Open GIS Consortium Inc.

Date: 17-MAY-2001

Reference number of this OpenGIS® project document: OGC 02-059

Version: 1.0.0

Category: OpenGIS® Implementation Specification

Status: Adopted Specification

Editor: Panagiotis A. Vretanos

Filter Encoding Implementation Specification

Document type: **OpenGIS® Publicly Available Standard**
Document stage: **Request for Comment**
Document language: **English**

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.
Jonathan Clark
1881 Campus Commons Drive
Reston, VA 20191
U.S.A
jrclark@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 or to the WWW Mapping SIG chair:

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

Allan Doyle (WWW Mapping SIG Chair)
International Interfaces, Inc.
948 Great Plain Ave. PMB-182
Needham, MA 02492 USA
+1 781 433 2695
adoyle@intl-interfaces.com

Additional contributors

Rob Atkinson (Social Change Online) rob@socialchange.net.au
Craig Bruce (CubeWerx) csbruce@cubwerx.com
Jonathan Clark (Intergraph) jrclark@intergraph.com
Adrian Cuthbert (SpotOn MOBILE) adrian@spotonmobile.com
Paul Daisey (U.S. Census) pdaisey@geo.census.gov
Ignacio Guerrero (Intergraph) IGuerrer@ingr.com
Sandra Johnson (Mapinfo) Sandra_Johnson@mapinfo.com
Edric Keighan (CubeWerx) ekeighan@cubewerx.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
Brian May (CubeWerx) bmay@cubewerx.com
Richard Martell (Galdos Systems Ltd.) rmartell@galdosinc.com
Aleksander Milanovic (Galdos Systems Ltd.) amilanovic@galdosinc.com
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
Daniel Specht (TEC) specht@tec.army.mil
James T. Stephens (Lockheed Martin) james.t.stephens@lmco.com
Glenn Stowe (CubeWerx) gstowe@cubwerx.com
Milan Trninic (Galdos Systems Inc.) mtrninic@galdosinc.com
Peter Woodsford (Laser-Scan Ltd.) peterw@lsl.co.uk
Arliss Whitesize (BAE Systems) Arliss.Whiteside@baesystems.com

iv. Revision history

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 section 2.2 and 3.2 of Topic 13, "Catalog Services".

vi. Future work

This specification defines a feature identifier as an identifier that is unique within the context of the server serving the feature. The specification also describes how such a local identifier can be made globally unique by combining it with the URL of the service serving the feature. However what is described in this specification is informative. A future work item is to normatively describe how globally unique identifiers should be generated.

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

Annexes B and C are 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 constraints 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] Enloe, Yonsook, Nebert, Doug, Stphens, Larry (eds.), "OpenGIS® Implementation Specification #99-051s: Catalog Interface Implementation Specification, Version 1.0", 1999

[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., Cuthbert, A., Lake, R., and Martell, R. (eds.), "OpenGIS Implementation Specification #02-009: OpenGIS® Geography Markup Language (GML) Implementation Specification, version 2.1.1", April 2002

[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 sections 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 feature properties encoded as XML elements or attributes.

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. 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 simple and complex properties of an object (encoded as elements or attributes), consider the fictitious GML feature *Person* defined by the following XML Schema document:

```
<?xml version="1.0" ?>
<schema
   targetNamespace="http://www.cubewerx.com/myns"
   xmlns:myns="http://www.cubewerx.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/2.1/feature.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 name="sin" type="xsd:anyURI" use="required" />
                  </complexType>
               </element>
               <element name="Location"
                        type="gml:PointPropertyType"
                        nillable="true"/>
               <element name="Address" type="myns:AddressType" nillable="true"/>
            </sequence>
            <attribute name="sin" type="xsd:anyURI" use="required"/>
         </extension>
      </complexContent>
   </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 *Address* is a complex property of type *AddressType*. An example instance of the feature *Person* might be:

