the differences between GML and Oracle topology models.
- Any query that contains a predicate based on the node, edge or face will have to be rewritten to use raw SQL.

6.4 Lossless Topology Representations in Oracle Spatial 10.2g

The explicit encoding of a universal faces together with the winged edge encoding (an edge plus next left, next right, previous left, and previous right edges) yields a lossless

planar topology representation in Oracle Spatial 10g. Lossless topology representations in GML topology requires at a minimum, the specification of a consistent cyclic ordering (e.g. counter-clockwise) of directed edges around a node, and a way to encode a universal face. Note that the consistent cyclic ordering of directed edges around each node *can* be maintained for a GML dataset in an application domain, although it is not required in GML. Also, a universal face *can* be determined implicitly *if* all non-universal faces are encoded in the dataset (again not required in GML)—in this case the boundary of the universal face U consists of the oppositely oriented external boundary edges of the complement, $\overline{U}$. If these two conditions are maintained, a lossless planar topology representation of GML is achievable and this can be mapped to a lossless planar topology representation in Oracle. However this would require some computational overhead in order to determine the Oracle winged edge encoding from the cyclic ordering of GML directed edges incident at each node, and the Oracle universe face encoding from the complement $\overline{U}$ derived from the GML encoding of all faces.

6.5 Summary and Conclusions

Information losses are to be expected in the direct mapping of GML topology to Oracle Spatial 10g topology due to a limitation in Oracle 10.2g (and previous versions) that no more than planar topology is supported. Consequently, gml:TopoSolid cannot be mapped to Oracle topology, rather it must be mapped to a user-defined table as described in 6.3.2.1.4 *Supporting GML Features, Topology Expressions and TopoSolid*. Similarly, Oracle does not support any of the GML topological "expressions": TopoPoint, TopoCurve, TopoSurface, and TopoVolume, nor does it support more than two faces adjacent to an edge, only two adjacent faces can be supported—the left and the right face. The standard GML properties that are inherited on all GML topology primitives, such as metadata, name, and description also cannot be represented by the Oracle topology model—again user-defined tables must be used as described in 6.3.2.1.4. However, the alternative approach described in 6.3.2.1.4 has the drawback of precluding these topologies from being used by topological operations that act on native Oracle topology.

The lack of support in GML topology for universal faces and for cyclic orderings of directed edges around nodes can lead to planar topology representations that are not lossless, even though Oracle is capable of capturing enough information to support lossless topology representations in the plane. The silver lining is that GML geometry captures the information required for a lossless planar topology representation, which can be recovered from geometry in Oracle Spatial 10.2g. Note also that minor enhancements to GML topology can enable the encoding of lossless planar topology representations without reference to GML geometries.

7 Part 2: Integraph Topology Study

As part of this study, Intergraph will respond to the paper titled “Impacts of Topology on WFS transactions: Observations from OWS2 Information Interoperability Thread”. It is Intergraph’s intent to investigate the problems and benefits of including topology within a GML document both persistently and on-the-fly. Following a thorough look at GML topology and how it interacts with other elements of the OGC web services, Intergraph and Galdos will put forth possible solutions.

7.1 Overview

The basis for this study is in response to the paper written by John Vincent and David Danko titled, “Impacts of Topology on WFS Transactions: Observations from OWS2 Information Interoperability Thread” (OGC 04-101). In this paper, questions were raised as to the effectiveness of embedding topology within a GML formatted document. The observations noted were as follows:

1. Most clients ignore topology – preferring to rebuild it, if necessary, when needed. This point stressed the situation in which the “WFS services generate topology information on the fly from feature geometry properties, and encode this topology in GML and build the required links into the feature geometry properties, only to have the clients follow the links to the topology and extract only the geometry definitions, ignoring the topological relationships”. The paper also stressed the fact that this was not representative of all ingesting applications.

2. In a feature based transaction service, modifying a single feature geometry that is part of a topological relationship can cause conceptual and design difficulties. What appears as a simple geometric change can potentially ripple through the geometry definitions of many other feature instances. The question was then raised as to whether the service or client was responsible for maintaining the topological relationships.

