Open Geospatial Consortium Inc.

Date: 3 May 2005

Reference number of this OpenGIS® project document: OGC 04-095

Version: 1.1.0

Category: OpenGIS® Implementation Specification

Status: Adopted Specification

Editor: Panagiotis A. Vretanos

OpenGIS® Filter Encoding Implementation Specification

Document type: OpenGIS® Publicly Available Implementation Specification
Document stage: Final
Document language: English

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

i. Preface

This document was originally part of version 0.0.10 of the Web Feature Server (WFS) Implementation Specification [14]. It was decided that the contents of this specification would be put into their own document since the Filter encoding described herein can be used by a broad range of services that require the ability to express predicates in XML. Such services include Web Feature Service, Web Coverage Service, Gazetteer, Web Registries, etc…

The predicate language defined in this document is based on the productions for the Common Query Language (CQL) found in the OpenGIS® Catalog Interface Implementation Specification V1.0 [2]. The spatial operators included in this specification are derived from [2] and from the OpenGIS® Simple Features Specification For SQL, Revision 1.1[2].

ii. Submitting organizations

The following companies submitted this specification to the OGC as a Request for Comment:

CubeWerx Inc.
Edric Keighan
200 Rue Montcalm, Suite R-13
Hull, Quebec
Canada J8Y 3B5
ekeighan@cubewerx.com

Intergraph Corp.
John T. Vincent
1881 Campus Commons Drive
Reston, VA 20191
U.S.A
jtvincen@intergraph.com

IONIC Software
Serge Margoulies
128 Avenue de l'Observatoire
B-4000 LIEGE
Belgium
Serge.Margoulies@ionicsoft.com

Laser-Scan Ltd.
Peter Woodsford
101 Cambridgbe Science Park
Milton Road
Cambridge CB4 0FY
U.K.
peterw@lsl.co.uk

iii. Submission contact points

All questions regarding this submission should be directed to the Editor:

Panagiotis (Peter) A. Vretanos
CubeWerx, Inc.
200 Rue Montcalm, Suite R-13
Hull, Quebec J8Y 3B5 CANADA
+1 416 701 1985
pvretano@cubewerx.com

Additional contributors

Rob Atkinson (Social Change Online) rob@socialchange.net.au
Craig Bruce (CubeWerx) csbruce@cubwerx.com
Paul Daisey (U.S. Census) Paul.W.Daisey@census.gov
Ignacio Guerrero (Intergraph) IGuerrer@ingr.com
Edric Keighan (CubeWerx) ekeighan@cubewerx.com
Marin Kyle (Galdos System Inc.) mkyle@galdosinc.com
Ron Lake (Galdos Systems Inc.) rlake@galdosinc.com
Jeff Lansing (Polexis) jeff@polexis.com
Seb Lessware (Laser-Scan Ltd.) sebl@lsl.co.uk
Marwa Mabrouk (ESRI) mmabrouk@esri.com
Serge Margoulies (Ionic) Serge.Margoulies@ionicsoft.com
Richard Martell (Galdos Systems Ltd.) rmartell@galdosinc.com
Aleksander Milanovic
Dimitri Monie (Ionic) dimitri.monie@ionicsoft.com
Paul Pilkington (Laser-Scan Ltd.) paulpi@lsiva.com
Keith Pomakis (CubeWerx) pomakis@cubewerx.com
Lou Reich (NASA) louis.i.reich@gsfc.nasa.gov
Carl Reed (Open GIS Consortium) creediii@mindspring.com
Martin Schaefer (Cadcorp Ltd.) martins@cadcorpdev.co.uk
Bernard Snyers (Ionic) Bernard.Snyers@ionicsoft.com
James T. Stephens (Lockheed Martin) james.t.stephens@lmco.com
Glenn Stowe (CubeWerx) gstowe@cubwerx.com
Milan Trninic (Galdos Systems Inc.) mtrninic@galdosinc.com
Arliss Whitesize (BAE Systems) Arliss.Whiteside@baesystems.com
Tim Wilson (Galdos System Inc.) twilson@gladosinc.com
Peter Woodsford (Laser-Scan Ltd.) peterw@lsl.co.uk

iv. Revision history

1.1.0	Edit for V1.1.0. Added SORTBY encoding. Abstracted id encoding to deprecate fid and use gml:id instead. Major rebuild of the filter capabilities document to make if more extensible and add support for geometry operands. Update based on comment in 03-082.
0.0.0	Address RFC comments.
0.0.7	Reformat document for RFC Re-Submission; Add use of XPath expressions for refering to complex attributes.
0.0.6	Prepare for RFC Submission
0.0.5	Define a capabilities section.

0.0.4	Add support for arithmetic expressions.
0.0.3	Add support for functions.
0.0.2	Correct typographic errors.
0.0.1	First version derived from the Open GIS Web Feature Server Specification [14].

v. Changes to the OpenGIS® Abstract Specification

No further revisions to the OGC Abstract Specification are required. This specification represents an implementation of a predicate language required to support the discover, query of geospatial resources as discussed in clauses 2.2 and 3.2 of Topic 13, “Catalog Services”.

Foreword

Attention is drawn to the possibility that some of the elements of this standard may be the subject of patent rights. Open GIS Consortium Inc. shall not be held responsible for identifying any or all such patent rights. However, to date, no such rights have been claimed or identified.

This version of the specification cancels and replaces all previous versions.

Normative annexes

Annex B is normative..

Introduction

This document defines an XML encoding for filter expressions based on the BNF definition of the OpenGIS® Common Catalog Query Language as described in the OpenGIS® Catalog Interface Implementation Specification, Version 1.0 [2].

Filter Encoding Implementation Specification

1 Scope

A *filter expression* is a construct used to constrain the property values of an object type for the purpose of identifying a subset of object instances to be operated upon in some manner.

The intent of this document is to describe an XML encoding of the OGC Common Catalog Query Language (CQL) [2] as a system neutral representation of a query predicate. Using the numerous XML tools available today, such an XML representation can be easily validated, parsed and then transformed into whatever target language is required to retrieve or modify object instances stored in some a persistent object store. For example, an XML encoded filter could be transformed into a WHERE clause for a SQL SELECT statement to fetch data stored in a SQL-based relational database. Similarly, and XML encoded filter expression could be transformed into an XPath or XPointer expression for fetching data from XML documents.

A large class of OpenGIS® web based service require the ability to express filter expressions in XML.

Relation to other OGC web services

The filter encoding described in this document is a common component that can be used by a number of OGC web services. Any service that requires the ability to query objects from a web-accessible repository can make use of the XML filter encoding described in this document. For example, a web feature service may use the XML filter encoding in a *GetFeature* operation to define query constraints. Other services based of the web feature service, such as **Gazetteer** or the **Web Registry Service**, could also make use of this filter encoding.

2 Conformance

Conformance with this specification shall be checked using all the relevant tests specified in Annex B (normative). The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in ISO 19105: Geographic information — Conformance and Testing.

3 Normative references

[1] Bradner, Scott, "RFC 2119 Key words for use in RFCs to Indicate Requirement Levels," March 1997, ftp://ftp.isi.edu/in-notes/rfc2119.txt .

[2] Percivall, George, ed., "The OpenGIS® Abstract Specification, Topic 12: Service Architecture", 2002

[3] Kottman, C., ed., "The OpenGIS® Abstract Specification, Topic 13: Catalog Services", Version 4, 1999

[4] Nebert, Douglas, Whiteside, Arliss (eds), "OpenGIS® Implementation Specification #04-021r2: Catalog Services Specification, Version 2.0"

[5] OpenGIS® Implementation Specification #99-049, "OpenGIS® Simple Features Specification For SQL, Revision 1.1", May 1999

[6] Vretanos, Panagiotis (ed.), "OpenGIS® Implementation Specification #01-067: Filter Encoding Implementation Specification", May 2001

[7] Bray, Paoli, Sperberg-McQueen, eds., "Extensible Markup Language (XML) 1.0", 2nd edition, October 2000, W3C Recommendation, http://www.w3.org/TR/2000/REC-xml.