```
<?xml version="1.0" ?>
<myns:Person
   sin="111222333"
   xmlns:myns="http://www.opengis.net/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:coordinates>15,15</gml:coordinates></gml:Point>
   </myns:Location>
   <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:Person>
```

Using XPath [10] expressions, each property of a *Person* feature can be referenced. Table 1 shows the XPath expressions that may be used to reference all the properties of the *Person* feature as well as the corresponding value of each property.

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

XPath Expression	Alternate XPath Expression	Property Value
LastName	Person/LastName	Smith
FirstName	Person/FirstName	Fred
Age	Person/Age	35
Sex	Person/Sex	Male
Spouce/@sin	Person/Spouce/@sin	444555666
Location	Person/Location	<gml:Point> <gml:coordinates>15,15</gml:coordinates> </gml:Point>
Address/StreeNumber	Person/Address/StreetNumber	5
Address/StreetName	Person/Address/StreetName	Main St.
Address/City	Person/Address/City	SomeCity
Address/Province	Person/Address/Province	SomeProvince
Address/PostalCode	Person/Address/Postal_Code	X1X 1X1
Address/Country	Person/Address/Country	Canada
Person/@sin	Person/@sin	111222333

Notice that each relative location paths may begin with the root element name of the property being referenced or with the root element name of the object, *Person.* 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 property which is of node type *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).

[1] SIN = Social Insurance Number

7 Filter

7.1 Introduction

A filter is any valid 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 speification.

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:FeatureId" maxOccurs="unbounded"/>
   </xsd:choice>
</xsd:complexType>
```

The elements <**logicalOps**>, <**comparisonOps**> and <**spatialOps**> are substitution groups for logical, spatial and comparison operators. In addition, using the <**FeatureId**> element, a filter can conveniently encode a reference to one or more enumerated feature instances.

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:Box"/>
         </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"/>
            <xsd:element ref="gml:Box"/>
         </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 **<Overlap>** 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:Box** 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 in some manner.

The Bounding Box (BBOX) is a set of four comma-separated decimal, scientific notation, or integer values (if integers are provided where floating point is needed, the decimal point is assumed at the end of the number). These values specify the minimum X, minimum Y, maximum X, and maximum Y ranges, in that order, expressed in units of the SRS of the feature type(s) begin queried. The four bounding box values indicate the outside edges of a rectangle, as in Figure 5; minimum X is the left edge, maximum X the right, minimum Y the bottom, and maximum Y the top.

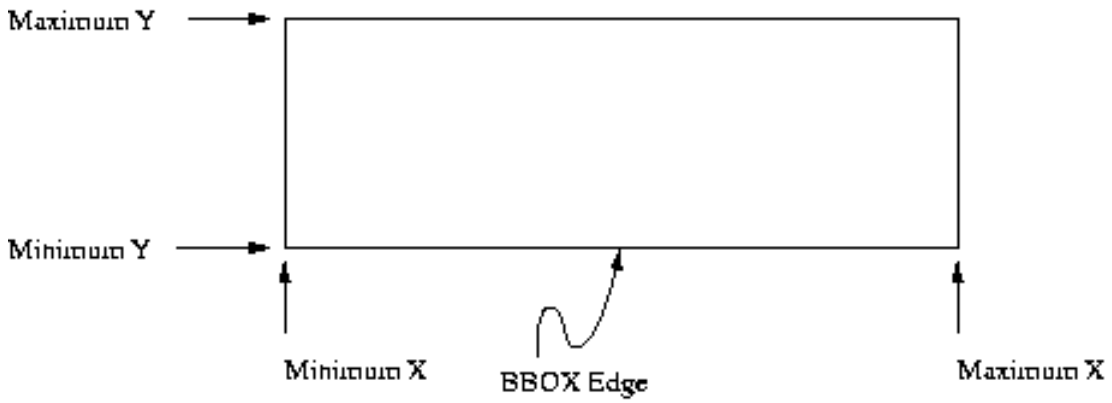

Figure 5 — Bounding Box representation

The semantics of the other operators **Equals**, **Disjoint**, **Touches**, **Within**, **Overlaps**, **Crosses**, **Intersects**, and **Contains** are defined in section 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 semantics of these operators are defined in section 4 of the OpenGIS Catalog Interface Implementation Specification [4].

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: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="escape" type="xsd:string" use="required"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="PropertyIsNullType">
   <xsd:complexContent>
      <xsd:extension base="ogc:ComparisonOpsType">
         <xsd:choice>
            <xsd:element ref="ogc:PropertyName"/>
            <xsd:element ref="ogc:Literal"/>
         </xsd:choice>
      </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. 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 Feature identifiers

11.1 Introduction

A feature identifier is meant to represent a unique identifier for a geographic feature instance within the context of the web service that is serving the feature.

11.2 Encoding

The <**FeatureId**> element is defined by the following XML Schema fragment:

```
<xsd:element name="FeatureId" type="ogc:FeatureIdType"/>
<xsd:complexType name="FeatureIdType">
   <xsd:attribute name="fid" type="xsd:anyURI" use="required"/>
</xsd:complexType>
```

The <**FeatureId**> element can be used to reference instances of geographic features in filter expressions and other XML documents.

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 geometry.xsd schema 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 Schema geometry.xsd.