3. Additional concerns were raised as to the original design objectives for a WFS. This objective was “to expose a feature model against which transactions could be performed on individual feature instances or, in some cases, groups of feature instances while hiding the details of the underlying implementation from the user”. However, this again causes problems in a transaction-based service because it would require that the client have a more intimate knowledge about the underlying data. For example, did the client request enough of the data to adequately re-build the topology; or was the data retrieved from a persistent topology data store or built on-the-fly topology and how should the edited features be returned?

7.2 Overview of topology

Topology can be defined as the mathematical study of spatial relationships. Its purpose is to capture and retain information concerning a geographic element’s spatial and thematic

relationships with its neighbouring elements. The topology model of a dataset must remain invariant under continuous deformations of the dataset. This means that if the geometry of the data set is changed in size or is otherwise "stretched without being torn", the topology model must remain the same.

One point of clarification that needs to be addressed before beginning is whether the topology being referenced is primitive-based or feature-based. GML, through its inheritance from ISO19107, should be dealing with primitive-based topology. But, as some vendors have a very feature-centric design, the concept of "feature topology" creeps into the encoding of topology within their GML. For the sake of this study, discussions will reflect primitive-based topology as it relates the data content (e.g. if roads and rivers are requested together, then the topology will be constructed for both).

There are two methodologies used in creating topology for a given dataset. The first is known as persistent topology. Persistent topology is created and maintained as part of the dataset. If an element is modified, the underlying topology is automatically modified as well. The second methodology is called "on-the-fly". This method can best be described as one in which the topology is created on an as-needed basis. If an application needs topology, it is the responsibility of that application to create it. Along with other topology topics, this paper will examine whether the topology should be persisted (Persistent) in the GML data or created only when needed (On-the fly).

Persistent topology has a number of benefits that make it an attractive inclusion in one's data model. Some of those benefits include, but may not be limited to:

- Faster data retrieval
- More consistent data
- Built-in Quality Control (QC) providing data integrity
- Easier spatial analysis of the data.

However, persistent topology can also provide some disadvantages, including:

- Lack of consistency between GML vendors causes various "flavors" of GML topology.
- Complexity of building topology often requires splitting the primitives that make up a given feature (e.g. where they intersect with other primitives).
- Size of GML data file can become large and bulky to pass over networks.

These issues, and possibly others, are why so many GML implementations (both ingest and creation) choose to avoid or ignore topological elements defined within the GML specification.

7.3 Response to the Paper

The first summary point the paper pointed out was the fact that most WFS clients ignore topology. This seems to be due to a couple of reasons. One is obvious – a given client might not have any need for topology in the application use of the data. Such clients are only interested in the features and not how they are spatially related. The second reason is that certain clients, especially GIS applications, might have the ability to generate topology on their own. However, the majority of the capabilities provided by the GIS applications do not need the topology – only certain functionality. Therefore, it is more efficient for these clients to build the topology only when a function requiring topology is invoked.

With all of this said, it should be noted that encoding topology within the GML data typically will not impact any of the clients mentioned above. They are simply designed to recognize specific elements and ignore the topological ones.

The paper also addressed issues with encoding topology in a WFS-T service. Since topology defines the spatial relationship between feature instances, modifying a single geometry can actually affect a number of other geometric instances. For example, the client requests a road network in the left of Figure 7.2-1.

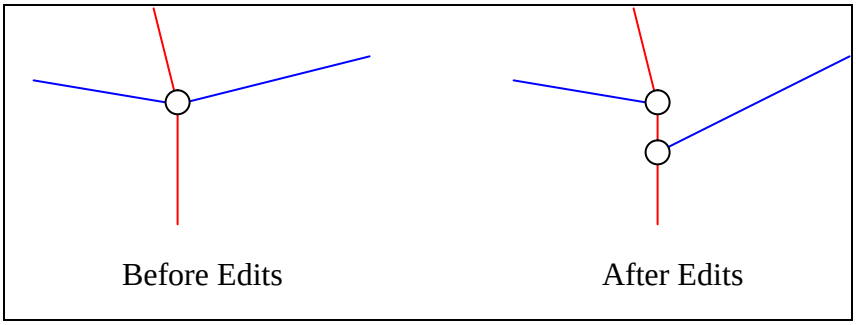


Figure 7.2-1 Data Edits and Impacts on Topology.