[8] Beech, David, Maloney, Murry, Mendelson, Noah, Thompson, Harry S., "XML Schema Part 1: Structures", May 2001, W3C Recommendation, http://www.w3c.org/TR/xmlschema-1.

[9] Bray, Hollander, Layman, eds., "Namespaces In XML", January 1999, W3C Recommendation, http://www.w3.org/TR/2000/REC-xml-names.

[10] Clark, James, DeRose, Steve, "XML Path Language (XPATH), Version 1.0", November 1999, W3C Recommendation, http://www.w3c.org/TR/XPath.

[11] Berners-Lee, T., Fielding, N., and Masinter, L., "Uniform Resource Identifiers (URI): Generic Syntax", IETF RFC 2396, http://www.ietf.org/rfc/rfc2396.txt.

[12] Cox S., Daisey P., Lake, R., Portele C., Whiteside A. (eds.), "OpenGIS Implementation Specification #02-023r4: OpenGISâ Geography Markup Language (GML) Implementation Specification, version 3.1.1", January 2005

[13] Fielding et. al., "Hypertext Transfer Protocol - HTTP/1.1," IETF RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt.

4 Terms and definitions

4.1
operation
specification of a transformation or query that an object may be called to execute [2]

4.2
interface
a named set of operations that characterize the behavior of an entity [2]

4.3
service
a distinct part of the functionality that is provided by an entity through interfaces [2]

4.4
service instance
an actual implementation of a service; service instance is synonymous with server

4.5
client
a software component that can invoke an operation from a server

4.6
request
an invocation by a client of an operation.

4.7
response
the result of an operation returned from a server to a client.

4.8
capabilities XML
service-level metadata describing the operations and content available at a service instance

4.9
spatial reference system
as defined in ISO19111

4.10
opaque
not visible, accessible or meaningful to a client application

4.11
filter expression processor
a component of a system that process a filter expression as defined in this specification

4.12
property
a facet or attribute or an object referenced by a name

4.13
function
a function is a named procedure that accepts zero or more arguments, performs a distinct computation and returns a single result

5 Conventions

5.1 Normative verbs

In the sections labeled as normative, the key words "**must**", "**must not**", "**required**", "**shall**", "**shall not**", "**should**", "**should not**", "**recommended**", "**may**", and "**optional**" in this document are to be interpreted as described in Internet RFC 2119 [1].

5.2 Abbreviated terms

CQL	Common Catalog Query Language
EPSG	European Petroleum Survey Group
GIS	Geographic Information System
GML	Geography Markup Language
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
MIME	Multipurpose Internet Mail Extensions
OGC	Open GIS Consortium
OWS	OGC Web Service
URL	Uniform Resource Locator
WFS	Web Feature Service
XML	Extensible Markup Language

6 Properties

6.1 Introduction

This specification assumes that general objects are composed of simple and/or complex or aggregate non-geometric properties. This specification further assumes that the properties of an object are mapped to XML elements (or nested sets of elements for complex objects) or XML attributes where the name of the element or attribute is the name of the property being mapped.

6.2 Property names

The **<PropertyName>** element is used to encode the name of any property of an object. The property name can be used in scalar or spatial expressions to represent the value of that property for a particular instance of an object.

Since this specification assumes that objects are encoded in XML, property names must also be valid element or attribute names as described in the Extensible Markup Language (XML) 1.0 [7] specification. In addition, property names may be qualified with a namespace prefix in which case the name must conform to the Namespaces In XML [9] specification. The following definition is taken from clauses 2 & 3, of that document:

Names and Tokens

```
[4]  NCName ::=  (Letter | '_') (NCNameChar)*
/*  An XML Name, minus the ":" */
[5]  NCNameChar ::=  Letter | Digit | '.' | '-' | '_' | CombiningChar | Extender
[6]  QName ::=  (Prefix ':')? LocalPart
[7]  Prefix ::=  NCName
[8]  LocalPart ::=  NCName
```

The definitions of the components Letter, Digit, CombiningChar and Extender are defined in annex B of [7].

Examples

Examples of valid property names are:

Age, Temperature, _KHz, INWATERA_1M.WKB_GEOM

Examples of invalid property names are:

+Domain, 123_SomeName

6.3 Property references

6.3.1 Introduction

Simple properties may be are referenced using their name. However, since objects can also include complex or aggregate non-geometric properties a problem arises concerning how such properties should be referenced in the various places where property references are required in filter expressions.

In order to handle property references in a consistent manner, a filter expression processor **must** use the subset of XPath [10] expressions defined in this document for referencing simple properties and the properties and sub-properties of objects with complex or aggregate non-geometric properties or properties encoded as XML attributes.

6.1.2 XPath expressions

Properties of an object may be mapped, in XML, to elements or attributes of elements. Thus it is required that a filter expression be able to address properties encoded as XML elements or attributes.

The XML Path Language [10] specification is a language for addressing parts of a XML document, or in the case of this specification, for referencing XML elements and attributes within the description of a feature.

This specification does not require that a filter expression processor support the full XPath language. In order to keep the implementation entry cost as low as possible, this specification mandates that a filter expression processor **must** support the following subset of the XPath language:

1. A filter expression processor **must** support *abbreviated relative location* paths.

2. Relative location paths are composed of one or more *steps* separated by the path separator ‘/’.

3. The first step of a relative location path **may** correspond to the root element of the object property being referenced **or** to the root element of the object with the next step corresponding to the root element of the object property being referenced.

4. Each subsequent step in the path **must** be composed of the abbreviated form of the *child::* axis specifier and the name of the object property encoded as the principal node type of *element*. The abbreviated form of the *child::* axis specifier is to simply omit the specifier from the location step.

5. Each step in the path may optionally contain a predicate composed of the predicate delimiters '[' and ']' and a number indicating which child of the context node is to be selected. This allows feature properties that may be repeated to be specifically referenced.

6. The final step in a path may optionally be composed of the abbreviated form of the *attribute::* axis specifier, '@', and the name of an object property encoded as the principal node type of *attribute*.

Example

To practically illustrate the use of XPath expressions for referencing the XML elements and attributes within the description of a feature consider the fictitious complex feature *Person* defined by the following XML Schema document:

```
<?xml version="1.0" ?>
<schema
   targetNamespace="http://www.someserver.com/myns"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   version="1.0">

   <import namespace="http://www.opengis.net/gml"
           schemaLocation="../gml/3.1.0/base/gml.xsd"/>

   <element name="Person" type="myns:PersonType"
            substitutionGroup="gml:_Feature"/>
   <complexType name="PersonType">
      <complexContent>
         <extension base="gml:AbstractFeatureType">
            <sequence>
               <element name="lastName" nillable="true">
                  <simpleType>
                     <restriction base="string">
                        <maxLength value="30"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="firstName" nillable="true">
                  <simpleType>
                     <restriction base="string">
                        <maxLength value="10"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="age" type="integer" nillable="true"/>
               <element name="sex" type="string"/>
               <element name="spouse">
                  <complexType>
                     <attribute ref="xlink:href" use="required" />
                  </complexType>
               </element>
               <element name="location"
                        type="gml:PointPropertyType"
```

```
                        nillable="true"/>
                <element name="mailAddress"
                        type="myns:AddressPropertyType" nillable="true"/>
                <element name="Phone" type="xsd:string"
                        minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
            <attribute name="SIN" type="xsd:ID" use="required"/>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="AddressPropertyType">
      <sequence>
         <element name="Address"
                  type="myns:AddressType" minOccurs="0" />
      </sequence>
   </complexType>
   <complexType name="AddressType">
      <sequence>
         <element name="streetName" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
               </restriction>
            </simpleType>
         </element>
         <element name="streetNumber" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="10"/>
               </restriction>
            </simpleType>
         </element>
         <element name="city" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
               </restriction>
            </simpleType>
         </element>
         <element name="province" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
               </restriction>
            </simpleType>
         </element>
         <element name="postalCode" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="15"/>
               </restriction>
            </simpleType>
         </element>
         <element name="country" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
               </restriction>
            </simpleType>
         </element>
      </sequence>
   </complexType>
</schema>
```