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 section 11.

16 Filter capabilities

This section of the document defines a capabilities schema that can be included in the capabilities document of services that use filter encoding. The documents 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 two categories: spatial capabilities and scalar 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="ogc:Spatial_CapabilitiesType"/>
         <xsd:element name="Scalar_Capabilities"
                      type="ogc:Scalar_CapabilitiesType"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

A service that supports spatial filtering would include a spatial capabilities section. Spatial capabilities include the ability to filter spatial data 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 fragment:

```
<xsd:complexType name="Spatial_CapabilitiesType">
   <xsd:sequence>
      <xsd:element name="Spatial_Operators"
                   type="ogc:Spatial_OperatorsType"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Spatial_OperatorsType">
   <xsd:choice maxOccurs="unbounded">
      <xsd:element ref="ogc:Equals"/>
      <xsd:element ref="ogc:Disjoint"/>
      <xsd:element ref="ogc:Touches"/>
      <xsd:element ref="ogc:Within"/>
      <xsd:element ref="ogc:Overlaps"/>
      <xsd:element ref="ogc:Crosses"/>
      <xsd:element ref="ogc:Intersect"/>
      <xsd:element ref="ogc:Contains"/>
      <xsd:element ref="ogc:DWithin"/>
      <xsd:element ref="ogc:Beyond"/>
      <xsd:element ref="ogc:BBOX"/>
   </xsd:choice>
</xsd:complexType>
```

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:Logical_Operators"/>
      <xsd:element name="Comparison_Operators"
                   type="ogc:Comparison_OperatorsType"/>
      <xsd:element name="Arithmetic_Operators"
                   type="ogc:Arithmetic_OperatorsType"/>
   </xsd:choice>
</xsd:complexType>
```

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

The **<Comparison_Operators>** element is used to indicate what types of comparison operators are supported by a service. The **<Simple_Comparisons>** 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 **<Arithmetic_Operators>** element is used to indicate what arithmetic operators the a service can support. The contained element **<Simple_Arithmetic>** 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 Section 15, *Encoding Functions*.

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>
   <Spatial_Capabilities>
      <Spatial_Operators>
         <BBOX />
         <Equals />
         <Disjoint />
         <Intersect />
         <Touches />
         <Crosses />
         <Within />
         <Contains />
         <Overlaps />
         <Beyond />
      </Spatial_Operators>
   </Spatial_Capabilities>
   <Scalar_Capabilities>
      <Logical_Operators />
      <Comparison_Operators>
         <Simple_Comparisons />
         <Like />
         <Between />
         <NullCheck />
      </Comparison_Operators>
      <Arithmetic_Operators>
         <Simple_Arithmetic />
         <Functions>
            <Function_Names>
               <FunctionName nArgs="1">MIN</FunctionName>
               <FunctionName nArgs="1">MAX</FunctionName>
               <FunctionName nArgs="1">SIN</FunctionName>
               <FunctionName nArgs="1">COS</FunctionName>
               <!--.
                    . … more functions defined here …
                    .
               -->
            </Function_Names>
         </Functions>
      </Arithmetic_Operators>
   </Scalar_Capabilities>
```

```
</Filter_Capabilities>
```

Example

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

```
<Filter_Capabilities>
   <Spatial_Capabilities>
      <Spatial_Operators>
         <BBOX />
      </Spatial_Operators>
   </Spatial_Capabilities>
   <Scalar_Capabilities>
      <Logical_Operators />
      <Comparison_Operators>
         <Simple_Comparisons />
         <Like />
         <Between />
         <NullCheck />
      </Comparison_Operators>
      <Arithmetic_Operators>
         <Simple_Arithmetic />
      </Arithmetic_Operators>
   </Scalar_Capabilities>
</Filter_Capabilities>
```

Specifically, this service support 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:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
            <gml:coordinates>13.0983,31.5899 35.5472,42.8143</gml:coordinates>
         </gml:Box>
      </Disjoint>
   </Not>
</Filter>
```

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

```
<Filter>
   <BBOX>
      <PropertyName>Geometry</PropertyName>
      <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
         <gml:coordinates>13.0983,31.5899 35.5472,42.8143</gml:coordinates>
      </gml:Box>
   </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:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326”>
               <gml:coordinates>13.0983,31.5899 35.5472,42.8143</gml:coordinates>
            </gml:Box>
         </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.

