GeoConnections
GéoConnexions
Canada

Canadian Geospatial Data Infrastructure
Infrastructure canadienne de données géospatiales

BRITISH COLUMBIA
Ministry of Sustainable Resource Management

galdos
systems inc

GML Foundation Project

Developing and Managing GML Application Schemas

A Best Practices Guide prepared by Galdos Systems Inc.™

Requested and Funded by:
The Ministry of Sustainable Resource Management and GeoConnections

TR2003-232-01

May 15, 2003

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

Contact: David S. Burggraf
Phone: (604) 484-2769
Fax: (604) 484-2755
E-mail: burggraf@galdosinc.com

Table of Contents

1 Introduction

This Best Practices Guide describes a set of schema design guidelines and their supporting rationale. The objective of these guidelines is not to dictate rules, but rather to shed light on the different sides of each design issue so that an application schema designer can make intelligent design decisions regarding the handling of GML within geospatial applications. The intent of this guide is to make it easier for application developers to produce schemas that they can work with.

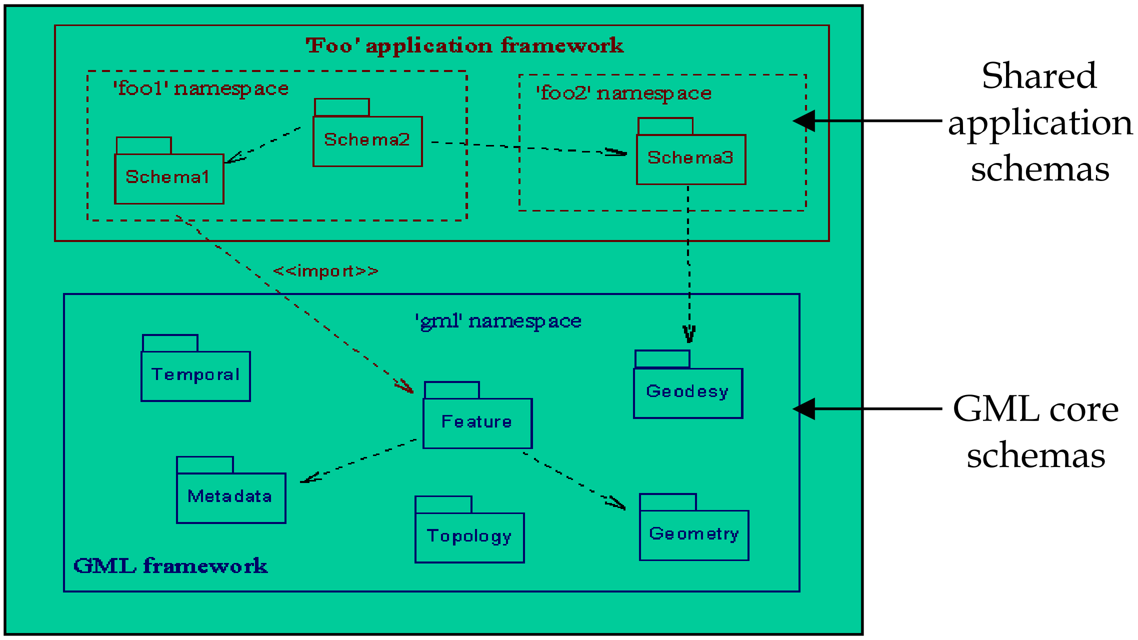
GML is a markup language that is used to encode both spatial and non-spatial geographic information. By building on broader Internet standards from the World Wide Web Consortium (W3C), GML is used to express geographic information in a manner that can be readily shared on the Internet. GML is not the first meta-language used to describe geographic information, but it is the first to be widely accepted within the GIS community. Other formats have been developed as a means to store and exchange spatial and temporal geographic information, however supporting tools to validate and reference the information were often not available. One of the advantages of using GML is that it enables one to leverage the whole world of XML technologies. In particular, GML builds on eXtensible Markup Language (XML), XML Schema, XLink, and XPointer. GML data can also be easily mixed with non-spatial data. By providing an open, standard, meta-language for geospatial constructs, information can be semantically shared across various domains like forestry, tourism, archaeology, geology and telecommunications.

GML adopts the International Organization for Standardization (ISO) notion of a feature as described by the OGC Abstract Specification, available online at http://www.opengis.org/techno/abstract/99-105r2.pdf. Features describe real world entities and are the fundamental objects used in GML. GML features can be concrete and tangible, or abstract and conceptual. For example, GML features could be: rivers, buildings, boundaries, or distributions of quantities over geographical areas (coverages). GML features are described in terms of their properties, which can include spatial (geometric or topological), temporal or other non-spatiotemporal descriptions of the feature. For instance, GML can describe the location, shape and extent of geographic objects as well as properties such as colour, speed, and density, some of which may depend on time. One could describe a natural disaster such as a flood as a dynamic feature in GML, whose properties such as extent, shape, water temperature, and expansion rate are all recorded at various times in some time interval. As it is impossible to describe all features and predict their usage beforehand, the GML core schemas do not contain definitions of concrete features. Rather, concrete features must be defined in GML Application Schemas, which are created by users such as database administrators.

This guide provides an overview of GML. It is assumed that the reader is acquainted with XML schema. While the information in this guide is intended to be self-contained, the reader will occasionally be referred to external sources such as the GML Implementation Specification and the GML Technical Guide. However, important details from these sources will be explained in this text where it is instructive to do so.

2 The GML Model

One of the primary objectives of GML is to provide a language for expressing geographic objects in a manner that is shareable over the Internet. GML provides a set of core schema components (e.g. features, geometry, topology, temporal, etc) together with a simple semantic model between objects and properties that is similar to Entity-Relationship diagrams or the class/property model of RDF (Resource Description Framework). Using the GML model and its schema components, users can describe the geographic types, whether concrete or conceptual, that are used within their application domain. The set of objects is created in the form of one or more GML Application Schemas, that is XML Schemas that make use of the GML schema components, and which comply with the GML semantic model and syntactic rules. A key benefit of GML is that the application schemas can be published and shared over the Internet, something that would be critical to any regional, national or international information infrastructure. XML Schema as the implementation encoding of GML, is ideally suited for this purpose, as opposed to say UML for which such sharing is not possible in any meaningful way.



GML is based on a feature framework. Domain experts create domain specific objects that make up the vocabulary of a domain. Such objects might include, for example, roads, dykes, rivers, buildings, escarpments, eskers, landslides, and faults. Features in GML are XML elements, with explicit names (e.g. `<trm:Road>`) that are described by their child property elements. Feature properties can have spatial and temporal values or not. However, unlike traditional GIS geometry based entities, we do not talk about “the geometry” of a feature, but rather about geometric properties that express geometric aspects of the feature. A GML feature can have multiple geometric properties each representing different aspects of the object in question, or none at all. We might, for example, describe a dyke by the centreline of its visible portion on the exposed surface, or by the bounding planes that demarcate it as a solid structure from surrounding rock units, or both.

2.1 The Object/Property Rule

An understanding of the object-property model is necessary to understand GML. The GML object/property model is partially based on the class-property model of the Resource Description Framework (RDF), which can be used as a reference point for understanding the GML model. Online resources about RDF are available at http://www.w3.org/TR/rdf-schema/ and http://www.w3.org/TR/REC-rdf-syntax/.

The object/property model is encoded in GML by declaring a type and then assigning properties to that type. These property XML elements are called child elements of the type.

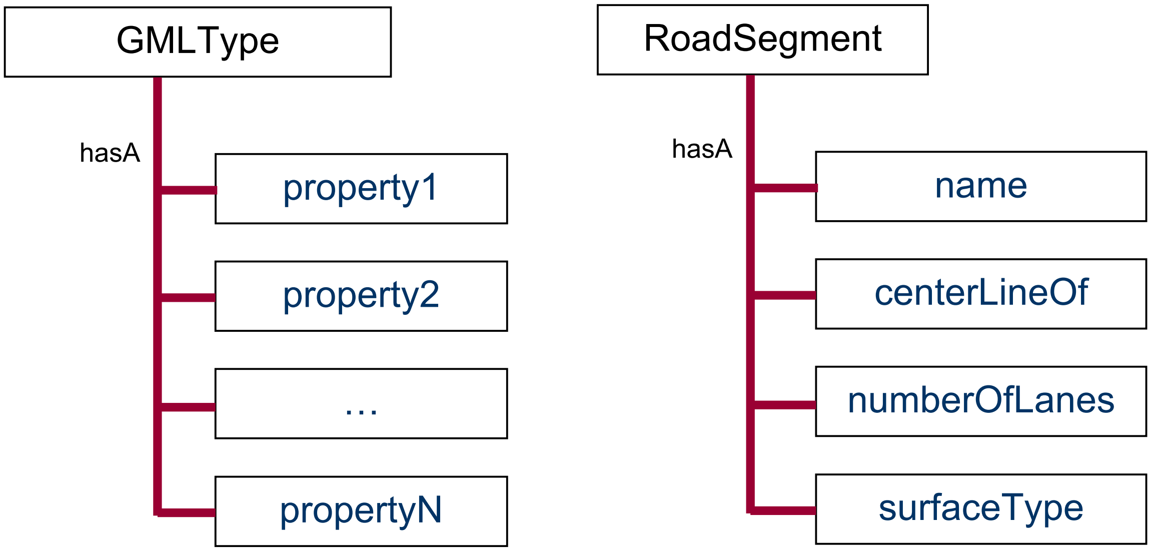


Figure 2-1: A GML Type is described by its property children

Note that in Figure 2-1 the GML Types are in written in UpperCamelCase and the properties are in lowerCamelCase. This convention is used in GML to distinguish between properties and types.

In contrast to legacy GIS approaches, a Feature is not defined primarily as a geometric object, but as a meaningful object that might have some properties that are geometric and other properties that are not. For example, Figure 2-1 shows that a `RoadSegment` feature has three properties that describe its physical characteristics and one property that describes its name.

Properties can be used to describe relationships (associations) between two GML objects, where the property name should provide the name of the relationship or possibly the name of the role that the target (or source) value plays in the relationship. This is illustrated in Figure 2-2.

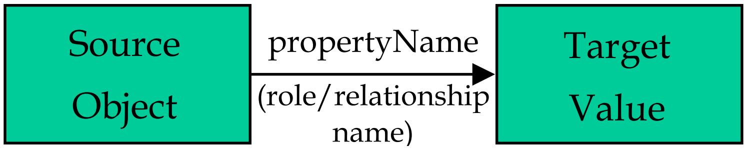


Figure 2-2: A property name in GML should provide the name of the relationship or possibly the name of the role that the target (or source) value plays in the relationship

For example, Figure 2-1 shows that the `RoadSegment` has a `centerLineOf` property that may contain a geometry object such as a `Curve` or `LineString`. The property name `centerLineOf` in this case represents the role that the target value (`Curve`) plays with respect to the source value in the relationship and is illustrated in Figure 2-3 (see Section 2.2 for a further discussion of role names vs relationship names). Note that it is possible to describe several different relationships between the `RoadSegment` and other geometries. A more general name, such as `geometry`, should not be used in place of the name `centerLineOf` in the `RoadSegment` definition because this can cause confusion if other geometry-valued properties are added. For example, another property of the `RoadSegment` called `extentOf` whose value is a `Polygon` could describe its surface extent, and a property called `centerOf` that is `Point` valued could locate the center of its surface extent.

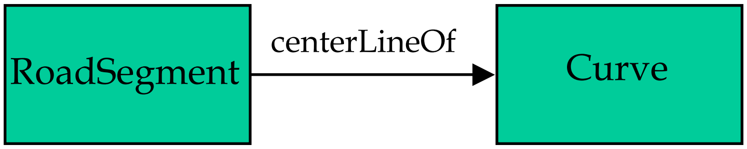


Figure 2-3: The relationship between a feature type and a geometry type is expressed by the property centerLineOf

A GML object cannot directly contain another GML object, but must have a property whose value is the other object. The following example shows a correct sample instance of the `RoadSegment` feature in GML:

```
<app:RoadSegment gml:id = "RS1">
   <app:name>Hanbury Road North</app:name>
   <gml:centerLineOf>
      <gml:LineString gml:id="L1">
         <gml:coordinates>1,2 2,3 3,4 4,0</gml:coordinates>
      </gml:LineString>
   </gml:centerLineOf>
   <app:numberOfLanes>2</gml:numberOfLanes>
   <app:surfaceType>Asphalt</gml:surfaceType>
</app:RoadSegment>
```

In this example, `app:RoadSegment` refers to the `RoadSegment` feature as defined in an application schema that resides in the application namespace. The prefix “app” is an arbitrary string that represents the application namespace and serves only as a place-holder for the application namespace (see Section 5.1 for a more detailed treatment of namespaces). The `RoadSegment` instance is uniquely identified by the ID attribute, `gml:id`.

Remote Properties

The values of GML properties are defined either inline or remotely. In the previous `RoadSegment` sample instance, for example, all of the property values are defined inline. Remote values, on the other hand, are not inline children of the property element, as shown in the following example:

```
<app:RoadSegment gml:id = "RS1">
   <app:name>Hanbury Road North</app:name>
   <gml:centerLineOf xlink:href="#L1"/>
   <app:numberOfLanes>2</gml:numberOfLanes>
   <app:surfaceType>Asphalt</gml:surfaceType>
</app:RoadSegment>
```

The `centerLineOf` property references a `LineString` geometry defined elsewhere in the instance document by using an `xlink:href` attribute. The `xlink:href` attribute, from the XLinks schema, refers to a GML type in the same document whose unique identifier is `gml:id="L1"`. Note that the `centerLineOf` property does not have separate opening and closing tags, because the value is indicated by an attribute, and there is no inline value provided. The "/" symbol at the end of the tag indicates that the tag is closed.

In GML, remote property values can reference data instances that are located within the same instance document, as shown above, or in another document that may reside anywhere on the Internet. In the above example, the hash symbol "#" in front of `L1` indicates that the reference points to an instance that is located within a document. The absence of a filename preceding "#" indicates that the reference is to a location in the same document.

The Role of Attributes in GML

In GML instances, properties of features and other objects are typically expressed by elements. For example, the following grammar is not valid GML:

```
<app:RoadSegment gml:id = "RS1" numberOfLanes="2" surfaceType = "Asphalt"/>
```

Although it is not invalid in XML schema to use the attributes as shown above, GML processing software will not interpret these attributes as properties of the GML feature. In GML instances, only a few attributes such as `gml:id`, `srsName`, and `xlink:href`, are used and these are used only as supportive constructs, rather than carrying object properties. Section 4.3 contains another example and further discussion about the use of attributes.

2.2 Feature Relationships

A property of a feature whose value is another feature expresses a relationship between the two features. The property name should provide the name of the feature relationship or possibly the name of the role that the target (or source) feature plays in the relationship. For example, in the following simple instance a `Road` crosses a `Creek`:

```
<app:Road gml:id = "R1">
   <app:name>Robert’s Creek Road</app:name>
   <app:crosses>
      <app:Creek gml:id = "C1">
         <app:name>Robert’s Creek</app:name>
      </app:Creek>
   </app:crosses>
</app:Road>
```

The `crosses` property name in this case provides the name of the role that the source (`Road`) feature plays in the relationship. On the other hand, one can also say that the `Creek` is crossed by the `Road`. The `crossedBy` property name in the following instance provides the name of the role that the target (`Creek`) feature plays in the relationship. This inverse relationship can be encoded as follows:

```
<app:Road gml:id = "R1">
   <app:name>Robert’s Creek Road</app:name>
   <app:crosses>
      <app:Creek gml:id = "C1">
         <app:name>Robert’s Creek</app:name>
         <app:crossedBy xlink:href = "#R1"/>
      </app:Creek>
   </app:crosses>
</app:Road>
```

More generally, a bi-directional property such as `intersects` can be used whose name is used to express the relationship without designating a role of the source or target feature as shown in the following sample instance:

```
<app:Road gml:id = "R1">
   <app:name>Robert’s Creek Road</app:name>
   <app:intersects>
      <app:Creek gml:id = "C1">
         <app:name>Robert’s Creek</app:name>
         <app:intersects xlink:href = "#R1"/>
      </app:Creek>
   </app:intersects>
</app:Road>
```

3 New components in GML3

Many new components have been added to version 3.0 of GML since version 2.1.2. There were only 3 core schemas in GML2; feature.xsd, geometry.xsd and xlinks.xsd, whereas there are twenty-eight core schemas in GML3.0. The model is the same respecting features and their properties, however, GML 3.0 supports many new kinds of objects, including topology, temporal components, dynamic features, coverages and coordinate reference systems. This chapter provides an overview of the new components in GML3.

3.1 Additional Geometry types

GML supports three-dimensional geometry models consisting of types that represent points, curves, surfaces and solids. The geometry primitives supported in GML2 were `Point`, `LineString`, `LinearRing`, `Box` and `Polygon` as well as the corresponding aggregates: `MultiPoint`, `MultiLineString`, `MultiPolygon`. GML3 supports many new types in the geometry schemas including: `Arc`, `Circle`, `CubicSpline`, `Ring`, `OrientableCurve`, `OrientableSurface` and `Solid`. In addition to the aggregates, such as `MultiCurve`, `MultiSurface`, etc., GML3 also supports the following composites: `CompositeCurve`, `CompositeSurface`, and `CompositeSolid`. The main difference between a geometry aggregate and a composite in GML is that the geometry members of a composite are all connected, where the members of an aggregate can be disjoint. Note that in GML, the geometry members of a composite do not "live or die" with the composite, they can exist independently of the composite.

Rather than having all the geometry types and elements in GML 3.0 lumped into one schema, the different types are factored into the following five schemas:

1. `geometryBasic0d1d.xsd`
2. `geometryBasic2d.xsd`
3. `geometryAggregates.xsd`
4. `geometryPrimitives.xsd`
5. `geometryComplex.xsd`

The first three schemas contain the most commonly used geometry components (the linear geometries) and are backwards compatible with GML 2. The last two geometry schemas contain most of the new types and elements including many non-linear geometries.

Interpolated Curves and Curve Segments

The most commonly used one-dimensional geometry type in GML is the `LineString`. A `LineString` in GML is a curve that is linearly interpolated between a finite set of points called control points.

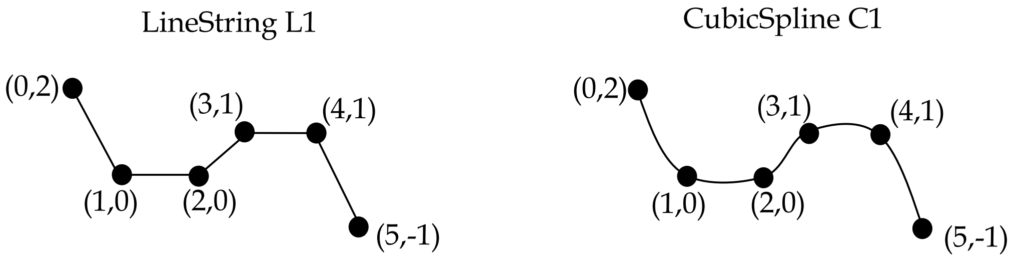


Figure 3-1: Linear versus non-linear geometries. A (piecewise) linear curve can be represented by a `LineString` (left). A non-linear `CubicSpline` curve is shown on the right.

The `LineString` in Figure 3-1 can be encoded in GML as follows:

```
<gml:LineString gml:id="L1">
   <gml:pos>0 2</pos>
   <gml:pos>1 0</pos>
   <gml:pos>2 0</pos>
   <gml:pos>3 1</pos>
   <gml:pos>4 1</pos>
   <gml:pos>5 -1</pos>
</gml:LineString>
```

Note that each control point in the encoding above is recorded in a `<pos>` element, which is of `gml:DirectPositionType`. The control points of `L1` can alternatively be encoded more compactly as "coordinate tuples" using the `<coordinates>` element as follows:

```
<gml:LineString gml:id="L1">
   <gml:coordinates decimal="." cs="," ts="&#x20;">
      0,2 1,0 2,0 3,1 4,1 5,-1
   </gml:coordinates>
</gml:LineString>
```

Note that the attributes `decimal`, `cs` and `ts` are used to format the decimal, coordinate separator and tuple separator, respectively. In this case, the tuple separator is set to a single white space by specifying its hexadecimal code: `ts="&#x20;"`. In the case that the attribute values of `decimal`, `cs` and `ts` are not specified, the default values are ".", "," and "&#x20;", respectively. GML3 supports other non-linear interpolation methods for curves including circular, elliptical, and conic interpolation methods. GML3 also supports a well-known family of interpolated `CurveSegments` called `Bsplines`, of which `Bezier` polynomial splines and `CubicSplines` are particular examples. A `gml:CurveSegment` has a uniform interpolation type and always occurs as a segment of a `gml:Curve`. A `gml:Curve` has a `segments` property whose value is an array of `gml:CurveSegments`. The cubic spline curve `C1` in Figure 3-1 is encoded in GML as follows:

```
<gml:Curve gml:id="C1">
   <gml:segments>
      <gml:CubicSpline>
         <gml:coordinates>0,2 1,0 2,0 3,1 4,1 5,-1</gml:coordinates>
         <gml:vectorAtStart>1 -3</gml:vectorAtStart>
         <gml:vectorAtEnd>1 -3</gml:vectorAtEnd>
      </gml:CubicSpline>
   <gml:segments>
<gml:Curve>
```

Notice that in addition to providing the control points of the `CubicSpline`, the vectors encapsulated by the `vectorAtStart` and `vectorAtEnd` properties are necessary to uniquely determine the cubic spline going through the given set of control points.

Furthermore, note that the curves `L1` and `C1` have an inherent direction that is determined by the order of the control points. A `Curve` that traverses `C1` in the opposite direction—from (5,-1) to (0,2)—can be defined using `gml:OrientableCurve` and is encoded as follows:

```
<gml:OrientableCurve gml:id="C2" orientation="-">
   <gml:baseCurve xlink:href=”#C1”/>
</gml:OrientableCurve>
```

GML3 supports other non-linear `CurveSegments` such as `Arc`, `ArcString` and `Circle`. A `gml:Arc` is just a connected subset of a circle that is encoded using circular interpolation between three control points. For example, the semicircle that passes through the points (1,0), (0,1), (-1,0) would be encoded as follows:

```
<gml:Arc>
   <gml:coordinates>1,0 0,1 –1,0</gml:coordinates>
</gml:Arc>
```

An `ArcString` consists of one or more `Arcs` that are contiguous (connected at endpoints) and a `Circle` is an `Arc` that closes up to form a loop. Example encodings of an `ArcString` and a `Circle` are as follows:

```
<gml:ArcString>
   <gml:coordinates>1,0 0,1 –1,0 –2,-1 –3,0</gml:coordinates>
</gml:ArcString>

<gml:Circle>
   <gml:coordinates>1,0 0,1 -1,0</gml:coordinates>
</gml:Circle>
```

Notice that the number of control points in an `ArcString` must be an odd integer greater than or equal to three—three for the first `Arc` and two more for each additional `Arc`.

Surfaces, Surface Patches and Solids

Surfaces and solids are also supported in GML3. The boundary of a `gml:Solid` is a `gml:CompositeSurface`, which consists of one or more surface members that enclose the `Solid` without overlap or open gaps. Figure 3-2 shows an example of a solid half-ball `S1` having a composite surface as its boundary consisting of two surfaces, the lower hemisphere `H1` and the equatorial plane `P1`.

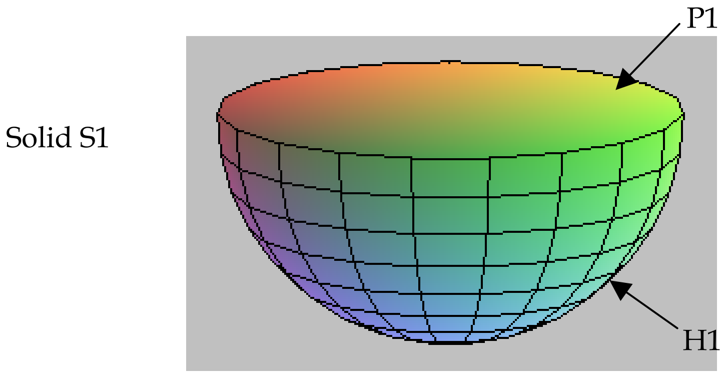


Figure 3-2: A `Solid` bounded by a `CompositeSurface`

A `gml:Surface` such as the lower hemisphere `H1` is encoded as a collection of surface patches, where each patch member is usually a small (flat) polygon. In this case, the surface looks like the surface of a cut gem. However, a more accurate representation of `H1` can be defined in an application schema using spherical surface interpolation. An example of this is given in the next section. The following example shows a sample encoding of the `Solid S1`. Note that a collection of small polygonal patches are used to represent the upper hemispherical `Surface H1`:

```
<gml:Solid gml:id=”S1”>
   <gml:exterior>
      <gml:CompositeSurface>
         <gml:surfaceMember>
            <gml:Surface gml:id="H1">
               <gml:patches>
                  <gml:PolygonPatch>
                     <gml:exterior>
                        <gml:LinearRing>
                          <gml:coordinates>
                     1.,0.,0. .95,.31,0 .94,.30,-.14 .94,0,-.14 1.,0.,0.
                        </gml:coordinates>
                        </gml:LinearRing>
                     </gml:exterior>
                  </gml:PolygonPatch>
                  <gml:PolygonPatch>
                      ...
                  </gml:PolygonPatch>
                  ...
               </gml:patches>
            </gml:Surface>
         </gml:surfaceMember>
         <gml:surfaceMember>
            <gml:Surface gml:id="P1">
               <gml:patches>
                  <gml:PolygonPatch>
                     <gml:exterior>
                        <gml:LinearRing>
                        <gml:coordinates>
.95, .31, 0, .81 ,.58,0, .60,.80,0, .36,.93,0, 0.,1.,0, -.36,.93,0, -.60,.80,0,
-.81,.58,0, -.95,.31,0, -1.,0.,0, -.95,-.31,0, -.81,-.58,0, -.60,-.80,0, -.36,-
.93,0, 0.,-1.,0, .36,-.93,0, .60,-.80,0, .81,-.58,0, .95,-.31,0, 1.,0.,0.
                        </gml:coordinates>
                        </gml:LinearRing>
                     </gml:exterior>
                  </gml:PolygonPatch>
               </gml:patches>
            </gml:Surface>
         </gml:surfaceMember>
      <gml:CompositeSurface>
   </gml:exterior>
</gml:Solid>
```

Spherical Surface Interpolation

Geometry types that are not provided in GML 3.0 can be created in an application schema. One rule that must followed is that the new geometry type must derive, directly or indirectly, from AbstractGeometryType. We recommend that you avoid deriving anything from the top-level gml:AbstractGeometryType, if possible, since this provides very limited information to GML-aware software. Instead try to choose the most specific geometry type possible in constructing your derived geometry types. For example, if your geometry is some type of surface, then derive from gml:AbstractSurfaceType or in the case that the surface uses a single uniform interpolation method it may be more appropriate to derive from gml:AbstractSurfacePatchType.

A gml:SurfacePatch is of gml:AbstractSurfacePatchType and is just a higher dimensional analogue of gml:CurveSegment, i.e. it has a uniform interpolation type and always occurs as a patch of a gml:Surface. Figure 3-3 and the following schema fragment provide an example of a new type of surface patch called SphericalCap, which is essentially a piece of a sliced sphere.

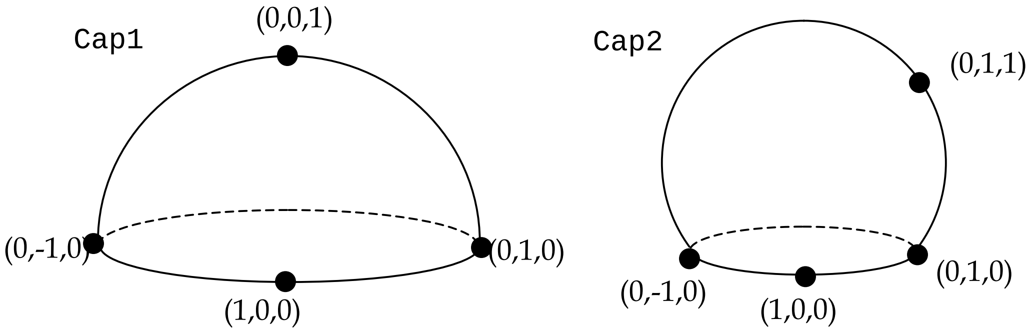


Figure 3-3: Two examples of a SphericalCap

```
  <element name="SphericalCap" type="app:SphericalCapType"
substitutionGroup="gml:_SurfacePatch"/>
  <!-- ============================================================ -->
  <complexType name="SphericalCapType">
     <complexContent>
        <extension base="gml:AbstractSurfacePatchType">
           <sequence>
              <choice>
                 <annotation>
                    <documentation>
                       The number of tuples in the coordinate list must be four.
                    </documentation>
                 </annotation>
                 <choice minOccurs="4" maxOccurs="4">
                    <element ref="gml:pos"/>
                    <element ref="gml:pointRep"/>
                 </choice>
                 <element ref="gml:coordinates"/>
              </choice>
```

```
          </sequence>
          <attribute name="interpolation" type="gml:SurfaceInterpolationType"
fixed="spherical"/>
       </extension>
    </complexContent>
  </complexType>
```

The SphericalCap is a higher dimensional analogue of Arc. The attribute "interpolation" specifies the interpolation mechanism used for this surface patch. The value of this attribute is fixed to "spherical", i.e. the interpolation method will return a subset of points on a sphere that lie on one side of a circle which forms the boundary points of the spherical cap. The first 3 points are the control points of the boundary circle of the spherical cap, where circular interpolation is used. The fourth point is the last control point needed to uniquely determine the spherical cap. Spherical interpolation through the four points is used to complete the spherical cap. Note that the first 3 points cannot lie on a common line (not collinear) and the 4 points together cannot lie on a common plane (not coplanar). Sample instances of the SphericalCaps Cap1 and Cap2 in Figure 3-3 might be as follows:

```
<gml:Surface gml:id="Cap1">
   <gml:patches>
       <gml:SphericalCap>
          <gml:coordinates>0,-1,0 1,0,0 0,1,0 0,0,1</gml:coordinates>
       </gml:SphericalCap>
   </gml:patches>
</gml:Surface>

<gml:Surface gml:id="Cap2">
   <gml:patches>
       <gml:SphericalCap>
          <gml:coordinates>0,-1,0 1,0,0 0,1,0 0,1,1</gml:coordinates>
       </gml:SphericalCap>
   </gml:patches>
</gml:Surface>
```

3.2 Topology

Topology is a branch of mathematics that describes the properties of spatial objects that remain unchanged after “twisting” and “stretching” (continuous deformation). In GML, spatial topology is modeled using basic building blocks—nodes, edges, faces, and solids—called topology primitives, together with a description of their connective relationships to one another. The GML topology primitive types, Node, Edge, Face, and TopoSolid, are often used to represent the geometry primitives, Point, Curve, Surface, and Solid, respectively. The connectedness of nodes, coincidence of edges, and adjacency of faces and solids are some of the connective relationships between primitives that topology is concerned with. Unlike GML geometry, topology does not encode any coordinate values—so <pos> and <coordinates> tags are absent. The topology model is not concerned with the position of nodes, direction of edges, nor the shape of faces and solids.

Nodes, Edges and Faces

Figure 3-4 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)

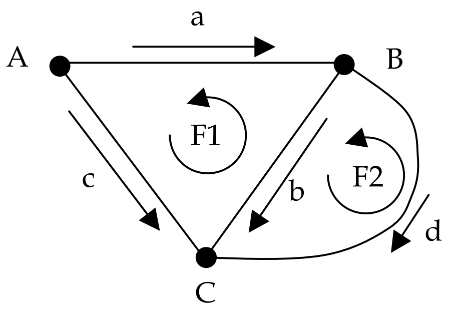


Figure 3-4: A Simple Two-dimensional Topology Network

In GML 3.0, the nodes can be encoded as follows:

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

The `Nodes` are only required to be uniquely identified using the `gml:id` attribute. The `Edges` can be encoded as follows:

```
<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 of the edge.

Note that each `Edge` is directed by its start and end node and 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 3.0 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. The following example shows how to encode this path from A to B as a TopoCurve:

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

In the GML topology model, 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/IS 19107 *Spatial Schema*) as indicated by the arrow surrounding F1 and F2 in Figure 3-4. 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, when traversed counter-clockwise, corresponds to the directed edges in the set {c,-b,-a}. The faces are encoded in GML 3.0 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 3-4, 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 coboundary of b. In GML 3, the Edge primitive has an optional property called directedFace, whose value is a Face that is in the co-boundary of the Edge. To distinguish between the face on the left of b and the face on the right of b, each of the coboundary faces is assigned 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 coboundary 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 coboundary information in addition to its boundary information 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 Node B from Figure 3-4 can be 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>
```

More Complex Topology Primitives

The coboundary 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 3-5.