Note that the property *mailAddress* has a complex value given by its type **myns:AddressType**. An example instance of the feature **Person** might be:

```
<?xml version="1.0"?>
<myns:Person
   SIN="111222333"
   xmlns:myns="http://www.someserver.com/myns"
   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/myns Person.xsd">

   <myns:lastName>Smith</myns:lastName>
```

```
    <myns:firstName>Fred</myns:firstName>
    <myns:age>35</myns:age>
    <myns:sex>Male</myns:sex>
    <myns:spouse SIN="444555666" />
    <myns:location>
       <gml:Point><gml:pos>15 15</gml:pos></gml:Point>
    </myns:location>
    <myns:mailAddress>
       <myns:Address>
          <myns:streetName>Main St.</myns:streetName>
          <myns:streetNumber>5</myns:streetNumber>
          <myns:city>SomeCity</myns:city>
          <myns:province>SomeProvince</myns:province>
          <myns:postalCode>X1X 1X1</myns:postalCode>
          <myns:country>Canada</myns:country>
       </myns:Address>
    </myns:mailAddress>
    <myns:phone>416-123-4567</myns:phone>
    <myns:phone>416-890-1234</myns:phone>
</myns:Person>
```

Using XPath [10] expressions, each XML element within the description of a *Person* feature that is the root element of an XML document can be referenced as follows (omitting the namespace qualifiers for the sake of clarity):

Table 1: XPath expressions and property values for *Person* example

XPath Expression in Person element context	Alternate XPath Expression in XML document context	Property Value
lastName	Person/lastName	Smith
firstName	Person/firstName	Fred
age	Person/age	35
sex	Person/sex	Male
spouce/@sSIN	Person/spouce/@SIN	444555666
location	Person/location	<gml:Point> <gml:pos>15,15 </gml:pos> </gml:Point>
mailAddress/Address/streetNumber	Person/mailAddress/Address/streetNumber	5
mailAddress/Address/streetName	Person/mailAddress/Address/streetName	Main St.
mailAddress/Address/city	Person/mailAddress/Address/city	SomeCity
mailAddress/Address/province	Person/mailAddress/Address/province	SomeProvince
mailAddress/Address/postalCode	Person/mailAddress/Address/postalCode	X1X 1X1
mailAddress/Address/country	Person/mailAddress/Address/country	Canada
Person/@SIN	Person/@SIN	111222333
phone[1]	Person/phone[1]	416-123-4567
phone[2]	Person/phone[2]	416-890-1234

Notice that in this instance, each relative location path begins with the root element of the feature property being referenced. This simply corresponds to the name of the feature property. Optionally, each XML element within the description may be referenced with the relative location path beginning with root element of the feature (i.e. the name of the feature type). Thus the *lastName* property could be reference as *Person/lastName*, the *City* element could be referenced as *Person/mailAddress/Address/city* and so on.

Each step of the path is composed of the abbreviated *child::* axis specifier (i.e. the axis specifier *child::* is omitted) and the name of the specified XML element within the description which is of node type *element*.

The cardinality of the <**phone**> element is greater than 1 and *phone* property values appear multiple times. The specific element being referenced is indicated using the predicates *[1]* and *[2]*. The predicate *[1]* is used to indicate the first occurrence of the *Phone* element. The predicate *[2]* is used to indicate the second occurrence of the *phone* element.

In addition, the **SIN**[1] attribute on the **<Person>** and **<spouse>** elements is referenced using the following XPath [10] expressions:

```
Person/@SIN
Person/spouse/@SIN
```

In these cases the final step of the path contains the abbreviated axis specifier *attribute::* (i.e. @) and the node type is *attribute* (i.e. **SIN** in this case).

7 Filter

7.1 Introduction

A filter is any valid predicate expression that can be formed using the elements defined in this specification. The root element **<Filter>** contains the expression which is created by combining the elements defined in this specification.

7.2 Encoding

The root element of a filter expression, <**Filter**>, is defined by the following XML Schema fragment:

```
<xsd:element name="Filter" type="ogc:FilterType"/>
<xsd:complexType name="FilterType">
   <xsd:choice>
      <xsd:element ref="ogc:spatialOps"/>
      <xsd:element ref="ogc:comparisonOps"/>
      <xsd:element ref="ogc:logicOps"/>
      <xsd:element ref="ogc:_Id" maxOccurs="unbounded"/>
   </xsd:choice>
</xsd:complexType>
```

The elements <**logicalOps**>, <**comparisonOps**> and <**spatialOps**> are substitution groups for logical, spatial and comparison operators. Logical operators may be used to combine spatial operators and comparison operators in one filter expression. The **<_Id**> element is the head of a substitution group for object identifiers and is defined in clause 11.

[1] SIN = Social Insurance Number

8 Spatial operators

8.1 Introduction

A spatial operator determines whether its geometric arguments satisfy the stated spatial relationship. The operator evaluates to TRUE if the spatial relationship is satisfied. Otherwise the operator evaluates to FALSE.

8.2 Encoding

The XML encoding for spatial operators is defined by the following XML Schema fragment:

```
<xsd:element name="Equals"
             type="ogc:BinarySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Disjoint"
             type="ogc:BinarySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Touches"
             type="ogc:BinarySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Within"
             type="ogc:BinarySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Overlaps"
             type="ogc:BinarySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Crosses"
             type="ogc:BinarySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Intersects"
             type="ogc:BinarySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Contains"
             type="ogc:BinarySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="DWithin"
             type="ogc:DistanceBufferType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Beyond"
             type="ogc:DistanceBufferType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="BBOX"
             type="ogc:BBOXType" substitutionGroup="ogc:spatialOps"/>

<xsd:complexType name="SpatialOpsType" abstract="true"/>
<xsd:element name="spatialOps" type="ogc:SpatialOpsType" abstract="true"/>

<xsd:complexType name="BinarySpatialOpType">
   <xsd:complexContent>
      <xsd:extension base="ogc:SpatialOpsType">
         <xsd:sequence>
            <xsd:element ref="ogc:PropertyName"/>
            <xsd:choice>
               <xsd:element ref="gml:_Geometry"/>
               <xsd:element ref="gml:Envelope"/>
            </xsd:choice>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

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

<xsd:complexType name="DistanceBufferType">
   <xsd:complexContent>
      <xsd:extension base="ogc:SpatialOpsType">
         <xsd:sequence>
            <xsd:element ref="ogc:PropertyName"/>
            <xsd:element ref="gml:_Geometry"/>
            <xsd:element name="Distance" type="ogc:DistanceType"/>
         </xsd:sequence>
```

```
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="DistanceType" mixed="true">
   <xsd:attribute name="units" type="xsd:anyURI" use="required"/>
</xsd:complexType>
```

As defined in this specification, spatial operators are used to test whether the value of a geometric property, referenced using the name of the property, and a literal geometric value satisfy the spatial relationship implied by the operator. For example, the **<Overlaps>** operator evaluates whether the value of the specified geometric property and the specified literal geometric value spatially overlap. Literal geometric values are expressed using GML [12].

The **<BBOX>** element is defined as a convenient and more compact way of encoding the very common bounding box constraint based on the **gml:Envelope** geometry. It is equivalent to the spatial operation **<Not><Disjoint>** … **</Disjoint></Not>** meaning that the <**BBOX>** operator should identify all geometries that spatially interact with the box. If the optional <**propertyName**> element is not specified, the calling service must determine which spatial property is the spatial key and apply the BBOX operator accordingly. For feature types that has a single spatial property, this is a trivial matter. For feature types that have multiple spatial properties, the calling service either knows which spatial property is the spatial key or the calling service generates an exception indicating that the feature contains multiple spatial properties and the <**propertyName**> element must be specified. Of course a client application always has the options of avoiding the exceptions by calling the **DescribeFeatureType** operation to get a description of the feature type.