In the example of Figure 7.2-1, a simple movement of the endpoint for one blue edge will result at a minimum of breaking the intersecting red edge into two segments. Since topology represents the relationship between neighbouring geometries, decisions will need to be made as to whether the entire intersection should be moved, which affects all four edges and the node, or just the addition of a new intersection node. Who is responsible for knowing this information – the client or the WFS-T server? The problem here is the fact that the application maintaining the underlying data is completely separate from the application making the edits.

This question leads into the final summary point made in the paper. It pointed out that the original design objectives of a WFS was “to expose a feature model against which transactions could be performed on individual feature instances or, in some cases, groups of feature instances while hiding the details of the underlying implementation from the user”. Topology is constructed based on a set of geometric instances. This can be based on a given feature type such as roads, it can be based on a grouping of feature types such

as a transportation theme, or it can be based on a subset of these groupings such as the interstates only. Based on the geometries, the topology can range from very simple to very complex. Therefore, if a user is hidden from "the details of the underlying implementation", then how does the user maintain the topology without knowing the basis on which it was built?

Some GML creation tools currently have the capability to build topology inline with the request. This means that if the user requests the transportation theme, then topology is built on all the feature instances with the transportation theme – if the user request only roads, then topology is built on all road features. This solves the issue of allowing the user to know the basis on which the topology is built. However, it still does not solve the problems encountered within a WFS-T.

In summary, transaction based WFS services are not well suited for encoding topology within the GML it creates. It is virtually impossible for the client to handle the overhead of maintaining the topology and then submit it back to the WFS-T for storage in a way that the work of the WFS-T is simply to store the returned information. The burden must fall on the WFS-T to receive the returned features and interpret how to properly store them in the underlying database. The underlying storage may or may not include topology in its data model. If not, the WFS-T also has the added complexity of building topology when data is requested and then resolving the returned features such that the appropriate updates can be made without topology. The following section addresses the difference between storing GML with encoded topology verses simply encoding it in the GML being transferred.

7.4 GML: Transfer Format vs. Storage Format

During this investigation, a number of opinions about the need for topology were reviewed. Many users of data (both GML formatted data and non-GML formatted data) indicated the importance of topology and the necessity of its availability. However, many of the application developers seemed to have a different opinion indicating that topology should be left to the individual application – if topology is needed, then it can be built for that application. In reviewing cases for this study, it was noticed that one of the reasons for the difference in opinions dealt with how the data was being used. It was apparent that when data reached an "endpoint", the need for topology became more evident. The following serve as examples of what is meant by an endpoint:

- If the data was being pulled from a data store such as a database, then topology was important to ensure data integrity and provide faster queries.
- If the data was being stored in a database, then topology provided some form of quality assurance.
- If the data had been sent to an end user, then topology provided more data consistency and allowed for enhanced spatial analysis capabilities.

The option for including topology is obviously favorable in the environments involving endpoints because there is only one point of interface. This means that data is being

pulled from or pushed to a database through an interface defined by the database itself. If the data is sent to an end user, the end user requires the topology and accepts the overhead as necessary. In addition, the end user is the final point of transfer.

On the other hand, if the data format is being used as a transfer mechanism, all indications are that small and fast are the requirements - small because it makes for a faster transfer.

This is an important distinction to make when discussing the use of topology encoded within a GML document. For the most part, GML is used as a transfer format. When users indicate that topology is highly desired in a given environment, the context of the data usage must be defined first. There are exceptions were data is being stored in GML simply because it can be more openly retrieved by many participants. However, as we in the OGC know, GML is the foundation for transferring feature data over the web.

7.5 Complexity and Duplication Added by Topology

As was mentioned in the previous section, the verbosity of GML is a very important factor, especially when being used as a transfer format. However, the encoding of topology within the GML will definitely add to its size as well as its complexity. For example, if we apply the geometries used earlier, we see that the original data could be simplified to two simple linear geometries. However, by applying topology and breaking the edges at the intersecting points, we double the number of edges by creating two red and two blue line features. In addition, it also adds a new geometric node element to the intersect location.