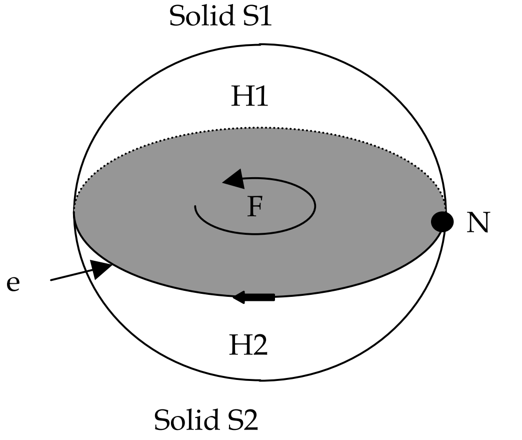


Figure 3-5: Two Solid half-balls in the Co-Boundary of a Face

The solid upper half-ball S1 and the solid lower half-ball S2 together form the solid ball shown in Figure 3-5. The boundary of S1 consists of two faces, representing the upper hemisphere H1 and the equatorial plane F. The Solid 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 3-6 shows a more complicated use of two-dimensional 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.

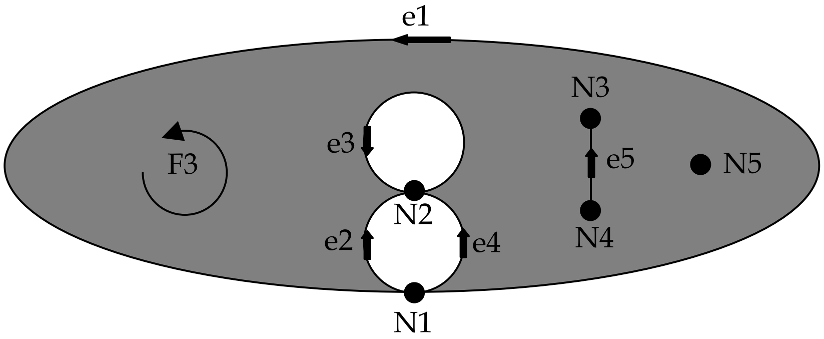


Figure 3-6: A More Complex Boundary Configuration of a Face

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 ring 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 GML3 Topology but they are in GML3 Geometry. The reason for this is to allow for an efficient encoding of topology (beyond the absence of coordinates) by minimizing the level of nesting in instance documents. Note also that any number of boundary rings can intersect at a common Node, but boundary rings are not allowed to intersect along a common Edge. The Face F3 is encoded in GML 3 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>
```

Topology and Geometry

In GML 3.0, there is symmetry between the geometric and topological primitives. The geometry primitives—`Point`, `Curve`, `Surface`, and `Solid`—are often referred to 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 3-7 as shown.

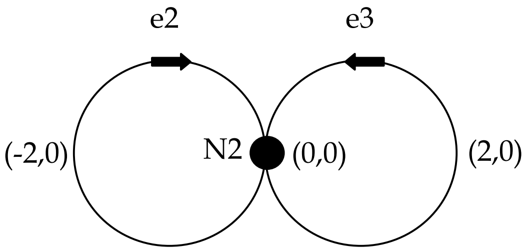


Figure 3-7: Topology with Geometry

The simple topology network can be encoded including the geometric information as follows:

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

3.3 Coverages

A GML coverage is essentially a distribution function, defined on a domain, which usually consists of a set of geometry elements. For example, the geometry elements may be a set of polygons in a surface tessellation, a set of curves segments along curve, a discrete set of points, or a rectified grid. The range of the distribution function could take on values such as elevation, temperature, pressure, rock type, or reflectance as shown in Figure 3-8.

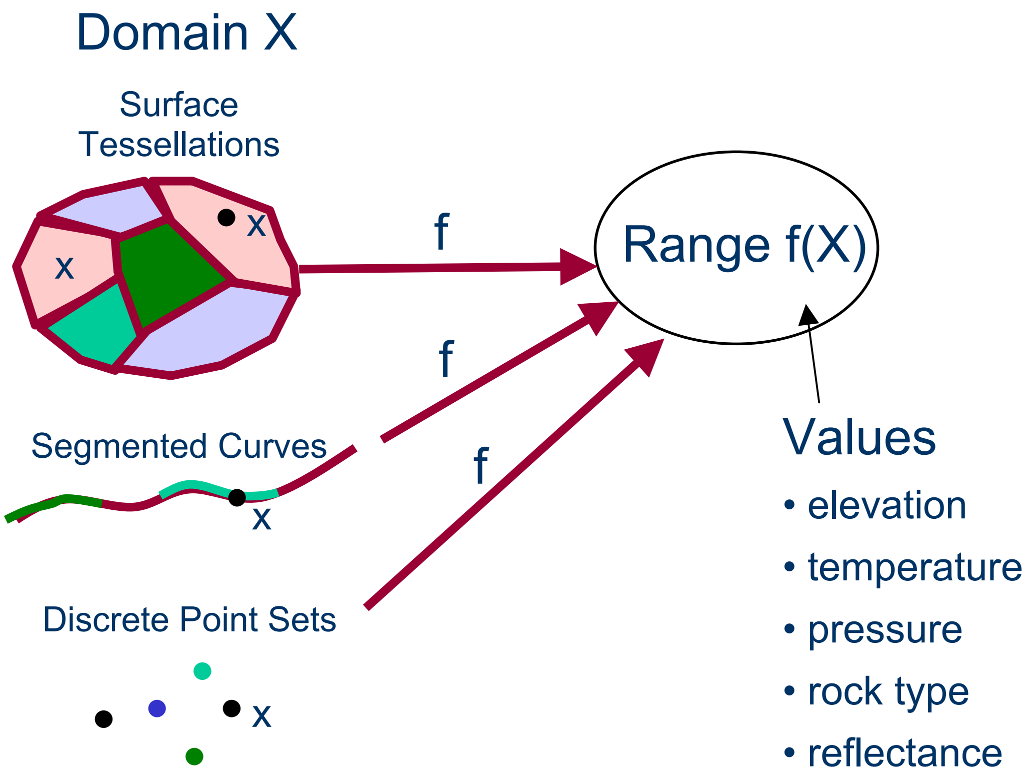


Figure 3-8: Coverages are Distribution Functions defined on some Domain.

There are three components that define a GML coverage—the domain, range, and coverage function. Each of these components are encoded separately and are contained by the three properties: `domainSet`, `rangeSet` and `coverageFunction`, respectively. The GML spatial coverage model is represented by the Entity Relationship (ER) diagram in Figure 3-9.

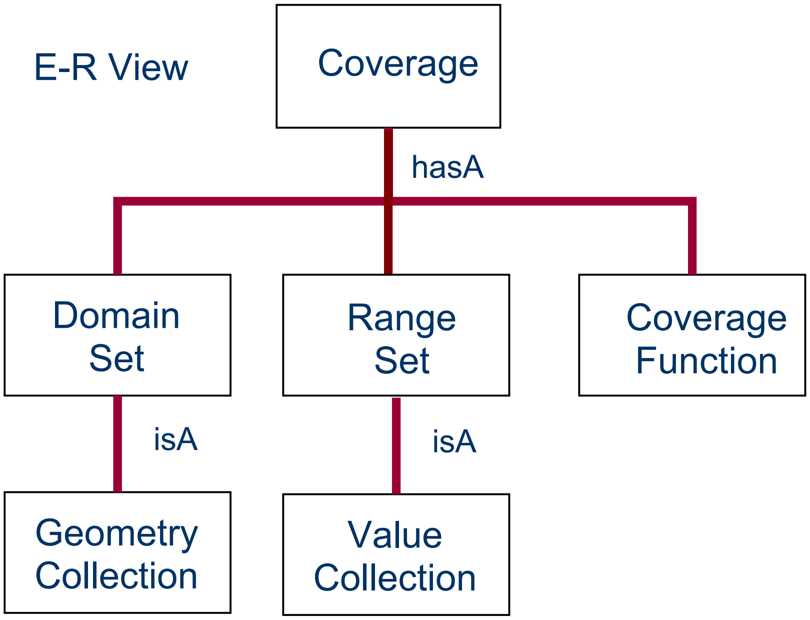


Figure 3-9: Entity Relationship (ER) View of a Coverage

Domain Set

In a GML instance, the value of `domainSet` will often be a GML geometry aggregate, such as `MultiPoint`, `MultiCurve`, `MultiPolygon`, or a `Grid` such as `RectifiedGrid`.

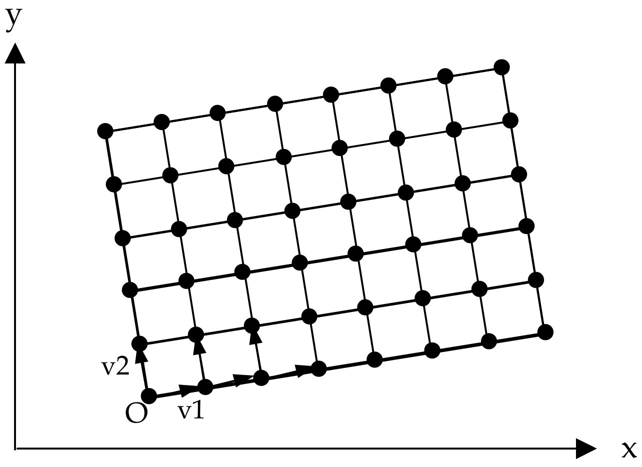


Figure 3-10: A Two-Dimensional Rectified Grid with Offset Vectors v1 and v2

In Figure 3-10, the origin is denoted as `O`, and the offset vectors are **`v1`** and **`v2`.** The origin and offset vectors must all be of the same dimension—usually 2D or 3D. The GML `RectifiedGrid` has the properties: `limits`, `axisName`, `origin` and `offsetVector`. The value of `limits` is a `GridEnvelope`, `which` has two properties called `low` and `high`.

The value of `low` is an integer list, for example [`n1`,`n2`], that represents the “lower left” corner `O+n1`**`v1`**`+n2`**`v2`** of the grid. Similarly, `high` represents the “upper right” corner of the grid. The following listing shows two sample encodings of a `domainSet` containing a GML `RectifiedGrid` and a `MultiPoint`, respectively:

```
<gml:domainSet>
   <gml:RectifiedGrid dimension="2">
      <gml:limits>
         <gml:GridEnvelope>
            <gml:low>0 0</gml:low>
            <gml:high>9 6</gml:high>
         </gml:GridEnvelope>
      </gml:limits>
      <gml:axisName>x</gml:axisName>
      <gml:axisName>y</gml:axisName>
      <gml:origin>
         <gml:Point gml:id="O"
srsName=”urn:epsg:v6.1:coordinateReferenceSystem:4267”>
            <gml:pos>2,1</gml:pos>
         </gml:Point>
      </gml:origin>
      <gml:offsetVector srsName=”urn:epsg:v6.1:coordinateReferenceSystem:4267”>1,
0.3</gml:offsetVector>
      <gml:offsetVector srsName=”urn:epsg:v6.1:coordinateReferenceSystem:4267”>-0.3,
1</gml:offsetVector>
   </gml:RectifiedGrid>
</gml:domainSet>

<gml:domainSet>
   <gml:MultiPoint srsName=”urn:epsg:v6.1:coordinateReferenceSystem:4267”>
      <gml:pointMembers>
         <gml:Point>
            <gml:pos>1 0</gml:pos>
         </gml:Point>
         <gml:Point>
            <gml:pos>0 1</gml:pos>
         </gml:Point>
         <gml:Point>
            <gml:pos>3 1</gml:pos>
         </gml:Point>
         <gml:Point>
            <gml:pos>1 2</gml:pos>
         </gml:Point>
         <gml:Point>
            <gml:pos>3 4</gml:pos>
         </gml:Point>
         <gml:Point>
            <gml:pos>1 5</gml:pos>
         </gml:Point>
      </gml:pointMembers>
   </gml:MultiPoint>
</gml:domainSet>
```

Range Set

The range set can be encoded as an aggregate value, a data block or a binary file. An aggregate value encoding is the most verbose; the binary encoding is the most efficient, and the data block encoding lies somewhere in between the two extremes.

This section provides examples of the three different `rangeSet` encodings with temperature and pressure values. Note that the temperature and pressure values in these examples are defined in a GML application schema.

Encoding the Range Set as an Aggregate Value

Any of the GML value aggregates can be used to encode range data as an aggregate value. Some commonly used aggregate values defined in the GML core schema `valueObjects.xsd` are as follows:

1. A (set of) `ValueArray(s)` in which the members of each array are homogeneously typed values
2. A member of the gml:`_ScalarValueList` substitution group, such as `CategoryList` or `CountList`

The following example shows how temperature and pressure values can be encoded as a set of value arrays:

```
<gml:rangeSet>
   <gml:ValueArray>
      <gml:valueComponents>
         <app:Temp uom="urn:x-si:v1999:uom:degreesC">0</app:Temp>
         <app:Temp uom="urn:x-si:v1999:uom:degreesC">24</app:Temp>
         <app:Temp uom="urn:x-si:v1999:uom:degreesC">17</app:Temp>
         <app:Temp uom="urn:x-si:v1999:uom:degreesC">19</app:Temp>
               ...
      </gml:valueComponents>
   </gml:ValueArray>
   <gml:ValueArray>
      <gml:valueComponents>
         <app:Pressure uom="urn:x-si:v1999:uom:kPa">101.1</app:Pressure>
         <app:Pressure uom="urn:x-si:v1999:uom:kPa">101.2</app:Pressure>
         <app:Pressure uom="urn:x-si:v1999:uom:kPa">101.3</app:Pressure>
         <app:Pressure uom="urn:x-si:v1999:uom:kPa">101.4</app:Pressure>
               ...
      </gml:valueComponents>
   </gml:ValueArray>
</gml:rangeSet>
```

Encoding the Range Set as a Data Block

A `gml:DataBlock` consists of two properties, `rangeParameters` and `tupleList`, whose values describe the range of a coverage. The value of `rangeParameters` is of `gml:_Value` type from valueObjects.xsd, which describes the quantities in the `tupleList`, including the units of measure used.

The following example shows how temperature and pressure values can be encoded in a `DataBlock`:

```
<gml:rangeSet>
   <gml:DataBlock>
      <gml:rangeParameters>
         <gml:CompositeValue>
            <gml:valueComponents>
               <app:Temp uom="urn:x-si:v1999:uom:degreesC"/>
            </gml:valueComponent>
            <gml:valueComponent>
               <app:Pressure uom="urn:x-si:v1999:uom:kPa"/>
            </gml:valueComponents>
         </gml:CompositeValue>
      </gml:rangeParameters>
   <gml:tupleList>0,101.1 24,101.2 17,101.3 19,101.4  ...</gml:tupleList>
   </gml:DataBlock>
</gml:rangeSet>
```

Note that the aggregate value object `CompositeValue` is the target of `rangeParameters` in the instance above, which has a `valueComponents` property. `Temp` and `Pressure` are value objects derived from `gml:MeasureType` that are encapsulated by the `valueComponents` property tags. The `tupleList` property contains a list of coordinate tuples, where the entries of the coordinate tuples provide the quantities of the range parameters.

Encoding the Range Set as a Binary File

The most efficient encoding of quantities in the range set is the binary file encoding provided by `gml:File`. The following example shows how to use `gml:File` to record the temperature and pressure values:

```
<gml:rangeSet>
   <gml:File>
      <gml:rangeParameters>
         <gml:CompositeValue>
            <gml:valueComponents>
               <app:Temp uom="urn:x-si:v1999:uom:degreesC"/>
               <app:Pressure uom="urn:x-si:v1999:uom:kPa"/>
            </gml:valueComponents>
         </gml:CompositeValue>
      </gml:rangeParameters>
      <gml:fileName>weather.dat</gml:fileName>
      <gml:fileStructure>Record Interleaved</gml:fileStructure>
   </gml:File>
</gml:rangeSet>
```

The `rangeParameters` property is the same as that used for the `DataBlock` example. The properties `fileName` and `fileStructure`, contain the location, name and structure of the binary file in which the data is stored. Note that the fields within each record of the file correspond to the `valueComponents` of the `CompositeValue` in the given order.

Another sample instance of `rangeSet` that makes use of different range parameters is as follows:

```
<gml:rangeSet>
   <gml:File>
      <gml:rangeParameters>
         <gml:CompositeValue>
            <gml:valueComponents>
               <app:SoilType codeSpace=“urn:x-NID:v0.1:soils:123”>/>
               <app:SoilMoisture uom="urn:x-IHSDM:v2.05a:uom:percent"/>
            </gml:valueComponents>
         </gml:CompositeValue>
      </gml:rangeParameters>
      <gml:fileName>soil.dat</gml:fileName>
      <gml:fileStructure>Record Interleaved</gml:fileStructure>
   </gml:File>
</gml:rangeSet>
```

Coverage Function

The value of `gml:coverageFunction` is a choice between `gml:MappingRule` or `gml:GridFunction`. The `MappingRule` is of type `gml:StringOrRefType`, which is a string or a URI reference to a mapping rule defined elsewhere. The mapping rule by default is linear if not specified, where linear mapping assigns the first geometry element (in document order) in the domain to the first value in the range, and so on.

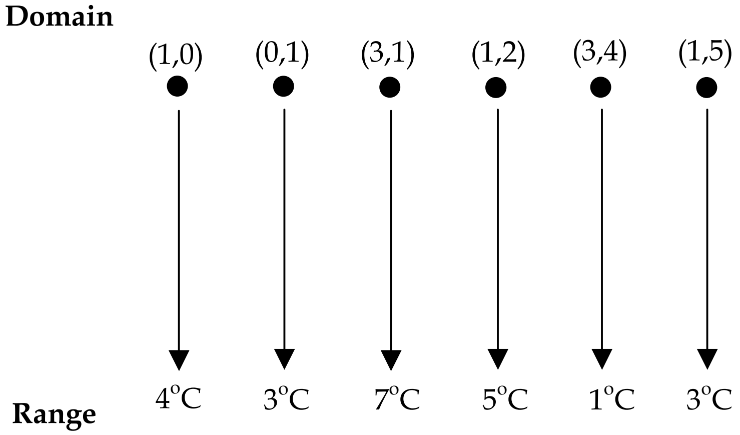


Figure 3-11: Linear Mapping Rule for a `MultiPoint` Temperature Coverage

Grid Functions

The `GridFunction` is used instead of `MappingRule` for grid coverages and has two properties, `sequenceRule` and `StartPoint`. The value of `startPoint` is an integer list that represents the first grid post in the grid to be traversed, the default start value is the value of `low` in `GridEnvelope`. The value of `sequenceRule` is one of the strings in the enumerated list sequence rules: `Linear`, `Boustrophedonic`, `Cantor-diagonal`, `Spiral`, `Morton`, or `Hilbert`. The `sequenceRule` property also has an optional `order` attribute whose value specifies one of the increment orders in the enumerated list: `"+x+y"`, `"+y+x"`, `"x-y"`, `"-x-y"`, in the case where the grid is 2-dimensional. The increment order of `"+x+y"` means that the grid points are traversed from lower to higher on the `x`-axis and from lower to higher on the `y`-axis. For example, Figure 3-12 and Figure 3-13 both illustrate the linear sequence rule but with different increment orders.

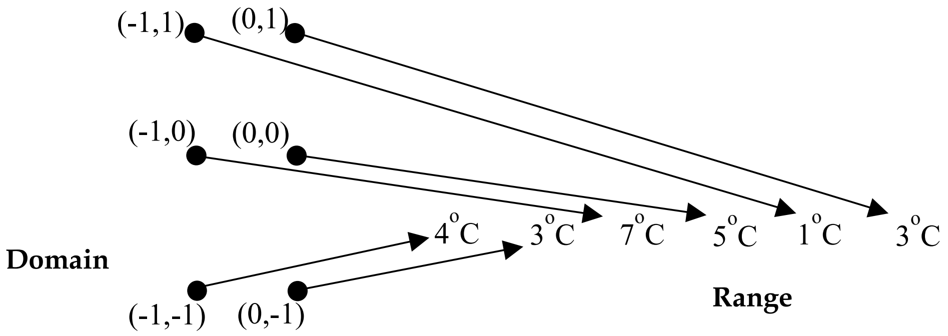


Figure 3-12: Linear Grid Function for a Grid Coverage with "+x+y" Increment Order

Note that in Figure 3-12, the starting point (-1,-1) is the same as the value of `low` in the corresponding `GridEnvelope`. The increment order "+x-y" means the grid points are traversed from lower to higher on the x-axis and from higher to lower on the y-axis as illustrated in Figure 3-13.

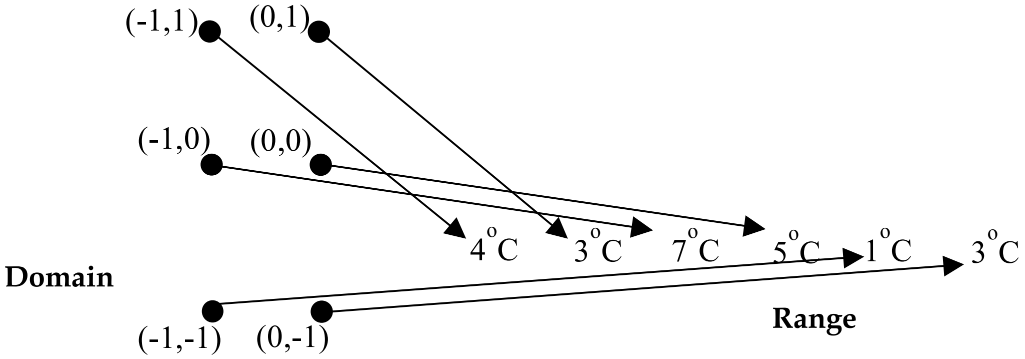


Figure 3-13: Linear Grid Function for a Grid Coverage with "+x-y" Increment Order

As shown in Figure 3-13, the starting point (-1,1) is the not the same as the value of `low` in the corresponding `GridEnvelope`. In this case the `startPoint` property is required as shown in the following instance:

```
<gml:domainSet>
   <gml:Grid dimension="2">
      <gml:limits>
         <gml:GridEnvelope>
            <gml:low>-1 -1</gml:low>
            <gml:high>0 1</gml:high>
         </gml:GridEnvelope>
      </gml:limits>
      <gml:axisName>x</gml:axisName>
      <gml:axisName>y</gml:axisName>
   </gml:Grid>
</gml:domainSet>
<rangeSet>...</rangeSet>
<coverageFunction>
   <GridFunction>
      <sequenceRule order="+x-y">Linear</sequenceRule>
      <startPoint>0 1</startPoint>
   </GridFunction>
<coverageFunction>
```

Note that if the `startPoint` was not specified, the default starting point would have been the value of `low`, (-1,-1).

Encoding a Rectified Grid

The following sample instance shows the entire temperature and pressure coverage, `TempPressure`, with range data encoded as a `DataBlock`:

```
<app:TempPressure>
   <gml:rectifiedGridDomain>
```

```
       <gml:RectifiedGrid dimension="2">
          <gml:limits>
             <gml:GridEnvelope>
                <gml:low>-1 -1</gml:low>
                <gml:high>2 2</gml:high>
             </gml:GridEnvelope>
          </gml:limits>
          <gml:axisName>u</gml:axisName>
          <gml:axisName>v</gml:axisName>
          <gml:origin>
             <gml:Point gml:id="O"
srsName="urn:epsg:v6.1:coordinateReferenceSystem:4267">
                <gml:coordinates>25,27</gml:coordinates>
             </gml:Point>
          </gml:origin>
          <gml:offsetVector>1, 0.2</gml:offsetVector>
          <gml:offsetVector>-0.2, 1</gml:offsetVector>
       </gml:RectifiedGrid>
   </gml:rectifiedGridDomain>
   <gml:rangeSet>
       <gml:DataBlock>
          <gml:rangeParameters>
             <gml:CompositeValue>
                <gml:valueComponents>
                   <app:Temp uom="urn:x-si:v1999:uom:degreesC"/>
                   <app:Pressure uom="urn:x-si:v1999:uom:kPa"/>
                </gml:valueComponent>
             </gml:CompositeValues>
          </gml:rangeParameters>
          <gml:tupleList>3,101.2 5,101.3 7,101.4 11,101.5 13,101.6 17,101.7
19,101.7 23,101.8 29,101.9 31,102.0 37,102.1 41,102.2 43,102.3 47,102. 53,102.5
59,102.6</gml:tupleList>
       </gml:DataBlock>
   </gml:rangeSet>
</app:TempPressure>
```

Example: Gridded Digital Elevation Model (DEM)

This example is of a gridded DEM encoded in GML as a `RectifiedGridCoverage` with range data encoded as a `File`. The schema is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.opengis.net/app" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml" xmlns:app="http://www.opengis.net/app" elementFormDefault="qualified"
version="2.06">
  <!-- include constructs from the GML observation schema -->
  <import namespace="http://www.opengis.net/gml" schemaLocation="../GML-3.00/base/gml.xsd"/>
  <!-- ==============================================================
    global declarations
  ================================================================= -->
  <element name="GriddedDEM" type="app:GriddedDEMType" substitutionGroup="gml:_Coverage"/>
  <!-- ==============================================================-->
  <complexType name="GriddedDEMType">
      <annotation>
          <documentation>The gridded DEM files shall conform to the National Topographic System (NTS) for
1:250,000 scale. The 1:250,000 scale NTS quadrangles are each 1 degree of latitude by 2 degrees of longitude in
size. Data for each file will extend at least 250 metres past the range rectangle fully contained by the NTS
mapsheet neat line. This implies the adjacent sheets will contain exact duplicate data in the overlapping areas. All
data is presented on the Universal Transverse Mercator Coordinate System based on the 1983 North American
Datum. The vertical datum is mean sea level as established by the Geodetic Survey of Canada. All data will be
interpolated using a linear interpolation process. Extrapolation processes will not be used except in ocean areas on
the perimeter of the data set. TRIM DEM data will be used to interpolate a grid at an even 25 metre spacing. The
eastings and northings will be gridded so they are evenly divisible by 25 metres. The eastings and northings of the
outside extents of the data shall be evenly divisible by 100 metres. Please note that this requirement, in
conjunction with the 250 metre margin requirement, may mean the actual extents of the data will be greater that
either requirement would otherwise demand.
```

```
</documentation>
      </annotation>
      <complexContent>
          <restriction base="gml:AbstractCoverageType">
              <sequence>
                  <element ref="gml:metaDataProperty" minOccurs="0" maxOccurs="unbounded"/>
                  <element ref="gml:description" minOccurs="0"/>
                  <element ref="gml:name" minOccurs="0" maxOccurs="unbounded"/>
                  <element ref="gml:boundedBy" minOccurs="0"/>
                  <element ref="gml:location" minOccurs="0"/>
                  <element ref="gml:rectifiedGridDomain"/>
                  <element ref="app:rangeSet"/>
                  <element ref="gml:coverageFunction" minOccurs="0"/>
              </sequence>
          </restriction>
      </complexContent>
  </complexType>
  <!-- ============================================================ -->
  <element name="rangeSet" type="app:FileRangeSetType" substitutionGroup="gml:rangeSet"/>
  <!-- ============================================================ -->
  <complexType name="FileRangeSetType">
      <complexContent>
          <restriction base="gml:RangeSetType">
              <sequence>
                  <element ref="app:File"/>
              </sequence>
          </restriction>
      </complexContent>
  </complexType>
  <!-- ============================================================ -->
  <element name="File">
      <complexType name="FileType">
          <annotation>
              <documentation>Elevations will be stored as pixels in signed 16 bit integer binary format. Negative
values are considered to be valid elevations. Elevations are to be stored in the file with the top left (northwest
corner) pixel first, followed by the remaining elevation values of the row in sequential order. Rows of
elevations/pixels are to be stored from the top to the bottom (southward) direction. No invalid values are allowed in
the dataset, unless the boundaries of the Province are encountered. In this case, an invalid value will have an
elevation value of -9999 metres. Each file will be named according to the NTS 1:250,000 naming convention and a
three digit extension. All gridded DEM files will have the extension .GRD. The elevation's/pixel's easting and
northing position will be positioned on the northwest corner of the pixel, however the elevation value shall be
calculated on the position of the centre of the pixel. Breakline data shall be given more weight in the surface
modeling process than DEM Points (mass points). Indefinite DEM Points shall be given less weight than DEM
Points. All elevations interpolated are to be stored to the nearest metre.
 </documentation>
          </annotation>
          <sequence>
              <element ref="gml:rangeParameters"/>
              <element name="fileName" type="anyURI"/>
              <element name="fileDate" type="date"/>
              <element name="fileStructure" type="gml:FileValueModelType"/>
              <element name="mimeType" type="anyURI" minOccurs="0"/>
              <element name="compression" type="anyURI" minOccurs="0"/>
          </sequence>
      </complexType>
  </element>
  <!-- =============================================================-->
  <element name="Elevation" substitutionGroup="gml:_Value">
      <complexType>
          <simpleContent>
              <extension base="string">
                  <attribute name="uom" type="anyURI" use="required"/>
              </extension>
          </simpleContent>
      </complexType>
  </element>
```

```
</schema>
```

A sample instance of the gridded DEM could be as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<GriddedDEM dimension="2" xmlns="http://www.opengis.net/app" xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/app
GriddedDEM.xsd">
   <gml:rectifiedGridDomain>
        <gml:RectifiedGrid dimension="2" srsName="urn:epsg:v6.1:coordinateReferenceSystem:4267">
             <gml:limits>
                  <gml:GridEnvelope>
                       <gml:low>430000 5430000</gml:low>
                       <gml:high>570000 5540000</gml:high>
                  </gml:GridEnvelope>
             </gml:limits>
             <gml:axisName>x</gml:axisName>
             <gml:axisName>y</gml:axisName>
             <gml:origin>
                  <gml:Point gml:id="O">
                       <gml:pos>430000 5430000</gml:pos>
                  </gml:Point>
             </gml:origin>
             <gml:offsetVector>25 0</gml:offsetVector>
             <gml:offsetVector>0 25</gml:offsetVector>
        </gml:RectifiedGrid>
   </gml:rectifiedGridDomain>
   <rangeSet>
        <File>
             <gml:rangeParameters>
                  <Elevation uom="urn:x-si:v1999:uom:metres"/>
             </gml:rangeParameters>
             <fileName>92g.grd</fileName>
             <fileDate>1996-03-23</fileDate>
             <fileStructure>Record Interleaved</fileStructure>
        </File>
   </rangeSet>
   <gml:coverageFunction>
        <gml:GridFunction>
             <gml:sequenceRule order="+x-y">Linear</gml:sequenceRule>
             <gml:startPoint>430000 5540000</gml:startPoint>
        </gml:GridFunction>
   </gml:coverageFunction>
</GriddedDEM>
```

Non-spatial Coverages

This section includes sample instances of coverages over a feature domain (`FishInventoryCoverage`) and a temporal domain (`MultiTimeCoverage`) along with the corresponding application schemas.