The semantics of the other operators **Equals**, **Disjoint**, **Touches**, **Within**, **Overlaps**, **Crosses**, **Intersects**, and **Contains** are defined in subclause 3.2.19.2 of the OpenGIS® Simple Feature Specification for SQL [5].

The spatial operators **DWithin** and **Beyond** test whether the value of a geometric property is within or beyond a specified distance of the specified literal geometric value. Distance values are expressed using the **<Distance>** element. The content of the **<Distance>** element represents the magnitude of the distance and the **units** attribute is used to specify the units of measure. The **units** attribute is of type *anyURI* so that it may be used to reference a units dictionary. For example, the following XML fragment:

```
<Distance unit="http://www.uomdict.com/uom.html#meters">10</Distance>
```

encodes a distance value of 10 meters. The units of measure dictionary should be encoded in GML.

8.3 SRS handling of literal geometries in Filter expressions

The SRS of literal geometries used with Filter expressions shall be handled as follows:

- If the literal geometry SRS matches one of the <**DefaultSRS**> values or <**OtherSRS**> of the queried feature type, then the WFS is free to transform either the literal geometry or feature geometries for purposes of processing the query only. Any matching features must be returned in the requested and supported SRS.

If the literal geometry SRS is unspecified, then the literal geometry SRS shall be interpreted to be equivalent to the <**DefaultSRS**> value of the queried feature type.

- If the literal geometry SRS does not match the <**DefaultSRS**> value, or any <**OtherSRS**> value of the queried feature type, then an exception shall be returned.

9 Comparison operators

9.1 Introduction

A comparison operator is used to form expressions that evaluate the mathematical comparison between two arguments. If the arguments satisfy the comparison then the expression evaluates to TRUE. Otherwise the expression evaluates to FALSE.

9.2 Encoding

The XML encoding for comparison operators is defined by the following XML Schema fragment:

```
<xsd:element name="PropertyIsEqualTo"
             type="ogc:BinaryComparisonOpType"
substitutionGroup="ogc:comparisonOps"/>
<xsd:element name="PropertyIsNotEqualTo"
             type="ogc:BinaryComparisonOpType"
substitutionGroup="ogc:comparisonOps"/>
<xsd:element name="PropertyIsLessThan"
             type="ogc:BinaryComparisonOpType"
substitutionGroup="ogc:comparisonOps"/>
<xsd:element name="PropertyIsGreaterThan"
             type="ogc:BinaryComparisonOpType"
substitutionGroup="ogc:comparisonOps"/>
<xsd:element name="PropertyIsLessThanOrEqualTo"
             type="ogc:BinaryComparisonOpType"
substitutionGroup="ogc:comparisonOps"/>
<xsd:element name="PropertyIsGreaterThanOrEqualTo"
             type="ogc:BinaryComparisonOpType"
substitutionGroup="ogc:comparisonOps"/>
<xsd:element name="PropertyIsLike"
             type="ogc:PropertyIsLikeType" substitutionGroup="ogc:comparisonOps"/>
<xsd:element name="PropertyIsNull"
             type="ogc:PropertyIsNullType" substitutionGroup="ogc:comparisonOps"/>
<xsd:element name="PropertyIsBetween"
             type="ogc:PropertyIsBetweenType" substitutionGroup="ogc:comparisonOps"/>

<xsd:complexType name="ComparisonOpsType" abstract="true"/>
<xsd:element name="comparisonOps" type="ogc:ComparisonOpsType" abstract="true"/>

<xsd:complexType name="BinaryComparisonOpType">
   <xsd:complexContent>
      <xsd:extension base="ogc:ComparisonOpsType">
         <xsd:sequence>
            <xsd:element ref="ogc:expression" minOccurs="2" maxOccurs="2"/>
         </xsd:sequence>
         <xsd:attribute name="matchCase" type="xsd:boolean"
                        use="optional" default="true"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="PropertyIsLikeType">
   <xsd:complexContent>
      <xsd:extension base="ogc:ComparisonOpsType">
         <xsd:sequence>
            <xsd:element ref="ogc:PropertyName"/>
            <xsd:element ref="ogc:Literal"/>
         </xsd:sequence>
```

```
            <xsd:attribute name="wildCard" type="xsd:string" use="required"/>
            <xsd:attribute name="singleChar" type="xsd:string" use="required"/>
            <xsd:attribute name="escapeChar" type="xsd:string" use="required"/>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="PropertyIsNullType">
      <xsd:complexContent>
         <xsd:extension base="ogc:ComparisonOpsType">
            <xsd:sequence>
               <xsd:element ref="ogc:PropertyName" maxOccurs="1"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="PropertyIsBetweenType">
      <xsd:complexContent>
         <xsd:extension base="ogc:ComparisonOpsType">
            <xsd:sequence>
               <xsd:element ref="ogc:expression"/>
               <xsd:element name="LowerBoundary" type="ogc:LowerBoundaryType"/>
               <xsd:element name="UpperBoundary" type="ogc:UpperBoundaryType"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="LowerBoundaryType">
      <xsd:choice>
         <xsd:element ref="ogc:expression"/>
      </xsd:choice>
   </xsd:complexType>

   <xsd:complexType name="UpperBoundaryType">
      <xsd:sequence>
         <xsd:element ref="ogc:expression"/>
      </xsd:sequence>
   </xsd:complexType>
```

The Common Catalog Query Language [4] defines a standard set of comparison operators (=,<,>,>=,<=,<>). These comparison operators are encoded using the complex type **BinaryComparisonOpType**. This type definition includes the **matchCase** attribute which is of type Boolean and controls whether string comparisons are caseless or not. A value of true means that string comparisons also match case. This is the default value. A value of false means that string comparisons are performed caselessly.

In addition to the standard set of comparison operators, this specification defines the elements **<PropertyIsLike>, <PropertyIsBetween>** and **<PropertyIsNull>**.

The **<PropertyIsLike>** element is intended to encode a character string comparison operator with pattern matching. The pattern is defined by a combination of regular characters, the **wildCard** character, the **singleChar** character, and the **escapeChar** character. The **wildCard** character matches zero or more characters. The **singleChar** character matches exactly one character. The **escapeChar** character is used to escape the meaning of the **wildCard**, **singleChar** and **escapeChar** itself.

The **<PropertyIsNull>** element encodes an operator that checks to see if the value of its content is NULL. A NULL is equivalent to no value present. The value 0 is a valid value and is not considered NULL.

The **<PropertyIsBetween>** element is defined as a compact way of encoding a range check. The lower and upper boundary values are inclusive.

10 Logical operators

10.1 Introduction

A logical operator can be used to combine one or more conditional expressions. The logical operator AND evaluates to TRUE if all the combined expressions evaluate to TRUE. The operator OR operator evaluates to TRUE is any of the combined expressions evaluate to TRUE. The NOT operator reverses the logical value of an expression.