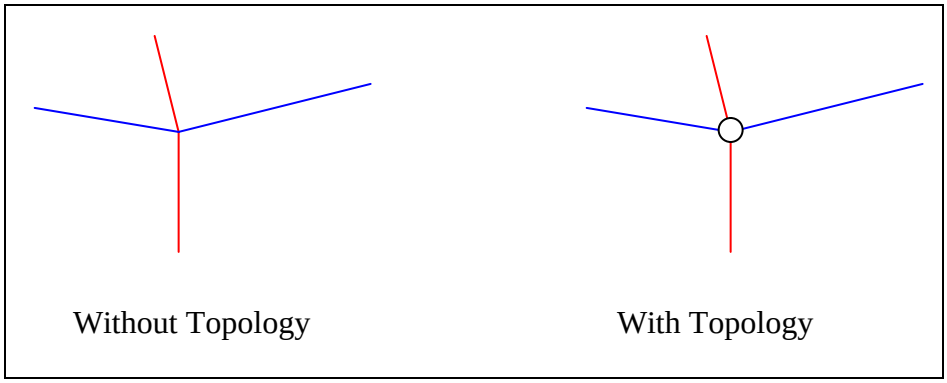


Figure 7.4-1 Introduction of Nodes at Intersections.

Topology encoding adds a number of complexities when building the geometries that are topologically correct. For example, the following represents a single feature from the LFC165 “Route (Maritime)” feature type with no topology.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<gml:FeatureCollection xmlns="http://www.opengis.net/ows-3/nga/MSD3"
xmlns:gml="http://www.opengis.net/gml" xmlns:MSD3="http://www.opengis.net/ows-
3/nga/MSD3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/ows-3/nga/MSD3 ./MSD3.xsd">
    <gml:boundedBy>
        <gml:Envelope srsName="urn:ogc:def:crs:EPSG:6.6:4326">
            <gml:lowerCorner>37.4883472 126.5849368 0</gml:lowerCorner>
            <gml:upperCorner>37.4932724 126.587602 0</gml:upperCorner>
        </gml:Envelope>
    </gml:boundedBy>
    <gml:featureMember>
        <MSD3:LFC165 gml:id="LFC165.1">
            <MSD3:position>
                <gml:LineString srsName="urn:ogc:def:crs:EPSG:6.6:4326">
                    <gml:posList srsDimension="3">37.493268 126.5849368 0
                    37.4932688 126.5853864 0 37.4932724 126.587602 0 37.4883472
                    126.58585 0 </gml:posList>
                </gml:LineString>
            </MSD3:position>
            <MSD3:atn>2</MSD3:atn>
            <MSD3:bds>0</MSD3:bds>
            <MSD3:brr>UNK</MSD3:brr>
            <MSD3:brs
                uom="http://zx10.ingr.com/msd3/schema/dictionaries/UnitDictionary.x
                ml#deg">0</MSD3:brs>
            <MSD3:dan>N/A</MSD3:dan>
            <MSD3:dof
                uom="http://zx10.ingr.com/msd3/schema/dictionaries/UnitDictionary.x
                ml#deg">>0</MSD3:dof>
            <MSD3:drp>UNK</MSD3:drp>
            <MSD3:hdi>12</MSD3:hdi>
            <MSD3:hdp
                uom="http://zx10.ingr.com/msd3/schema/dictionaries/UnitDictionary.x
                ml#m">0</MSD3:hdp>
            <MSD3:nam>Unknown</MSD3:nam>
            <MSD3:rtt>0</MSD3:rtt>
            <MSD3:txt>N_P</MSD3:txt>
            <MSD3:vdc>0</MSD3:vdc>
            <MSD3:vdr>N_P</MSD3:vdr>
            <MSD3:orig_source_info>N_P</MSD3:orig_source_info>
            <MSD3:originating_source>N_P</MSD3:originating_source>
            <MSD3:originator>N_P</MSD3:originator>
            <MSD3:scale>-32766</MSD3:scale>
        </MSD3:LFC165>
    </gml:featureMember>
</gml:FeatureCollection>
```

The above example is simple in that one can quickly identify the geometries and properties belonging to the given feature. Once we encode the topology for that single feature, the GML becomes as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<gml:FeatureCollection xmlns="http://www.opengis.net/ows-3/nga/MSD3"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:gml="http://www.opengis.net/gml"
xmlns:MSD3="http://www.opengis.net/ows-3/nga/MSD3"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/ows-3/nga/MSD3 ./MSD3.xsd">
    <gml:boundedBy>
        <gml:Envelope srsName="urn:ogc:def:crs:EPSG:6.6:4326">
            <gml:lowerCorner>37.4883472 126.5849368 0</gml:lowerCorner>
            <gml:upperCorner>37.4932724 126.587602 0</gml:upperCorner>
        </gml:Envelope>
    </gml:boundedBy>
    <gml:featureMember>
        <MSD3:LFC165 gml:id="LFC165.1">
            <MSD3:position>
                <gml:LineString srsName="urn:ogc:def:crs:EPSG:6.6:4326">
                    <gml:posList srsDimension="3">37.493268 126.5849368 0
