INTERNATIONAL STANDARD

ISO 19143

First edition
2010-10-15

Geographic information — Filter encoding

Information géographique — Codage de filtres

Reference number
ISO 19143:2010(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

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

ISO 19143 was prepared by Technical Committee ISO/TC 211, *Geographic information/Geomatics*, in collaboration with the Open Geospatial Consortium Inc. (OGC).

Introduction

Filter encoding was originated within the OGC.

A fundamental operation performed on a set of data or resources is that of querying in order to obtain a subset of the data which contains certain desired information that satisfies some query criteria and which is also, perhaps, sorted in some specified manner.

The term “projection clause” is used to describe an encoding for specifying which subset of resource properties are presented in the response to a query.

The term “filter or selection clause” is used to describe an encoding of predicates which are typically used in query operations to specify how data instances in a source dataset should be filtered to produce a result set. Each data instance in the source set is evaluated using the filter expression. The overall filter expression always evaluates to true or false. If the expression evaluates to true, the data instance satisfies the expression and is marked as being in the result set. If the overall filter expression evaluates to false, the data instance is not in the result set. Thus, the net effect of evaluating a filter expression is a set of data or resource identifiers which satisfy the predicates in the expression.

The term “sorting clause” is used to describe an encoding for specifying how the data in a response is ordered prior to being presented.

Such encodings are considered system neutral because using the numerous XML tools available today, XML encoded projection, selection and sorting clauses can be easily validated, parsed and then transformed into whatever target query language is required to retrieve or modify resources stored in some persistent object store. For example an XML encoded query composed of a projection, selection and sorting clauses can be transformed into a SQL “SELECT … FROM … WHERE … ORDER BY …” statement to fetch data stored in a SQL-based relational database. Similarly, the same XML encoded query expression can just as easily be transformed into an XQuery expression in order to retrieve data from XML document.

The XML and KVP encodings of projection, selection and sorting clauses described in this International Standard are common components which can be used together or as individually by a number of web services. Any service that requires the ability to query objects from a web-accessible repository can make use of the XML and KVP encodings of a query expression described in this International Standard. For example the GetFeature operation, defined in ISO 19142, uses the elements derived from definitions in this International Standard to encode query expressions.

Geographic information — Filter encoding

1 Scope

This International Standard describes an XML and KVP encoding of a system neutral syntax for expressing projections, selection and sorting clauses collectively called a query expression.

These components are modular and intended to be used together or individually by other standards which reference this International Standard.

EXAMPLE 1 ISO 19142 makes use of some or all of these components.

This International Standard defines an abstract component, named AbstractQueryExpression, from which other specifications can subclass concrete query elements to implement query operations.

This International Standard also defines an additional abstract query component, named AbstractAdhocQueryExpresison, which is derived from AbstractQueryExpression and from which other specifications can subclass concrete query elements which follow the following query pattern:

> An abstract query element from which service specifications can subclass a concrete query element that implements a query operation that allows a client to specify a list of resource types, an optional projection clause, an optional selection clause, and an optional sorting clause to query a subset of resources that satisfy the selection clause.

This pattern is referred to as an ad hoc query pattern since the server is not aware of the query until it is submitted for processing. This is in contrast to a stored query expression, which is stored and can be invoked by name or identifier.

This International Standard also describes an XML and KVP encoding of a system-neutral representation of a select clause. The XML representation is easily validated, parsed and transformed into a server-specific language required to retrieve or modify object instances stored in some persistent object store.

EXAMPLE 2 An XML encoded filter can 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 can be transformed into an XPath or XPointer expression for fetching data from XML documents.

This International Standard defines the XML encoding for the following predicates.

a) A standard set of logical predicates: and, or and not.

b) A standard set of comparison predicates: equal to, not equal to, less than, less than or equal to, greater than, greater than or equal to, like, is null and between.

c) A standard set of spatial predicates: equal, disjoint, touches, within, overlaps, crosses, intersects, contains, within a specified distance, beyond a specified distance and BBOX.

d) A standard set of temporal predicates: after, before, begins, begun by, contains, during, ends, equals, meets, met by, overlaps and overlapped by.

e) A predicate to test whether the identifier of an object matches the specified value.

This International Standard defines the XML encoding of metadata that allows a service to declare which conformance classes, predicates, operators, operands and functions it supports. This metadata is referred to as Filter Capabilities.