10.2 Encoding

The XML encoding for the logical operators AND, OR and NOT is defined by the following XML Schema fragment:

```
<xsd:element name="And" type="ogc:BinaryLogicOpType" substitutionGroup="ogc:logicOps"/>
<xsd:element name="Or"  type="ogc:BinaryLogicOpType" substitutionGroup="ogc:logicOps"/>
<xsd:element name="Not" type="ogc:UnaryLogicOpType" substitutionGroup="ogc:logicOps"/>
<xsd:element name="logicOps" type="ogc:LogicOpsType" abstract="true"/>
<xsd:complexType name="LogicOpsType" abstract="true"/>
<xsd:complexType name="BinaryLogicOpType">
   <xsd:complexContent>
      <xsd:extension base="ogc:LogicOpsType">
         <xsd:choice minOccurs="2" maxOccurs="unbounded">
            <xsd:element ref="ogc:comparisonOps"/>
            <xsd:element ref="ogc:spatialOps"/>
            <xsd:element ref="ogc:logicOps"/>
         </xsd:choice>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="UnaryLogicOpType">
   <xsd:complexContent>
      <xsd:extension base="ogc:LogicOpsType">
         <xsd:sequence>
            <xsd:choice>
               <xsd:element ref="ogc:comparisonOps"/>
               <xsd:element ref="ogc:spatialOps"/>
               <xsd:element ref="ogc:logicOps"/>
            </xsd:choice>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

The elements <**And**>, <**Or**> and <**Not**> can be used to combine scalar, spatial and other logical expressions to form more complex compound expressions.

The <**comparisonOps**> and <**spatialOps**> elements are abstract elements that can be substituted for the comparison operators [sec 9] (<,>,=,etc…) and the spatial operators [sec 8].

11 Object identifiers

11.1 Introduction

A object identifier is meant to represent a unique identifier for an object instance within the context of the web service that is serving the object. This specification does not define a specific element for identifying objects but instead defines the abstract element **<_Id>** as the head of an XML substitution group that may be used to define an object identifier element for specific object types.

For example, the <**GmlObjectId**> element is defined in this specification to reference GML 3 feature instances and component elements of GML 3 feature instances. Other specifications may need to define identifier elements for their specific object types. A web object service might, for instance, define an <**ObjectId**> element while a web catalog service might define a <**RecordId**> element.

The <**FeatureId**> element introduced in version 1.0.0 of this specification identifies feature instances from GML version 2 documents and data services using the **fid** attribute. With GML version 3.0, the **fid** attribute has been deprecated in favor of a new **gml:id** attribute that may used to identify GML complex type elements in addition to features, such as geometries, topologies, and complex attributes. A new <**GmlObjectId**> element is introduced here to identify elements in GML version 3 documents and data services. The <**FeatureId**> element is retained for backward compatibility, but deprecated in this version, meaning that it may be removed from a subsequent version of this specification without further notice.

11.2 Encoding

The following XML schema fragment defined the abstract element **<_Id**> as well as the <**GmlObjectId**> element introduced above. The <**FeatureId**> element is defined for backward compatibility but is deprecated in this version of the specification.

```
<xsd:element name="_Id" type="ogc:AbstractIdType" abstract="true"/>
<xsd:element name="FeatureId" type="ogc:FeatureIdType" substitutionGroup="ogc:_Id"/>
<xsd:element name="GmlObjectId" type="ogc:GmlObjectIdType" substitutionGroup="ogc:_Id"/>
<xsd:complexType name="AbstractIdType" abstract="true"/>
<xsd:complexType name="FeatureIdType">
   <xsd:complexContent>
      <xsd:extension base="ogc:AbstractIdType">
         <xsd:attribute name="fid" type="xsd:ID" use="required"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="GmlObjectIdType">
   <xsd:complexContent>
      <xsd:extension base="ogc:AbstractIdType">
         <xsd:attribute ref="gml:id" use="required"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

Using the <**GmlObjectId**> or <**FeatureId**> elements that may substitute for the **<_Id**> element, a filter can conveniently encode a reference to one or more enumerated feature instances, or to feature components identified by **gml:id** attributes, including geometries, topologies, and complex attributes. However, note that all of the **<_Id**> elements in a <**Filter**> must be either GmlObjectIds or FeatureIds; the two types cannot be used together.

12 Expressions

12.1 Introduction

An expression is a combination of one or more symbols that evaluate to single boolean value of true or false. In this specification, valid symbols are encoded using XML elements and expressions are formed by nesting elements to form XML fragments that validate against the schemas in ANNEX A.

12.2 Encoding

An *expression* can be formed using the elements:

<Add>, <Sub>, <Mul>, <Div>, <PropertyName>, <Literal> and <**Function**>.

They all belong to the *substitution group* **expression** which means that any of them can be used wherever an expression is called for. In addition, the XML fragments formed by combining these elements are themselves *expressions* and can be used wherever an *expression* is called for.

The <**expression**> element is an *abstract* element which means that it does not really exist and its only purpose is to act as a placeholder for the elements and combinations of elements that can be used to form expressions.

The XML Schema fragment that defines the abstract <**expression>** element is:

```
<xsd:element name="expression" type="ogc:ExpressionType" abstract="true"/>
<xsd:complexType name="ExpressionType" abstract="true"/>
```

13 Arithmetic operators

13.1 Introduction

The elements defined in this section are used to encode the fundamental arithmetic operations of addition, subtraction, multiplication and division. Arithmetic operators are binary operators meaning that they accept two arguments and evaluate to a single result.

13.2 Encoding

The XML encoding for arithmetic expressions is defined by the following XML Schema fragment:

```
<xsd:element name="Add"
             type="ogc:BinaryOperatorType"
             substitutionGroup="ogc:expression"/>
<xsd:element name="Sub"
             type="ogc:BinaryOperatorType"
             substitutionGroup="ogc:expression"/>
<xsd:element name="Mul"
             type="ogc:BinaryOperatorType"
             substitutionGroup="ogc:expression"/>
<xsd:element name="Div"
             type="ogc:BinaryOperatorType"
             substitutionGroup="ogc:expression"/>
<xsd:complexType name="BinaryOperatorType">
   <xsd:complexContent>
      <xsd:extension base="ogc:ExpressionType">
         <xsd:sequence>
            <xsd:element ref="ogc:expression" minOccurs="2" maxOccurs="2"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

The <**Add**> element encodes the operation of addition and contains the arguments which can be any *expression* that validates according to this specification.

The <**Sub**> element encodes the operation of subtraction where the second argument is subtracted from the first. The <**Sub>** element contains the argument when can be any *expression* that validates according to this specification.

The <**Mul**> element encodes the operation multiplication. The **<Mul>** element contains the two argument to be multiplied which can be any *expression* that validates according to this specification.

The <**Div**> element encodes the operation of division where the first argument is divided by the second argument. The **<Div>** element contains the arguments which can be any valid *expression* that validates according to this specification. The second argument or expression cannot evaluate to zero.

14 Literals

14.1 Introduction

This section defines how the filter encoding defined in this specification encodes literal values. A literal value is any part of a statement or expression that is to be used exactly as it is specified, rather than as a variable or other element.

14.2 Encoding

The following XML Schema fragment defines the **<Literal>** element:

```
<xsd:element name="Literal"
             type="ogc:LiteralType"
             substitutionGroup="ogc:expression"/>
<xsd:complexType name="LiteralType">
   <xsd:complexContent mixed="true">
      <xsd:extension base="ogc:ExpressionType">
         <xsd:sequence>
            <xsd:any minOccurs="0"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

The **<Literal>** element is used to encode literal scalar and geometric values.

Literal geometric values must be encoded as the content of the **<Literal>** element, according to the schemas of GML as described in the Geography Markup Language (GML) Implementation Specification [12]. That is to say that any literal GML geometry instance must validate against the XML Schemas defined for GML3.

15 Functions

15.1 Introduction

This section defines the encoding of single value functions using the <**Function>** element. A function is a named procedure that performs a distinct computation. A function may accept zero or more arguments as input and generates a single result.