```
<FishInventoryCoverage xmlns:app="http://www.opengis.net/examples" xmlns="http://www.opengis.net/examples"
xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/examples
FeatureCoverage.xsd">
     <multiFeatureDomain>
          <FeatureBag>
               <gml:featureMember xlink:href="urn:x-msrm:v0.1:hydrographyDataSet#Lake0001"/>
               <gml:featureMember xlink:href="urn:x-msrm:v0.1:hydrographyDataSet#Lake0001"/>
               <gml:featureMember xlink:href=" urn:x-msrm:v0.1:hydrographyDataSet#River0011 "/>
               <gml:featureMember xlink:href=" urn:x-msrm:v0.1:hydrographyDataSet#Stream0031"/>
          </FeatureBag>
     </multiFeatureDomain>
     <gml:rangeSet>
```

```
        <gml:DataBlock>
            <gml:rangeParameters>
                <gml:CompositeValue>
                    <gml:valueComponents>
                        <FishCount/>
                        <FishSpecies/>
                    </gml:valueComponents>
                </gml:CompositeValue>
            </gml:rangeParameters>
            <gml:tupleList>350,SteelHeadTrout 301,PinkSalmon 23,PinkSalmon 45,SockeyeSalmon</gml:tupleList>
        </gml:DataBlock>
    </gml:rangeSet>
</FishInventoryCoverage>
```

The following schema defines the `FishInventoryCoverage` and its supporting components:

```
<schema targetNamespace="http://www.opengis.net/examples" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:app="http://www.opengis.net/examples" elementFormDefault="qualified" version="0.0">
    <import namespace="http://www.opengis.net/gml" schemaLocation="../../base/gml.xsd"/>
    <!-- ============================================================ -->
    <element name="FishInventoryCoverage" type="app:MultiFeatureCoverageType" substitutionGroup="gml:_Coverage"/>
    <!-- ============================================================ -->
    <complexType name="MultiFeatureCoverageType">
        <annotation>
            <documentation>A discrete coverage type whose domain is defined by a bag of features.
            </documentation>
        </annotation>
        <complexContent>
            <restriction base="gml:AbstractCoverageType">
                <sequence>
                    <element ref="app:multiFeatureDomain"/>
                    <element ref="gml:rangeSet"/>
                    <element ref="gml:coverageFunction" minOccurs="0"/>
                </sequence>
            </restriction>
        </complexContent>
    </complexType>
    <!-- ============================================================ -->
    <element name="multiFeatureDomain" type="app:MultiFeatureDomainType" substitutionGroup="gml:domainSet"/>
    <!-- ============================================================ -->
    <complexType name="MultiFeatureDomainType">
        <complexContent>
            <extension base="app:FeatureDomainSetBaseType">
                <sequence minOccurs="0">
                    <element ref="app:FeatureBag"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <!-- ============================================================ -->
    <complexType name="FeatureDomainSetBaseType" abstract="true" >
        <complexContent>
            <restriction base="gml:DomainSetType">
                <sequence/>
            </restriction>
        </complexContent>
    </complexType>
    <!-- ========== Concrete "Collection" supertype ======================== -->
    <element name="FeatureBag" substitutionGroup="gml:_GML">
      <complexType>
    <annotation>
                <documentation>An aggregate of Features, which is not itself a Feature.</documentation>
           </annotation>
           <complexContent>
            <extension base="gml:AbstractGMLType">
                <sequence>
```

```
          <element ref="gml:featureMember" minOccurs="0" maxOccurs="unbounded"/>
          <element ref="gml:featureMembers" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
   </complexType>
  </element>
  <!-- ==============================================================-->
  <element name="FishCount" type="gml:CountType" substitutionGroup="gml:_Value"/>
  <!-- ==============================================================-->
  <element name="FishSpecies" type="gml:CodeType"  substitutionGroup="gml:_Value"/>

</schema>
```

The following sample instance of `MultiTimeCoverage` gives an example of a coverage defined over a temporal-valued domain:

```
<MultiTimeCoverage xmlns="http://www.opengis.net/app"
xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.opengis.net/app MultiTimeCoverage.xsd">
  <multiTimeDomain>
     <MultiTimeInstant>
        <timeMembers>
           <gml:TimeInstant gml:id=”T1”>
              <gml:timePosition>1999-12-2T11:01-08:00</gml:timePosition>
           </gml:TimeInstant>
           <gml:TimeInstant gml:id=”T2”>
              <gml:timePosition>1999-12-2T11:11:00-08:00</gml:timePosition>
           </gml:TimeInstant>
           <gml:TimeInstant gml:id=”T3”>
              <gml:timePosition>1999-12-2T11:21:00-08:00</gml:timePosition>
           </gml:TimeInstant>
        </timeMembers>
     </MultiTimeInstant>
  </multiTimeDomain>
  <gml:rangeSet>
     <gml:ValueArray>
        <gml:valueComponents>
           <TemperatureReading uom="urn:x-si:v1999:uom:degreesC">3</TemperatureReading>
           <TemperatureReading uom="urn:x-si:v1999:uom:degreesC">5</TemperatureReading>
           <TemperatureReading uom="urn:x-si:v1999:uom:degreesC">7</TemperatureReading>
        </gml:valueComponents>
     </gml:ValueArray>
  </gml:rangeSet>
</MultiTimeCoverage>
```

The following schema defines the `MultiTimeCoverage` and its supporting components:

```
  <element name="MultiTimeCoverage" type="app:MultiTimeCoverageType"
substitutionGroup="gml:_Coverage"/>
  <!-- ========================================================== -->
  <complexType name="MultiTimeCoverageType">
     <annotation>
        <documentation>A discrete coverage type whose domain is defined by a collection
of timePositions
        </documentation>
     </annotation>
     <complexContent>
        <restriction base="gml:AbstractCoverageType">
           <sequence>
              <element ref="app:multiTimeDomain"/>
              <element ref="gml:rangeSet"/>
              <element ref="gml:coverageFunction" minOccurs="0"/>
           </sequence>
        </restriction>
     </complexContent>
  </complexType>
  <!-- ========================================================== -->
  <element name="multiTimeDomain" type="app:MultiTimeDomainType"
substitutionGroup="gml:domainSet"/>
  <!-- ========================================================== -->
  <complexType name="MultiTimeDomainType">
     <complexContent>
        <restriction base="gml:DomainSetType">
```

```
           <sequence minOccurs="0">
              <element ref="app:MultiTimeInstant"/>
           </sequence>
        </restriction>
     </complexContent>
  </complexType>
  <!-- ========================================================== -->
  <element name="MultiTimeInstant" type="app:MultiTimeInstantType"
substitutionGroup="gml:_TimeObject"/>
  <!-- ========================================================== -->
  <complexType name="MultiTimeInstantType">
     <complexContent>
        <restriction base="gml:AbstractTimeType">
           <sequence>
              <element ref="app:timeMember" minOccurs="0" maxOccurs="unbounded"/>
              <element ref="app:timeMembers" minOccurs="0"/>
           </sequence>
        </restriction>
     </complexContent>
  </complexType>
  <!-- ========================================================== -->
  <element name="timeMember" type="gml:TimeInstantPropertyType"/>
  <!-- ========================================================== -->
  <element name="timeMembers" type="app:TimeInstantArrayType"/>
  <!-- ============================================================-->
  <complexType name="TimeInstantArrayType">
     <complexContent>
        <restriction base="gml:ArrayAssociationType">
           <sequence>
              <element ref="gml:TimeInstant" minOccurs="0" maxOccurs="unbounded"/>
           </sequence>
        </restriction>
     </complexContent>
  </complexType>
```

3.4 Coordinate Reference Systems

A Coordinate Reference System (CRS) describes the spatial context of coordinates. A CRS consists of a Coordinate System (CS) that is related to the real world by a datum. In practice, a coordinate reference system is often referenced from a CRS dictionary by a geometry element in a data instance. For that purpose, a GML attribute called srsName is built into the base geometry type and inherited by every concrete geometry type.

The srsName attribute has anyURI type and the value of the attribute is a URI that may point to an entry in a CRS dictionary. The following example shows a geometry instance that references a location independent CRS dictionary using a URN.

```
<MultiPoint gml:id=”MP1” srsName="urn:epsg:v6.1:coordinateReferenceSystem:4267">
   <pointMembers>
      <Point gml:id=”P1”>
         <coordinates>23378428,25959999</coordinates>
      </Point>
      <Point gml:id=”P2”>
         <coordinates>43411784,25959743</coordinates>
      </Point>
   </pointMembers>
</MultiPoint>
```

If the srsName attribute is not provided on a geometry element (such as on the Point geometries above) then the CRS must be specified as part of the larger context this geometry is part of, for example on the aggregate MultiPoint. See Sections 6.1 and 6.7 for further discussion of this topic and best practice recommendations.

Creating New CRS Dictionary Entries

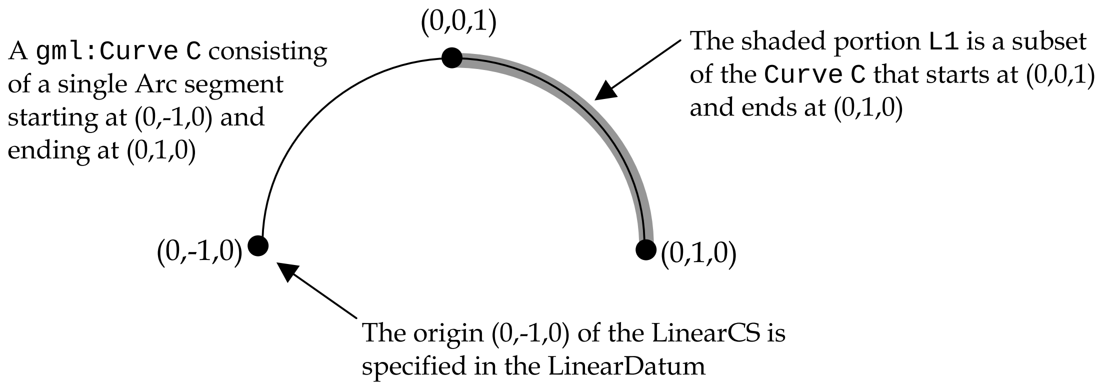
A CRS dictionary entry can be created in a GML instance document using the GML types defined in the CRS and supporting schemas. Some of the commonly used types provided in the CRS schema are `GeographicCRSType`, `GeocentricCRSType`, `EngineeringCRSType` and `ProjectedCRSType`. This section provides a sample GML instance of a CRS Dictionary, identified by EPSG codes of some common reference systems used in BC. For example: BC Albers using NAD83 datum. The CRS instances are identified by EPSG codes by setting the value “EPSGxyzw” to the `gml:id` attribute. Note that the lexical constraint on the XML Schema ID attribute prohibits the use of the colon symbol (:) in the value of `gml:id`. The corresponding application schema follows the sample instance.

```
<Dictionary xmlns="http://www.opengis.net/gml" xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:msrm="http://www.msrm.gov.bc.ca/schema/base" xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/gml
..\GML-3.00\base\coordinateReferenceSystems.xsd" gml:id="urn.msrm.CRSDictionary.v0.1">
  <description>Dictionary of standard MSRM CRSs.</description>
  <name>MSRM CRS Dictionary</name>
  <dictionaryEntry>
      <GeographicCRS gml:id="EPSG62696405">
          <srsName>NAD 83 (deg)</srsName>
          <srsID>
              <code>62696405</code>
              <codeSpace>EPSG</codeSpace>
              <version>0.1</version>
          </srsID>
          <validArea>
              <description>North America</description>
          </validArea>
          <usesEllipsoidalCS>
              <EllipsoidalCS gml:id="EPSG6405">
                  <remarks>Axis order is by element order.</remarks>
                  <csName>ellipsoidal</csName>
                  <csID>
                      <code>6405</code>
                      <codeSpace>EPSG</codeSpace>
                      <version>6.0</version>
                  </csID>
                  <usesAxis>
                      <CoordinateSystemAxis gml:id="EPSG9901" uom="urn:x-epsg:v0.1:uom:degree">
                          <remarks>Axis order = 1</remarks>
                          <axisName>Geodetic latitude</axisName>
                          <axisID>
                              <code>9901</code>
                              <codeSpace>EPSG</codeSpace>
                              <version>6.0</version>
                          </axisID>
                          <axisAbbrev>Lat</axisAbbrev>
                          <axisDirection>north</axisDirection>
                      </CoordinateSystemAxis>
                  </usesAxis>
                  <usesAxis>
                      <CoordinateSystemAxis gml:id="EPSG9902" uom="urn:x-epsg:v0.1:uom:degree">
                          <remarks>Axis order = 2</remarks>
                          <axisName>Geodetic longitude</axisName>
                          <axisID>
                              <code>9902</code>
                              <codeSpace>EPSG</codeSpace>
                              <version>6.0</version>
```

```
                            </axisID>
                            <axisAbbrev>Long</axisAbbrev>
                            <axisDirection>east</axisDirection>
                        </CoordinateSystemAxis>
                    </usesAxis>
                </EllipsoidalCS>
            </usesEllipsoidalCS>
            <usesGeodeticDatum>
                <GeodeticDatum gml:id="EPSG6269">
                    <datumName>North American Datum 1983</datumName>
                    <datumID>
                        <code>6269</code>
                        <codeSpace>EPSG</codeSpace>
                        <version>0.1</version>
                    </datumID>
                    <usesPrimeMeridian>
                        <PrimeMeridian gml:id="EPSG8901">
                            <meridianName>Greenwich</meridianName>
                            <meridianID>
                                <code>8901</code>
                                <codeSpace>EPSG</codeSpace>
                                <version>6.0</version>
                            </meridianID>
                            <greenwichLongitude>
                                <angle uom="urn:x-epsg:v0.1:uom:degree">0</angle>
                            </greenwichLongitude>
                        </PrimeMeridian>
                    </usesPrimeMeridian>
                    <usesEllipsoid>
                        <Ellipsoid gml:id="EPSG7019">
                            <ellipsoidName>GRS 1980</ellipsoidName>
                            <ellipsoidID>
                                <code>7019</code>
                                <codeSpace>EPSG</codeSpace>
                                <version>6.0</version>
                            </ellipsoidID>
                            <semiMajorAxis uom="urn:x-si:v1999:uom:metre">6378137</semiMajorAxis>
                            <secondDefiningParameter>
                                <inverseFlattening uom="urn:x-
bagug:v0.1:dictionary:ifu">298.257222101</inverseFlattening>
                            </secondDefiningParameter>
                        </Ellipsoid>
                    </usesEllipsoid>
                </GeodeticDatum>
            </usesGeodeticDatum>
        </GeographicCRS>
    </dictionaryEntry>
    <dictionaryEntry>
        <ProjectedCRS gml:id="MSRM0001">
            <srsName>BC Projected CRS Equal-Area, Central Meridian 139W</srsName>
            <srsID>
                <code>0002</code>
                <codeSpace>MSRM</codeSpace>
                <version>0.1</version>
                <remarks>Originally administered by Geographic Data BC (GDBC).</remarks>
            </srsID>
            <validArea>
                <description>Intended for use between the northern and southern limits of British Columbia,
bounded EW by -140W and -138W degrees longitude.</description>
            </validArea>
            <baseCRS xlink:href="#EPSG62696405" xlink:title="NAD 83 (deg)"/>
            <definedByConversion>
                <Conversion gml:id="X-MSRMC0001">
                    <coordinateOperationName>BC Albers Equal-Area</coordinateOperationName>
                    <coordinateOperationID>
                        <code>C0001</code>
            <codeSpace>MSRM</codeSpace>
            <version>0.1</version>
          </coordinateOperationID>
          <usesMethod xlink:href="urn:x-msrm:v0.1:operationMethod:X-MSRMOM0001" xlink:title="BC
Albers Equal-Area"/>
          <usesValue>
            <value uom="urn:x-epsg:v0.1:uom:degree">45.0</value>
            <valueOfParameter xlink:href="urn:epsg:v6.1:coordinateOperationParameter:8801"
xlink:title="Latitude of natural origin"/>
          </usesValue>
          <usesValue>
            <value uom="urn:x-epsg:v0.1:uom:degree">-126.0</value>
            <valueOfParameter xlink:href="urn:epsg:v6.1:coordinateOperationParameter:8802"
xlink:title="Longitude of natural origin"/>
          </usesValue>
          <usesValue>
            <value uom="urn:x-epsg:v0.1:uom:degree">50.0</value>
            <valueOfParameter xlink:href="urn:epsg:v6.1:coordinateOperationParameter:8823"
xlink:title="Latitude of 1st standard parallel"/>
          </usesValue>
          <usesValue>
            <value uom="urn:x-epsg:v0.1:uom:degree">58.5</value>
            <valueOfParameter xlink:href="urn:epsg:v6.1:coordinateOperationParameter:8824"
xlink:title="Latitude of 2nd standard parallel"/>
          </usesValue>
          <usesValue>
            <value uom="urn:x-si:v1999:uom:metre">400000</value>
            <valueOfParameter xlink:href="urn:epsg:v6.1:coordinateOperationParameter:8826"
xlink:title="Easting at false origin"/>
          </usesValue>
          <usesValue>
            <value uom="urn:x-si:v1999:uom:metre">-100000</value>
            <valueOfParameter xlink:href="urn:epsg:v6.1:coordinateOperationParameter:8827"
xlink:title="Northing at false origin"/>
          </usesValue>
        </Conversion>
      </definedByConversion>
      <usesCartesianCS>
        <CartesianCS gml:id="EPSG4400">
          <csName>Cartesian</csName>
          <csID>
            <code>4400</code>
            <codeSpace>EPSG</codeSpace>
            <version>6.0</version>
          </csID>
          <usesAxis xlink:href="urn:x-epsg:v6.0:coordinateSystemAxis:EPSG9906" xlink:title="Easting"/>
          <usesAxis xlink:href="urn:x-epsg:v6.0:coordinateSystemAxis:EPSG9907" xlink:title="Northing"/>
        </CartesianCS>
      </usesCartesianCS>
    </ProjectedCRS>
  </dictionaryEntry>
</Dictionary>
```

Defining New CRS Components

GML allows for the creation of stand-alone support components for a CRS such as a CS or datum, for instance. Suppose we want to define a new datum called `LinearDatum` to be used with `gml:EngineeringCRS` and the coordinate system `gml:LinearCS`. Part of the motivation for defining such a datum (used by `gml:EngineeringCRS`) might be to define a parameterized curve based on a `gml:Curve`. For example, the shaded portion `L1` of the curve `C` shown in Figure 3-14 is such a "parameterized" curve:



A `gml:Curve C` consisting of a single Arc segment starting at (0,-1,0) and ending at (0,1,0)

The shaded portion `L1` is a subset of the `Curve C` that starts at (0,0,1) and ends at (0,1,0)

The origin (0,-1,0) of the LinearCS is specified in the LinearDatum

Figure 3-14: The `LinearDatum` specifies the `gml:Curve` on which the CRS is based

A sample instance of the "parameterized" curve `L1` could be as follows:

```
<LineString gml:id="L1" srsName="urn:x-opengis:0.1:EngineeringCRS:1234">
      <pos>1.570796327</pos>
<!-- This must be interpreted as 1.570796327 meters along curve past the origin -->
      <pos>3.141592654</pos>
<!—3.141592654 meters along curve past the origin -->
</LineString>
```

A sample instance corresponding to the value of `srsName` above might be as follows.

```
<!-- ============================================================= -->
<EngineeringCRS gml:id="1234">
      <gml:crsID>
            <gml:name/>
      </gml:crsID>
      <gml:usesCS>
            <LinearCS>
                  <gml:csID/>
                  <gml:usesAxis>
                        <gml:CoordinateSystemAxis uom="urn:x-si:v1999:unit:metre">
                              <gml:axisAbbrev>s</gml:axisAbbrev>
                              <gml:axisDirection/>
                        </gml:CoordinateSystemAxis>
                  </gml:usesAxis>
            </LinearCS>
      </gml:usesCS>
      <gml:usesDatum>
            <LinearDatum>
                  <datumID/>
                  <usesCurve xlink:href="#C"/>
                  <origin uom="urn:x-si:v1999:unit:metre">0</origin>
                  <gml:parameterization>Arclength</parameterization>
            </LinearDatum>
      </gml:usesDatum>
</EngineeringCRS>
<!-- ============================================================= -->
```

The schema definition of the `LinearDatum` could be as follows:

```
<element name="LinearDatum" type="app:LinearDatumType" substitutionGroup="gml:EngineeringDatum"/>
<!-- ============================================================= -->
<complexType name="LinearDatumBaseType">
      <annotation>
            <documentation>Restricts EngineeringDatumType by removing the realizationEpoch property</documentation>
      </annotation>
```

```
        <complexContent>
            <restriction base="gml:EngineeringDatumType">
                <sequence>
                    <element name="datumID" type="gml:ExtendedIdentifierType">
                        <annotation>
                            <documentation>Identification of this datum. </documentation>
                        </annotation>
                    </element>

                   <element ref="gml:anchorPoint" minOccurs=”0”/>

                   <element ref="gml:validArea" minOccurs=”0”/>

                </sequence>
            </restriction>
        </complexContent>
    </complexType>
    <!-- ============================================================ -->
    <complexType name="LinearDatumType">
        <annotation>
            <documentation> </documentation>
        </annotation>
        <complexContent>
            <extension base="app:LinearDatumBaseType">
                <sequence>
                    <element name="usesCurve" type="gml:CurvePropertyType"/>
                    <element name="origin" type="gml:PointPropertyType" minOccurs="0"/>
                    <element name="parameterization" type="ParametizationType" default="Arclength" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <!-- ============================================================== -->
    <simpleType name="ParameterizationType">

        <restriction base="string">
            <enumeration value="Arclength"/>
            <enumeration value="Normalized"/>
        </restriction>
    </simpleType>
```

The value of `usesCurve` is the `gml:Curve`, for example the `Curve C`, that the CRS is based on. The value of the optional property `origin` is a `gml:Point` that anchors the `linearCS` to the `Curve C` and corresponds to the origin (0-value) on the single axis used by the `linearCS`. The parameterization value “Normalized” indicates that the portion of the curve from the `origin` to the end is (re)parameterized from 0 to 1, where 0 corresponds to the origin and 1 corresponds to the end of the curve. In essence the parameterized curve with `parameterization` value “Normalized” can be thought as ‘percentage' along the curve. If the value of the `parameterization` property in `LinearDatum` were change from `Arclength` to `Normalized`, the corresponding instance encoding of `L1` would be as follows:

```
<LineString gml:id="L1" srsName="urn:x-opengis:0.1:EngineeringCRS:1234">
        <pos>0.5</pos>
<!-- This must be interpreted as half-way (50%) along the curve past the origin -->
      <pos>1</pos>
<!-- This would be interpreted as the end of the curve (100% along the curve) -->
</LineString>
```

3.5 Temporal Components and Dynamic Features

In GML 3.0, you can model features that are moving objects or have dynamic properties. For example, forest-fire or flood boundaries, moving vehicles and evolving disaster situations. This section summarizes the temporal components that play a role in describing the dynamic properties of features.

Temporal Primitives

The two geometric time primitives provided in GML 3.0 are `TimeInstant` and `TimePeriod`. `TimeInstant` is used to represent positions in time, while `TimePeriod` represents temporal length or duration. An example instance demonstrating the use of `TimeInstant` is as follows:

```
<gml:TimeInstant gml:id="T1">
  <gml:timePosition>
     2003-01-01T12:34-08:00
  </gml:timePosition>
</gml:TimeInstant>
```

The `timePosition` value "2003-01-01T12:34-08:00" follows the format defined in the ISO 8601 reference system (Gregorian Calendar), where "2003-02-13" represents the date "12:28" represents the hour and minutes and "-8:00" indicates the time zone, i.e. eight hours behind Greenwich time (Pacific Standard Time).

The `TimePeriod` primitive is typically used to encode an interval of time between two `TimeInstants`. In GML instances, a `TimePeriod` must have either a `begin` or an `end` property. Each property has a child `TimeInstant` element that provides the time values for the beginning or end of the period. If only one of these properties is provided, then a `duration` must also be included to implicitly specify the position of the other one. For example, an interval that begins at 2003-01-01 and has a `duration` of six days ends at 2003-01-07.

```
<gml:TimePeriod>
   <gml:begin>
      <gml:TimeInstant>
         <gml:timePosition>
            2002-01-01T21:29-08:00
         </gml:timePosition>
      </gml:TimeInstant>
   </gml:begin>
   <gml:end>
      <gml:TimeInstant>
         <gml:timePosition xlink:href="#T1"/>
      <gml:TimeInstant>
   </gml:end>
</gml:TimePeriod>
```

Temporal Reference Systems

Temporal reference systems, such as calendars, provide the time measurements for temporal objects. The GML default temporal reference system is the same as the one defined in ISO 8601—the Gregorian calendar with Coordinated Universal Time (UTC). Other reference systems—such as the Global Positioning System (GPS) calendar and the Julian calendar—can also be used.

To reference a calendar, you need to include the frame attribute with one of the temporal objects discussed above. The following example shows how to reference the Gregorian calendar from a gml:TimePosition property:

```
<gml:TimeInstant>
   <gml:timePosition frame="#ISO-8601">...</gml:timePosition>
</gml:TimeInstant>
```

If the timePosition property does not specify a value for the frame attribute, then the property automatically references the Gregorian calendar, the default temporal reference system in GML. The frame attribute is also used to reference temporal ordinal reference systems and temporal coordinate reference systems.

Dynamic Features

Dynamic features and feature collections can be defined in GML using DynamicFeatureType and DynamicFeatureCollectionType. These types are derived from AbstractFeatureType and FeatureCollectionType, respectively. Both a dynamic feature and a dynamic feature collection have a history property that may be used to express its temporal development. The history property of a dynamic feature contains a sequence of time slices that capture the evolution of the feature over time. A gml:TimeSlice object contains the dynamic properties of the feature. To define a dynamic feature, such as a moving bus, in an application schema, you can derive from DynamicFeatureType, as shown in the following element declaration and content model for a Bus feature:

```
<element name="Bus" type="app:BusType"
substitutionGroup="gml:_Feature"/>

<complexType name="app:BusType">
    <complexContent>
       <extension base="gml:DynamicFeatureType">
          <sequence>
             <element name="routeNumber" type="positiveInteger"/>
             <element name="maxPassengers" type="positiveInteger"/>
          </sequence>
       </extension>
   </complexContent>
</complexType>
```

In addition to inheriting the history property from DynamicFeatureType, the BusType also specifies additional static properties, routeNumber and maxPassengers. Note that to define a feature collection with dynamic properties, you can derive by extension from DynamicFeatureCollectionType. A sample Bus instance might be as follows:

```
<Bus gml:id="B999">
  <history>
     <TimeSlice gml:id="TS-0">
        <gml:timeStamp>
           <gml:TimeInstant>
              <gml:timePosition>2003-02-24T14:22-8:00</gml:timePosition>
           </gml:TimeInstant>
        </gml:timeStamp>
        <gml:location>
```

```
          <gml:Point srsName="urn:epsg:v6.1:coordinateReferenceSystem:4267">
             <gml:pos>0,100</pos>
          </gml:Point>
       </gml:location>
       <numPassengers>25</numPassengers>
    </TimeSlice>
    <TimeSlice gml:id="TS-1">
       <gml:timeStamp>
          <gml:TimeInstant>
             <gml:timePosition>2003-02-24T15:11-8:00</gml:timePosition>
          </gml:TimeInstant>
       </gml:timeStamp>
       <gml:location>
          <gml:Point srsName="urn:epsg:v6.1:coordinateReferenceSystem:4267">
             <gml:pos>0,0</pos>
          </gml:Point>
       </gml:location>
       <numPassengers>20</numPassengers>
    </TimeSlice>
    <TimeSlice gml:id="TS-2">
       <gml:timeStamp>
          <gml:TimeInstant>
             <gml:timePosition>2003-02-24T15:22-8:00</gml:timePosition>
          </gml:TimeInstant>
       </gml:timeStamp>
       <gml:location>
          <gml:Point srsName="urn:epsg:v6.1:coordinateReferenceSystem:4267">
             <gml:pos>100,0</pos>
          </gml:Point>
       </gml:location>
       <numPassengers>15</numPassengers>
    </TimeSlice>
 </history>
 <routeNumber>44</routeNumber>
 <maxPassengers>60</maxPassengers>
</Bus>
```

3.6 Units of Measure

In many geo-spatial applications, it is important to be able to associate units of measure to quantities. In practice, units of measure are often referenced from a dictionary. For that purpose, a GML attribute called uom is provided. The uom attribute is of anyURI type and the value of the attribute is a URI that points to an entry in a unit dictionary. The following example shows an instance that references a unit dictionary.

```
<app:RailCar gml:id="RC1">
   <app:length uom="urn:x-si:v1999:uom:metre">10</app:length>
       ...
</app:RailCar>
```

The reference in the uom attribute can be interpreted in two ways: as a valid URI pointing to a unit of measure dictionary entry, or as an identifier that represents a well-known unit, such as the units in the SI system.

The corresponding content model of RailCarType in an application schema fragment might look like the following:

```
<element name = “RailCar”  type = “app:RailCarType” substitutionGroup =
“gml:_Feature”/>

<complexType name = “RailCarType”>
   <complexContent>
      <extension base =”gml:AbstractFeatureType”>
         <element name = “length”  type = “app:LengthType”
         ... other feature properties
      </extension>
   </complexContent>
</complexType>

<complexType name = “LengthType”>
   <simpleContent>
      <extension base =”float”>
         <sequence/>
      </extension>
   </simpleContent>
   <attribute name=”uom” type=”anyURI” use=”required”/>
</complexType>
```

GML 3.0 provides a set of schema components for defining units of measure and units of measure dictionaries. Units of measure dictionaries are based on the dictionary model that is defined in `dictionary.xsd`.

Base Unit Dictionaries

Base units are independent units of measure—that is, they cannot be a combination of other units. They are the most commonly used units for fundamental quantities, such as length, mass and time. Table Table 3-1 lists the seven base units and their corresponding quantity types, as defined by the International System of Units (SI).

Table 3-1: SI Base Units and Quantity Types

Base Unit	Quantity Type
Metre	Length
Kilogram	Mass
Second	Time interval
Ampere	Electric current
Kelvin	Thermodynamic temperature
Mole	Amount of substance
Candela	Luminous intensity

A sample dictionary entry for the base unit "metre" might look like:

```
<gml:Dictionary gml:id="unitsDictionary">
   ...
<gml:dictionaryEntry>
   <gml:DefinitionCollection gml:id="SIBaseUnits">
      <gml:description>The Base Units from the SI units
   system.</gml:description>
      <gml:name>SI Base Units</gml:name>
         <gml:dictionaryEntry>
            <gml:BaseUnit gml:id="metre">
               <gml:description>...</gml:description>
               <gml:name codeSpace="http://www.bipm.fr/
            en/3_SI/base_units.html">metre</gml:name>
               <gml:name xml:lang="en/US">meter</gml:name>
               <gml:quantityType>length</gml:quantityType>
               <gml:catalogSymbol codeSpace="http://www.bipm.fr/
            en/3_SI/base_units.html">m</gml:catalogSymbol>
               <gml:unitsSystem
   xlink:href="http://www.bipm.fr/en/3_SI"/>
            </gml:BaseUnit>
         </gml:dictionaryEntry>
         ...
      </DefinitionCollection>
   </gml:dictionaryEntry>
</gml:Dictionary>
```

The rules and types for creating more general units of measures (derived units) are beyond the scope of this guide but are covered in detail in the GML 3.0 Technical Guide.

4 GML Modeling Rules

GML builds on XML Schema, and makes heavy use of XML Schema mechanisms to define GML types and to achieve extensibility. As a result, there are a few normative rules for GML that must be adhered to by all GML Application Schema developers.

4.1 Feature Types

In GML, geographic features are represented exclusively by global XML element declarations in the GML application schema. Features cannot be represented as attributes. Note that as a consequence of this rule, GML data instances contain elements whose names specify a feature type. For example the global element declaration of a Road feature could be as follows:

```
<element name="Road" substitutionGroup="gml:_Feature"/>
```

The content models of GML feature types must derive, directly or indirectly from gml:AbstractFeatureType. It is recommended that the feature element is also made substitutable for the GML abstract feature element gml:_Feature. This permits the GML feature to be substituted anywhere that a gml:_Feature can be used, such as in a FeatureCollections and as values of GML properties (i.e. as a participant in a feature relationship). For example the content model of Road could be as follows:

```
<complexType>
   <complexContent>
      <extension base="gml:AbstractFeatureType">
         <sequence>
            <element name="centerLineOf" type="gml:CurvePropertyType"/>
               ...
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

4.2 Spatial Types

As with features, spatial types (Point, Node, etc) are represented in GML exclusively by gobal XML element declarations in the GML Application Schema. You can define new spatial or temporal types, for instance geometry types, by deriving them from base geometry types or by deriving them from gml:AbstractGeometryType. We recommend that you avoid deriving anything from the top-level gml:AbstractGeometryType, since this provides very limited information to GML-aware software. Always try to choose the most specific geometry type possible in constructing your derived geometry types. For example, if your geometry is some type of curve (in GML 3.0), then derive from gml:AbstractCurveType or, if possible, one of its concrete subclasses, such as gml:LineString.

4.3 Spatial Properties

In GML, properties of features (or other GML types) are represented by XML element declarations and not by attributes. For example, the following instance encoding would be invalid in GML:

```
<app:Ellipse gml:id=”e1” semiminor = “5” semimajor = “7” center =”50,20”/>
```

The following shows a correct instance encoding for this geometry in GML.

```
<app:Ellipse gml:id = “e1”>
   <app:semiminor>5</app:semiminor>
   <app:semimajor>7</app:semimajor>
   <app:center>50,20</app:center>
</app:Ellipse>
```

However, the best practice recommendation for encoding this geometry is to use the existing GML `centerOf` property and to encode the `Ellipse` instance as shown in the following example:

```
<app:Ellipse gml:id = “e1”>
   <app:semiminor>5</app:semiminor>
   <app:semimajor>7</app:semimajor>
   <gml:centerOf>
      <gml:Point  gml:id = “p1”>
         <gml:coordinates>50,20</gml:coordinates>
      </gml:Point>
   </gml:centerOf>
</app:Ellipse>
```

It is strongly recommended that all spatial derived types build, wherever possible, on the built-in GML spatial types.

5 Managing GML Application Schemas

5.1 Namespaces

Consider the sentence "Will will will his boat to his daughter." The word "will" is used three times in the sentence with different meanings and is causing a "name collision". A human reader can distinguish the three different meanings of the word "will" from the context of the sentence and in doing so resolves the name collision. Similarly, a name collision in XML schema is caused by the repeated used of an element name in a schema. One reason to use namespaces in XML schema is to provide "context" and avoid name collisions. The use of namespaces also promotes the re-use of elements from other domains and allows a reader to visually identify the origin of each element/attribute in an instance document.

A schema can be viewed as a collection (vocabulary) of type definitions and element declarations whose names belong to a particular namespace called a target namespace. A target namespace is declared in the application schema using the `targetNamespace` attribute of the `schema` element from XML Schema, as shown below:

```
<schema targetNamespace="http://www.gov.bc.ca/schema"
xmlns:app="http://www.gov.bc.ca/schema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
```

The value of the targetNamespace is any URI, which does not necessarily signify a physical target, meaning that there may be nothing at the end of the URI. A namespace should be interpreted as a unique string that serves as an identifier only. The attribute xmlns:app is also anyURI valued and is used to declare the namespace of the application domain and associates the prefix "app" to the namespace. The prefix "app" is an arbitrary string, usually just a few characters long that represents the application namespace "http://www.tourism.net/tourism". It is important to note that the prefix serves only as a placeholder for the application namespace within the schema document. The namespace prefixes "xsd" and "gml" are also declared as other namespaces that are referenced from the schema document using the attributes xmlns:xsd and xmlns:gml, respectively. By setting the value of the elementFormDefault attribute to "qualified", the namespace prefixes of elements are required in instance documents.