                    37.4932688 126.5853864 0 37.4932724 126.587602 0 37.4883472
                    126.58585 0 </gml:posList>
                </gml:LineString>
            </MSD3:position>
            <MSD3:topology>
                <gml:TopoCurve>
                    <gml:directedEdge orientation="+">
                        <gml:Edge gml:id="E1">
                            <gml:directedNode orientation="-">
                                <gml:Node gml:id="N1">
                                    <gml:directedEdge orientation="-"
                                    xlink:href="#E1"/>
                                    <gml:pointProperty>
                                        <gml:Point
                                        srsName="urn:ogc:def:crs:EPSG:6.6:4326">
                                            <gml:pos srsDimension="3">37.493268
                                            126.5849368 0</gml:pos>
                                        </gml:Point>
                                    </gml:pointProperty>
                                </gml:Node>
                            </gml:directedNode>
                            <gml:directedNode orientation="+">
                                <gml:Node gml:id="N2">
                                    <gml:directedEdge orientation="+"
                                    xlink:href="#E1"/>
                                    <gml:pointProperty>
```

```
                                    <gml:Point
                                    srsName="urn:ogc:def:crs:EPSG:6.6:4326">
                                        <gml:pos srsDimension="3">37.4883472
                                        126.58585 0</gml:pos>
                                    </gml:Point>
                                </gml:pointProperty>
                            </gml:Node>
                        </gml:directedNode>
                        <gml:curveProperty>
                            <gml:LineString
                            srsName="urn:ogc:def:crs:EPSG:6.6:4326">
                                <gml:posList srsDimension="3">37.493268
                                126.5849368 0 37.4932688 126.5853864 0
                                37.4932724 126.587602 0 37.4883472 126.58585 0
                                </gml:posList>
                            </gml:LineString>
                        </gml:curveProperty>
                    </gml:Edge>
                </gml:directedEdge>
            </gml:TopoCurve>
        </MSD3:topology>
        <MSD3:atn>2</MSD3:atn>
        <MSD3:bds>0</MSD3:bds>
        <MSD3:brr>UNK</MSD3:brr>
        <MSD3:brs
            uom="http://zx10.ingr.com/msd3/schema/dictionaries/UnitDictionary.x
            ml#deg">0</MSD3:brs>
        <MSD3:dan>N/A</MSD3:dan>
        <MSD3:dof
            uom="http://zx10.ingr.com/msd3/schema/dictionaries/UnitDictionary.x
            ml#deg">0</MSD3:dof>
        <MSD3:drp>UNK</MSD3:drp>
        <MSD3:hdi>12</MSD3:hdi>
        <MSD3:hdp
            uom="http://zx10.ingr.com/msd3/schema/dictionaries/UnitDictionary.x
            ml#m">0</MSD3:hdp>
        <MSD3:nam>Unknown</MSD3:nam>
        <MSD3:rtt>0</MSD3:rtt>
        <MSD3:txt>N_P</MSD3:txt>
        <MSD3:vdc>0</MSD3:vdc>
        <MSD3:vdr>N_P</MSD3:vdr>
        <MSD3:orig_source_info>N_P</MSD3:orig_source_info>
        <MSD3:originating_source>N_P</MSD3:originating_source>
        <MSD3:originator>N_P</MSD3:originator>
        <MSD3:scale>-32766</MSD3:scale>
    </MSD3:LFC165>
</gml:featureMember>
```