15.2 Encoding

The following XML Schema fragment defines the **<Function>** element:

```
<xsd:element name="Function"
             type="ogc:FunctionType"
             substitutionGroup="ogc:expression"/>
<xsd:complexType name="FunctionType">
   <xsd:complexContent>
      <xsd:extension base="ogc:ExpressionType">
         <xsd:sequence>
            <xsd:element ref="ogc:expression"
                         minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="name" type="xsd:string" use="required"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

A function is composed of the name of the function, encoded using the attribute **name**, and zero or more arguments contained within the **<Function>** element. The arguments themselves are *expressions* discussed in clause 11.

16 Filter capabilities

The filterCapabilities.xsd schema defines a capabilities document section that shall be instantiated in the capabilities document of services that use filter encoding. The filter capabilities document section describes what specific filter capabilities are supported by a service. For example, a web feature service that uses filter encoding would include this fragment in its capabilities document to advertise what filter capabilities it supports.

Filter capabilities are divided into three categories: spatial capabilities, scalar capabilities and id capabilities. The following XML Schema fragment defines the root element of the filter capabilities:

```
<xsd:element name="Filter_Capabilities">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="Spatial_Capabilities"
                      type="ows:Spatial_CapabilitiesType"/>
         <xsd:element name="Scalar_Capabilities"
                      type="ows:Scalar_CapabilitiesType"/>
         <xsd:element name="Id_Capabilities"
                      type="ows:Id_CapabilitiesType"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

A service that supports spatial filtering shall include a spatial capabilities section in the capabilities document that it returns. Spatial capabilities include the ability to filter spatial data of specified geometry types based on the definition of a bounding box (BBOX) as well as the ability to process the spatial operators defined by CQL [4] and the Simple Features for SQL [5] specification: Equals, Disjoint, Touches, Within, Overlaps, Crosses, Intersects, Contains, DWithin and Beyond. Spatial capabilities are encoded according to the following XML Schema fragments:

```
   <xsd:element name="Spatial_Capabilities"
            type="ows:Spatial_CapabilitiesType"/>
   <xsd:complexType name="Spatial_CapabilitiesType">
      <xsd:sequence>
         <xsd:element name="GeometryOperands"
                      type="ogc:GeometryOperandsType"/>
```

```
            <xsd:element name="SpatialOperators"
                         type="ogc:SpatialOperatorsType"/>
         </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="GeometryOperandsType">
         <xsd:sequence>
            <xsd:element name="GeometryOperand"
                         type="GeoemtryOperandType" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:complexType>
      <xsd:simpleType name="GeometryOperandType">
         <xsd:restriction base="xsd:QName">
            <xsd:enumeration value="gml:Envelope"/>
            <xsd:enumeration value="gml:Point"/>
            <xsd:enumeration value="gml:LineString"/>
            <xsd:enumeration value="gml:Polygon"/>
            <xsd:enumeration value="gml:ArcByCenterPoint"/>
            <xsd:enumeration value="gml:CircleByCenterPoint"/>
            <xsd:enumeration value="gml:Arc"/>
            <xsd:enumeration value="gml:Circle"/>
            <xsd:enumeration value="gml:ArcByBulge"/>
            <xsd:enumeration value="gml:Bezier"/>
            <xsd:enumeration value="gml:Clothoid"/>
            <xsd:enumeration value="gml:CubicSpline"/>
            <xsd:enumeration value="gml:Geodesic"/>
            <xsd:enumeration value="gml:OffsetCurve"/>
            <xsd:enumeration value="gml:Triangle"/>
            <xsd:enumeration value="gml:PolyhedralSurface"/>
            <xsd:enumeration value="gml:TriangulatedSurface"/>
            <xsd:enumeration value="gml:Tin"/>
            <xsd:enumeration value="gml:Solid"/>
         </xsd:restriction>
      </xsd:simpleType>
      <xsd:complexType name="SpatialOperatorsType">
         <xsd:sequence>
            <xsd:element name="SpatialOperator"
                         type="SpatialOperatorType"
                         maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="SpatialOperatorType">
         <xsd:sequence>
            <xsd:element name="GeometryOperands"
                         type="ogc:GeometryOperandsType"/>
         </xsd:sequence>
         <xsd:attribute name="name" type="SpatialOperatorNameType"/>
      </xsd:complexType>
      <xsd:simpleType name="SpatialOperatorNameType">
         <xsd:restriction base="xsd:string">
            <xsd:enumeration value="BBOX"/>
            <xsd:enumeration value="Equals"/>
            <xsd:enumeration value="Disjoint"/>
            <xsd:enumeration value="Intersect"/>
            <xsd:enumeration value="Touches"/>
            <xsd:enumeration value="Crosses"/>
            <xsd:enumeration value="Within"/>
            <xsd:enumeration value="Contains"/>
            <xsd:enumeration value="Overlaps"/>
            <xsd:enumeration value="Beyond"/>
         </xsd:restriction>
</xsd:simpleType>
```

A service that implements this specification shall list the spatial operators and geometry operand types that it supports as the content of the <**SpatialCapabilities**> element of its filter capabilities document. Geometry operands are listed using the <**GeometryOperands**> element. Geometry operands may be defined globally (as the first child of the <**Spatial_Capabilities**> element) indicating that all spatial operators know how process the specified operands or locally for each spatial operator (as the content of the <**SpatialOperator**> element) indicating that the specific operator knows how to process the specified operands.

Scalar capabilities include the ability to process logical expressions, comparisons and arithmetic operations including the ability to process a list of named functions. The following XML Schema defines how scalar capabilities are encoded:

```
<xsd:complexType name="Scalar_CapabilitiesType">
   <xsd:choice maxOccurs="unbounded">
      <xsd:element ref="ogc:LogicalOperators"/>
      <xsd:element name="ComparisonOperators"
                   type="ogc:ComparisonOperatorsType"/>
      <xsd:element name="ArithmeticOperators"
                   type="ogc:ArithmeticOperatorsType"/>
   </xsd:choice>
</xsd:complexType>
```

The **<LogicalOperators>** element is used to indicate that the filter can process ***And, Or*** and ***Not*** operators.

The **<ComparisonOperators>** element is used to indicate what types of comparison operators are supported by a service. The **<SimpleComparisons>** element is used to indicate that the operators **=, <, <=, >, >=** are supported. The elements **<Like>, <Between>** and **<NullCheck>** are used to indicate that the service can support the operators LIKE, BETWEEN and NULL.

The **<ArithmeticOperators>** element is used to indicate what arithmetic operators the a service can support. The contained element **<SimpleArithmetic>** is used to indicate that the service can support addition, subtraction, multiplication and division. The contained element **<Functions>** is used to list the function names that are supported and the number of arguments each function requires. Function arguments are encoded as nested elements within the **<Function>** tag as defined in clause 15, *Encoding Functions*.

Id capabilities include the ability to refer to elements in a GML version 3 data source using an ogc:GmlObjectId with a gml:id attribute. Any elements with a gml:id attribute, including features, geometries, topologies, and complex attributes, may be so referenced.

For backward compatibility, Id capabilities also include the ability to refer to features using an ogc:FeatureId with a fid attribute. Only features with a GML version 2 gml:fid attribute may be so referenced. Note that features in a GML version 3 data source may have gml:fid attributes for backwards compatibility with GML version 2, in addition to gml:id attributes introduced in GML version 3. This ability is deprecated and may be removed from a subsequent version of this specification without further notice.