Import vs. Include

If the schema to be referenced lies outside the target namespace, then it must be imported. If the schema to be referenced resides within the target namespace, then if must be included. The following schema fragment shows how to include a supporting schema, schema2.xsd, in the target namespace, and import a GML core schema, `feature.xsd`:

```
<include schemaLocation="C:/schemas/schema2.xsd"/>
<import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.00/base/feature.xsd"/>
```

In this example, the path to `schema2.xsd`—as indicated by the `schemaLocation` attribute value in the include element—is a URI reference to a local directory. The path can also be to a remote repository, as shown in the schemaLocation value of the import element. Note that the import element specifies that the components described in `feature.xsd` are associated with the GML namespace "http://www.opengis.net/gml". To ensure XML Schema validity, this namespace identifier must match the target namespace specified in the schema that is being imported, meaning the target namespace in the `feature.xsd` must also be "http://www.opengis.net/gml". Figure 5-1 graphically illustrates the use of the include and import elements.

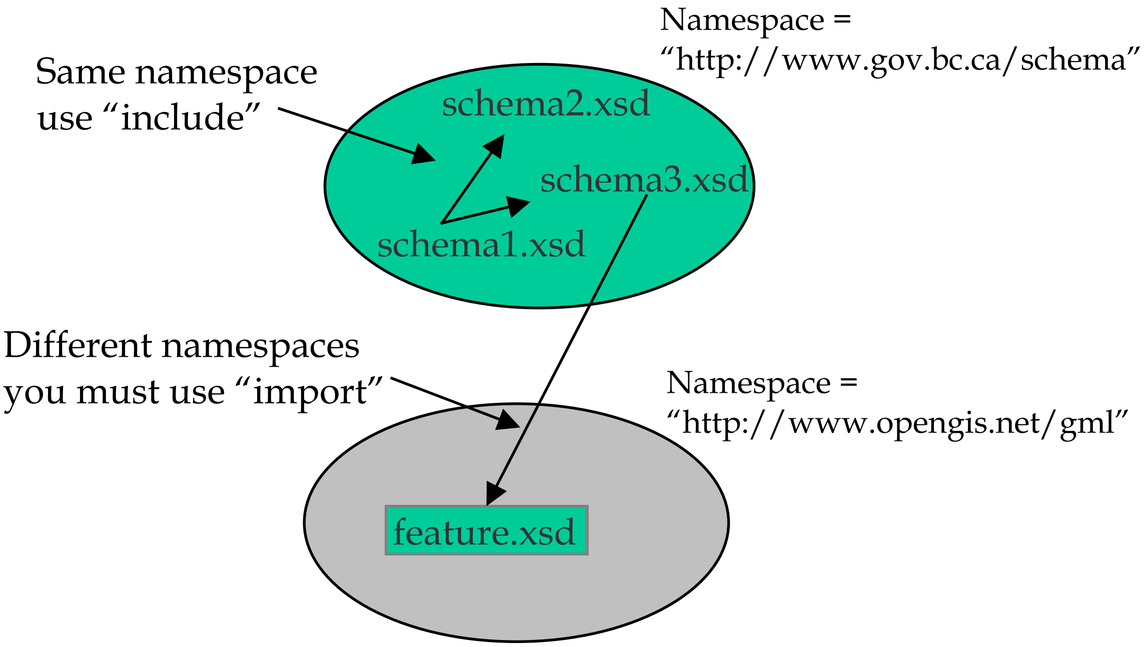


Figure 5-1: In a heterogeneous namespace design, schemas in the same namespace as schema1.xsd are included. Otherwise, they are imported.

5.2 Versioning

It is clear that XML schemas will evolve over time and it is important to capture the schema's version. The simplest way to capture the schema's version is to use the schema version attribute. For example the version attribute value is set to 3.00 as shown in the schema declaration of one of the GML 3 core schemas:

```
<schema targetNamespace="http://www.opengis.net/gml" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:sch="http://www.ascc.net/xml/schematron" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" elementFormDefault="qualified" version="3.00">
```

Although versioning the schema using the built in attribute is straightforward, the validator ignores the attribute and is therefore not an enforceable constraint.

Alternatively, schema versioning can be accomplished by changing the name/location of the schema. This mimics the convention that many people use for naming their files so that they know which version is the most current (for example, appending the version number or date to end of file name). As instance documents must specify the name/location of the schema in the schemaLocation attribute the constraint is enforceable by the validator. The value of the schemaLocation consists of pairs of the form (name space, location) each of which are of `anyURI` type. The location value can be a physical URL and consumers of the application schema can then obtain the correct version of the schema from this location. It is possible to use URNs for the schemaLocation, which would be more useful, but the current lack of tools/applications that support URN resolvers make it impractical.

Schema Versioning Best Practice

The best practice recommendation for schema versioning is to use a combination of both of the abovementioned approaches and is outlined as follows:

1. Version schemas using the built-in XMLSchema version attribute,
2. Uniquely identify the schema document using the following tags:

<annotation><appinfo source="URNValue">filename.xsd</appinfo></annotation>

This is required to link the schemaLocation URL (or URN) to the correct schema within server/client tools. A sample from one of the core GML3 schemas is as follows:

```
<annotation>
    <appinfo source="urn:opengis:specification:gml:schema:xsd:geometryBasic0d1d:v3.00">
        geometryBasic0d1d.xsd
    </appinfo>
</annotation>
```

3. Establish and publish physical URL directory/file structures (i.e. on a webserver) for hosting schema versions.
4. Inform consumers that the schemaLocation should NOT be treated as optional - it provides useful/required versioning information. Schema instances must include the correct URL to indicate the schema version that validates the instance.

5.3 Schema Registries/Repositories

GML application schemas are anticipated to be increasingly deployed on the Internet as the standard framework for sharing geographic data. As more schemas become available, it will be necessary to provide more sophisticated mechanisms for managing schemas. Schema registries can fill this need by providing:

1. Online access to the GML application schemas for the appropriate domain
2. Administrative functions for the schema administrator, such as authorization and versioning
3. Classification and other means for helping a user locate a schema

Schema registries will eventually be the source for all of the GML application schemas that are supported by different geospatial web services. Clients, providers and other services use the schemas that are stored on the GML application schema registry. These schemas provide the clients and providers with the feature types for request and response messages. Note that the schema registry and service registry are both metadata registries that can be part of a Catalogue. The Catalogue is a key component in a common service architecture that manages shared resources and facilitates the discovery of resources within an open and distributed system.

6 Best Practices for Developing GML Application Schemas

A critical design issue facing the GML authors was how to provide type extensibility. GML does not and should not provide definitions for concrete feature types such as Roads, Rivers or Bridges. GML's role is to provide the mechanism for expressing such types of geographic features. This is accomplished in GML through the use of XML Schema. Section 6.1 provides a very brief synopsis of the Best Practice guidelines and will be elaborated upon in the following Sections of this Chapter.

6.1 Summary of Best Practice Guidelines

Features	Make the feature directly or indirectly substitutable for "`gml:_Feature`" (rationale - participation within a WFS) Feature Inheritance: Given Feature (class A) derived from Feature (class B) . Ensure that 1. The XML element for class B is directly or indirectly substitutable for "`gml:_Feature`" 2. The complex type for class B derives directly or indirectly from `gml:AbstractFeatureType`. 3. The XML element for class A has the `substitutionGroup` attribute = "B" (with appropriate namespace prefix). The complex type for class A derives from the complex type for class B.
Feature Collections	Similar to Features above. Make the feature collection directly or indirectly substitutable for "`gml:_FeatureCollection`" (rationale - participation within a WFS)
Associations/ Properties	For each association generate an XML element with a name that represents the association relationship/role name, following the Association Type Pattern (or derive the complex type of this XML element from `gml:AssociationType`). GML property names should be used to name roles, the property name does not need to indicate the content of the value.
Customizing GML Properties	Properties such as `gml:featureMember`, `gml:metaDataProperty`, etc, can be customized in different ways. The best practice recommendation is to assign an appropriate custom property name (for example replace `featureMember` with `cityMember`), but leave the property type definition unchanged. The substitution group mechanism allows the intended target values of the custom property to be used with the existing property type

	definition. See Section 6.3 Feature Collections for a more detailed discussion in the context of the `featureMember` property.
Geometry	Derive from the base geometry types in GML or from `gml:AbstractGeometryType`. The best practice recommendation is to avoid deriving anything from the top-level `gml:AbstractGeometryType` whenever possible, since this provides very limited information to GML-aware software. Always try to choose the most specific geometry type possible in constructing your derived geometry types. For example, if your geometry is some type of curve (in GML 3.0), then derive from `gml:AbstractCurveType` or, if possible, one of its concrete subclasses, such as `gml:LineString`.
Topology, Temporal, Coverages, etc.	Similar to Geometry. Derive from base types or from abstract GML types. The best practice recommendation is to avoid deriving anything from the top-level abstract type since this provides very limited information to GML-aware software. Always try to choose the most specific GML type possible in constructing your derived types.
Local vs global element declarations	Declare objects globally (rationale – the type of the entity should not be modified). Note that this is mandatory for features and feature collections in GML. Declare property elements locally unless they are reused extensively (rationale – schema legibility)
Global vs. local type definitions	Object Type definition - For reasons of legibility where a schema author does not expect or intend to derive from the type, define it locally - Leave the object type unnamed. Property type definition - should be global if it is intended to be reused (rationale – schema readability) - should be named - Keep the type definition simple (rationale – it cleanly separates the property and the value) Global elements - The declaration of global elements of a complex type is necessary to allow a document instance to include such elements at the root level of a document - Elements included in complex types by reference to global elements support derivation by restriction in another namespace, allowing restriction of cardinality, and/or replacement by a member of a substitution group Local child elements - prevent derivation by restriction in another namespace if the child element is included in the derived type. Derivation by

	restriction is only possible if the local child elements are removed (left out).
Substitution groups in app schemas in general	Should reflect the data model being used. -Choice group (con) increase coupling between schemas. -Choice group (pro) can mimic multiple inheritances.
Schema Modularization	Modularize as necessary to organize elements into logically related sets. - test: should be able to create a single wrapper schema that will include all element and types from all schemas within the name space without any name collisions. This test will aid correct handling of name spaces within modularized schema structures.
Creating new namespaces	If schemas are modified on different schedules or under the authority of different custodians (to avoid name conflicts and to distinguish different vocabularies).
Metadata	Metadata may pertain to multiple levels: feature, feature collection, data set, thematic layer, etc. GML metadata properties are usually concerned with common feature-level descriptors (e.g., identifier, description) and annotations; the type definitions should always be derived from `gml:AbstractMetaDataType` and the metadata element should be made substitutable for the GML abstract metadata element `gml:_MetaData`. This permits the metadata to be substituted as the value of the built in metadata property on all GML types that derive by extension from `AbstractGMLType`. **Metadata handling within property type definitions:** - Wherever possible use `gml:MetaDataPropertyType`. It is not recommended that types be restricted to only allow specific application metadata objects (rationale: doing otherwise is cumbersome and unnecessary, as substitution is controlled by metadata object element declaration)
Derive by extension vs. restriction	Derive by extension to - add elements to contents to complex types, for recording additional data about that item. - specify standard contents and content patterns for selected elements and attributes, as needed to improve interoperability. Derive by restriction to - restrict multiplicity to eliminate flexibility not needed or to avoid confusion - use a different element name, to be more easily understood in that specific application, primarily for elements that will be instantiated many times. In some cases, restricting an existing concrete element can be done by extending the abstract element from, which it is derived, by

	adding somewhat different but corresponding extensions.
Import of schema subcomponents	This is a general XML Schema problem. When a GML application namespace imports another namespace (such as GML), it should import the "top-level" schema, which in turn includes all the necessary schema subcomponents.
Hierarchies vs. flatter designs	Hierarchal design - (pro) deriving from base types reduces the need to repeat common property elements - the attribute value `final="true"` can be used to truncate the hierarchy Flat design - (pro) easier to map to/from relational databases
Composition vs. aggregation	Use the untyped AssociationType pattern, and describe or define further semantics of Aggregation or Composition within annotation, or using formal Schematron syntax. Note Galdos is pursuing a GML change request to have AssociationTypes carry an explicit, optional attribute that would declare the association type.
Annotation	Use liberally for type definitions. Clarify in plain English any rules for the use of this type, for example, explicitly state if an association type is meant to be interpreted as aggregation. We encourage the use of formal syntax (schematron) to enforce or capture business rules.
SRS Handling	Use a URN as the value of `srsName` (rationale—location independent identifiers are preferable over URLs that constantly need to be updated).

6.2 Features

In GML there are no constraints on how specific the types should be when you develop application schemas. You could, for example, decide that there is only a single generic feature type and encode it as follows:

```
<element name = “GenericFeature”>
   <complexType>
      <complexContent>
         <extension base =”gml:AbstractFeatureType”>
         … feature properties
         </extension>
      </complexContent>
   </complexType>
</element>
```

Instances of this feature can have different property values as indicated in the following examples:

```
<app:GenericFeature gml:id = “p1”>
   <app:material>ashphalt</app:material>
   <app:age>30</app:age>
   <gml:centerLineOf> … </gml:centerLineOf>
```

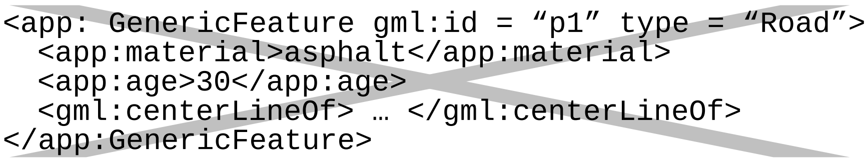
```
</app:GenericFeature>


<app:GenericFeature gml:id = “p2”>
   <app:material>water</app:material>
   <app:age>90</app:age>
   <gml:centerLineOf> … </app:centerLineOf>
</app:GenericFeature>
```

Both of these examples are perfectly valid GML, even if the semantics are not clearly defined (for example, what kinds of features are they?).

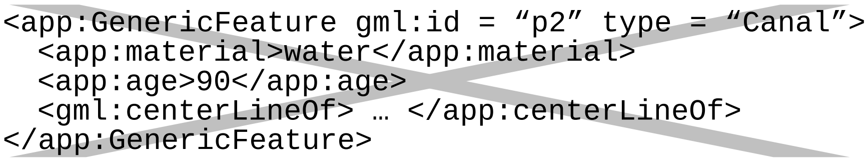
On the other hand if you wish to denote distinct semantic types such as `river` and `road`, it is INVALID GML to use a `type` attribute as shown in the following examples.

Example of Invalid GML Using a Type Attribute



```
<app: GenericFeature gml:id = “p1” type = “Road”>
  <app:material>asphalt</app:material>
  <app:age>30</app:age>
  <gml:centerLineOf> … </gml:centerLineOf>
</app:GenericFeature>
```

Example of Invalid GML Using a Type Attribute



```
<app:GenericFeature gml:id = “p2” type = “Canal”>
  <app:material>water</app:material>
  <app:age>90</app:age>
  <gml:centerLineOf> … </app:centerLineOf>
</app:GenericFeature>
```

You can “narrow” the type denoted by the feature type name element (`<app: GenericFeature>`) by using qualifying properties, as shown in the following example.

```
<app:GenericFeature gml:id = “p2”>
   <app:type>Canal</app:type>
   <app:material>water</app:material>
   <app:age>90</app:age>
   <gml:centerLineOf> ... </app:centerLineOf>
</app:GenericFeature>
```

For this encoding to be valid, however, all of the other property children of the feature type name element (`<app:GenericFeature>`) must be meaningful properties for all values of the `type` property. An XML Schema-Aware processor that performs grammar-based validation cannot be used to conduct this kind of rule-based validation (that is, it is necessary to enforce higher-level constraints among nodes).

The GML best practice recommendation is that you select a new feature type name element in each case, as shown in the following sample instances:

```
<app:Canal id = “p2”>
   <app:material>water</app:material>
   <app:age uom="urn:x-si:v1999:uom:year">90</app:age>
   <gml:centerLineOf> ... </app:centerLineOf>
</app:Canal>

<app:Road id = “p2”>
   <app:material>ashphalt</app:material>
   <app:age uom="urn:x-si:v1999:uom:year">30</app:age>
```

```
    <gml:centerLineOf> ... </app:centerLineOf>
</app:Road>
```

The following schema fragment defines the corresponding content model of `Road` according to the Best Practice Guidelines:

```
<element name="Road" substitutionGroup="gml:_Feature">
  <complexType>
    <complexContent>
      <extension base="gml:AbstractFeatureType">
        <sequence>
          <element name="material" type="string" minOccurs="0"/>
          <element name="age" type="gml:MeasureType" minOccurs="0"/>
          <element name="divided" type="boolean" default="false" minOccurs="0"/>
          <element name="travelDirection" type="app:TravelDirectionClassification" minOccurs="0"/>
          <element name="underConstruction" type="boolean" default="false" minOccurs="0"/>
          <element name="centerLineOf" type="gml:CurvePropertyType"/>
          <element name="surfaceExtent" type="gml:SurfacePropertyType" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</element>
```

Note that declaration of the element name `Road` must be global. The value of `substitutionGroup` is set to `gml:_Feature`. The type definition of `Road` (contained between the `<complexType>` tags) in this case is local and unnamed (the `name` attribute is absent on `complexType`). This prohibits any further derivation from `Road` type. The properties names of `Road` are represented by local element names. The property type definition `app:TravelDirectionClassification` is a globally defined, named, simple type as the following schema fragment shows:

```
<simpleType name="TravelDirectionClassification">
  <restriction base="string">
    <enumeration value="oneWay"/>
    <enumeration value="twoWay"/>
  </restriction>
</simpleType>
```

If it is anticipated that another feature will derive from `Road` by extension or restriction, the type definition of `Road` must be a globally defined, named complex type:

```
<element name="Road" type="app:RoadType" substitutionGroup="gml:_Feature"/>
<complexType name="RoadType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
        <element name="material" type="string" minOccurs="0"/>
        <element name="age" type="gml:MeasureType" minOccurs="0"/>
        <element name="divided" type="boolean" default="false" minOccurs="0"/>
        <element name="travelDirection" type="app:TravelDirectionClassification" minOccurs="0"/>
        <element name="underConstruction" type="boolean" default="false" minOccurs="0"/>
        <element name="centerLineOf" type="gml:CurvePropertyType"/>
        <element name="surfaceExtent" type="gml:SurfacePropertyType" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.3 Feature Collections

A feature collection is itself a feature so the best practice recommendation from Section 6.2 apply equally well to feature collections. A feature collection has one or more `gml:featureMember` properties each having `gml:_Feature` as its value, or a `gml:featureMembers` property whose value is an array of `gml:_Features.` The `gml:featureMember` property (and similarly for any complex valued property defined in GML) can be customized at different levels ranging from merely changing the property name to changing the property type definition. The best practice recommendation is to assign an appropriate custom property name (for example replace featureMember with cityMember), but leave the property type definition unchanged. The intended values of the custom property are then required to derive from `gml:AbstractFeatureType` and substitute for `gml:_Feature`. The rationale for this best practice recommenation is to keep the schema design simple. Three different levels of customization of the `gml:featureMember` property are discussed in the following examples.

Example 1

Consider a feature collection called `City` that may have feature members such as a `Street` or `Building` as shown in the following instance:

```
<City>
  <gml:boundedBy>...</gml:boundedBy>
  <cityMember>
      <Street>...</Street>
  </cityMember>
  <cityMember>
      <Building>...</Building>
  </cityMember>
</City>
```

The element declarations for `City`, `cityMember`, `Street` and `Building` following the best practice recommendation are as follows:

```
<element name="City" type="gml:FeatureCollectionType"
   substitutionGroup="gml:_FeatureCollection"/>

<element name="cityMember" type="gml:FeaturePropertyType"
   substitutionGroup="gml:featureMember"/>

<element name="Street" substitutionGroup="gml:_Feature">
   ...
</element>

<element name="Building" substitutionGroup="gml:_Feature">
   ...
</element>
```

The advantage of this schema design is that it is simple. The purpose of setting the substitution group value of `cityMember` to `gml:featureMember` is to allow GML aware software to detect that `cityMember` is a feature membership association as opposed to some other feature association. Furthermore, by setting the substitution group to `gml:featureMember`, it is not necessary when defining `City` to change the content model of `gml:FeatureCollection`. One potential drawback is that the allowed values of `cityMember` are not restricted to specific features, rather any feature in the substitution group `gml:_Feature` is allowed.

Example 2

One way to restrict the set of allowed feature values of `cityMember` using XML schema is to create a new substitution group as shown in the following schema fragment:

```
<element name="cityMember" type="app:CityMemberType"
   substitutionGroup="gml:featureMember"/>

<complexType name="CityMemberType">
   <complexContent>
      <restriction base="gml:FeaturePropertyType">
         <sequence>
            <element ref="app:_CityFeature" minOccurs="0"/>
         </sequence>
         <attributeGroup ref="gml:AssociationAttributeGroup"/>
      </restriction>
   </complexContent>
</complexType>

<element name="_CityFeature" type="gml:AbstractFeatureType"
   abstract="true" substitutionGroup="gml:_Feature"/>

<element name="Street" substitutionGroup="app:_CityFeature">
   ...
</element>

<element name="Building" substitutionGroup="app:_CityFeature">
   ...
</element>
```

The schema in this case is more complex, but it does add a feature filter to the `cityMember` property by allowing only the restricted substitution class headed by `gml:_CityFeature`. If at some later time, it is necessary to add another feature type to the list of allowable feature values of `cityMember`, one just needs to define the new feature type, possibly in a different namespace, and set the value of `substitutionGroup` to `gml:_CityFeature`. If the `City` element is declared, as in Example 1, to be of type `gml:FeatureCollectionType`, then the `gml:featureMember` property can be used in addition to `cityMember` as shown in the following instance:

```
<City>
  <gml:boundedBy>...</gml:boundedBy>
  <cityMember>
      <Street>...</Street>
  </cityMember>
  <featureMember>
      <IceBerg>...</IceBerg>
  </featureMember>
</City>
```

The following content model of City does not allow the use of the gml:featureMember property, however this adds further complexity to the schema document:

```
<element name="cityMember" type="app:CityMemberType"
   substitutionGroup="gml:featureMember"/>

 <element name="City" substitutionGroup="gml:_FeatureCollection">
   <complexType>
      <complexContent>
        <restriction base="gml:AbstractFeatureCollectionType">
           <sequence>
             <element ref="gml:metaDataProperty" minOccurs="0"
                maxOccurs="unbounded"/>
             <element ref="gml:description" minOccurs="0"/>
             <element ref="gml:name" minOccurs="0"
                maxOccurs="unbounded"/>
             <element ref="gml:boundedBy"/>
             <element ref="gml:location" minOccurs="0"/>
             <element name="cityMember" type=”app:CityMemberType”
                minOccurs="0"/>
           </sequence>
        </restriction>
      </complexContent>
   </complexType>
 </element>
```

Example 3

A finer feature filter can be achieved by explicitly listing the allowed features in a choice group in the content model of CityMemberType:

```
<element name="cityMember" type="app:CityMemberType"
   substitutionGroup="gml:featureMember"/>

<complexType name="CityMemberBaseType">
   <complexContent>
      <restriction base="gml:FeaturePropertyType"/>
   </complexContent>
</complexType>

<complexType name="CityMemberType">
   <complexContent>
      <extension base="app:CityMemberBaseType">
        <choice>
           <element ref="app:Street"/>
           <element ref="app:Building"/>
             ...
        </choice>
      </extension>
   </complexContent>
</complexType>

<element name="Street">
   ...
</element>

<element name="Building">
```

```
    ...
</element>
```

The schema design is more complex than the first case even though the substitution group mechanism is used less heavily than in the second case. The purpose of restricting out the content of `gml:FeaturePropertyType` in `CityFeatureBaseType` is to ensure that `cityMember` can substitute for `gml:featureMember` by validly deriving its type definition from `gml:FeaturePropertyType`. Again, the rationale for doing this is to allow GML aware software to recognise that `cityMember` is a feature membership association. One potential disadvantage of this schema design is its lack of extensibility. If at some later time, it is necessary to add another feature type to the list of allowable feature values of `cityMember`, the content model of `CityMemberType` would have to be modified.

6.4 Metadata

Different categories of metadata can be distinguished:

- General - describes a data resource as a whole
- Lifecycle - pertains to the history and current state of the resource
- Meta-metadata - info about the metadata instance itself
- Technical - domain-specific info pertaining to data usage and quality.
- Rights – access rights or use restrictions
- Relation - links to related resources (e.g. services that provide the data)
- Annotation
- Classification

GML metadata properties are usually concerned with common feature-level descriptors (e.g., identifier, description) and annotations; the type definitions should always be derived from `gml:AbstractMetaDataType` and the metadata element should be made substitutable for the GML abstract metadata element `gml:_MetaData`. This permits the metadata to be substituted as the value of the built in metadata property on all GML types that derive by extension from `AbstractGMLType`. For example the content model of a metadata type called `Annotation` can be encoded as follows:

```
<xsd:element name="Annotation" type="saf:AnnotationType" substitutionGroup="saf:_MetaData"/>
<!-- ============================================================ -->
<xsd:complexType name="AnnotationType" mixed="true">
    <xsd:annotation>
        <xsd:documentation/>
    </xsd:annotation>
    <xsd:complexContent mixed="true">
        <xsd:extension base="gml:AbstractMetaDataType">
            <xsd:sequence>
                <xsd:element ref="saf:textOrSymbol" minOccurs="0"/>
                <xsd:element ref="saf:spatialReferencing" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Note that the `Annotation` element must be declared globally in order to use the substitution mechanism. Note also that the `Annotation` element does not directly substitute for `gml:_MetaData`, it does so indirectly, as the abstract element `saf:_MetaData` substitutes for `gml:_MetaData`. This is shown in following element declaration for `saf:_MetaData`:

```
    <element name="_MetaData" type="gml:AbstractMetaDataType" abstract="true"
substitutionGroup="gml:_MetaData"/>
```

The element `saf:_MetaData` was created for the purpose of extensibility. Over time it may become necessary to alter the value of the metadata type, in which case a new metadata type can be created and used in place of `Annotation` simply by making the new metadata type substitutable for `saf:_MetaData`. The following content model for a new `MetaDataPropertyType` would not allow for this type of extensibility:

```
<complexType name="MetaDataPropertyType">
   <complexContent>
      <restriction base="gml:MetaDataPropertyType">
         <sequence>
            <element ref="saf:Annotation " minOccurs="0"/>
         </sequence>
         <attributeGroup ref="gml:AssociationAttributeGroup"/>
      </restriction>
   </complexContent>
</complexType>
```

The `Annotation` element would never appear as the root element of an instance document, rather it would be the value of a property. GML 3 provides a simple mechanism (`gml:MetaDataPropertyType`) for including feature or feature collection metadata either by reference (using XLink attributes) or by value (in-line). Higher-level metadata are typically stored elsewhere and accessed through catalogue services; GML metadata properties may reference such catalogue entries (using the XLink attributes contained in `gml:AssociationAttributeGroup`). A property whose value is `saf:Annotation` might be called `annotationComponent` and have the following element declaration:

```
  <element name="annotationComponent" type="gml:MetaDataPropertyType"
substitutionGroup="gml:metaDataProperty"/>
```

A sample instance of `Annotation` metadata is shown in Listing 8-1.

6.5 Modelling Features

In some cases, GML provides more than one way to encode something, and this may be a source of initial confusion, especially when you need to model complex features. A complex feature, such as an airport, may itself be composed of multiple features, such as the terminal buildings, taxiways, runways, control towers, and aprons. This can be readily modelled in GML as a complex feature with the above component features. However, you can also model the airport as a simple feature (that is, not composed of other features) but with complex geometry (for example, a Multi-Geometry) with geometric components for each of the aprons, buildings, runways, control towers and taxiways. Which should you do? The answer depends on what we want to achieve.

If you are concerned only with the airport itself and with none of its component parts, you may elect to adopt the complex geometry approach and model the airport as a multi-polygon or multi-geometry, as shown in the following example:

```
<app:Airport gml:id = “a1”>
   <gml:multiExtentOf>
      <gml:MultiPolygon srsName=“urn:epsg:v6.1:coordinateReferenceSystem:4267”>
         <gml:polygonMember>
            <gml:Polygon> ... </gml:Polygon>
         </gml:polygonMember>
         <gml:polygonMember>
            <gml:Polygon> ... </gml:Polygon>
         </gml:polygonMember>
         <gml:polygonMember>
            <gml:Polygon> ... </gml:Polygon>
         </gml:polygonMember>
      </gml:MultiPolygon>
   <gml:multiExtentOf>
<app:passengerCapacity>100000</passengerCapacity>
</app:Airport>
```

On the other hand, if you want to disaggregate the airport and focus on a particular level of description of the component parts (for example, runway length, operating hours), then you should create these as independent features and construct the airport as a feature collection, as shown below:

```
<app:Airport gml:id = “a1”>
   <gml:boundedBy>
      <gml:Envelope srsName=“urn:epsg:v6.1:coordinateReferenceSystem:4267”>
         <gml:pos>0 0</gml:pos>
         <gml:pos>1000 1000</gml:pos>
      </gml:Envelope>
   </gml:boundedBy>
   <gml:featureMember>
      <app:Terminal gml:id = “p1”>
         <gml:extentOf>
            <gml:Polygon> ... </gml:Polygon>
         </gml:extentOf>
         <app:area uom="urn:x-si:v1999:uom:m^2">45000</app:area>
         <app:numberOfGates>20</app:numberOfGates>
      </app:Terminal>
   </gml: featureMember >
   <gml:featureMember>
      <app:RunWay gml:id = “p2”>
         <gml:extentOf>
            <gml:Polygon> ... </gml:Polygon>
         </gml:extentOf>
         <app:length uom="urn:x-si:v1999:uom:metre">5000</app:length>
         <app:direction>NE</app:direction >
      </app:RunWay>
   </gml: featureMember >
   <gml:featureMember>
      <app:ControlTower gml:id = “p3”>
         <gml:extentOf>
            <gml:Polygon> ... </gml:Polygon>
         </gml:extentOf>
         <app:height uom="urn:x-si:v1999:uom:metre">30</app:height >
      </app:ControlTower>
   </gml:featureMember >
   <app:passengerCapacity>2000000</passengerCapacity>
</app:Airport>
```

6.6 Features with Spatial Valued Properties

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 simple 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.

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 6-1, but this is not necessarily a requirement.

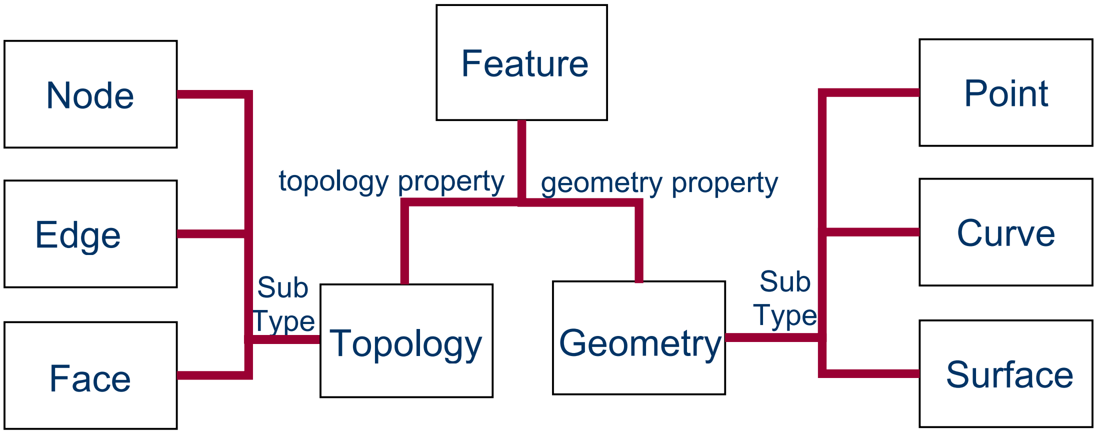


Figure 6-1: A feature with two-dimensional spatial extent, having both geometry and topology valued properties.

As GML Topology is separate from GML Geometry, features may have a stand-alone topology model. It is also possible for a feature to have a geometry model that is embedded in its topology model as shown in Figure 6-2.

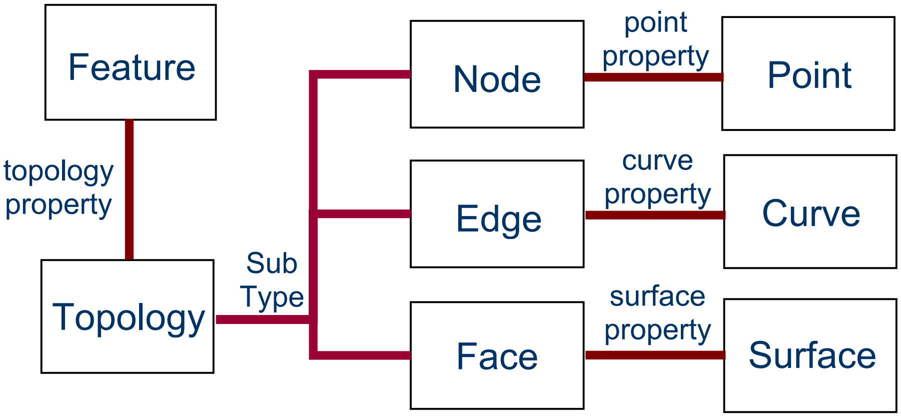


Figure 6-2: A feature with two-dimensional spatial extent, having topology valued properties with embedded geometry valued properties.