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

Example 6

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

```
<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>100</LowerBoundary>
      <UpperBoundary>200</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#4326">
         <gml:outerBoundaryIs>
            <gml:LinearRing>
               <gml:coordinates> ... </gml:coordinates>
            </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>
         <Polygon name="1" srsName="EPSG:4326">
            <outerBoundaryIs>
               <LinearRing>
                  <coordinates>
                     -98.5485,24.2633 ...
                  </coordinates>
               </LinearRing>
            </outerBoundaryIs>
         </Polygon>
      </Within>
      <PropertyIsBetween>
         <PropertyName>DEPTH</PropertyName>
         <LowerBoundary>400</LowerBoundary>
         <UpperBoundary>800</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/Address/City</PropertyName>
         <Literal>Toronto</Literal>
      </PropertyIsEqualTo>
   </And>
</Filter>
```

ANNEX B – Filter schema definitions (Normative)

B.1 Introduction

The schemas contained in this section are the normative XML Schema definitions for expressions, filters and filter capabilities. The schema **filter.xsd** should be imported into any schema that intends to use the filter encoding described in this specification. The schema, **filterCapabilities.xsd** should be imported into any capabilities schema that intends to advertise the capabilities of a particular filter processing module.

B.2 expr.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
   targetNamespace="http://www.opengis.net/ogc"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified">

  <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:element name="PropertyName"
               type="ogc:PropertyNameType"
               substitutionGroup="ogc:expression"/>
  <xsd:element name="Function"
               type="ogc:FunctionType"
               substitutionGroup="ogc:expression"/>
  <xsd:element name="Literal"
               type="ogc:LiteralType"
               substitutionGroup="ogc:expression"/>

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

  <xsd:complexType name="ExpressionType" abstract="true"/>
  <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>
  <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>
  <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>
  <xsd:complexType name="PropertyNameType">
    <xsd:complexContent mixed="true">
      <xsd:extension base="ogc:ExpressionType"/>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

B.3 filter.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
   targetNamespace="http://www.opengis.net/ogc"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified">

   <xsd:include schemaLocation="expr.xsd" />
   <xsd:import namespace="http://www.opengis.net/gml"
               schemaLocation="../../gml/2.1/geometry.xsd"/>

   <!-- ============================================= -->
   <!-- FILTER EXPRESSION                             -->
   <!-- ============================================= -->
   <xsd:element name="FeatureId" type="ogc:FeatureIdType"/>
   <xsd:element name="Filter" type="ogc:FilterType"/>

   <!-- ============================================= -->
   <!-- COMPARISON OPERATORS                          -->
   <!-- ============================================= -->
   <xsd:element name="comparisonOps"
                type="ogc:ComparisonOpsType" abstract="true"/>
   <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"/>

   <!-- ============================================= -->
   <!-- SPATIAL OPERATORS (sec 3.2.19.2 99-049)       -->
   <!-- ============================================= -->
   <xsd:element name="spatialOps"
                type="ogc:SpatialOpsType" abstract="true"/>
   <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"/>
<!-- These operatons are from sec 4.2 of OpenGIS Catalog Interface -->
<xsd:element name="DWithin"
            type="ogc:DistanceBufferType"
            substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Beyond"
            type="ogc:DistanceBufferType"
            substitutionGroup="ogc:spatialOps"/>
<!-- This is a convenience operator to allow simple BBOX queries -->
<xsd:element name="BBOX"
            type="ogc:BBOXType"
            substitutionGroup="ogc:spatialOps"/>
<xsd:complexType name="SpatialOpsType" abstract="true"/>

<!-- ============================================== -->
<!-- LOGICAL OPERATORS                              -->
<!-- ============================================== -->
<xsd:element name="logicOps"
            type="ogc:LogicOpsType" abstract="true"/>
<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:complexType name="LogicOpsType" abstract="true"/>

<!-- ============================================== -->
<!-- COMPLEX TYPES                                  -->
<!-- ============================================== -->
<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:FeatureId" maxOccurs="unbounded"/>
   </xsd:choice>
</xsd:complexType>
<xsd:complexType name="FeatureIdType">
   <xsd:attribute name="fid" type="xsd:anyURI" use="required"/>
</xsd:complexType>
<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: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="escape" type="xsd:string" use="required"/>
```