Example

The following example shows a capabilities fragment for a service that supports all filtering capabilities defined in this document include a list of named functions:

```
<Filter_Capabilities>
      <ogc:Filter_Capabilities
         xmlns:ogc="http://www.opengis.net/ogc"
         xmlns:gml="http://www.opengis.net/gml"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.opengis.net/ogc
http://www.pvretano.com/wfs1.1/1.1/schemas/filter/1.1.0/filterCapabilities.xsd">
      <ogc:Spatial_Capabilities>
         <ogc:GeometryOperands>
            <ogc:GeometryOperand>gml:Envelope</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Point</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:LineString</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Polygon</ogc:GeometryOperand>
```

```
            <ogc:GeometryOperand>gml:ArcByCenterPoint</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:CircleByCenterPoint</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Arc</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Circle</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:ArcByBulge</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Bezier</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Clothoid</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:CubicSpline</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Geodesic</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:OffsetCurve</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Triangle</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:PolyhedralSurface</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:TriangulatedSurface</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Tin</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Solid</ogc:GeometryOperand>
         </ogc:GeometryOperands>
         <ogc:SpatialOperators>
            <ogc:SpatialOperator name="BBOX"/>
            <ogc:SpatialOperator name="Equals"/>
            <ogc:SpatialOperator name="Disjoint"/>
            <ogc:SpatialOperator name="Intersect"/>
            <ogc:SpatialOperator name="Touches"/>
            <ogc:SpatialOperator name="Crosses"/>
            <ogc:SpatialOperator name="Within"/>
            <ogc:SpatialOperator name="Contains"/>
            <ogc:SpatialOperator name="Overlaps"/>
            <ogc:SpatialOperator name="Beyond"/>
         </ogc:SpatialOperators>
      </ogc:Spatial_Capabilities>
      <ogc:Scalar_Capabilities>
         <ogc:LogicalOperators/>
         <ogc:ComparisonOperators>
           <ogc:ComparisonOperator>LessThan</ogc:ComparisonOperator>
           <ogc:ComparisonOperator>GreaterThan</ogc:ComparisonOperator>
           <ogc:ComparisonOperator>LessThanEqualTo</ogc:ComparisonOperator>
           <ogc:ComparisonOperator>GreaterThanEqualTo</ogc:ComparisonOperator>
           <ogc:ComparisonOperator>EqualTo</ogc:ComparisonOperator>
           <ogc:ComparisonOperator>NotEqualTo</ogc:ComparisonOperator>
           <ogc:ComparisonOperator>Like</ogc:ComparisonOperator>
           <ogc:ComparisonOperator>Between</ogc:ComparisonOperator>
           <ogc:ComparisonOperator>NullCheck</ogc:ComparisonOperator>
         </ogc:ComparisonOperators>
         <ogc:ArithmeticOperators>
            <ogc:SimpleArithmetic/>
            <ogc:Functions>
               <ogc:FunctionNames>
                  <ogc:FunctionName nArgs="1">MIN</ogc:FunctionName>
                  <ogc:FunctionName nArgs="1">MAX</ogc:FunctionName>
                  <ogc:FunctionName nArgs="1">SIN</ogc:FunctionName>
                  <ogc:FunctionName nArgs="1">COS</ogc:FunctionName>
                  <ogc:FunctionName nArgs="1">TAN</ogc:FunctionName>
               </ogc:FunctionNames>
            </ogc:Functions>
         </ogc:ArithmeticOperators>
      </ogc:Scalar_Capabilities>
      <ogc:Id_Capabilities>
         <ogc:EID/>
         <ogc:FID/>
      </ogc:Id_Capabilities>
   </ogc:Filter_Capabilities>
```

Example

The following example is a filter capabilities document for a service that support a limited set of filter capabilities:

```
      <ogc:Filter_Capabilities
         xmlns:ogc="http://www.opengis.net/ogc"
         xmlns:gml="http://www.opengis.net/gml"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.opengis.net/ogc
http://www.pvretano.com/wfs1.1/1.1/schemas/filter/1.1.0/filterCapabilities.xsd">
      <ogc:Spatial_Capabilities>
         <ogc:GeometryOperands>
            <ogc:GeometryOperand>gml:Envelope</ogc:GeometryOperand>
         </ogc:GeometryOperands>
         <ogc:SpatialOperators>
            <ogc:SpatialOperator name="BBOX"/>
         </ogc:SpatialOperators>
      </ogc:Spatial_Capabilities>
      <ogc:Scalar_Capabilities>
         <ogc:LogicalOperators/>
```

```
        <ogc:ComparisonOperators>
          <ogc:ComparisonOperator>LessThan</ogc:ComparisonOperator>
          <ogc:ComparisonOperator>GreaterThan</ogc:ComparisonOperator>
          <ogc:ComparisonOperator>LessThanEqualTo</ogc:ComparisonOperator>
          <ogc:ComparisonOperator>GreaterThanEqualTo</ogc:ComparisonOperator>
          <ogc:ComparisonOperator>EqualTo</ogc:ComparisonOperator>
          <ogc:ComparisonOperator>NotEqualTo</ogc:ComparisonOperator>
          <ogc:ComparisonOperator>Like</ogc:ComparisonOperator>
          <ogc:ComparisonOperator>Between</ogc:ComparisonOperator>
          <ogc:ComparisonOperator>NullCheck</ogc:ComparisonOperator>
        </ogc:ComparisonOperators>
        <ogc:ArithmeticOperators>
           <ogc:SimpleArithmetic/>
        </ogc:ArithmeticOperators>
     </ogc:Scalar_Capabilities>
     <ogc:Id_Capabilities>
        <ogc:EID/>
        <ogc:FID/>
     </ogc:Id_Capabilities>
  </ogc:Filter_Capabilities>
```

Specifically, this service supports the BBOX spatial operator, logical operations, comparison operations and simple arithmetic.

ANNEX A - Examples (Informative)

A.1 Introduction

This annex contains a number of examples of filters. Since filter are meant to be part of larger schemas, these examples represent XML fragments that would likely be embedded in another XML document such as web feature service request.

A.2 Examples

Example 1

A simple non-spatial filter checking to see if *SomeProperty* is equal to 100.

```
<Filter>
   <PropertyIsEqualTo>
      <PropertyName>SomeProperty</PropertyName>
      <Literal>100</Literal>
   </PropertyIsEqualTo>
</Filter>
```

Example 2

A simple non-spatial filter comparing a property value to a literal. In this case, the DEPTH is checked to find instances where it is less than 30 - possibly to identify areas that need dredging.

```
<Filter>
   <PropertyIsLessThan>
      <PropertyName>DEPTH</PropertyName>
      <Literal>30</Literal>
   </PropertyIsLessThan>
</Filter>
```

Example 3

This example encodes a simple spatial filter. In this case we are finding all features that have a geometry that spatially interacts with the specified bounding box. The expression NOT DISJOINT is used to exclude all features that do not interact with the bounding box; in other words identify all the features that interact with the bounding box in some way.

```
<Filter>
   <Not>
      <Disjoint>
         <PropertyName>Geometry</PropertyName>
         <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
            <gml:lowerCorner>13.0983 31.5899</gml:lowerCorner>
            <gml:upperCorner>35.5472 42.8143</gml:upperCorner>
         </gml:Envelope>
      </Disjoint>
   </Not>
</Filter>
```

An alternative encoding of this filter could have used to **<BBOX>** element:

```
<Filter>
   <BBOX>
      <PropertyName>Geometry</PropertyName>
      <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
         <gml:lowerCorner>13.0983 31.5899</gml:lowerCorner>
         <gml:upperCorner>35.5472 42.8143</gml:upperCorner>
```

```
      </gml:Envelope>
   </BBOX>
</Filter>
```

Example 4

In this example we combine examples 2 and 3 with the logical operator AND. The predicate is thus interpreted as seeking all features that interact with the specified bounding box and have a DEPTH value of less than 30 meters.

```
<Filter>
   <And>
      <PropertyIsLessThan>
         <PropertyName>DEPTH</PropertyName>
         <Literal>30</Literal>
      </PropertyIsLessThan>
      <Not>
         <Disjoint>
            <PropertyName>Geometry</PropertyName>
            <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
               <gml:lowerCorner>13.0983 31.5899</gml:lowerCorner>
               <gml:upperCorner>35.5472 42.8143</gml:upperCorner>
            </gml:Envelope>
         </Disjoint>
      </Not>
   </And>
</Filter>
```

Example 5

This example encodes a simple filter block identifying a specific feature instance of the feature type TREESA_1M. Any operation that included this filter block would be applied to that specific feature. For a filter applied to a GML version 3 data store, the filter block is:

```
<Filter>
   <GmlObjectId gml:id="TREESA_1M.1234"/>
</Filter>
```

For a filter applied to a GML version 2 data store, the filter block is:

```
<Filter>
   <FeatureId fid="TREESA_1M.1234"/>
</Filter>
```

Example 6

A **<Filter>** element can also be used to identify an enumerated set of feature instances or feature components. In this case, any operation that included this filter block would be limited to the feature instances or feature components listed within the **<Filter>** tags.