A feature with two-dimensional spatial extent that has both a topology and geometry model can be organized as shown in either Figure 6-1 or Figure 6-2. The best organizational strategy depends on what you want to achieve. If you plan to produce a map from the geometric data captured in the spatial model then the organization of Figure 6-1 would be preferable as it would be easier for an application to extract the geometric data. If you want to perform a query that may involve both topology and geometry, such as the optimal route query stated in the introduction of this section, then the organization of Figure 6-2 might be preferable, since it would be easier for an application to analyze the topological data and relate it to the geometric data. Furthermore, the organization of Figure 6-2 requires a single property to express the spatial extent in any given dimension, rather than two separate properties for topology and geometry.

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 inline as 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`.

Relating a Feature Collection’s Topology to the Topology of its Members

Suppose a city is modelled as a feature collection and includes the street segments and intersections as shown in Figure 6-3.

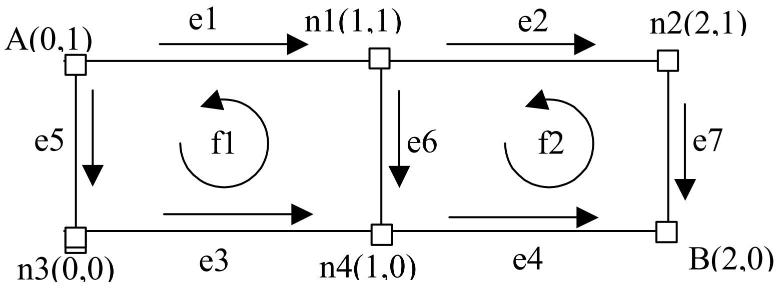


Figure 6-3: City Feature Collection consisting of street segments and intersections.

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 6-4 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>
```

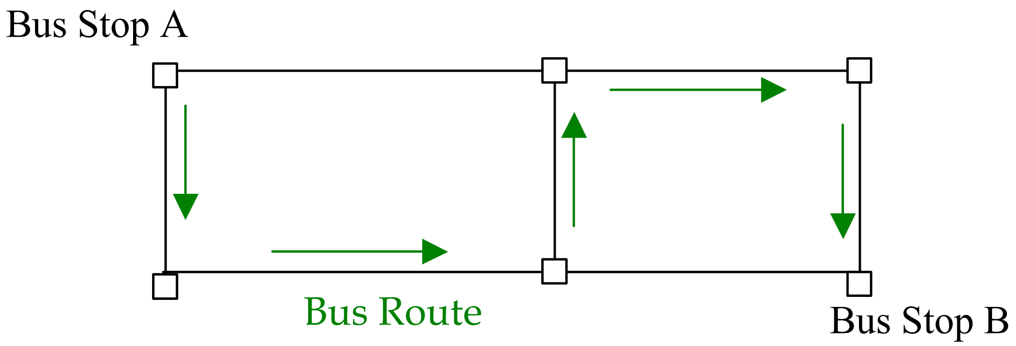


Figure 6-4: Bus Route Feature Member of the City Feature Collection

Distinguishing Topology Networks

To motivate this discussion, first consider the two different road networks in Figure 6-5.

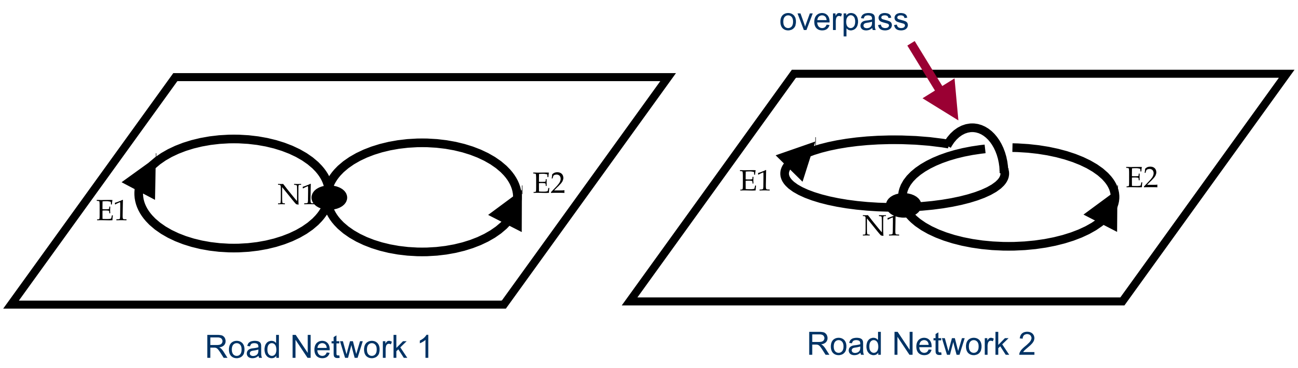


Figure 6-5: Two different road networks with the same number of Edges and Nodes. The graph that represents Road Network 1 is embedded in a plane, unlike Road Network 2, which has a bridge overpass that does not lie in the plane.

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

```
<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 6-5 is the cyclic order of the incident edges around the central node `N1`. This difference is reflected in the coboundary 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>
```

Although the coboundary information of `N1` is sufficient to distinguish between the two different topology networks in Figure 6-5, this is not the case for Topology Networks 1 and 2 in Figure 6-6 and Figure 6-7, respectively. Each of the planar topology networks shown in Figure 6-6 and Figure 6-7 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.

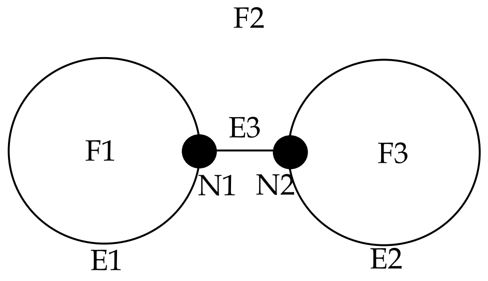


Figure 6-6: Topology Network 1

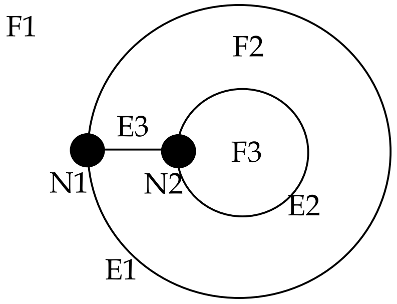


Figure 6-7: Topology Network 2

The GML 3.0 `TopoComplex` encoding of the topology models in Figure 6-6 and Figure 6-7 are indistinguishable even if the coboundary 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 [Kui95] A Lossless Representation of Topological Spatial Data, *Advances in Spatial Data* SSD 95, LNCS 951, p.1-13, 1995, by B. Kuijpers et al. If two non-equivalent topology networks are represented the same way then some information is lost in the representation. The representation introduced [Kui95] 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 6-8). The topological representation introduced in [Kui95] 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 at a node `n0` is a sequence of incident edges and adjacent faces arranged in counter-clockwise cyclic order around `n0` (following the convention used for coboundary edges). For example, the node observation at the node `N1` in topology network in either Figure 6-6 or Figure 6-7 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 GML3 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:

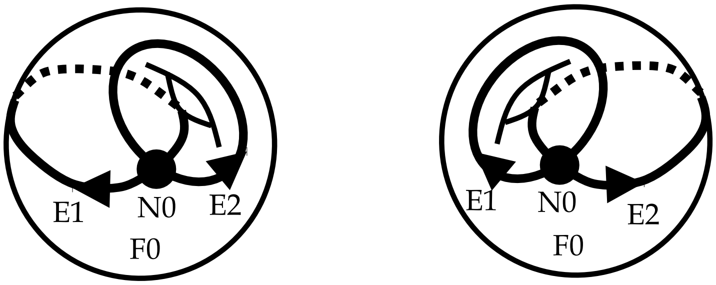
```
<Network2 gml:id=”N2”>
     ...
  <universalFace xlink:href="#F1"/>
  <nodeObservations>
     <NodeObservation>
       <node xlink:href="#N1"/>
       <edge xlink:href="#E1"/>
       <face xlink:href="#F1"/>
       <edge xlink:href="#E1"/>
       <face xlink:href="#F2"/>
       <edge xlink:href="#E3"/>
       <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 could be 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 above). The two topology networks shown in Figure 6-8 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 6-8: PLA Structures do not generalize to a lossless representation on more general surfaces. There is no universal face and only one node observation in each topology model shown.

One method that can be used to distinguish between the topology models in Figure 6-8 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 6-9.

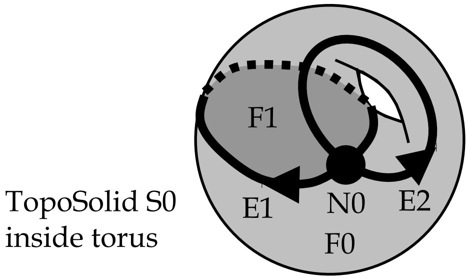


Figure 6-9: A TopoSolid S0 and a Face F1 is added to Topology Network 1 to form a 3-dimensional TopoComplex TM1

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 coboundary of `F1`. Even more can be gleaned from this encoding. The double occurrence of `TopoSolid S0` in the coboundary 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).

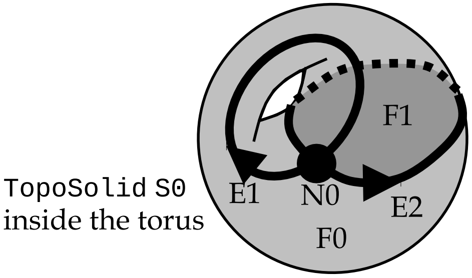


Figure 6-10: A TopoSolid S0 and a Face F1 is added to Topology Network 2 to form a 3-dimensional TopoComplex TM2

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 coboundary of `F1` indicates that `E1` is contractible to a point in the 3-dimensional topology and hence is the “tunnelling” edge in this case.

6.7 SRS Handling

We recommend using a URI-based persistent identifier scheme in order to unambiguously identify a spatial reference system in a location-independent manner; the value of the `srsName` attribute is then a Uniform Resource Identifier that does not need to be constantly updated to reference specific copies. Some existing schemes are listed below:

- Uniform Resource Name (URN) <http://www.ietf.org/rfc/rfc2141.txt>
- Persistent URL (PURL) <http://purl.oclc.org/docs/inet96.html>
- Digital Object Identifer (DOI) < http://www.ietf.org/internet-drafts/draft-paskin-doi-uri-03.txt>
- Archival Resource Key (ARK) < http://www.ietf.org/internet-drafts/draft-kunze-ark-05.txt>

All of the persistent identifier schemes listed above have multiple parts, where the first part identifies the URI scheme; the following parts typically include a namespace identifier or registrant code, followed by a resource name structured according to the rules laid out by the namespace authority. Some of the schemes require that namespaces be formally registered with a registration authority (e.g. URN namespaces must be registered with the Internet naming authority, IANA).

Suppose one wanted to employ URNs and had registered a URN namespace according to the procedures described in RFC 3406. If the namespace has not been formally registered, the namespace identifier (NID) must be prefixed with “x-“ (e.g., x-opengis). The following examples of resource identifiers provide hints concerning the kind of resource that is being referenced, but these are hints only to someone familiar with the rules for constructing an identifier, as stipulated by the namespace authority. Generally, resource identifiers are more or less opaque.

1. urn:NID:CoordinateReferenceSystem:1234
2. urn:NID:PrimeMeridian:6789
3. urn:NID:Operation:10101

The first part, `urn:`, specifies the URI scheme and `NID` represents the namespace identifier. The namespace-specific string (NSS) following the `NID` includes the resource type or category and a numeric code.

As noted in Section 3.4, if the `srsName` attribute is not provided on a geometry element (such as on the `Point` geometries above) then the CRS must be specified on the aggregate geometry that this geometry is part of. According to this rule, the CRS can be specified in the `srsName` value on the `MultiPoint` element and not on its `Point` members, since it is assumed that they have the same CRS as the `MultiPoint` aggregate. In the case that both the aggregate geometry and one of its member geometries specify different values of `srsName`, the value of `srsName` on the member geometry overrides that of the aggregate.

```
<MultiPoint gml:id=”MP1” srsName=“urn:x-epsg:CoordinateReferenceSystem:4269”>
   <pointMembers>
      <Point gml:id=”P1”>
         <coordinates> 23378428,25959999</coordinates>
      </Point>
      <Point gml:id=”P2” srsName=“urn:x-epsg:CoordinateReferenceSystem:4267”>
         <coordinates>43411784,25959743</coordinates>
      </Point>
   </pointMembers>
</MultiPoint>
```

However, one potential problem that can arise is that if the `Point P1` is referenced from elsewhere by an `xlink:href`, there is no CRS information to accompany the coordinates in `P1`. For example consider another `MultiPoint` instance that uses a different CRS:

```
<MultiPoint gml:id=”MP2” srsName=“urn:x-epsg:CoordinateReferenceSystem:4268”>
   <pointMember xlink:href=”#P1”/>
   <pointMember xlink:href=”#P2”/>
   <pointMember>
      <Point gml:id=”P3” srsName=“urn:x-epsg:CoordinateReferenceSystem:4267”>
         <coordinates>1,2</coordinates>
      </Point>
      <Point gml:id=”P4” srsName=“urn:x-epsg:CoordinateReferenceSystem:4270”>
         <coordinates>3,4</coordinates>
      </Point>
   </pointMember>
</MultiPoint>
```

Without any additional information, the CRS associated to the `Points P1` and `P2` will be misinterpreted. The best practice recommendation to avoid this scenario is to enter a valid CRS value for the `srsName` attribute on every instance of a geometry element.

6.8 Designing GML Application Schemas to work with a WFS

The OGC Web Feature Service (WFS)—one of the keys to GML deployment—became an adopted specification in January 2003. This OGC interface provides standardized access to a feature store and enables users to add, update or retrieve GML data, locally or across the Internet. When a client sends a request for information to the OGC WFS, the service sends a response message that includes geographic feature data in GML.

Data requests may be filtered using the OGC Common Query Language (CQL). Two possible examples of CQL expressions are: "Find all towns inside this state" and "Find all roads with three lanes that intersect the provincial border." GML is used to express parts of the request—such as GML feature type names, GML geometries and GML properties—in combination with the OGC CQL filter grammar. The response from the WFS is encoded in GML.

A client can determine the types of features supported by a given WFS through the capabilities interface. This can enable a WFS client to determine, for example, that a given WFS can serve geographic information on roads, power lines, substations, transforms, dams and power stations. The client can also request a detailed description of any group of these feature types.

General Guidelines

There are a few design strategies for GML application schemas that can be used to make them work better with a WFS:

1. Make each `Feature` visible to a WFS by deriving from `AbstractFeatureType` and setting the substitution group to `_Feature`. A WFS can manipulate only those features that are declared substitutable directly or indirectly with `gml:_Feature`.
2. Make each `FeatureCollection` visible to a WFS by deriving from `AbstractFeatureCollectionType` and setting the substitution group (directly or indirectly) to `gml:_FeatureCollection`.
3. Avoid putting a feature inside another feature that is not a feature collection. The WFS specification currently does not define how to handle this situation.
4. Avoid deriving from GML geometries, because some WFSs will not be able to recognize them as geometries.
5. Avoid using the deprecated `fid` (feature id) and `gid` (geometry id) attributes with GML-3. If the underlying feature schema is based on GML-3, fid and gid attributes might be ignored by a WFS.
6. To improve the efficient use of the network bandwidth, it is a good idea to define as many properties as possible to be optional (unless prohibited by your model). The mandatory properties are returned even when they are not explicitly requested. In contrast, optional properties can be weeded out by not specifying them in a request. If a feature type has many properties, consider making some/all of them optional.

7. Avoid using nested choices/sequences (e.g. choice within a choice or sequence within a choice) as this may confuse a WFS.

7 Mapping UML and SAIF to GML

The GeographicObject class represents the central concept in SAIF just as the Feature is the central concept in GML. The UML Diagrams of the GeographicObject model and the Feature model are shown in Figure 7-1 and Figure 7-2, respectively.

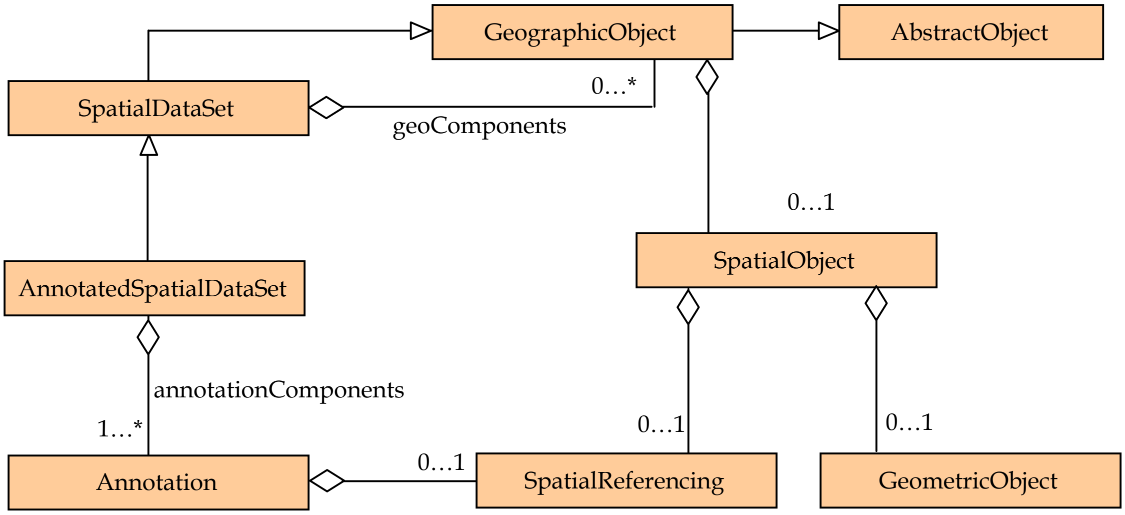


Figure 7-1: UML Diagram of the High Level GeographicObject Class Model in SAIF

The SpatialDataSet in SAIF is the analogue of the FeatureCollection in GML and the AnnotatedSpatialDataSet corresponds to a FeatureCollection with metadata. The SpatialObject in SAIF is essentially a GML geometry type with a Coordinate Reference System.

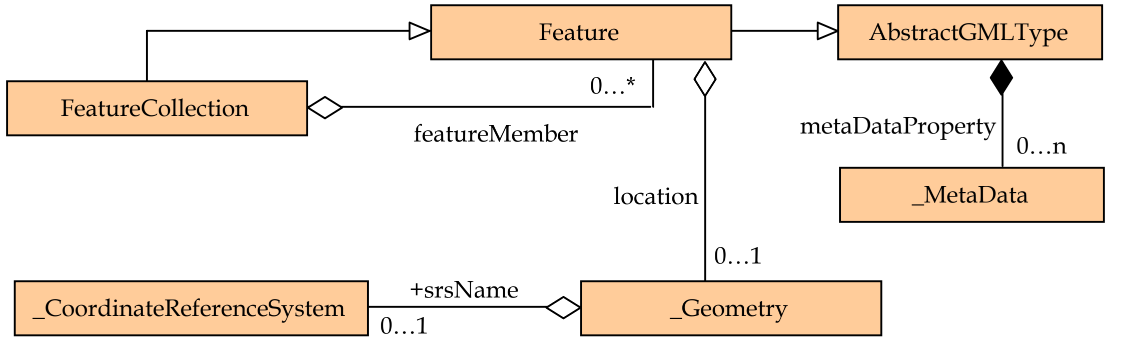


Figure 7-2: UML Diagram of the High Level Feature Model in GML

The material in this chapter can be used to develop a mapping from UML to GML and using the UML model for the GeographicObject, we will map SAIF to GML.

Note that properties in GML are UML associations. GML does not prescribe whether or not an association is an aggregation or a composition. The usual interpretation of XML element inclusion is that it implies composition but this is not always the case. Consider the following example:

```
<app:Building gml:id=”C1”>
   <app:height uom="urn:x-si:v1999:uom:metre">87</app:height>
   <app:frontsOn xlink:href=“#Georgia”/>
   <app:photoOf>
     <my:Photo gml:id =”p1”> ... </my:Photo>
 </app:photoOf>
</app:Building>
```

Note that the `photoOf` relationship is not a composition, since the photo and the building exist completely independent of one another.

Note conversely that an association expressed via an `xlink:href` could also be a composition and it would be up to an application to see that the composition semantics are enforced.

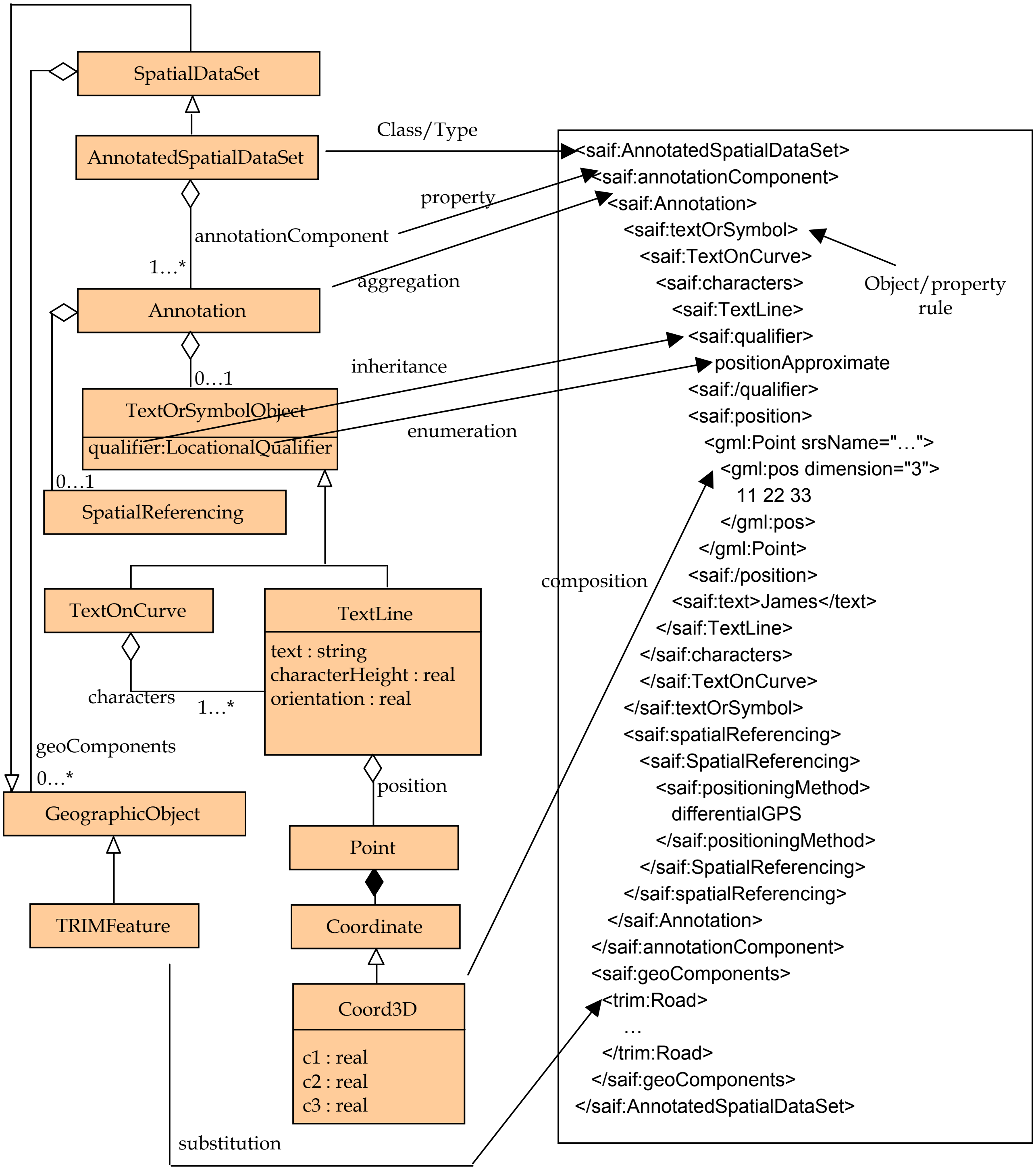
GML represents UML associations as GML property elements either with or without enclosed content. In the case of no enclosed content, the GML property element must have the xlink:href attribute to point to the value of the property.

We should note that GML is not a general encoding language for UML models and that this is the intent of XMI. GML is intended to encode the description of geographic objects. Since it is often useful to work in UML, some general rules for the mapping from UML to GML are clearly helpful.

The basic concepts for mapping UML (and SAIF) to GML can be summarized graphically in Figure 7-3.

Note in particular the following:

1. UML Classes are mapped to XML Elements.
2. UML Attributes are mapped to XML Elements or attributes.
3. UML Associations roles are mapped to GML properties (XML Elements) for both general Associations and Compositions.
4. UML inheritance corresponds to derivation by extension or restriction. In the case of restriction, the stereotype <<restriction>> would be present next to the inheritance arrow (generalization relationship) in the UML diagram.



```
<saif:AnnotatedSpatialDataSet>
 <saif:annotationComponent>
  <saif:Annotation>
   <saif:textOrSymbol>
    <saif:TextOnCurve>
     <saif:characters>
      <saif:TextLine>
       <saif:qualifier>
        positionApproximate
       <saif:/qualifier>
       <saif:position>
        <gml:Point srsName="…">
         <gml:pos dimension="3">
          11 22 33
         </gml:pos>
        </gml:Point>
       <saif:/position>
      <saif:text>James</text>
     </saif:TextLine>
    </saif:characters>
   </saif:TextOnCurve>
  </saif:textOrSymbol>
  <saif:spatialReferencing>
   <saif:SpatialReferencing>
    <saif:positioningMethod>
     differentialGPS
    </saif:positioningMethod>
   </saif:SpatialReferencing>
  </saif:spatialReferencing>
 </saif:Annotation>
 </saif:annotationComponent>
 <saif:geoComponents>
  <trim:Road>
   …
  </trim:Road>
 </saif:geoComponents>
</saif:AnnotatedSpatialDataSet>
```

Figure 7-3: Graphical Overview of the Mapping from UML and SAIF to GML

General Rules

The following general rules for the generation of GML from UML models are summarized in the following table:

Class	Generate an XML element with the same name as the class name, and a complex type definition with the same or related name to the element name.(e.g. "Annotation" becomes "AnnotationType".
Feature Class	For each Feature Class (e.g. Road) sub-typed from Feature, generate a global XML element (feature element) with the same name as the feature class name. Set the substitutionGroup attribute value for this XML element to "gml:_Feature". Generate a complex type with the same name or related name to the element name (e.g. RoadType). This complex type must have complex content and derive from gml:AbstractFeatureType.
Class Attribute	For each class attribute generate an XML element that is a child of the feature element for the associated class. The name of the XML element is the same as the name of the feature attribute in the UML model. For each UML class attribute having a primitive type use the corresponding XSD simple type for the content model of the generated XML element.
Class Association	For each association role generate an XML element with a name equal to the UML role name. Derive the complex type of this XML element from gml:AbstractAssociationType or follow the AssociationType Pattern.
Inheritance	Given Feature (class A) derived from Feature (class B) ensure that 4. The XML element for class B has the substitutionGroup attribute = "gml:_Feature" 5. The complex type for class B derives from gml:AbstractFeatureType . 6. The XML element for class A has the substitutionGroup attribute = "B" (with appropriate namespace prefix. 7. The complex type for class A derives from the complex type for class B.
Feature Collection Class	A Feature Collection Class is a Feature Class. Follow the same rules as for Feature Classes except that the complex type for the feature collection class must derive from gml:AbstractFeatureCollectionType.
Geometry Class	A Geometry Class is a Class derived from the class

	AbstractGeometry. For each geometry class (e.g. `Polygon`) sub-typed from AbstractGeometry, generate a global XML element (geometry element) with the same name as the geometry class name. Set the `substitutionGroup` attribute value for this XML element to "`gml:_Geometry`". Generate a complex type with the same name or related name to the element name. This complex type must have complexContent and derive from `gml:AbstractGeometryType`. (e.g. `MyPolygonType`)
Namespaces	Only GML core schemas can use the GML namespace (http://www.opengis.net/gml). All other schemas (GML Application Schemas) must declare a different target namespace.
Multiplicity	Multiplicity is controlled using the minOccurs and maxOccurs attribute.

Note that the declaration of global elements of a type is necessary in order to allow a document instance to include such elements at the root level of a document. Being able to do this is a requirement for GML if we are to use it to return query results from an OGC WFS (Web Feature Service).

Sample Mapping

The mapping from UML to GML introduced in the table above can be used to develop a mapping from SAIF to GML. We focus here on the `AnnotatedSpatialDataSet` model in SAIF and use this to demonstrate the mapping. Figure 7-3 shows the `AnnotatedSpatialDataSet` UML model on the left, the corresponding GML Instance on the right. The individual SAIF components are mapped to the corresponding GML Types in the following schema fragments, showing the equivalent GML of the UML model in Figure 7-3.