</gml:FeatureCollection>

We see the additions highlighted showing duplicated information as well as the addition of the start and end node points. Remember, this is a single feature instance. Now picture a dataset with a thousand roads. Every time one road intersects another, they must both be split into separate instances. Since an attempt is made to detail such items as the connecting nodes only once, it means that many of the instances must cross-reference the connected node geometry.

However, not only does topology make the GML more complex, but it also makes it substantially larger in size. For example, consider Table 7.4-1, which shows the size differences between various datasets that have been generated both with and without encoding topology. The data used in the table is the MSD3 data supplied by the sponsor.

Table 7.4-1 — Size Comparison of Small MSD3 Data Set with Topology.

Feature Name/Type	Number of Features	GML Size without topology	GML Size with topology
LFC165 / Line	1	1.6 KB	2.9 KB
AAA010 / Area	1	1.9 KB	4.0 KB
PGB063 / Point	1	1.1 KB	1.4 KB

Table 7.4-1 shows that, on average, the size of line features increased at a ratio of about 2:3 and area features increased at a ratio of about 1:2. The point features do not show a big increase but points have little impact on topology. Note that these numbers involve only a single instance of each feature type. As the number of features is increased, the more information that has to be stored - the ratio may stay the same, but the amount of additional information gets larger based on the increase in features. The Table 7.4-2 shows feature types with a larger number of features.

Table 7.4-2 — Size Comparison of Larger MSD3 Data Set with Topology.

Feature Name/Type	Number of Features	GML Size without topology	GML Size with topology
LAP030 / Line	1444	1.7 MB	2.9 MB
AAL015 / Area	857	1.5 MB	2.8 MB
PAL015 / Point	2888	4.0 MB	4.9 MB

It should be noted that in addition to this study, a parallel study is taking place that looks into performance of transferring the often bulky GML of large datasets across a network. Based on some of the items listed in this study, more effective ways of handling the issue of size may be identified and thus erase bulkiness as an area of concern. In a separate study of the performance of GML, Galdos and CubeWerx are looking into how GML can be transferred over networks and the internet in a more efficient manner.

It should also be noted that this study did not take into account the use of xlinks. The use of xlinks would make a large impact on the topology encoding size that is documented in that duplicate geometries would not be represented.

7.6 Summary

The summary of what we have seen is that the encoding of topology certainly adds both bulk and complexity to the base GML datasets. This is the question that must be answered: "Can the benefits of persisted topology justify the overhead of carrying it through the transfers over various networks?" Based on the answer to this question, one can determine the need for persisted topology. It is suspected that the answer to this question will likely vary based on the perspective of each individual's needs. Therefore, because there are varying perspectives, it is impossible for this paper to answer this question definitively for everyone. Several possible solutions are presented in the following section, but this is certainly not meant to be an exhaustive list.

7.7 Possible Solutions

Before discussing the possible solutions, it should first be noted that efforts have already taken place to define schemas that make topology elements optional. This optional tag will help the user to quickly identify the existence or absence of topology.

After researching the topic of topology as it relates to persistent and on-the-fly generation within the GML format, several conclusions were reached. Due to the varied uses for GML as well as user preferences, this list is by no means definitive.

1. The first option is to remove all topology references from GML. Because of the benefits that topology can provide, this is not recommended. However, this option would remove all restrictions that topology brings to GML.