A filter applied to a GML version 3 data store:

```
<Filter>
   <GmlObjectId gml:id="TREESA_1M.1234"/>
   <GmlObjectId gml:id="TREESA_1M.5678"/>
   <GmlObjectId gml:id="TREESA_1M.9012"/>
   <GmlObjectId gml:id="INWATERA_1M.3456"/>
   <GmlObjectId gml:id="INWATERA_1M.7890"/>
   <GmlObjectId gml:id="BUILTUPA_1M.4321”/>
</Filter>
```

A filter applied to a GML version 2 data store:

```
<Filter>
   <FeatureId fid="TREESA_1M.1234"/>
   <FeatureId fid="TREESA_1M.5678"/>
   <FeatureId fid="TREESA_1M.9012"/>
   <FeatureId fid="INWATERA_1M.3456"/>
   <FeatureId fid="INWATERA_1M.7890"/>
```

```
   <FeatureId fid="BUILTUPA_1M.4321”/>
</Filter>
```

Example 7

The following filter includes the encoding of a function. This filter identifies all features where the sin() of the property named DISPERSION_ANGLE is 1.

```
<Filter>
   <PropertyIsEqualTo>
      <Function name="SIN">
         <PropertyName>DISPERSION_ANGLE</PropertyName>
      </Function>
      <Literal>1</Literal>
   </PropertyIsEqualTo>
</Filter>
```

Example 8

This example encodes a filter that includes an arithmetic expression. This filter is equivalent to the expression A = B + 100.

```
<Filter>
   <PropertyIsEqualTo>
      <PropertyName>PROPA</PropertyName>
      <Add>
         <PropertyName>PROPB</PropertyName>
         <Literal>100</Literal>
      </Add>
   </PropertyIsEqualTo>
</Filter>
```

Example 9

This example encodes a filter using the BETWEEN operator. The filter identifies all features where the DEPTH is between 100 and 200 meters.

```
<Filter>
   <PropertyIsBetween>
      <PropertyName>DEPTH</PropertyName>
      <LowerBoundary><Literal>100</Literal></LowerBoundary>
      <UpperBoundary><Literal>200</Literal></UpperBoundary>
   </PropertyIsBetween>
</Filter>
```

Example 10

This example is similar to example 9 except that in this case the filter is checking to see if the SAMPLE_DATE property is within a specified date range. The dates are represented as described in the Annex B of the Web Map Service Implementation Specification [6].

```
<Filter>
   <Between>
      <PropertyName>SAMPLE_DATE</PropertyName>
      <LowerBoundary>
         <Literal>2001-01-15T20:07:48.11</Literal>
      </LowerBoundary>
      <UpperBoundary>
         <Literal>2001-03-06T12:00:00.00</Literal>
      </UpperBoundary>
   </Between>
</Filter>
```

Example 11

This example encodes a filter using the LIKE operation to perform a pattern matching comparison. In this case, the filter identifies all features where the value of the property named LAST_NAME begins with the letters "JOHN".

```
<Filter>
   <PropertyIsLike wildCard="*" singleChar="#" escapeChar="!">
      <PropertyName>LAST_NAME</PropertyName>
      <Literal>JOHN*</Literal>
   </PropertyIsLike>
</Filter>
```

Example 12

This example encodes a spatial filter that identifies all features that overlap a polygonal area of interest.

```
<Filter>
   <Overlaps>
      <PropertyName>Geometry</PropertyName>
      <gml:Polygon srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
         <gml:outerBoundaryIs>
            <gml:LinearRing>
               <gml:posList> … </gml:posList>
            </gml:LinearRing>
         </gml:outerBoundaryIs>
      </gml:Polygon>
   </Overlaps>
</Filter>
```

Example 13

In this example, a more complex scalar predicate is encoded using the logical operators AND and OR. The example is equivalent to the expression:

```
((FIELD1=10 OR FIELD1=20) AND (STATUS="VALID").
```

```
<Filter>
   <And>
      <Or>
         <PropertyIsEqualTo>
            <PropertyName>FIELD1</PropertyName>
            <Literal>10</Literal>
         </PropertyIsEqualTo>
         <PropertyIsEqualTo>
            <PropertyName>FIELD1</PropertyName>
            <Literal>20</Literal>
         </PropertyIsEqualTo>
      </Or>
      <PropertyIsEqualTo>
         <PropertyName>STATUS</PropertyName>
         <Literal>VALID</Literal>
      </PropertyIsEqualTo>
   </And>
</Filter>
```

Example 14

Spatial and non-spatial predicates can be encoded in a single filter expression. In this example, a spatial predicate checks to see if the geometric property *WKB_GEOM* lies within a region of interest defined by a polygon and a scalar predicate check to see if the scalar property *DEPTH* lies within a specified range. This encoding is equivalent to the expression :

```
(geometry INSIDE ``some polygon'') AND (depth between 400 and 800)
```

```
<Filter>
   <And>
      <Within>
         <PropertyName>WKB_GEOM</PropertyName>
         <gml:Polygon name="1" srsName="EPSG:63266405">
            <gml:outerBoundaryIs>
               <gml:LinearRing>
                  <gml:posList>-98.5485,24.2633 ...</gml:posList>
               </LinearRing>
            </outerBoundaryIs>
         </Polygon>
      </Within>
      <PropertyIsBetween>
```

```
            <PropertyName>DEPTH</PropertyName>
            <LowerBoundary><Literal>400</Literal></LowerBoundary>
            <UpperBoundary><Literal>800</Literal></UpperBoundary>
         </PropertyIsBetween>
      </And>
   </Filter>
```

Example 15

The following example restricts the active set of objects to those instances of the *Person* type that are older than 50 years old and live in Toronto. This filter expression uses an XPath expression to reference the complex attributes of the *Person* type.

```
   <Filter>
      <And>
         <PropertyIsGreaterThan>
            <PropertyName>Person/age</PropertyName>
            <Literal>50</Literal>
         </PropertyIsGreaterThan>
         <PropertyIsEqualTo>
            <PropertyName>Person/mailAddress/Address/city</PropertyName>
            <Literal>Toronto</Literal>
         </PropertyIsEqualTo>
      </And>
   </Filter>
```

Example 16

```
<ogc:Filter xmlns:ogc="http://www.opengis.net/ogc>
   <ogc:Neighbor>
      <ogc:PropertyName>GEOM</ogc:PropertyName>
      <gml:Point srsName="EPSG:26985">
         <gml:pos>385514.709250495 118401.34057266</gml:pos>
      </gml:Point>
      <ogc:Literal>1</ogc:Literal>
   </ogc:Neighbor>
</ogc:Filter>
```

ANNEX B – Filter schema definitions (Normative)

B.1 Introduction

In order to keep this document to a reasonable length, the normative schemas are not included inline but are attached to the archive package that includes this document. Optionally, the schemas may be obtained at http://schemas.opengeospatial.net/filter/1.1.0.

The files that make up the filter schemas are:

```
1.1.0/filter.xsd
1.1.0/filterCapabilities.xsd
1.1.0/expr.xsd
1.1.0/sort.xsd
1.1.0/examples/Filter_Capabilities_Sample.xml
1.1.0/examples/Filter_Capabilities_Small_Sample.xml
```

ANNEX C - Conformance tests

Specific conformance tests for a filter encoding processor have not yet been determined and will be added in a future revision of this specification.

At the moment, a filter encoding processor implementation must satisfy the following system characteristics to be minimally conformant with this specification:

1. An XML filter expression encoded according to this specification must validate relative to the normative schemas defined in Annex A.

2. A filter capabilities document encoded as defined in this document must validate relative to the schema defined in Annex A.

3. All clauses in the normative sections of this specification that use the keywords "must", "must not", "required", "shall", and "shall not" have been satisfied.

Bibliography

[14] Vretanos, Panagiotis (ed.), "OpenGIS® Implementation Specification #01-065: Web Feature Service Implementation Specification", July 2002

[15] National Center for Supercomputing Applications, "The Common Gateway Interface," http://hoohoo.ncsa.uiuc.edu/cgi/.

[16] Freed, N. and Borenstein N., "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", IETF RFC 2045, November 1996, http://www.ietf.org/rfc/rfc2045.txt.

[17] Internet Assigned Numbers Authority, http://www.isi.edu/in-notes/iana/assignments/media-types/.