Step 1: For each association role generate an XML element with a name equal to the UML role name and a complex type definition with a related name (unless there is a more suitable name such as "`MetaDataPropertyType`") to the element name following the Association Type Pattern:

```
  <!-- ============================================================ -->
  <element name="annotationComponent" type="saf:MetaDataPropertyType"
substitutionGroup="gml:metaDataProperty"/>
  <complexType name="MetaDataPropertyType">
      <complexContent>
          <restriction base="gml:MetaDataPropertyType">
              <sequence>
                  <element ref="saf:_MetaData" minOccurs="0"/>
              </sequence>
              <attributeGroup ref="gml:AssociationAttributeGroup"/>
          </restriction>
      </complexContent>
  </complexType>
```

```
<!-- ============================================================= -->
<element name="spatialReferencing" type="saf:SpatialReferencingPropertyType"/>
<complexType name="SpatialReferencingPropertyType">
    <sequence>
        <element ref="saf:SpatialReferencing" minOccurs="0"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
<!-- ================================================================ -->
<element name="textOrSymbol" type="saf:TextOrSymbolObjectPropertyType"/>
<complexType name="TextOrSymbolObjectPropertyType">
    <sequence>
        <element ref="saf:TextOrSymbolObject" minOccurs="0"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
<!-- ================================================================ -->
<element name="textOnCurve" type="saf:TextOnCurvePropertyType"/>
<complexType name="TextOnCurvePropertyType">
    <sequence>
        <element ref="saf:TextOnCurve" minOccurs="0"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
<!-- ================================================================ -->
<element name="characters" type="saf:TextLineListPropertyType"/>
<complexType name="TextLineListPropertyType">
    <annotation>
        <documentation>Order is important in this list type</documentation>
    </annotation>
    <sequence>
        <element ref="saf:TextLine" maxOccurs="unbounded"/>
    </sequence>
</complexType>
<!-- ================================================================ -->
<element name="textLine" type="saf:TextLinePropertyType"/>
<complexType name="TextLinePropertyType">
    <sequence>
        <element ref="saf:TextLine" minOccurs="0"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
<!-- ================================================================ -->
  <element name="position" type="gml:PointPropertyType"/>
```

Step 2: For each feature (feature collection) class, generate an XML element with same name as the class name, and a complex type definition with a related name to the element name. Set the `substitutionGroup` attribute value for this element to “`gml:_Feature`” (”`gml:_FeatureCollection`”). The complex type must have a `complexContent` element and derive from `gml:AbstractFeatureType` (`gml:AbstractFeatureCollectionType`). Note that some feature attributes were suppressed in the UML model and appear below:

```
  <element name="GeographicObject" type="saf:AbstractGeographicObjectType" abstract="true"
substitutionGroup="gml:_Feature">
   <annotation>
       <documentation>gml:Feature is used to model the GeographicObject in SAIF</documentation>
   </annotation>
  </element>
  <complexType name="AbstractGeographicObjectType">
   <annotation>
       <documentation>base type for AbstractTRIMFeatureType</documentation>
   </annotation>
   <complexContent>
       <extension base="gml:AbstractFeatureType">
           <sequence/>
       </extension>
   </complexContent>
  </complexType>
  <!-- ================================================================ -->
  <element name="SpatialDataSet" type="saf:SpatialDataSetType" substitutionGroup="gml:_FeatureCollection"/>
  <complexType name="SpatialDataSetType">
   <complexContent>
       <restriction base="gml:AbstractFeatureCollectionType">
           <sequence>
               <element ref="gml:metaDataProperty" minOccurs="0" maxOccurs="unbounded"/>
               <element ref="gml:boundedBy" minOccurs="0"/>
               <element ref="saf:geoComponent" minOccurs="0" maxOccurs="unbounded"/>
               <element ref="saf:geoComponents" minOccurs="0"/>
           </sequence>
       </restriction>
   </complexContent>
  </complexType>
```

Step 3: For all other UML classes, generate XML elements with same name as the class names, and complex type definitions with related names to the element names. Follow inheritance rules if applicable. For each class attribute generate an XML element that is a child of the associated class with the same name as the name of the attribute in the UML model. For each UML class attribute having a primitive type use the corresponding XSD simple type for the content model of the generated XML element. (Note some class attributes were suppressed in the UML model.)

```
  <!-- ============================================================== -->
  <element name="AnnotatedSpatialDataSet" type="saf:AnnotatedSpatialDataSetType"
substitutionGroup="saf:SpatialDataSet"/>
  <complexType name="AnnotatedSpatialDataSetType">
      <annotation>
          <documentation/>
      </annotation>
      <complexContent>
          <restriction base="saf:SpatialDataSetType">
              <sequence>
                  <element ref="saf:annotationComponent" maxOccurs="unbounded"/>
                  <element ref="gml:boundedBy" minOccurs="0"/>
                  <element ref="saf:geoComponent" minOccurs="0" maxOccurs="unbounded"/>
                  <element ref="saf:geoComponents" minOccurs="0"/>
              </sequence>
          </restriction>
      </complexContent>
  </complexType>
  <!-- ============================================================ -->
  <xsd:element name="Annotation" type="saif:AnnotationType" substitutionGroup="saif:_MetaData"/>
  <xsd:complexType name="AnnotationType" mixed="true">
      <xsd:complexContent mixed="true">
          <xsd:extension base="gml:AbstractMetaDataType">
              <xsd:sequence>
                  <xsd:element name="textOrSymbol" type="saif:TextOrSymbolObjectPropertyType"
minOccurs="0"/>
                  <xsd:element name="spatialReferencing" type="saif:SpatialReferencingPropertyType"
minOccurs="0"/>
              </xsd:sequence>
          </xsd:extension>
      </xsd:complexContent>
  </xsd:complexType>
  <!-- ============================================================ -->
```

```
<element name="SpatialReferencing" substitutionGroup="saf:_MetaData"/>
<complexType mixed="true">
    <complexContent mixed="true">
        <extension base="gml:AbstractMetaDataType">
            <sequence>
                <element ref="saf:hydrographic" minOccurs="0"/>
                <element ref="saf:control" minOccurs="0"/>
                <element ref="saf:transform" minOccurs="0"/>
                <element ref="saf:positioningMethod" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<!-- ============================================================ -->
<element name="TextOrSymbolObject" type="saf:TextOrSymbolObjectType"/>
<complexType name="TextOrSymbolObjectType">
    <annotation>
        <documentation/>
    </annotation>
    <complexContent>
        <extension base="gml:AbstractGMLType">
            <sequence>
                <element ref="saf:qualifier" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<!-- ============================================================ -->
<xsd:element name="TextOnCurve" substitutionGroup="saif:TextOrSymbolObject">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="saif:TextOrSymbolObjectType">
                <xsd:sequence>
                    <xsd:element name="characters" type="saif:TextLineArrayPropertyType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
<!-- ============================================================ -->
<element name="TextLine" type="saf:TextLineType" substitutionGroup="saf:TextOrSymbolObject"/>
<complexType name="TextLineType">
    <annotation>
        <documentation>orientation type numeric rather than double</documentation>
    </annotation>
    <complexContent>
        <extension base="saf:TextOrSymbolObjectType">
            <sequence>
                <element name="position" type="gml:PointPropertyType"/>
                <element name="text" type="string"/>
                <element name="characterHeight" type="double" minOccurs="0"/>
                <element name="stringWidth" type="double" minOccurs="0"/>
                <element name="orientation" type="double" minOccurs="0"/>
                <element ref="saf:alignment" minOccurs="0"/>
                <element name="fontName" type="string" minOccurs="0"/>
                <element name="string" type="string" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Step 4. Define supporting GML types as necessary:

```
 <!-- ============================================================ -->
 <element name="_MetaData" type="gml:AbstractMetaDataType" abstract="true"
substitutionGroup="gml:_MetaData">
```

```
    <annotation>
        <documentation>Abstract element, which acts as the head of a substitution group for packages of SAIF
feature MetaData properties.  </documentation>
    </annotation>
</element>
<!-- ============================================================== -->
<element name="qualifier" type="saf:LocationalQualifierType"/>
<simpleType name="LocationalQualifierType">
    <restriction base="string">
        <enumeration value="definite"/>
        <enumeration value="indefinite"/>
        <enumeration value="positionApproximate"/>
        <enumeration value="virtual"/>
    </restriction>
</simpleType>
<!-- =========================================================== -->
<element name="geoComponent" type="gml:FeaturePropertyType" substitutionGroup="gml:featureMember">
<element name="geoComponents" type="gml:FeatureArrayPropertyType"
substitutionGroup="gml:featureMembers">
<!-- ============================================================== -->
<element name="hydrographic" type="saf:HydrographicReferencePropertyType"/>
<complexType name="HydrographicReferencePropertyType">
    <sequence>
        <element ref="saf:HydrographiclReference" minOccurs="0"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
<element name="HydrographiclReference" type="saf:HydrographiclReferenceType"/>
<complexType name="HydrographiclReferenceType">
    <annotation>
        <documentation/>
    </annotation>
    <complexContent>
        <extension base="gml:AbstractGMLType">
            <sequence>
                <element ref="saf:tidal" minOccurs="0"/>
                <element ref="saf:sounding" minOccurs="0"/>
                <element name="remarks" type="string" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<!-- ============================================================== -->
<element name="control" type="saf:ControlPropertyType"/>
<!-- ============================================================== -->
<complexType name="ControlPropertyType">
    <sequence>
        <element ref="saf:Control" minOccurs="0"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
<!-- ============================================================== -->
<xsd:element name="Control">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="horizontalControl" type="saif:DataControlPropertyType" minOccurs="0"/>
            <xsd:element name="verticalControl" type="saif:DataControlPropertyType" minOccurs="0"/>
            <xsd:element name="mapDerived" type="saif:TilePropertyType" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<!-- ============================================================== -->
<element name="transform" type="saf:TransformPropertyType"/>
<!-- ============================================================== -->
<complexType name="TransformPropertyType">
    <sequence>
```

```
          <element ref="saf:Transform" minOccurs="0"/>
      </sequence>
      <attributeGroup ref="gml:AssociationAttributeGroup"/>
  </complexType>
  <!-- ================================================================= -->
  <element name="Transform" type="saf:TransformType"/>
  <!-- ================================================================= -->
  <complexType name="TransformType">
      <annotation>
          <documentation/>
      </annotation>
      <complexContent>
          <extension base="gml:AbstractGMLType">
              <sequence>
                  <element ref="saf:transformMatrix" minOccurs="0"/>
                  <element name="spatialOffset" type="gml:DirectPositionType" minOccurs="0"/>
                  <element name="remarks" type="string" minOccurs="0"/>
              </sequence>
          </extension>
      </complexContent>
  </complexType>
  <!-- ================================================================= -->
  <element name="transformMatrix" type="saf:TransformMatrixPropertyType"/>
  <!-- ================================================================= -->
  <complexType name="TransformMatrixPropertyType">
      <sequence>
          <element ref="saf:Matrix" minOccurs="0"/>
      </sequence>
      <attributeGroup ref="gml:AssociationAttributeGroup"/>
  </complexType>
  <!-- ================================================================= -->
  <element name="Matrix" type="saf:MatrixType"/>
  <!-- ================================================================= -->
  <complexType name="MatrixType">
      <annotation>
          <documentation>A general structure for two-dimensional matrices consisting of numeric primitives.
The structure for the matrix is standard such that the first value in the list refers to the
upper left position of the matrix and the last value to the lower right position. The
values are listed row by row, with each row running from left to right and starting with
the top row.</documentation>
      </annotation>
      <complexContent>
          <extension base="gml:AbstractGMLType">
              <sequence>
                  <element name="rows" type="positiveInteger"/>
                  <element name="columns" type="positiveInteger"/>
                  <element name="matrixValues" type="gml:doubleList"/>
              </sequence>
          </extension>
      </complexContent>
  </complexType>
  <!-- ================================================================= -->
  <element name="positioningMethod" type="saf:PositioningMethodType"/>
  <!-- ================================================================= -->
  <simpleType name="PositioningMethodType">
      <restriction base="string">
          <enumeration value="differentialGPS"/>
          <enumeration value="singlePositionGPS"/>
          <enumeration value="photogrammetry"/>
          <enumeration value="radioPositioning"/>
          <enumeration value="terrestrial"/>
          <enumeration value="astronomicObservation"/>
          <enumeration value="safroximate"/>
          <enumeration value="fromMap"/>
          <enumeration value="other"/>
      </restriction>
```

```
</simpleType>
<!-- ================================================================== -->
<element name="north" type="saf:NorthType"/>
<!-- ================================================================== -->
<simpleType name="NorthType">
    <restriction base="string">
        <enumeration value="geodetic"/>
        <enumeration value="grid"/>
    </restriction>
</simpleType>
<element name="horizontal" type="saf:HorizontalPropertyType"/>
<!-- ================================================================== -->
<complexType name="HorizontalPropertyType">
    <sequence>
        <element ref="saf:HorizontalText" minOccurs="0"/>
    </sequence>
</complexType>
<!-- ================================================================== -->
<element name="HorizontalText" type="saf:HorizontalTextType"/>
<!-- ================================================================== -->
<simpleType name="HorizontalTextType">
    <restriction base="string">
        <enumeration value="left"/>
        <enumeration value="centre"/>
        <enumeration value="right"/>
        <enumeration value="normalHorizontal"/>
        <enumeration value="continuousHorizontal"/>
    </restriction>
</simpleType>
<!-- ================================================================== -->
<element name="vertical" type="saf:VerticalPropertyType"/>
<!-- ================================================================== -->
<complexType name="VerticalPropertyType">
    <sequence>
        <element ref="saf:VerticalText" minOccurs="0"/>
    </sequence>
</complexType>
<element name="VerticalText" type="saf:VerticalTextType"/>
<!-- ================================================================== -->
<simpleType name="VerticalTextType">
    <restriction base="string">
        <enumeration value="top"/>
        <enumeration value="cap"/>
        <enumeration value="half"/>
        <enumeration value="base"/>
        <enumeration value="bottom"/>
        <enumeration value="normalVertical"/>
<!-- ================================================================== -->
<element name="verticalControl" type="saf:DataControlPropertyType"/>
<!-- ================================================================== -->
<element name="horizontalControl" type="saf:DataControlPropertyType"/>
<!-- ================================================================== -->
<!-- ================================================================== -->
<complexType name="DataControlPropertyType">
    <sequence>
        <element ref="saf:DataControl" minOccurs="0"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
<!-- ================================================================== -->
<element name="DataControl" type="saf:DataControlType"/>
<!-- ================================================================== -->
<complexType name="DataControlType">
    <annotation>
        <documentation/>
    </annotation>
    <complexContent>
        <extension base="gml:AbstractGMLType">
            <sequence>
                <element ref="saf:authority" minOccurs="0"/>
                <element name="adjustmentOrder" type="string" minOccurs="0"/>
                <element name="integrationStatus" type="string" minOccurs="0"/>
                <element ref="saf:collectionAgency" minOccurs="0"/>
                <element name="finalCollectionDate" type="dateTime" minOccurs="0"/>
                <element ref="saf:adjustmentAgency" minOccurs="0"/>
                <element name="finalAdjustmentDate" type="dateTime" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
        <enumeration value="continuousVertical"/>
    </restriction>
</simpleType>
<!-- ============================================================= -->
<element name="authority" type="saf:AgencyPropertyType"/>
<!-- ============================================================= -->
<element name="adjustmentAgency" type="saf:AgencyPropertyType"/>
<!-- ============================================================= -->
<complexType name="AgencyPropertyType">
    <sequence>
        <element ref="saf:Agency" minOccurs="0"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
<!-- ============================================================= -->
<complexType name="AgencyPropertyType">
    <sequence>
        <element ref="saf:Agency" minOccurs="0"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
<!-- ============================================================= -->
<element name="Agency" type="saf:AgencyType"/>
<!-- ============================================================= -->
<complexType name="AgencyType">
    <annotation>
        <documentation/>
    </annotation>
    <complexContent>
        <extension base="gml:AbstractGMLType">
            <sequence>
                <element name="contactPerson" type="string" minOccurs="0"/>
                <element name="agencyName" type="string" minOccurs="0"/>
                <element name="address" type="string" minOccurs="0"/>
                <element name="telephoneNumber" type="string" minOccurs="0"/>
                <element name="faxNumber" type="string" minOccurs="0"/>
                <element name="email" type="string" minOccurs="0"/>
                <element name="country" type="string" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

8 GML Application Schema Tools

One of the advantages of GML over conventional geographic data formats is that GML can make use of readily available XML tools and technology. Section 8.1 demonstrates that it is simple to create GML application schemas and instances using Altova's XML Integrated Development Environment (IDE). Although Altova's popular XMLSpy XML IDE has been selected for this purpose, there are several other XML editors. Section 8.2 is devoted to a survey of the currently popular XML tools and software components. Section 8.3 then follows with a survey of currently supported GML tools. Each of the surveys in Section 8.2 and Section 8.3 outline the features, limitations, and licensing/cost of the individual software components.

8.1 Creating GML Application Schemas with XML Spy

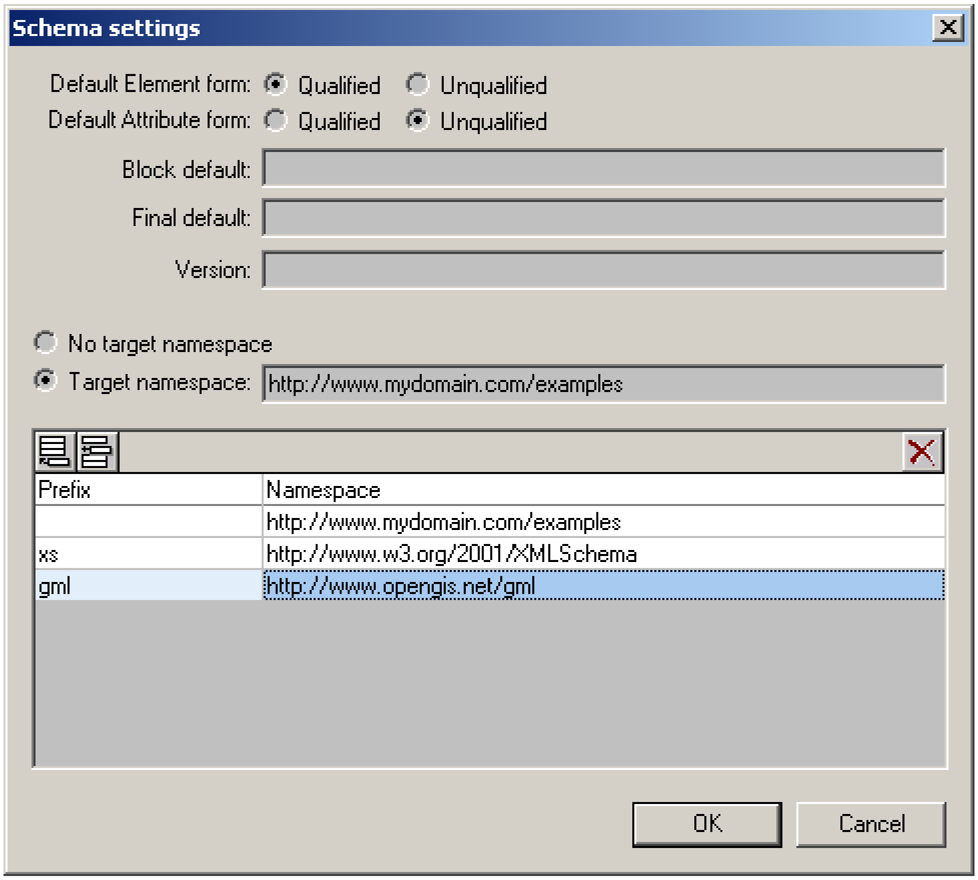
Getting Started

1. Start **XMLSpy** and select **File | New**.
2. Select **W3C XML Schema** as the file type and click **OK**.

This creates a dummy document with no schema content.

3. Select **Schema Design | Schema settings** to specify some basic schema parameters, including:
   - The `elementFormDefault` and `attributeFormDefault` attributes (use the default settings in the dialog box, as shown in Figure 8-1)
   - The target namespace
   - The application domain namespace prefix
   - The gml namespace and namespace prefix

Use the lower window of the **Schema settings** dialog box to define prefixes, as shown in Figure8-1. Make sure that you provide a prefix for the gml namespace (http://www.opengis.net/gml) and your selected application namespace (target namespace). Click the ▤ icon to add the gml prefix and namespace.



Default Element form: ● Qualified ○ Unqualified

Default Attribute form: ○ Qualified ● Unqualified

Block default:

Final default:

Version:

○ No target namespace

● Target namespace: http://www.mydomain.com/examples

Prefix	Namespace
	http://www.mydomain.com/examples
xs	http://www.w3.org/2001/XMLSchema
gml	http://www.opengis.net/gml

Figure 8.1: Setting Namespaces and Prefixes

4. Declare the target namespace and any other namespaces that will be used, click **OK**.

The **Schema Design** window appears. If it does not appear, select **View | Schema Design View.**

5. To import the GML feature schema (`feature.xsd`) or all of the gml core schemas (gml.xsd), click the Append icon (in the upper left corner of the schema window in XMLSpy) and select **Import** from the pop-up menu. You can get the gml core schemas at http://schemas.opengis.net/. This is shown in Figure 8-2.

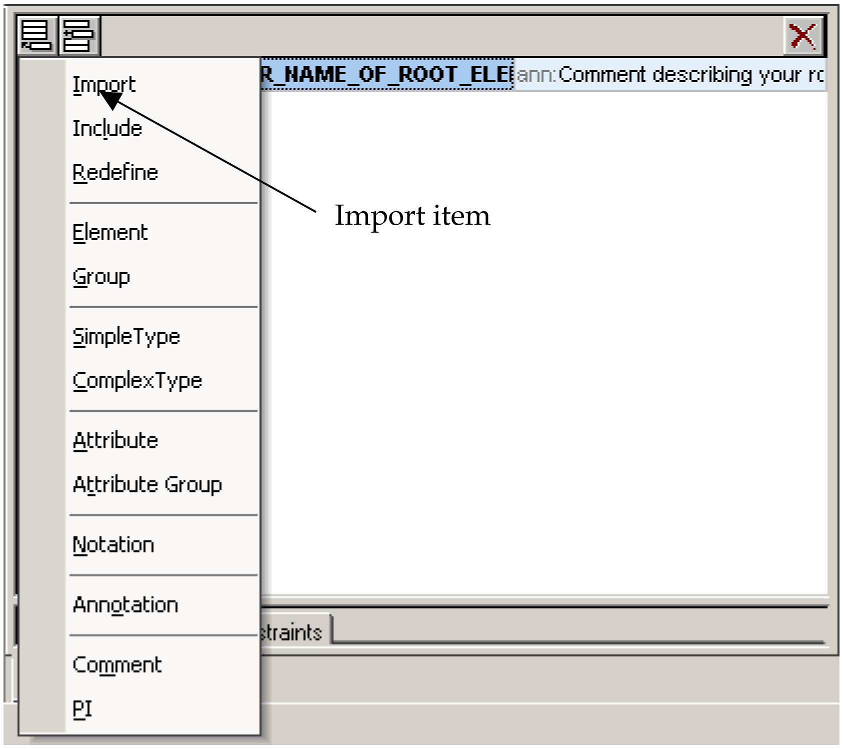


Figure 8-2: Importing the GML feature.xsd

6. In the alert box, enter the location of the `feature.xsd` file.
7. Add the GML namespace (http://www.opengis.net/gml) in the field marked **ns:** to the right of the schema location.

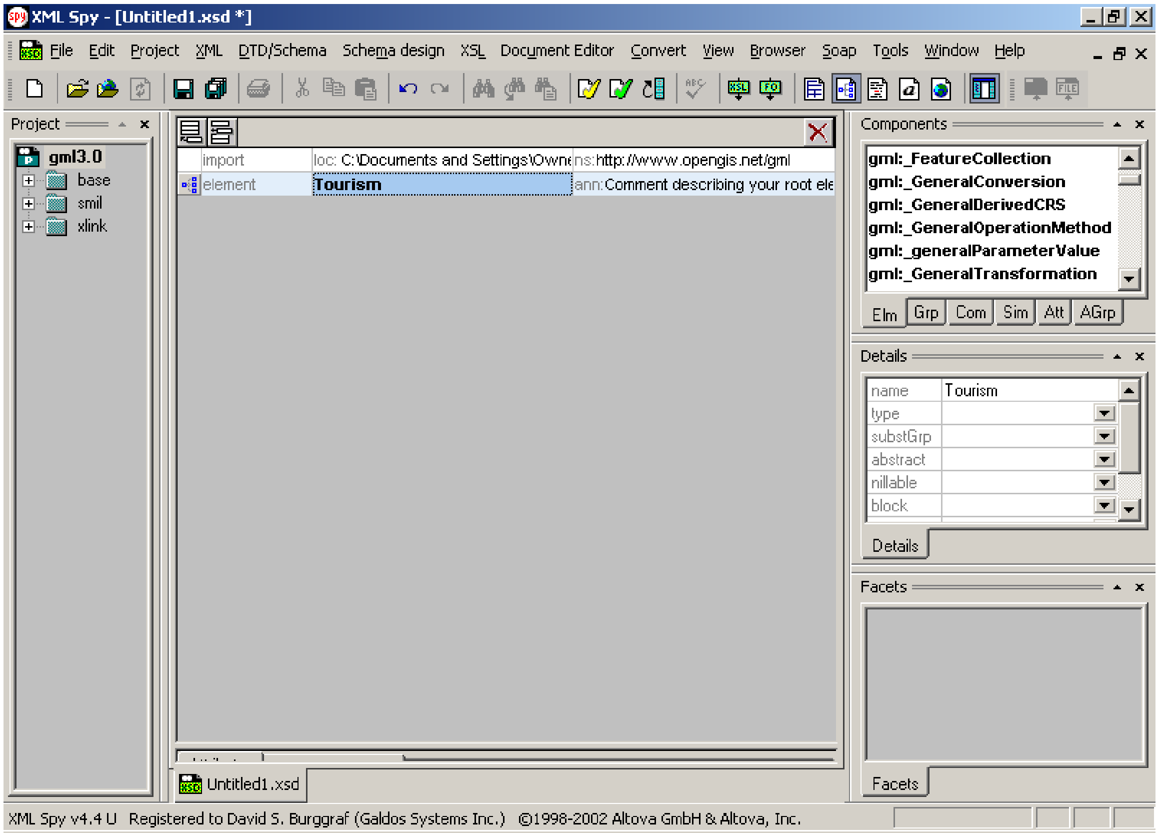
Note that you can refer to schema locations that are local or anywhere on the Internet.

Creating the Root Feature Collection

Many GML Application Schemas have a root Feature Collection that is a container for a collection of feature instances. To set this root element:

8. Rename the root element of the schema generated by XMLSpy.

Figure 8-3 shows an example where the root element is changed to `Tourism`.


XML Spy - [Untitled1.xsd *]
File Edit Project XML DTD/Schema Schema design XSL Document Editor Convert View Browser Soap Tools Window Help
Project
gml3.0
base
smil
xlink
Components
gml:_FeatureCollection
gml:_GeneralConversion
gml:_GeneralDerivedCRS
gml:_GeneralOperationMethod
gml:_generalParameterValue
gml:_GeneralTransformation
Elm Grp Com Sim Att AGrp
Details
Facets
Untitled1.xsd

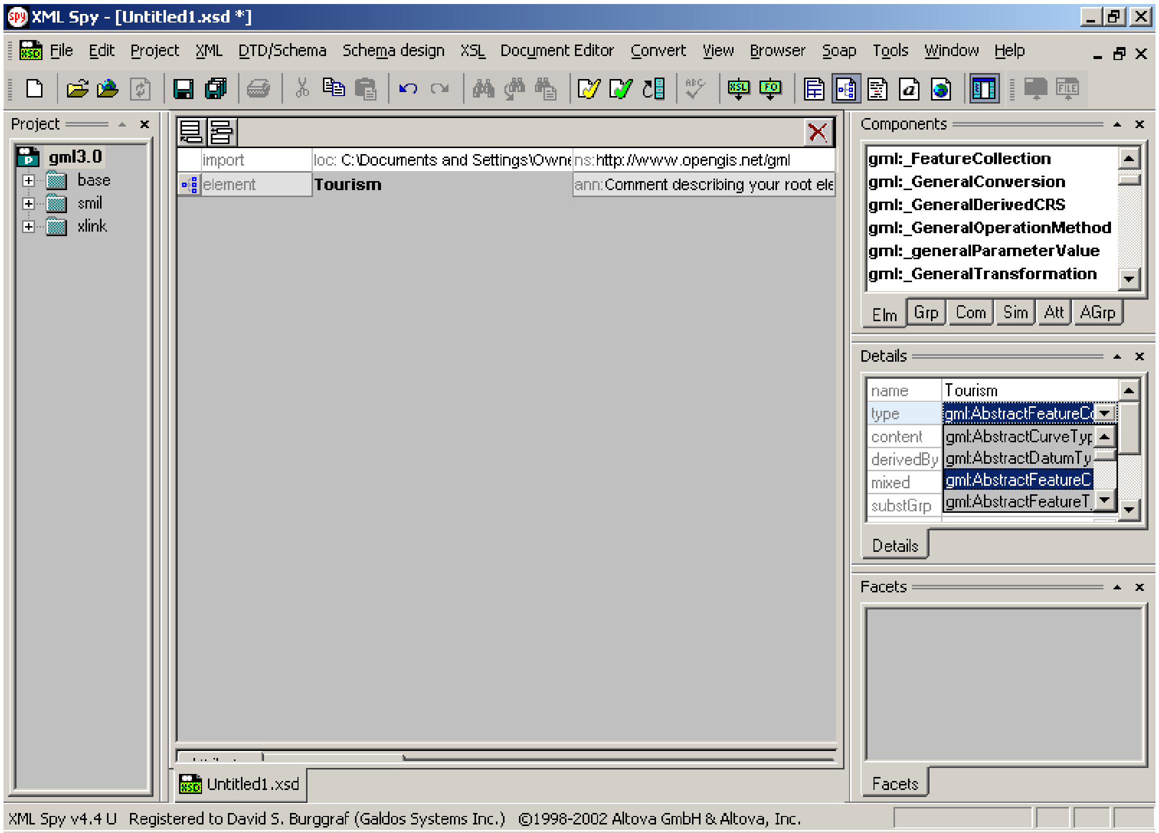
import loc: C:\Documents and Settings\Own ns:http://www.opengis.net/gml
element Tourism ann:Comment describing your root ele

name Tourism
type
substGrp
abstract
nillable
block

XML Spy v4.4 U Registered to David S. Burggraf (Galdos Systems Inc.) ©1998-2002 Altova GmbH & Altova, Inc.

Figure 8-3: Create the root Feature Collection by renaming the root element

9. To set the type of this element (for example, `Tourism`), select the element, select **type** in the **Details** window and then select `gml:AbstractFeatureCollectionType` in the **type** drop-down list (see Figure 8-4). Next click **SubstGrp** in the **Details** window and select `gml:_FeatureCollection`.



import | loc: C:\Documents and Settings\Own | ns:http://www.opengis.net/gml

element | Tourism | ann:Comment describing your root ele

gml:_FeatureCollection
gml:_GeneralConversion
gml:_GeneralDerivedCRS
gml:_GeneralOperationMethod
gml:_generalParameterValue
gml:_GeneralTransformation

name: Tourism
type: gml:AbstractFeatureC
content
derivedBy
mixed
substGrp

gml:AbstractCurveTyp
gml:AbstractDatumTy
gml:AbstractFeatureC
gml:AbstractFeatureT

XML Spy v4.4 U Registered to David S. Burggraf (Galdos Systems Inc.) ©1998-2002 Altova GmbH & Altova, Inc.

Figure 8-4: Set the type of the root Feature Collection to `AbstractFeatureCollectionType`

The root feature collection may derive from some other type, which in turn derives from `gml:AbstractFeatureCollectionType`. If this is the case, you should select this parent type instead of `gml:AbstractFeatureCollectionType`. If you want to create a new feature collection type that extends `gml:AbstractFeatureCollectionType`, define the type of the root element as `gml:AbstractFeatureCollectionType` (as outlined in step 2 of this procedure). Then extend it by adding new properties.

Creating Feature Types

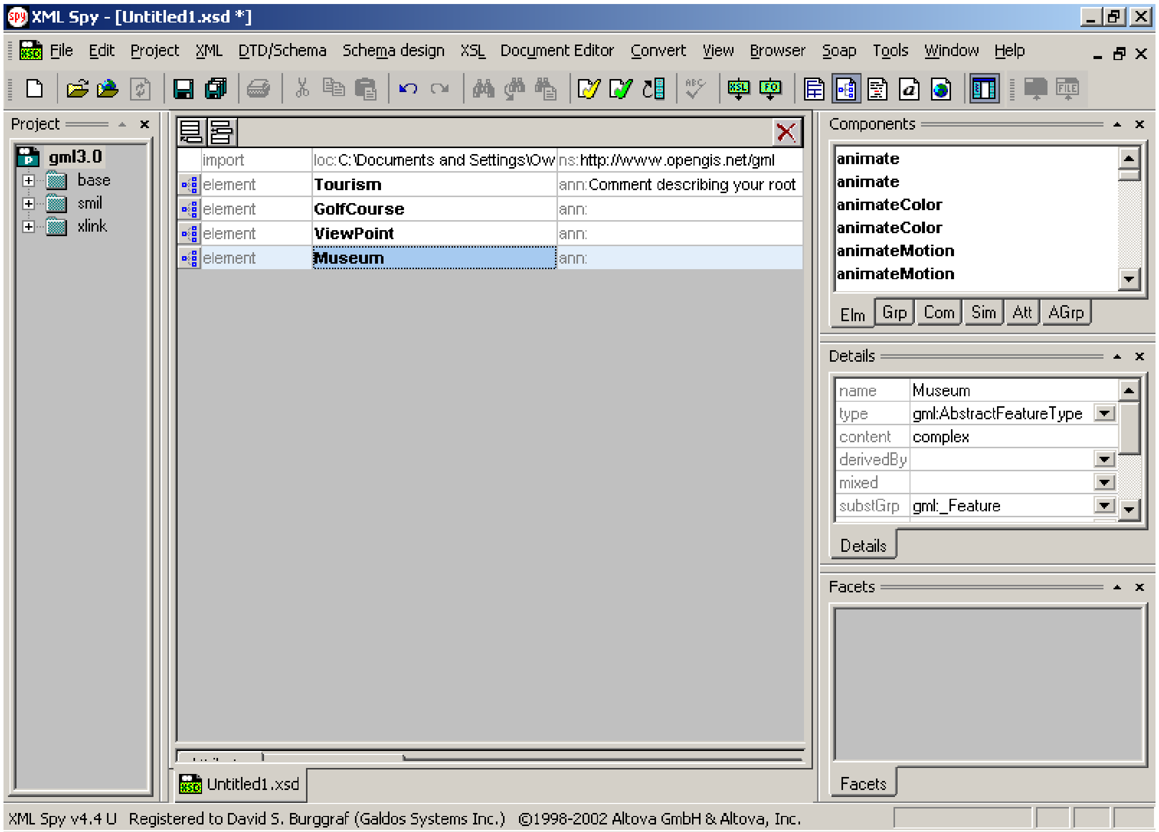
A `Feature` type is a kind of geographic feature. To create a new feature type:

10. Click the ☰ (Append) icon in the upper left corner of the **Schema design** window and select **Element** from the drop-down list.

11. Enter a name for the new element.

This name is the feature type (for example, Golf Course, ViewPoint, Museum).

12. In the **Details** window on the right side of the **Schema Design** window, select the **type** and **substitutionGroup**. Figure 8-5 shows an example of the Museum element, where the **type** is `gml:AbstractFeatureType` and the **substitutionGroup** is `gml:_Feature`.



import	loc:C:\Documents and Settings\Ow	ns:http://www.opengis.net/gml
element	Tourism	ann:Comment describing your root
element	GolfCourse	ann:
element	ViewPoint	ann:
element	Museum	ann:

name	Museum
type	gml:AbstractFeatureType
content	complex
derivedBy	
mixed	
substGrp	gml:_Feature

XML Spy v4.4 U Registered to David S. Burggraf (Galdos Systems Inc.) ©1998-2002 Altova GmbH & Altova, Inc.

Figure 8-5: Creating a Feature Type

Initially, the **type** in the **Details** window must be set to `gml:AbstractFeatureType`. When you add additional properties, XMLSpy creates an anonymous complex type and derives this anonymous type from `gml:AbstractFeatureType`.