2. Another option, based on the need for consistency, is to mandate that all GML generation tools be required to produce topology. The writer of this document recognizes this option is not practical for a number of reasons. The cost of including topology could become large and cumbersome for both exporters and importers since the cost of building and parsing GML can potentially include financial cost as well as performance cost. Although not a very practical approach, it will provide topology for those expecting topological relationships, while allowing clients to ignore the topology if it is not needed.

3. One practical solution is to continue to provide GML generators the ability to generate the GML containing topology only on user request. This can be accomplished through Boolean toggles set on a user interface, through API properties, or through an extended parameter on the WFS GetFeature request. The GML data would then include the topology and optional tags discussed at the beginning of this section based on the user preference. Although this solution does not address the issues with transaction based services, it does give those producing GML for clients expecting topological relationships a method to supply

the topology.

4. The final option presented here is to create topology via a Topology Building Service. Such a service would take a WFS URL as input, build the topology, and publish a new WFS URL serving the GML containing topology (see figure 7.6-1 below). The GML data received from the original endpoint would not include the optional tags whereas the GML data received through the service would. This option has several advantages. First, it would remove the burden of producing topology on each WFS. Furthermore, topology consistency would be achieved because all output would be re-directed through a standardized topology building service. In addition, it would give the clients that desire topology a method of building topology on WFS sites that do not offer topology while also allowing clients that do not want topology to connect directly to the original WFS. Also it would help to standardize the "flavors" of topology generated because the service itself would be standardized.

Figure 7.6-1 Topology Service

Another perceived benefit involves the use of transaction-based WFS services. A client could request data through a "non-topology" WFS and perform transactional edits on the data without topology being involved in any of the workflow. However, if the client wished to receive the dataset with topology enabled, the original dataset could be redirected through a topology service before being ingested (see figure 7.6-2 below). Even so, there would still be some complications if edits were made because the original WFS would be expected to know how to interpret the changed dataset that now contains encoded topology. So, even though it doesn't solve the entire problem, a Topology Building Service would allow a WFS-T to provide topology through an independent service.

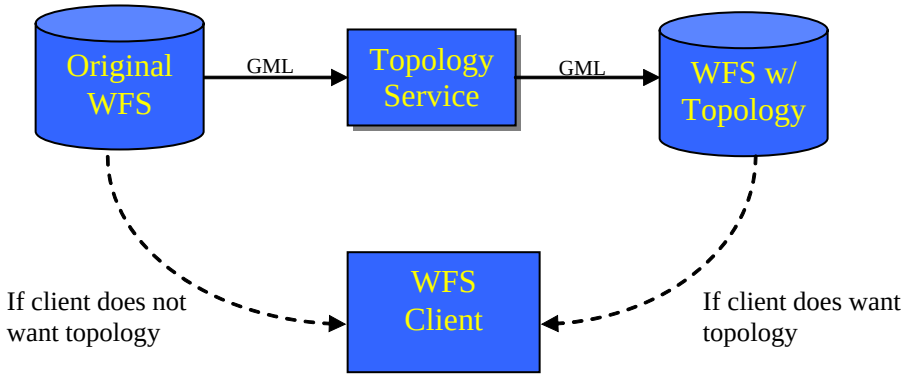


Figure 7.6-2 Optional Topology for any WFS

Note: An additional alternative to a Topology Service is extending the existing WFS specification. For example, one topic that is under investigation within WFS RWG is relaxing the constraint of WFS operations to include any GML object, thus allowing topology objects to be retrieved and manipulated. Such an improvement could end up in another conformance class for WFS.

Bibliography

[1] OGC 03-105r1, OpenGIS® Geography Markup Language (GML) Implementation Specification, Version 3.1.1, April 2004.

[2] Galdi, David, Spatial Data Storage and Topology in the Redesigned MAF/TIGER System, U.S. Census Bureau.

[3] Kuijpers, B., et al. A Lossless Representation of Topological Spatial Data, *Advances in Spatial Data* SSD 95, LNCS 951, p.1-13, 1995.

[4] Oracle® Spatial Topology and Network Data Models http://download-east.oracle.com/docs/cd/B19306_01/appdev.102/b14256/toc.htm

[5] Interface Standard for Vector Product Format, MIL-STD-2407, 28 June 1996.