```
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="PropertyIsNullType">
   <xsd:complexContent>
      <xsd:extension base="ogc:ComparisonOpsType">
         <xsd:choice>
            <xsd:element ref="ogc:PropertyName"/>
            <xsd:element ref="ogc:Literal"/>
         </xsd:choice>
      </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>
<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:Box"/>
            </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"/>
            <xsd:element ref="gml:Box"/>
         </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:string" use="required"/>
</xsd:complexType>
<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>
</xsd:schema>
```

B.4 filterCapabilities.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
   targetNamespace="http://www.opengis.net/ogc"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:ogc="http://www.opengis.net/ogc"
   elementFormDefault="qualified">

   <xsd:complexType name="Arithmetic_OperatorsType">
      <xsd:choice maxOccurs="unbounded">
         <xsd:element ref="ogc:Simple_Arithmetic"/>
         <xsd:element name="Functions" type="ogc:FunctionsType"/>
      </xsd:choice>
   </xsd:complexType>
   <xsd:element name="BBOX">
      <xsd:complexType/>
   </xsd:element>
   <xsd:element name="Between">
      <xsd:complexType/>
   </xsd:element>
   <xsd:element name="Beyond">
      <xsd:complexType/>
   </xsd:element>
   <xsd:complexType name="Comparison_OperatorsType">
      <xsd:choice maxOccurs="unbounded">
         <xsd:element ref="ogc:Simple_Comparisons"/>
         <xsd:element ref="ogc:Like"/>
         <xsd:element ref="ogc:Between"/>
         <xsd:element ref="ogc:NullCheck"/>
      </xsd:choice>
   </xsd:complexType>
   <xsd:element name="Contains">
      <xsd:complexType/>
   </xsd:element>
   <xsd:element name="Crosses">
      <xsd:complexType/>
   </xsd:element>
   <xsd:element name="Disjoint">
      <xsd:complexType/>
   </xsd:element>
   <xsd:element name="Equals">
      <xsd:complexType/>
   </xsd:element>
   <xsd:element name="Filter_Capabilities">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="Spatial_Capabilities"
                         type="ogc:Spatial_CapabilitiesType"/>
            <xsd:element name="Scalar_Capabilities"
                         type="ogc:Scalar_CapabilitiesType"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:complexType name="Function_NameType">
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:attribute name="nArgs" type="xsd:string" use="required"/>
         </xsd:extension>
```

```
      </xsd:simpleContent>
   </xsd:complexType>
   <xsd:complexType name="Function_NamesType">
      <xsd:sequence maxOccurs="unbounded">
         <xsd:element name="Function_Name" type="ogc:Function_NameType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="FunctionsType">
      <xsd:sequence>
         <xsd:element name="Function_Names" type="ogc:Function_NamesType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:element name="Intersect">
      <xsd:complexType/>
   </xsd:element>
   <xsd:element name="Like">
      <xsd:complexType/>
   </xsd:element>
   <xsd:element name="Logical_Operators">
      <xsd:complexType/>
   </xsd:element>
   <xsd:element name="NullCheck">
      <xsd:complexType/>
   </xsd:element>
   <xsd:element name="Overlaps">
      <xsd:complexType/>
   </xsd:element>
   <xsd:complexType name="Scalar_CapabilitiesType">
      <xsd:choice maxOccurs="unbounded">
         <xsd:element ref="ogc:Logical_Operators"/>
         <xsd:element name="Comparison_Operators"
                      type="ogc:Comparison_OperatorsType"/>
         <xsd:element name="Arithmetic_Operators"
                      type="ogc:Arithmetic_OperatorsType"/>
      </xsd:choice>
   </xsd:complexType>
   <xsd:element name="Simple_Arithmetic">
      <xsd:complexType/>
   </xsd:element>
   <xsd:element name="Simple_Comparisons">
      <xsd:complexType/>
   </xsd:element>
   <xsd:complexType name="Spatial_CapabilitiesType">
      <xsd:sequence>
         <xsd:element name="Spatial_Operators"
                      type="ogc:Spatial_OperatorsType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="Spatial_OperatorsType">
      <xsd:choice maxOccurs="unbounded">
         <xsd:element ref="ogc:BBOX"/>
         <xsd:element ref="ogc:Equals"/>
         <xsd:element ref="ogc:Disjoint"/>
         <xsd:element ref="ogc:Intersect"/>
         <xsd:element ref="ogc:Touches"/>
         <xsd:element ref="ogc:Crosses"/>
         <xsd:element ref="ogc:Within"/>
         <xsd:element ref="ogc:Contains"/>
         <xsd:element ref="ogc:Overlaps"/>
         <xsd:element ref="ogc:Beyond"/>
      </xsd:choice>
   </xsd:complexType>
   <xsd:element name="Touches">
      <xsd:complexType/>
   </xsd:element>
   <xsd:element name="Within">
      <xsd:complexType/>
   </xsd:element>
</xsd:schema>
```

ANNEX C - Conformance tests (Normative)

Specific conformance tests for a filter encoding processor have not yet been determine 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/.