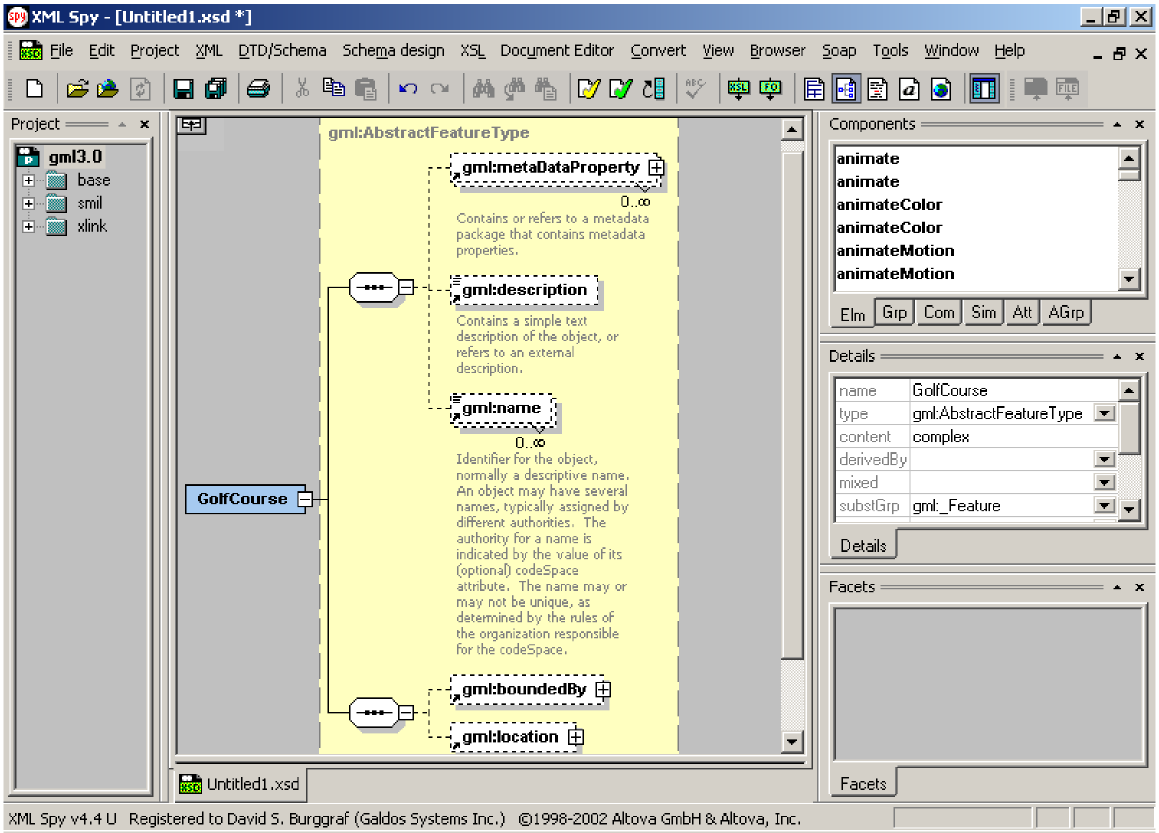
To create additional features (**types**), repeat steps 10 to 12. Once the feature name and **Details** have been set, add properties to the feature, as explained in the next section.

Adding Feature Properties

The simplest way to add properties to a feature is to select the feature and add a sequence of property elements. To do this:

13. In the **Schema Design** window, select the feature by clicking on the schema detail icon to the left of the feature.

This provides a graphical display of the schema of the selected feature type as shown in Figure 8-6.



gml:metaDataProperty: Contains or refers to a metadata package that contains metadata properties.

gml:description: Contains a simple text description of the object, or refers to an external description.

gml:name: Identifier for the object, normally a descriptive name. An object may have several names, typically assigned by different authorities. The authority for a name is indicated by the value of its (optional) codeSpace attribute. The name may or may not be unique, as determined by the rules of the organization responsible for the codeSpace.

name	GolfCourse
type	gml:AbstractFeatureType
content	complex
derivedBy	
mixed	
substGrp	gml:_Feature

XML Spy v4.4 U Registered to David S. Burggraf (Galdos Systems Inc.) ©1998-2002 Altova GmbH & Altova, Inc.

Figure 8-6: Graphical Display of the GolfCourse Feature Type

Note that in Figure 8-6, the feature type is already derived from `gml:AbstractFeatureType` and inherits the name, description, boundedBy properties and the gml:id attribute.

14. Before adding the first property, insert a sequence into the feature type as follows: right-click the feature type, for example GolfCourse, as shown in Figure . From the drop-down list select **Add Child** and then **Sequence**.

Note that the created sequence is automatically hi-lighted.

15. Add properties to the feature as follows:

a. Right-click the element, and then select **Add Child** and **Element** from the drop-down list.

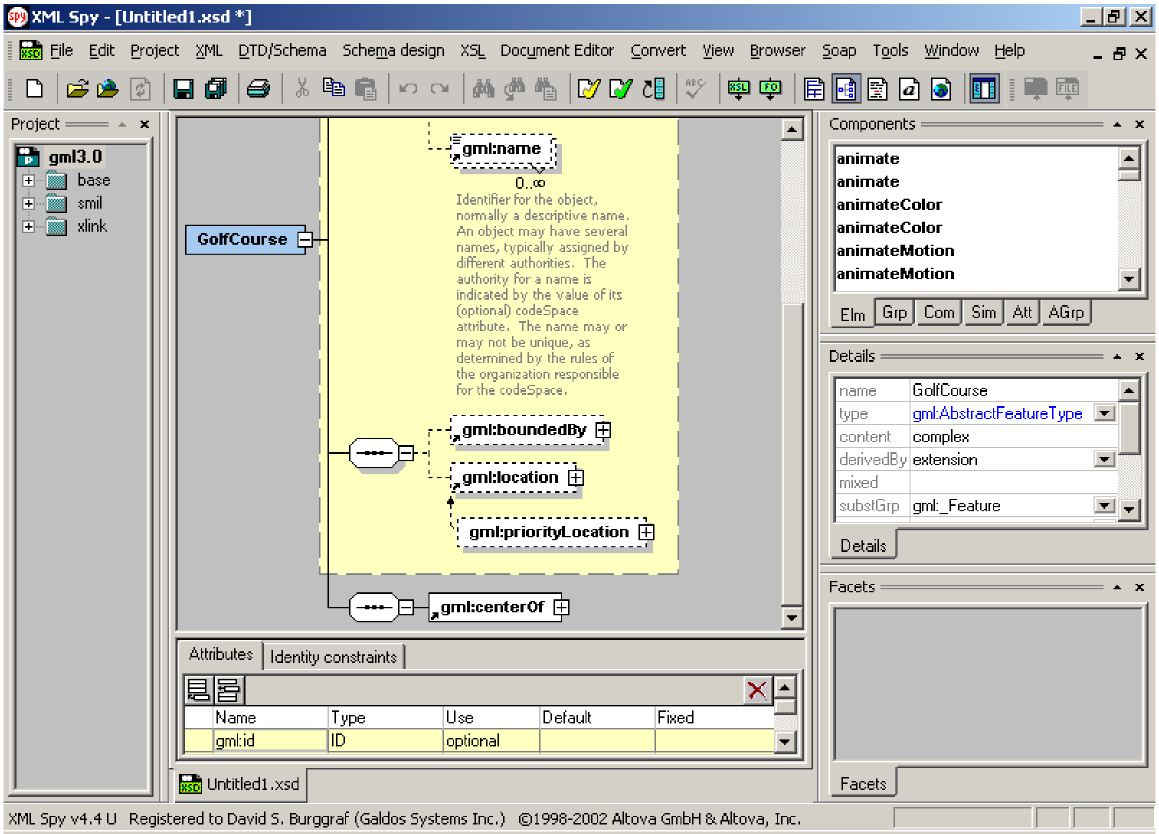
b. Select the name of the property from the drop-down list at the right side of the property graphic, for example gml:centerOf, as shown in Figure 8-7.

c. Set the **type** of the created property in the **Details** window on the right.

d. For XML Schema built-in properties, you can set the type by selecting it from the drop-down list.

Note that all simple types have the xs namespace prefix (for example, `xs:integer` or `xs:time`).

e. To use a gml property, select it from the drop-down list at the right of the hi-lighted element.



Identifier for the object, normally a descriptive name. An object may have several names, typically assigned by different authorities. The authority for a name is indicated by the value of its (optional) codeSpace attribute. The name may or may not be unique, as determined by the rules of the organization responsible for the codeSpace.

name	GolfCourse
type	gml:AbstractFeatureType
content	complex
derivedBy	extension
mixed	
substGrp	gml:_Feature

Name	Type	Use	Default	Fixed
gml:id	ID	optional		

XML Spy v4.4 U Registered to David S. Burggraf (Galdos Systems Inc.) ©1998-2002 Altova GmbH & Altova, Inc.

Figure 8-7: Adding Properties to the Feature

Recall that GML geometry properties include the following:

Geometry Property Name	Geometry Type
`gml:position`	`Point`
`gml:location`	`_Geometry`
`gml:centerOf`	`Point`
`gml:centerLineOf`	`Curve`
`gml:edgeOf`	`Curve`
`gml:extentOf`	`Surface`
`gml:coverage`	`Surface`

You can add additional properties to your feature type by repeating step 3 for each new property.

Create a simple type

You can create new simple properties by extending the XML Schema built-in types like `xs:integer` or `xs:float` or other simple types you created earlier.

To create a simple type:

1. Select the **Schema Design** view in XMLSpy and click on the ▤ (Append) icon in the upper left corner of the window.

This displays a drop-down list.

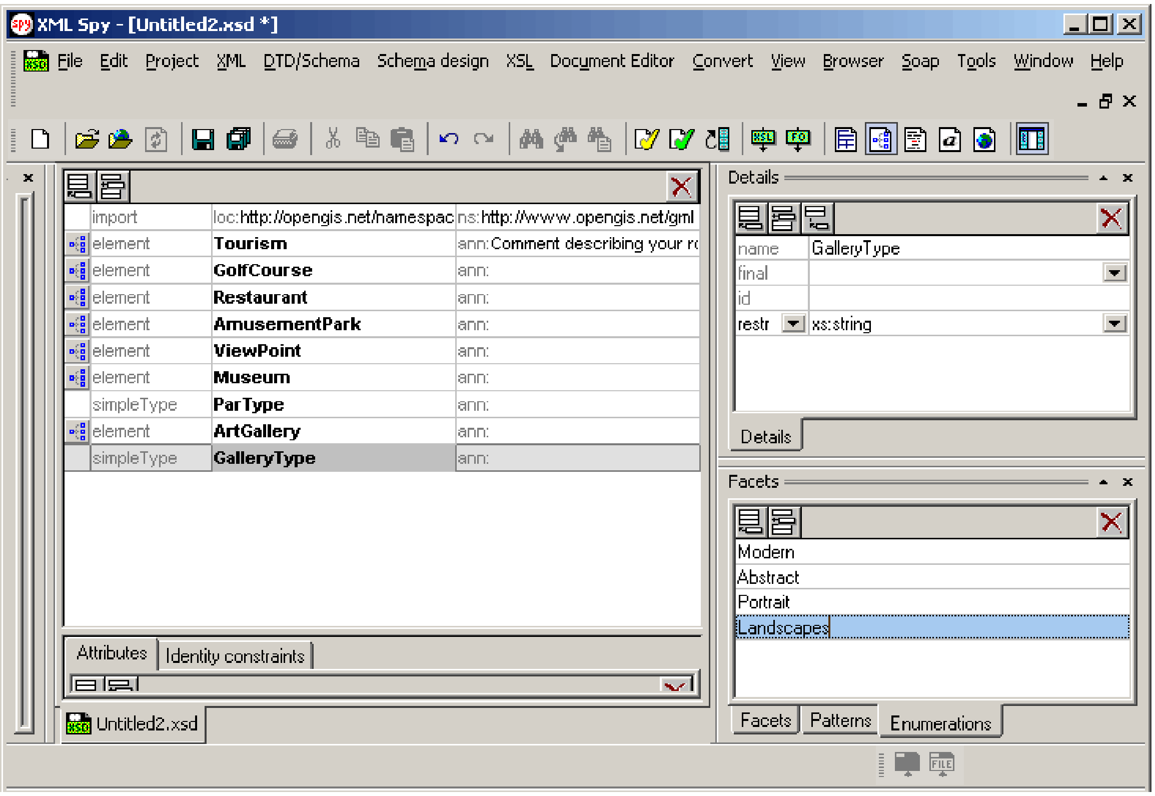
2. Select the **SimpleType** menu item.

A **SimpleType** appears in the main window, as shown in Figure 8-8.
Select one of the following to finish creating the simple type, based on the rules of XML Schema: **Union**, **List**, or **Restriction**. We use **restriction** here to illustrate how to create an enumerative simple type.

3. Select **restriction** in the **Details** window (this is the default).

4. Select the data type to be restricted.

In this example, select `xs:string` from the drop-down list to the right of **restriction** in the **Details** window.


XML Spy - [Untitled2.xsd *]
File Edit Project XML DTD/Schema Schema design XSL Document Editor Convert View Browser Soap Tools Window Help
Attributes
Identity constraints
Untitled2.xsd
Details
Facets
Facets Patterns Enumerations

import	loc:http://opengis.net/namespac	ns:http://www.opengis.net/gml
element	Tourism	ann:Comment describing your r
element	GolfCourse	ann:
element	Restaurant	ann:
element	AmusementPark	ann:
element	ViewPoint	ann:
element	Museum	ann:
simpleType	ParType	ann:
element	ArtGallery	ann:
simpleType	GalleryType	ann:

name	GalleryType
final	
id	
restr	xs:string

Modern
Abstract
Portrait
Landscapes

Figure 8-8: Creating Enumeration Values

Create Enumeration Values

5. Create each enumeration value in the **Facets** window by selecting the **Enumerations** tab at the bottom of the **Facets** window.

6. Click on the (Append) icon at the top left corner of the **Facets** window.

Whenever you click this icon, you add one enumeration value to the enumerated type.

7. Enter the value in the highlighted bar which appears each time you click on the ▤ icon, as shown in Figure 8-8.

You can continue to add new simple types in this manner.
To use these created simple types for your properties, just reference them in your property elements using the **type** drop-down list in the **Details** window.

Summary

Listing 6 shows an example of the created GML Application Schema.

Listing 6: Example of a GML Application Schema

```
 <?xml version="1.0" encoding="UTF-8"?>
 <xs:schema targetNamespace="http://www.ukusa.com/example"
 xmlns:exp="http://www.ukusa.com/example"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:gml="http://www.opengis.net/gml"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:import namespace="http://www.opengis.net/gml"
 schemaLocation="D:\Galdos\examples\project\gml2.1\feature.xsd"/>
   <xs:element name="Tourism"
 type="gml:AbstractFeatureCollectionType">
       <xs:annotation>
 <xs:documentation>Comment describing your root
 element</xs:documentation>
       </xs:annotation>
   </xs:element>
   <xs:element name="GolfCourse" substitutionGroup="gml:_Feature">
       <xs:complexType>
           <xs:complexContent>
               <xs:extension base="gml:AbstractFeatureType">
                   <xs:sequence>
                       <xs:element name="par" type="exp:ParType"/>
                       <xs:element name="noHoles" type="xs:integer"/>
                       <xs:element ref="gml:position"/>
                       <xs:element ref="gml:extentOf"/>
                   </xs:sequence>
               </xs:extension>
           </xs:complexContent>
       </xs:complexType>
   </xs:element>
   <xs:element name="Restaurant" substitutionGroup="gml:_Feature">
       <xs:complexType>
           <xs:complexContent>
               <xs:extension base="gml:AbstractFeatureType">
```

```
                    <xs:sequence>
                        <xs:element ref="gml:position"/>
      </xs:sequence>

                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="AmusementPark"
  substitutionGroup="gml:_Feature">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="gml:AbstractFeatureType">
                    <xs:sequence>
                        <xs:element ref="gml:position"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="ViewPoint" substitutionGroup="gml:_Feature">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="gml:AbstractFeatureType">
                    <xs:sequence>
                        <xs:element ref="gml:position"/>
                        <xs:element name="interest" type="xs:string"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:simpleType name="ParType">
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="68"/>
            <xs:maxInclusive value="72"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:element name="ArtGallery" substitutionGroup="gml:_Feature">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="gml:AbstractFeatureType">
```

```
               <xs:sequence>
                  <xs:element ref="gml:position"/>
                  <xs:element name="type" type="exp:GalleryType"/>
               </xs:sequence>
            </xs:extension>
         </xs:complexContent>
      </xs:complexType>
   </xs:element>
   <xs:simpleType name="GalleryType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="Modern"/>
         <xs:enumeration value="Abstract"/>
         <xs:enumeration value="Portrait"/>
         <xs:enumeration value="Landscapes"/>
      </xs:restriction>
   </xs:simpleType>
</xs:schema>
```

This section has shown that the creation and maintenance of GML Application Schemas can be greatly simplified by using the advanced editing features of XML Schema editors, such as XMLSpy. Next we will examine demonstrate how to create GML application instances using XMLSpy.

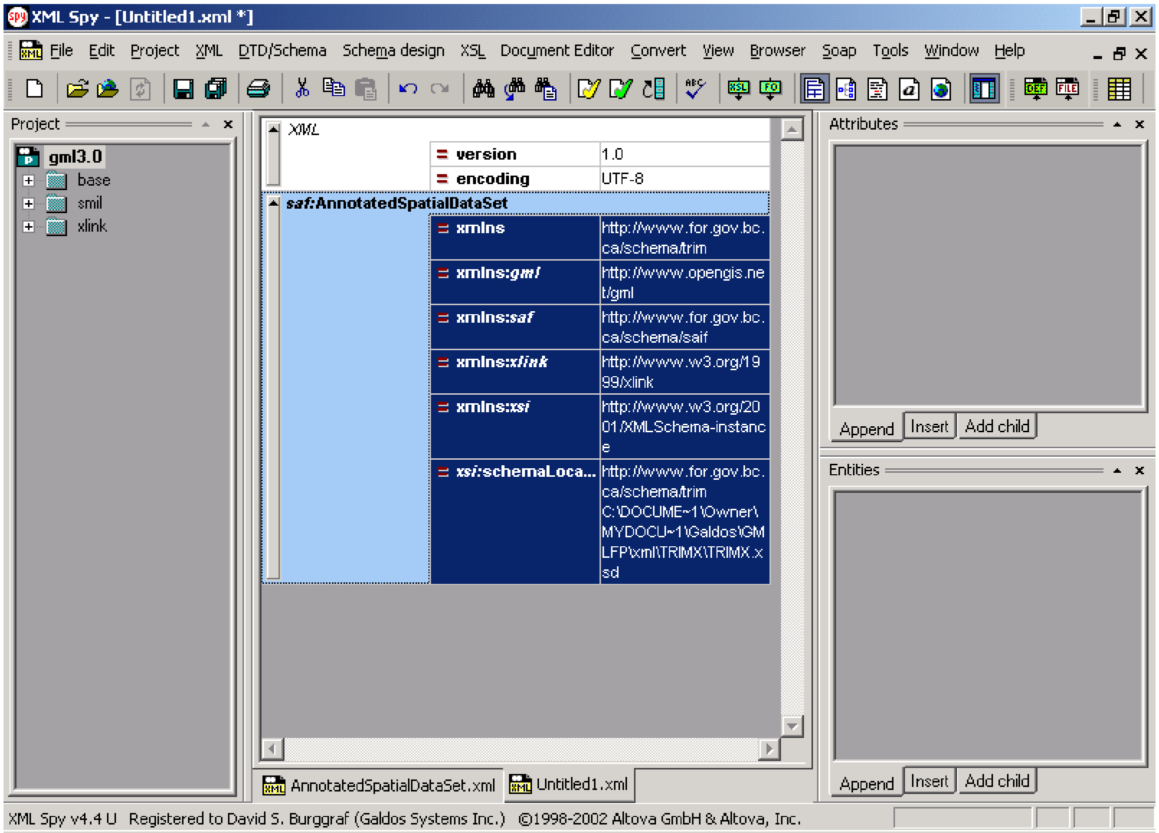
Creating GML Application Instances

1. Start **XMLSpy** and select **File | New**.
2. Select **XML Document** as the file type and click **OK**.

This creates a blank document with no schema content.

3. When you are prompted to decide whether to base the XML document on a DTD or Schema, select **Schema** and click **OK**.
4. Enter the GML Application Schema on which your instance document will be based (for example, TRIMX.xsd).
5. Once XMLSpy processes your schema, select a root element for the instance document.

This should be a feature collection (that is, something that derives from `gml:AbstractFeatureCollectionType` in your GML Application Schema).
After you select the root element, XMLSpy creates a nearly empty instance document as shown in Figure 8-9, where `AnnotatedSpatialDataSet` is the root element.


XML Spy - [Untitled1.xml *]
File Edit Project XML DTD/Schema Schema design XSL Document Editor Convert View Browser Soap Tools Window Help
Project
gml3.0
base
smil
xlink
Attributes
Append Insert Add child
Entities
Append Insert Add child
AnnotatedSpatialDataSet.xml
Untitled1.xml

XML

version	1.0
encoding	UTF-8

saf:AnnotatedSpatialDataSet

xmlns	http://www.for.gov.bc.ca/schema/trim
xmlns:gml	http://www.opengis.net/gml
xmlns:saf	http://www.for.gov.bc.ca/schema/saif
xmlns:xlink	http://www.w3.org/1999/xlink
xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLoca...	http://www.for.gov.bc.ca/schema/trim C:\DOCUME~1\Owner\MYDOCU~1\Galdos\GMLFP\xml\TRIMX\TRIMX.xsd

XML Spy v4.4 U Registered to David S. Burggraf (Galdos Systems Inc.) ©1998-2002 Altova GmbH & Altova, Inc.

Figure 8-9: Empty Instance Document

Entering Properties

6. You can now start entering properties of the root element. To do this:

    f. Select and right-click on the root element, for example, `saf:AnnotatedSpatialDataSet` (which is substitutable for `gml:_FeatureCollection`.)

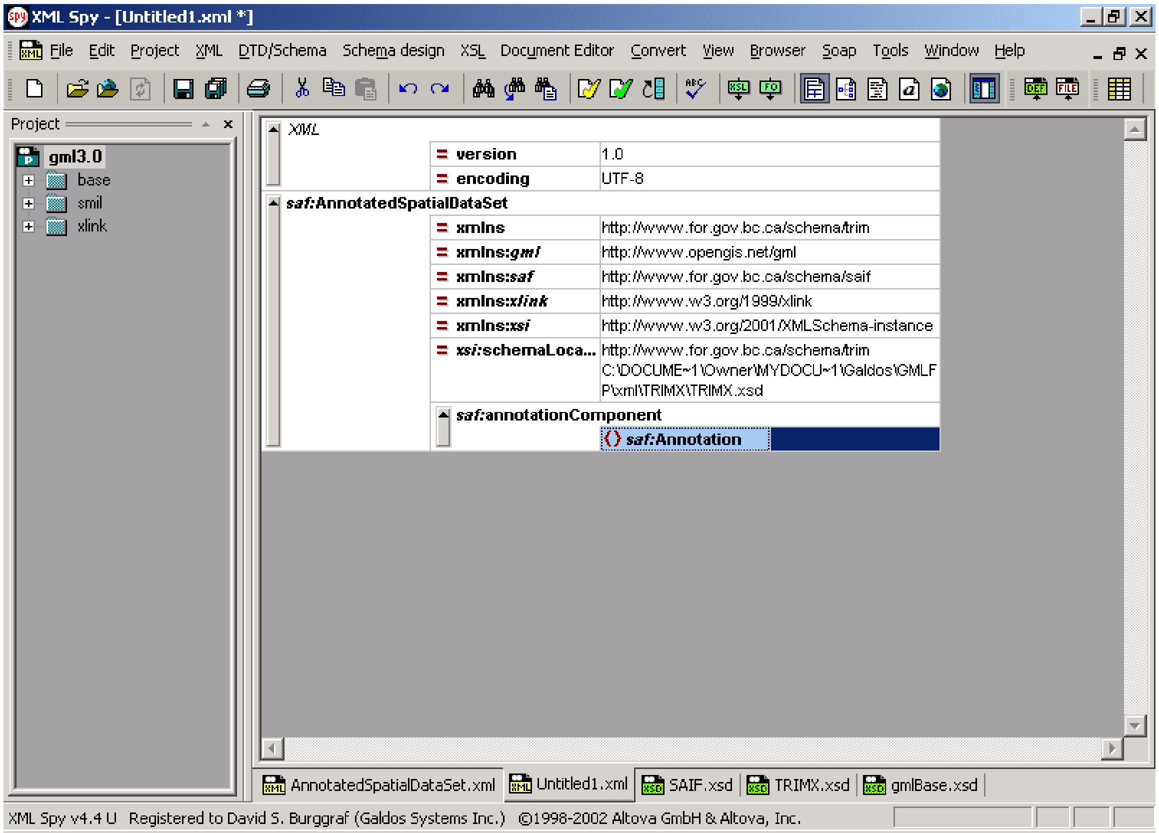
    g. Select **Add Child** and select **Element**.

    A list of valid child elements appears. In this example, only `saf:annotationComponent` appears (which is substitutable for `gml:metaDataProperty`)

    h. Select (or type) `saf:annotationComponent`.

7. To add an `saf:Annotation` (which is substitutable for `gml:_MetaData`) child element to the `saf:annotationComponent` element, right-click on the `saf:annotationComponent` element, select **Add Child** and select **Element**.

A list of possible property values appear

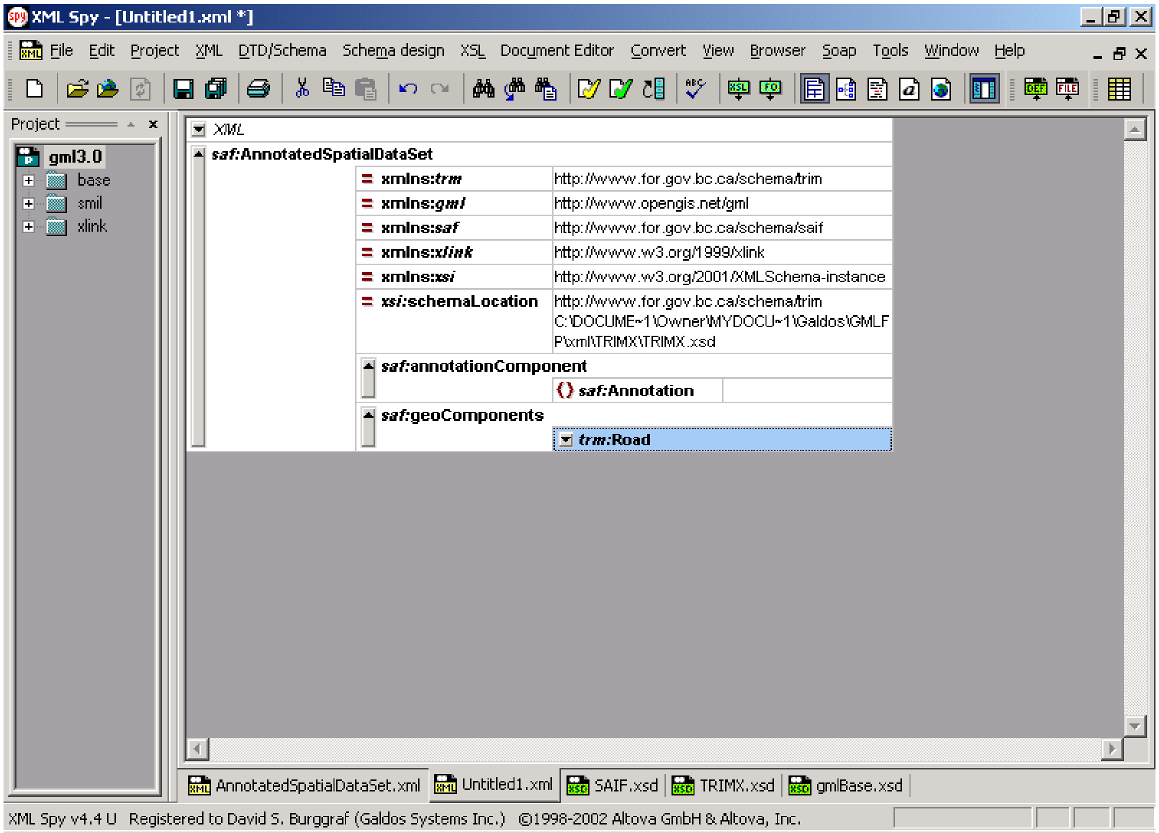

XML Spy - [Untitled1.xml *]
File Edit Project XML DTD/Schema Schema design XSL Document Editor Convert View Browser Soap Tools Window Help
Project
gml3.0
base
smil
xlink
AnnotatedSpatialDataSet.xml
Untitled1.xml
SAIF.xsd
TRIMX.xsd
gmlBase.xsd
XML Spy v4.4 U Registered to David S. Burggraf (Galdos Systems Inc.) ©1998-2002 Altova GmbH & Altova, Inc.

XML			
	= version	1.0	
	= encoding	UTF-8	
saf:AnnotatedSpatialDataSet			
	= xmlns	http://www.for.gov.bc.ca/schema/trim	
	= xmlns:gml	http://www.opengis.net/gml	
	= xmlns:saf	http://www.for.gov.bc.ca/schema/saif	
	= xmlns:xlink	http://www.w3.org/1999/xlink	
	= xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance	
	= xsi:schemaLoca...	http://www.for.gov.bc.ca/schema/trim C:\DOCUME~1\Owner\MYDOCU~1\Galdos\GMLFP\xml\TRIMX\TRIMX.xsd	
	saf:annotationComponent		
		() saf:Annotation	

Figure 8-10: Property Value is Entered

8. Select or type the property value of interest (for example, `saf:Annotation`) as shown in Figure 8-10.

Be sure not to select `gml:_MetaData` or `saf:_MetaData` as these are abstract elements and cannot appear in your GML instance document. Note that your instance document is not valid at this point, since you have not specified the values of the properties of the `saf:Annotation` element that was entered.


XML Spy - [Untitled1.xml *]
File Edit Project XML DTD/Schema Schema design XSL Document Editor Convert View Browser Soap Tools Window Help
Project
gml3.0
base
smil
xlink
XML
AnnotatedSpatialDataSet.xml Untitled1.xml SAIF.xsd TRIMX.xsd gmlBase.xsd
XML Spy v4.4 U Registered to David S. Burggraf (Galdos Systems Inc.) ©1998-2002 Altova GmbH & Altova, Inc.

saf:AnnotatedSpatialDataSet

= xmlns:trm	http://www.for.gov.bc.ca/schema/trim
= xmlns:gml	http://www.opengis.net/gml
= xmlns:saf	http://www.for.gov.bc.ca/schema/saif
= xmlns:xlink	http://www.w3.org/1999/xlink
= xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
= xsi:schemaLocation	http://www.for.gov.bc.ca/schema/trim C:\DOCUME~1\Owner\MYDOCU~1\Galdos\GMLFP\xml\TRIMX\TRIMX.xsd
saf:annotationComponent	saf:Annotation
saf:geoComponents	trm:Road

Figure 8-11: Sample of a New Feature

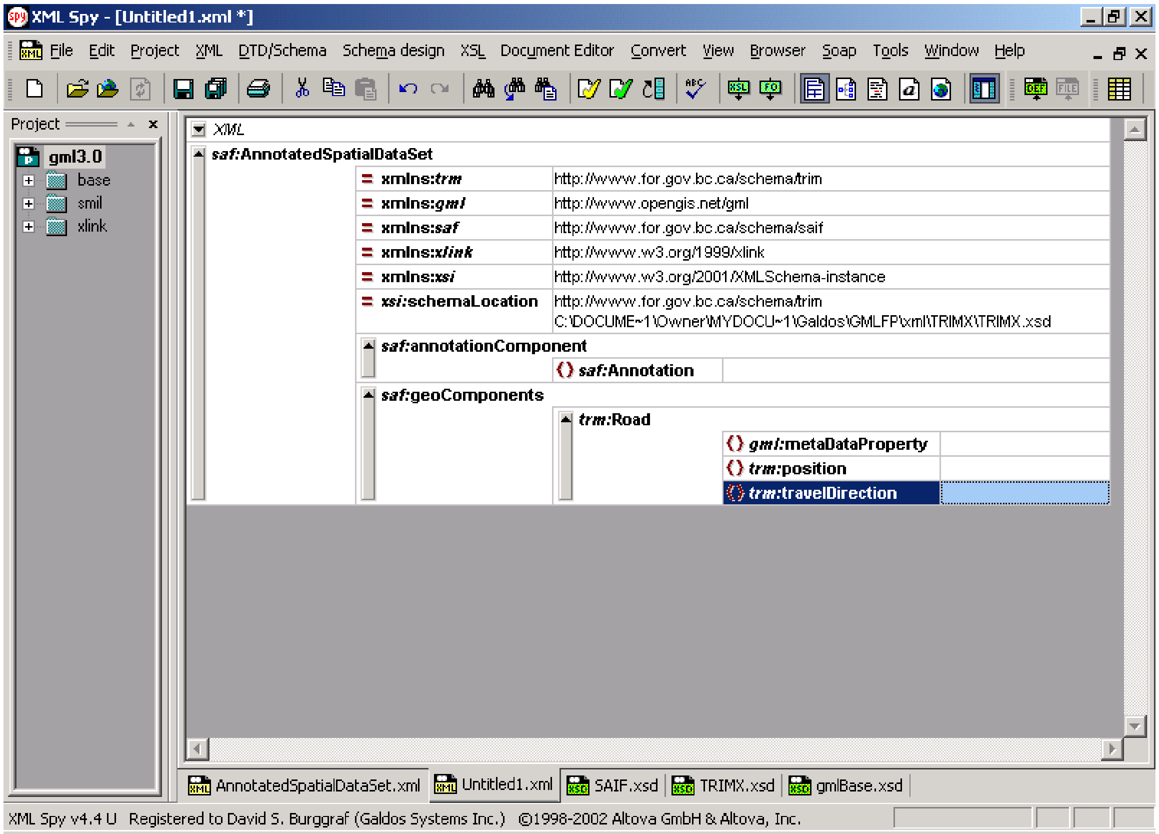
9. Repeat steps 6 to 8 to add new properties of `saf:AnnotatedSpatialDataSet` such as

Step 9 can be performed later if desired. The property `saf:geoComponents` (which is substitutable for `gml:featureMembers`) is added to the instance document as well as the value `trm:Road` (which is a TRIM feature) as shown in Figure 8-11.

To add properties to the TRIM feature:

10. Select `trm:Road` and click the associated down arrow to expand the feature in question.

This displays all mandatory properties of the feature. It does not display any of the optional properties of the feature. As shown in Figure8-12, the optional property `trm:travelDirection` is added to the feature. Notice that there is a box to the right of each property of the `trm:Road` feature.


XML Spy - [Untitled1.xml *]
File Edit Project XML DTD/Schema Schema design XSL Document Editor Convert View Browser Soap Tools Window Help
Project
gml3.0
base
smil
xlink
XML
AnnotatedSpatialDataSet.xml
Untitled1.xml
SAIF.xsd
TRIMX.xsd
gmlBase.xsd
XML Spy v4.4 U Registered to David S. Burggraf (Galdos Systems Inc.) ©1998-2002 Altova GmbH & Altova, Inc.

saf:AnnotatedSpatialDataSet

= xmlns:trm	http://www.for.gov.bc.ca/schema/trim
= xmlns:gml	http://www.opengis.net/gml
= xmlns:saf	http://www.for.gov.bc.ca/schema/saif
= xmlns:xlink	http://www.w3.org/1999/xlink
= xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
= xsi:schemaLocation	http://www.for.gov.bc.ca/schema/trim C:\DOCUME~1\Owner\MYDOCU~1\Galdos\GMLFP\xml\TRIMX\TRIMX.xsd

saf:annotationComponent
- saf:Annotation

saf:geoComponents
- trm:Road
  - gml:metaDataProperty
  - trm:position
  - trm:travelDirection

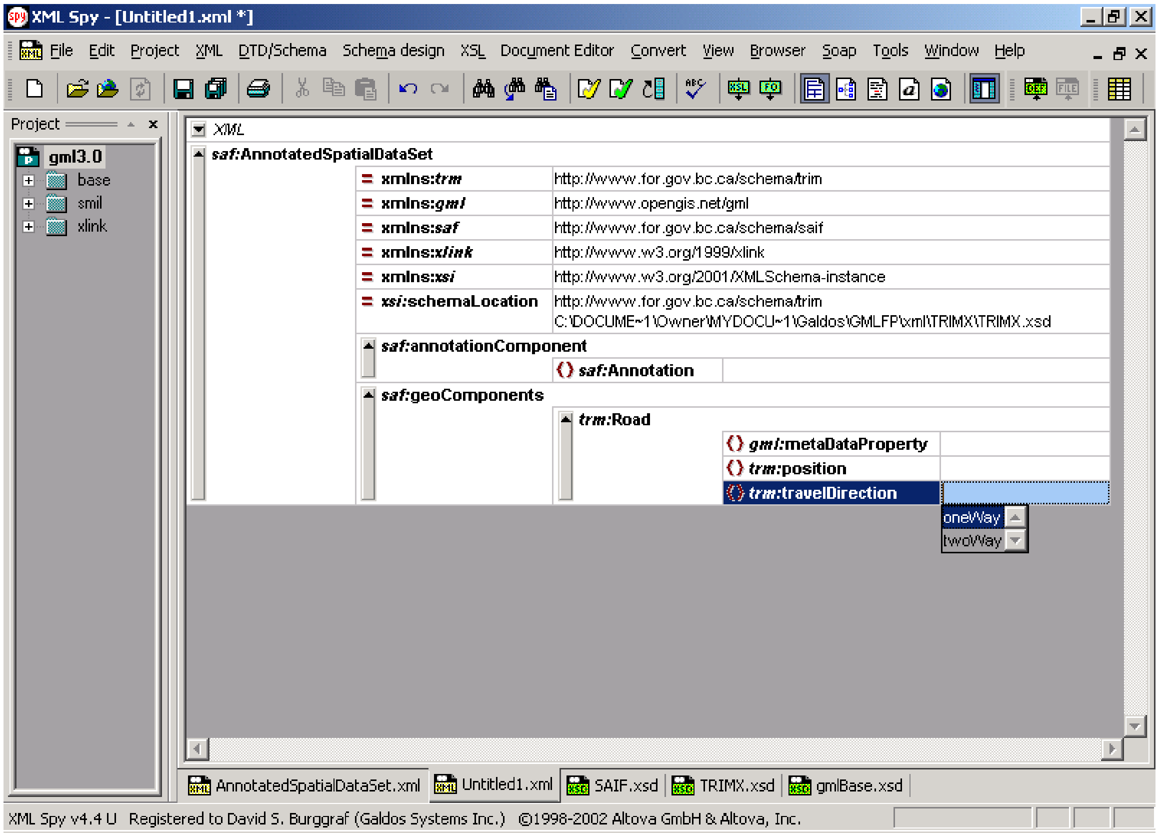
Figure 8-12: Feature Properties

11. Double-click inside this box to display a drop-down list of different values.

Note that nothing appears in this box if you select a complex-valued property (such as `gml:metaDataProperty`). A drop-down list appears if you select properties with enumerative data types (such as `trm:travelDirection`). This is shown in Figure 8-13.

12. Select the desired value from the drop-down list.

This adds values to the enumerative property.


XML Spy - [Untitled1.xml *]
File Edit Project XML DTD/Schema Schema design XSL Document Editor Convert View Browser Soap Tools Window Help
Project
gml3.0
base
smil
xlink
XML
AnnotatedSpatialDataSet.xml
Untitled1.xml
SAIF.xsd
TRIMX.xsd
gmlBase.xsd
XML Spy v4.4 U Registered to David S. Burggraf (Galdos Systems Inc.) ©1998-2002 Altova GmbH & Altova, Inc.

saf:AnnotatedSpatialDataSet

xmlns:trm	http://www.for.gov.bc.ca/schema/trim
xmlns:gml	http://www.opengis.net/gml
xmlns:saf	http://www.for.gov.bc.ca/schema/saif
xmlns:xlink	http://www.w3.org/1999/xlink
xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation	http://www.for.gov.bc.ca/schema/trim C:\DOCUME~1\Owner\MYDOCU~1\Galdos\GMLFP\xml\TRIMX\TRIMX.xsd

saf:annotationComponent
- saf:Annotation

saf:geoComponents
- trm:Road
  - gml:metaDataProperty
  - trm:position
  - trm:travelDirection
    - oneWay
    - twoWay

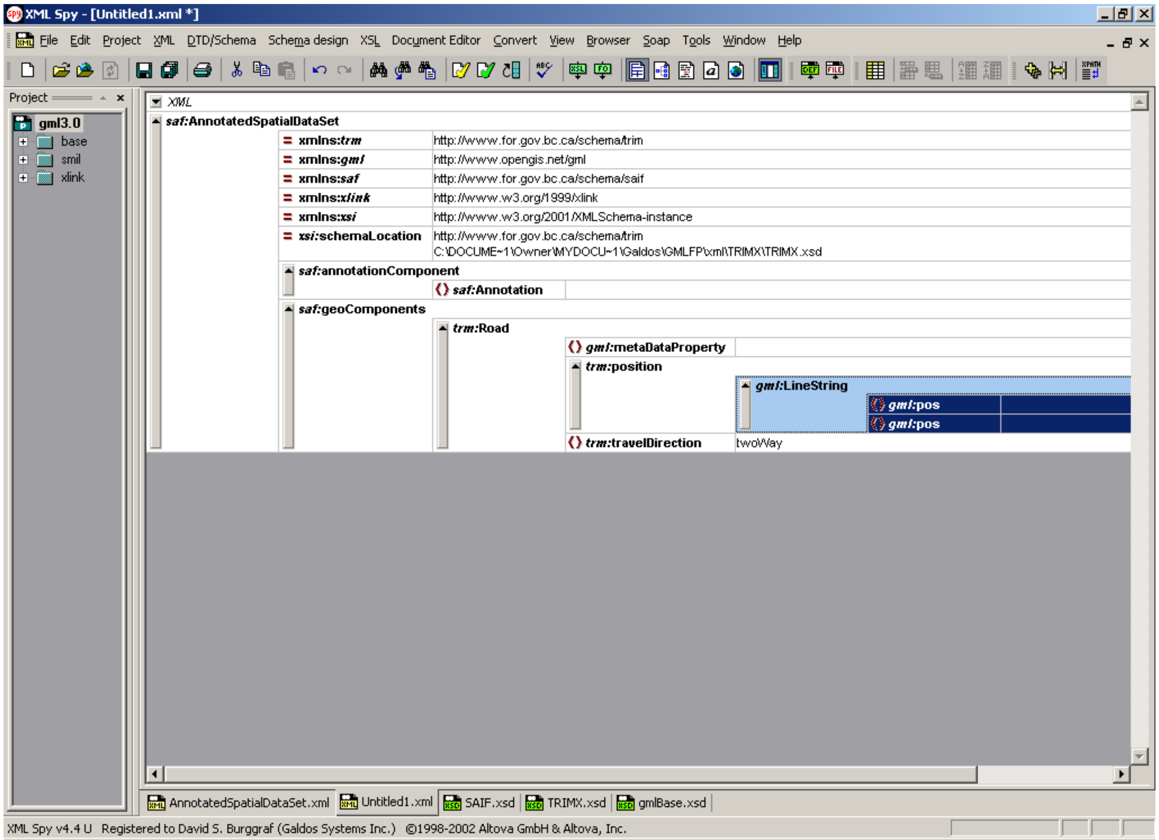
Figure 8-13: Enumerative Properties

To view the associated schema at any time, right-click anywhere on the grid window (where the instance is displayed) and select **Go to Definition**. This can be useful to check for optional properties or verify that you understand the associated content model.

Adding values for Complex-Valued or Text Properties

You can enter values in the text box to the right of the property. For example, you can complete the geometry of the `trm:position` property of the `trm:Road` by doing the following:

13. Repeat the above-mentioned pattern for creating properties until the `gml:pos` properties appear, as shown in Figure 8-14.


XML Spy - [Untitled1.xml *]
File Edit Project XML DTD/Schema Schema design XSL Document Editor Convert View Browser Soap Tools Window Help
Project
gml3.0
base
smil
xlink
AnnotatedSpatialDataSet.xml
Untitled1.xml
SAIF.xsd
TRIMX.xsd
gmlBase.xsd

XML

saf:AnnotatedSpatialDataSet

= xmlns:trm	http://www.for.gov.bc.ca/schema/trim
= xmlns:gml	http://www.opengis.net/gml
= xmlns:saf	http://www.for.gov.bc.ca/schema/saif
= xmlns:xlink	http://www.w3.org/1999/xlink
= xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
= xsi:schemaLocation	http://www.for.gov.bc.ca/schema/trim C:\DOCUME~1\Owner\MYDOCU~1\Galdos\GMLFP\xml\TRIMX\TRIMX.xsd

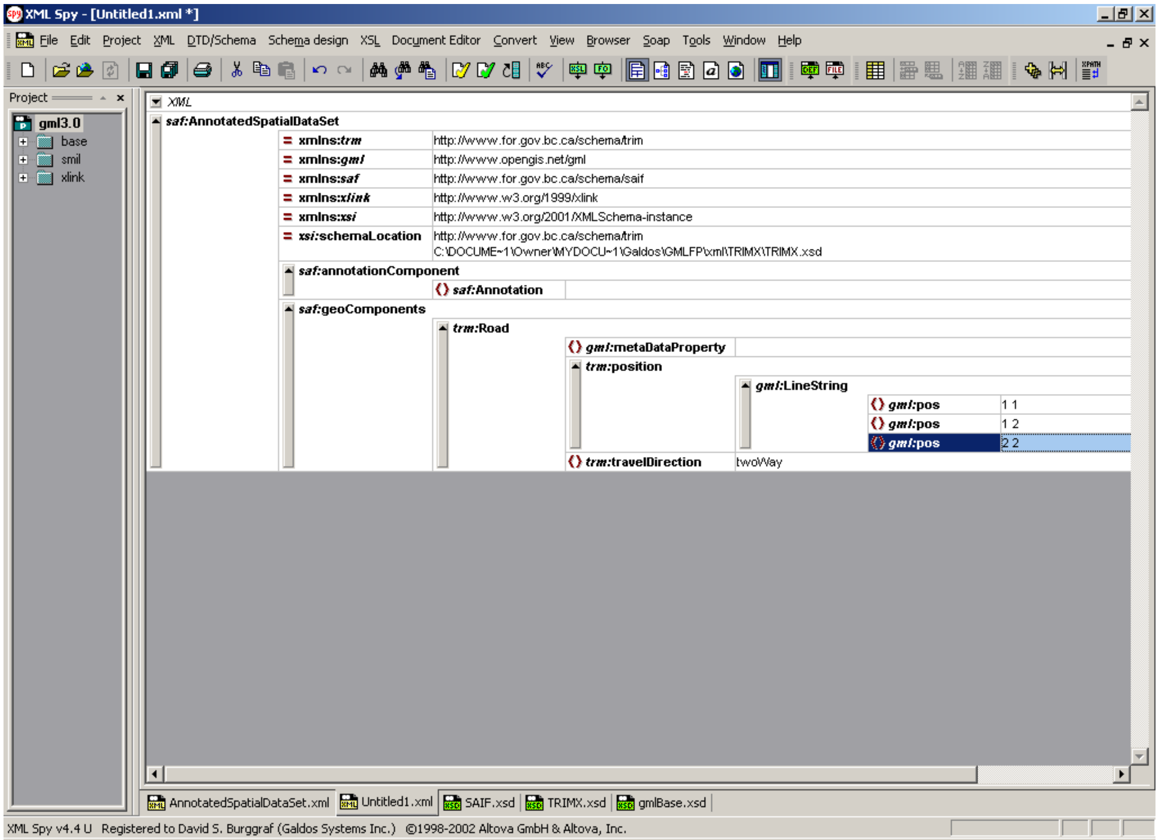
saf:annotationComponent
- saf:Annotation

saf:geoComponents
- trm:Road
  - gml:metaDataProperty
  - trm:position
    - gml:LineString
      - gml:pos
      - gml:pos
  - trm:travelDirection twoWay

XML Spy v4.4 U Registered to David S. Burggraf (Galdos Systems Inc.) ©1998-2002 Altova GmbH & Altova, Inc.

Figure 8-14: Expanding and entering geometry properties

14. Enter values of the `gml:pos` elements (add more `gml:pos` elements if necessary) by typing the values in the entry box to the right of the appropriate element as shown in Figure 8-15.


XML Spy - [Untitled1.xml *]
File Edit Project XML DTD/Schema Schema design XSL Document Editor Convert View Browser Soap Tools Window Help
Project
gml3.0
base
smil
xlink
AnnotatedSpatialDataSet.xml
Untitled1.xml
SAIF.xsd
TRIMX.xsd
gmlBase.xsd
XML Spy v4.4 U Registered to David S. Burggraf (Galdos Systems Inc.) ©1998-2002 Altova GmbH & Altova, Inc.

```
XML
saf:AnnotatedSpatialDataSet
    xmlns:trm            http://www.for.gov.bc.ca/schema/trim
    xmlns:gml            http://www.opengis.net/gml
    xmlns:saf            http://www.for.gov.bc.ca/schema/saif
    xmlns:xlink          http://www.w3.org/1999/xlink
    xmlns:xsi            http://www.w3.org/2001/XMLSchema-instance
    xsi:schemaLocation   http://www.for.gov.bc.ca/schema/trim
                         C:\DOCUME~1\Owner\MYDOCU~1\Galdos\GMLFP\xml\TRIMX\TRIMX.xsd
    saf:annotationComponent
                         saf:Annotation
    saf:geoComponents
                         trm:Road
                              gml:metaDataProperty
                              trm:position
                                   gml:LineString
                                        gml:pos   1 1
                                        gml:pos   1 2
                                        gml:pos   2 2
                              trm:travelDirection   twoWay
```

Figure 8-15: Entering feature geometry coordinates

Note that you can check if your instance document is well formed at any time by selecting **XML | Check well-formedness** or by pressing F7. You can also check if the instance document is valid relative to your GML Application Schema by selecting **XML | Validate** or by pressing F8.

Text View of Sample Instance Document

Listing 6 shows what the sample instance document should look like. Note that many of the elements are empty. These can be completed in the **View | Enhanced-Grid View** by repeating the above-mentioned procedures for entering features and properties. You can also work directly in the **View | Text View**, if you are familiar with XML. Check well formedness and validate at periodic intervals. This is much easier than debugging a complex instance document at a latter stage.

Listing 8-1: Example of a GML Instance

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by David S. Burggraf (Galdos Systems Inc.) --
>
<saf:AnnotatedSpatialDataSet xmlns:saf="http://www.for.gov.bc.ca/schema/saif"
xmlns:trm="http://www.for.gov.bc.ca/schema/trim" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.for.gov.bc.ca/schema/trim TRIMX.xsd
http://www.for.gov.bc.ca/schema/saif SAIF.xsd">
```

```
<saf:annotationComponent>
    <saf:Annotation>
        <saf:textOrSymbol>
            <saf:TextLine>
                <saf:qualifier>positionApproximate</saf:qualifier>
                <saf:position>
                    <gml:Point srsName=“urn:epsg:v6.1:coordinateReferenceSystem:4267”>
                        <gml:pos dimension="3">1111 2222 3333</gml:pos>
                    </gml:Point>
                </saf:position>
                <saf:text>James</saf:text>
            </saf:TextLine>
        </saf:textOrSymbol>
        <saf:spatialReferencing>
            <saf:SpatialReferencing>
                <saf:positioningMethod>differentialGPS</saf:positioningMethod>
            </saf:SpatialReferencing>
        </saf:spatialReferencing>
    </saf:Annotation>
</saf:annotationComponent>
<saf:geoComponents>
    <trm:Road>
        <gml:metaDataProperty>
            <trm:FeatureMetaData>
                <trm:updateHistory>
                    <trm:UpdateHistory>
                        <trm:admissionHistory>
                            <trm:date>2000-12-13</trm:date>
                            <trm:reasonForChange>additionModified</trm:reasonForChange>
                        </trm:admissionHistory>
                    </trm:UpdateHistory>
                </trm:updateHistory>
                <trm:exchangeInformation>
                    <trm:ExchangeInformation>
                        <trm:confidenceLevel>levelThree</trm:confidenceLevel>
                        <trm:confidenceReason>unknown</trm:confidenceReason>
                        <trm:dataSupplier>
                            <trm:DataSupplier>
                                <trm:category>category goes here</trm:category>
                                <trm:detail>details go here</trm:detail>
                            </trm:DataSupplier>
                        </trm:dataSupplier>
                        <trm:dataCaptureMethod>differentialGPS</trm:dataCaptureMethod>
                        <trm:dataAccuracy>
                            <trm:CoordinateAccuracy>
                                <trm:percentile>95</trm:percentile>
                                <trm:time uom="urn:x-si:v1999:uom:second">1012</trm:time>
                            </trm:CoordinateAccuracy>
                        </trm:dataAccuracy>
                        <trm:zValueDerived>unknown</trm:zValueDerived>
                    </trm:ExchangeInformation>
                </trm:exchangeInformation>
                <trm:exchangeFeature>true</trm:exchangeFeature>
            </trm:FeatureMetaData>
        </gml:metaDataProperty>
        <trm:position>
            <gml:LineString>
                <gml:pos>1 1</gml:pos>
                <gml:pos>1 2</gml:pos>
                <gml:pos>2 2</gml:pos>
            </gml:LineString>
        </trm:position>
    </trm:Road>
    <trm:AirfieldComposite>
        <trm:featureComponent xlink:href="#A1"/>
        <trm:featureComponent xlink:href="#A2"/>
```

```
      </trm:AirfieldComposite>
      <trm:Airfield gml:id="A1">
          <gml:metaDataProperty>
              <trm:FeatureMetaData>
                  <trm:updateHistory>
                      <trm:UpdateHistory>
                          <trm:admissionHistory>
                              <trm:date>2000-12-13</trm:date>
                              <trm:reasonForChange>additionModified</trm:reasonForChange>
                          </trm:admissionHistory>
                      </trm:UpdateHistory>
                  </trm:updateHistory>
                  <trm:exchangeInformation>
                      <trm:ExchangeInformation>
                          <trm:confidenceLevel>levelThree</trm:confidenceLevel>
                          <trm:confidenceReason>unknown</trm:confidenceReason>
                          <trm:dataSupplier>
                              <trm:DataSupplier>
                                  <trm:category>category goes here</trm:category>
                                  <trm:detail>details go here</trm:detail>
                              </trm:DataSupplier>
                          </trm:dataSupplier>
                          <trm:dataCaptureMethod>differentialGPS</trm:dataCaptureMethod>
                          <trm:dataAccuracy>
                              <trm:CoordinateAccuracy>
                                  <trm:percentile>95</trm:percentile>
                                  <trm:time uom="urn:x-si:v1999:uom:second">1012</trm:time>
                              </trm:CoordinateAccuracy>
                          </trm:dataAccuracy>
                          <trm:zValueDerived>unknown</trm:zValueDerived>
                      </trm:ExchangeInformation>
                  </trm:exchangeInformation>
                  <trm:exchangeFeature>true</trm:exchangeFeature>
              </trm:FeatureMetaData>
          </gml:metaDataProperty>
          <trm:positionQualifier>positionApproximate</trm:positionQualifier>
          <trm:position>
              <gml:OrientableCurve orientation="+"
srsName=“urn:epsg:v6.1:coordinateReferenceSystem:4267”>
                  <gml:baseCurve>
                      <gml:LineString>
                          <gml:pos dimension="3">1 2 3</gml:pos>
                          <gml:pos dimension="3">4 5 6</gml:pos>
                      </gml:LineString>
                  </gml:baseCurve>
              </gml:OrientableCurve>
          </trm:position>
          <trm:airfieldType>airport</trm:airfieldType>
      </trm:Airfield>
      <trm:Airfield gml:id="A2">
          <gml:metaDataProperty>
              <trm:FeatureMetaData>
                  <trm:updateHistory>
                      <trm:UpdateHistory>
                          <trm:admissionHistory>
                              <trm:date>2001-10-31</trm:date>
                              <trm:reasonForChange>additionNew</trm:reasonForChange>
                          </trm:admissionHistory>
                      </trm:UpdateHistory>
                  </trm:updateHistory>
                  <trm:exchangeInformation>
                      <trm:ExchangeInformation>
                          <trm:confidenceLevel>levelTwo</trm:confidenceLevel>
                          <trm:confidenceReason>unknown</trm:confidenceReason>
                          <trm:dataSupplier>
                              <trm:DataSupplier>
```

```
                                  <trm:category>category goes here</trm:category>
                                  <trm:detail>details go here</trm:detail>
                              </trm:DataSupplier>
                          </trm:dataSupplier>
                          <trm:dataCaptureMethod>monoRestitution </trm:dataCaptureMethod>
                          <trm:dataAccuracy>
                              <trm:CoordinateAccuracy>
                                  <trm:percentile>95</trm:percentile>
                                  <trm:time uom="urn:x-si:v1999:uom:second">1234</trm:time>
                              </trm:CoordinateAccuracy>
                          </trm:dataAccuracy>
                          <trm:zValueDerived>none</trm:zValueDerived>
                      </trm:ExchangeInformation>
                  </trm:exchangeInformation>
                  <trm:exchangeFeature>true</trm:exchangeFeature>
              </trm:FeatureMetaData>
          </gml:metaDataProperty>
          <trm:positionQualifier>indefinite</trm:positionQualifier>
          <trm:position>
              <gml:OrientableCurve orientation="-"
srsName=“urn:epsg:v6.1:coordinateReferenceSystem:4267”>
                  <gml:baseCurve>
                      <gml:LineString>
                          <gml:pos dimension="3">11 22 33</gml:pos>
                          <gml:pos dimension="3">44 55 66</gml:pos>
                      </gml:LineString>
                  </gml:baseCurve>
              </gml:OrientableCurve>
          </trm:position>
          <trm:airfieldType>airFacility</trm:airfieldType>
      </trm:Airfield>
  </saf:geoComponents>
</saf:AnnotatedSpatialDataSet>
```

8.2 XML Software Components

The term “component” is used for ‘low-level’ Java APIs that are useful for building applications that must accept, validate, and potentially return GML data and schemas.

XML Parsers / Validators

There are many open-source tools available for parsing, reading, writing and validating XML documents. The following articles provide useful information on their relative merits and performance:

- Document models, Part 1: Performance (http://www-106.ibm.com/developerworks/xml/library/x-injava/index.html)
- Java Document Model Usage (Part 2) (http://www-106.ibm.com/developerworks/xml/library/x-injava2/)

The XML parsing and validation tools included in this section are considered the most popular and/or developed within the Java XML community. All of these tools support namespace processing. Emerging XML parser tools such as XML Pull Parser, XOM, and dom4j, are not included as they are considered to be in early stages of development.

Apache Xerces2 Java Parser v2.4.0

http://xml.apache.org/xerces2-j/index.html

Features

- Fully conforming XML Schema processor.
- Supports both Simple API for XML (SAX) 2.0, Document Object Model (DOM) Level 2 Recommendations. Early DOM Level 3 support.
- Exposes Post-Validation Information Set (PSVI) through DOM or SAX.
- Supports JAXP 1.2.
- Modular component design (Xerces Native Interface).
- Based on Galdos' development experience using different XML Schema Parsers, Xerces2 is considered to provide the most exhaustive and correct validation of XML Schema rules.
- Anecdotal evidence suggests that Xerces2 is also the most widely used Java-based XML Schema parser within the GML community.
- Very active open-source development.

Limitations

- Standard issues with open-source: no support, no guarantees.

Licensing/Cost

- Standard open-source licensing / free.

JDOM v0.9beta

http://www.jdom.org/

Features

- A Java representation of an XML document, with a fairly straightforward API, that provides an alternative for manipulating XML Documents to SAX and DOM.
- Much easier to use than DOM, it provides an easy and effective interface for reading, manipulating, and writing XML.

Limitations

- JDOM is not an XML parser. It is a document object model that uses XML parsers (SAX or DOM) to build documents. As such, its performance is not as good as native SAX or DOM implementations.
- Requires a JAXP implementation of SAX or DOM (e.g. Xerces).
- XPath support is not yet complete.
- As it builds an in-memory document object model, memory requirements will be large when working with large documents (as opposed to SAX parsing).
- JDOM is still considered a beta-level product, although it has matured significantly in the past year.
- Standard issues with open-source: no support, no guarantees.

Licensing/Cost

- Standard open-source licensing / free.

XSLT Processors

Apache Xalan-Java version 2.5.0

http://xml.apache.org/xalan-j/index.html

Features

- Implements the W3C Recommendations for XSL Transformations (XSLT) Version 1.0 and the XML Path Language (XPath) Version 1.0.
- Is as an implementation of the TRaX (Transformation API for XML) interfaces, part of the Java API for XML Processing 1.1 Public Review 2. Underlying parser is configurable, with Xerces-Java Version 2 parser as the default.
- XSLT stylesheets transformation instructions may be compiled into Templates to avoid repetitive stylesheet parsing.
- Supports passing of parameters to stylesheets.
- Extensive extension function library.
- Optional XSLT Compiler (XSLTC) can compile XSLT stylesheets into lightweight and portable Java byte codes called translets.
- Uses the Bean Scripting Framework (BSF) to implement Java and script extensions, features EXSLT extensions, nodeset extension, multiple document output extensions and SQL extension.

Limitations

- No early support for XSLT Version 2.0.
- Standard issues with open-source: no support, no guarantees.

Licensing/Cost

- Standard open-source licensing / free.

Saxon v7.4

http://saxon.sourceforge.net/

Features

- Popular XSLT engine created and maintained by one of the leading XSLT developers and authors: Michael Kay.
- Leads the development race among XSLT processors, with early XSLT/XPath 2.0 beta support (version 7.3). This beta release integrates with JDOM.
- Many useful built-in extension functions.
- Excellent user documentation.

Limitations

- Compiled stylesheets are available only in the beta v7.3 release, and even are considered "a little fragile".
- Latest version requires JDK 1.4.

Licensing/Cost

- Standard open-source licensing / free. Subject to the Mozilla Public License Version 1.0.

Data Binding Frameworks

XML data binding for Java is a powerful alternative to XML document models for applications concerned mainly with the data content of documents. For a recent comparison of the two APIs described below, see http://www.javaworld.com/javaworld/jw-12-2001/jw-1228-jaxb.html.

Castor

A recent review of the Castor Data Binding Framework is available online at http://www-106.ibm.com/developerworks/xml/library/x-bindcastor/#resources.

Features

- XML Documents are represented internally using a Java object model.
- Since it abstracts many of the document details, data binding usually requires less memory than a document model approach.
- Provides a marshall (Object to Stream) and unmarshall (Stream to Object) framework for two-way mapping between the Java objects and an XML instance.
- Includes a Java XML Schema Object Model to represent XML Schema structures and types.
- Includes a Source Code Generator package that creates a set of Java classes from an XML application schema, as well as the necessary Class Descriptors used by the marshalling framework to obtain information about the generated classes. Currently the generated source files will need to be compiled.

Limitations

- Castor is best used in a context where GML application schemas are relatively fixed. This is due to the maintenance requirements for the generated Java object model and/or XML mapping files.
- Castor is considered to be in an advanced beta stage. Frequent updates from CVS daily snapshots are advisable for the most recent bug fixes.
- Castor is likely still not fully compliant with XML Schema, such that customization may be required to support all XML Schema structures used in GML.
- Our experience indicates that the Castor development cycle is relatively slow, meaning that the one bug that you really need fixed receives no attention.

Licensing/Cost

- Standard open-source licensing / free. Subject to the Mozilla Public License Version 1.0.

JAXB v1.0

http://java.sun.com/xml/jaxb/users-guide/index.html

Features

- In addition to the functionality described in Castor, JAXB also supports the validation of the Java representation of an XML document against schema constraints.
- JAXB is being developed through the Java Community Process (http://jcp.org/en/jsr/detail?id=031)

Limitations

- As with Castor, JAXB is best used when GML application schemas are relatively fixed, since mapping customization will almost always be necessary to resolve conflicts that could occur during derived-class generation, add Java-specific capabilities/info to the generated Java classes, and allow the application to adjust the binding to fit its needs.
- The JAXB specification and implementation have been very slow in coming, leading to Castor's popularity within the XML community. The production version 1.0 is not expected until Q1 CY2003.
- The current JAXB release 0.75 does not fully support XML Schema.

Licensing/Cost

- Sun Community Source Licensing / free.

8.3 GML Tools

GML Schema Parsers

GML4J v1.03 beta

GML4J is an experimental Java API for scanning GML-2 documents. Galdos made this API available as an open-source project in May 2001. The project page is at SourceForge https://sourceforge.net/projects/gml4j/. It has had 2600 downloads thus far, mostly by people eager to familiarize themselves with the programming aspect of GML.

Features

- It consists of two components: GML Schema Parser and GML Objects.
- GML Schema Parser is used to extract element-typing information from GML application schemas referenced in GML documents.
- GML Schema Parser is designed to work with any valid GML-2 application schema.
- GML Objects is a GML object model that lets users scan GML data as features, feature collections, geometries, geometry collections, feature properties, etc., instead of mere XML elements and attributes.

Limitations

- GML Schema Parser was designed to work for the most-frequently used XML Schema constructs, and therefore may not work well for the less-frequently used ones.
- GML Schema Parser is currently tightly coupled with GML Objects, which prevents independent schema management.
- GML Objects are tightly coupled with DOM, i.e. they depend on having the GML document pre-parsed to DOM.
- GML Objects do not support data updates.

Licensing/Cost

- Apache-style license/Free

Galdos Systems Inc.™ GMLSchemaParser

http://www.galdosinc.com/

GML Schema Parser is a Java library that parses GML-2 application schemas to provide information about GML features, geometries and other objects defined in the schemas. Functionality-wise it is similar to the GML Schema Parser part of GML4J, but unlike GML4J it is designed for use within a production environment.

Features

- Compiles schemas into a native schema object model.
- Caches schemas for better performance.
- Provides typing information about features, feature collections, geometries, geometry collections, feature properties, etc., including their type hierarchies.
- Provides access to feature properties and their types.

Limitations

- Does not support annotations and ID constraints.

Licensing/Cost

- Contact Galdos for details.

GeoTools / GeoServer Project

https://sourceforge.net/projects/geotools/

GeoTools is an open-source Java mapping toolkit for developing interactive geographical maps, which now includes the GeoServer project. GeoServer is a Java/PostGIS implementation of the 1.0 Web Feature Server specification. The GeoServer project team have allegedly developed a SAX-based GMLSchemaParser. Contact the GeoTools or GeoServer project managers for details.

List of Commercial XML/GML Tools

- **Breeze™ XML Binder v3.0**

http://www.breezefactor.com/overview.html

- **Oracle™ XML Parser for Java v2; XML Schema Processor for Java v9.2.0.4.0**

http://otn.oracle.com/docs/tech/xml/xdk_java/doc_library/Production9i/index.html

- **Oracle™ XSLT Processor**

http://otn.oracle.com/tech/xml/content.html

- **Oracle™ XML Class Generator for Java v9.2.0.4.0**

http://technet.oracle.com/docs/tech/xml/xdk_java/doc_library/Production9i/index.html

- **<oxygen/> XML Editor 2.0.1**

http://www.oxygenxml.com/index.html

- **Safe Software FME™ / FMEObjects™**

http://www.safe.com

- **Snowflake Software GoLoader**

http://www.snowflakesoft.co.uk/