2 Conformance

Few usage scenarios require the full implementation of this International Standard to work. Therefore, service providers may want to specify requirements for only the subset needed to fulfil their service. Or system developers may want to document which subset of this International Standard it is that that they have implemented and conform to. These named conformance classes help in specifying such subsets.

This International Standard defines conformance classes based on the operations and behaviour that a filter encoding service claims to implement. Table 1 indicates which behaviour shall be implemented for each of the conformance classes. The described behaviour shall be implemented for the corresponding conformance class, and the name of the paragraph of the actual detailed abstract test suite in Annex A.

Table 1 — FE conformance classes

Conformance class name	Operation or behaviour	Subclause of the abstract test suite
Query	Service that references this International Standard materializes a concrete query element that is substitutable for fes:AbstractQueryElement.	A.1
Ad hoc Query	Service that references this International Standard materializes a concrete query element that is substitutable for fes:AbstractAdhocQueryElement and materializes a concrete selection clause element that is substitutable for fes:AbstractSelectionClause and materializes a concrete projection clause element that is substitutable for fes:AbstractProjectionClause and materializes a concrete sorting clause element that is substitutable for fes:AbstractSortingClause.	A.2
Functions	Implements functions that are in addition to the operators defined in this International Standard.	A.3
Resource Identification	Implements the ResourceId operator with the rid parameter to allow predicates to be written that allow a specific resource to be queried.	A.4
Minimum Standard Filter	Implements the comparison operators: PropertyIsEqualTo, PropertyIsNotEqualTo, PropertyIsLessThan, PropertyIsGreaterThan, PropertyIsLessThanOrEqualTo, PropertyIsGreaterThanOrEqualTo. Implements the logical operators. Does not implement any additional functions.	A.5
Standard Filter	Implements all the comparison and logical operators and may implement one or more additional functions.	A.6
Minimum Spatial Filter	Implements only the BBOX spatial operator.	A.7
Spatial Filter	Implements the BBOX spatial operator and one or more of the other spatial operators.	A.8
Minimum Temporal Filter	Implements only the During temporal operator.	A.9
Temporal Filter	Implements the During temporal operator and one or more of the other temporal operators.	A.10
Version navigation	Implements ResourceId operator with the parameters that allow versions of resources to be queried (version, startTime, endTime).	A.11
Sorting	Implements sorting of the resources in a response.	A.12
Extended Operators	Implements additional operators not defined in this International Standard.	A.13
Minimum XPath	Implements the minimum required set of XPath capabilities.	A.14
Schema Element Function	Implements the schema-element() XPath function.	A.15

Other standards that include this International Standard shall declare what constitutes a “minimum” filter by declaring the minimum set of conformance classes from Table 1 that shall be implemented.

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 19108:2002, *Geographic information — Temporal schema*

ISO 19125-1:2004, *Geographic information — Simple feature access — Part 1: Common architecture*

ISO 19136:2007, *Geographic information — Geography Markup Language (GML)*

IETF RFC 2396, *Uniform Resource Identifiers (URN): Generic Syntax* (August 1998)

OGC 06-121r3, *OGC Web Services Common Specification*, OGC® Implementation Specification (9 February 2009)

W3C XML, *Extensible Markup Language (XML) 1.0 (Third edition)*, W3C Recommendation (4 February 2004)

W3C XML, *Namespaces, Namespaces in XML*, W3C Recommendation (14 January 1999)

W3C XML, *Path Language, XML Path Language (XPath) 2.0*, W3C Recommendation (23 January 2007)

W3C XML, *Schema Part 1, XML Schema Part 1: Structures*, W3C Recommendation (2 May 2001)

W3C XML, *Schema Part 2, XML Schema Part 2: Datatypes*, W3C Recommendation (2 May 2001)

4 Terms and definitions

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

4.1
attribute
⟨XML⟩ name-value pair contained in an **element**

[ISO 19136:2007, definition 4.1.3]

NOTE In this International Standard, an attribute is an XML attribute unless otherwise specified.

4.2
client
software component that can invoke an **operation** from a **server**

[ISO 19128:2005, definition 4.1]

4.3
coordinate
one of a sequence of n numbers designating the position of a point in n-dimensional space

[ISO 19111:2007, definition 4.5]

4.4
coordinate reference system
coordinate system that is related to an object by a datum

[ISO 19111:2007, definition 4.8]

4.5
coordinate system
set of mathematical rules for specifying how **coordinates** are to be assigned to points

[ISO 19111:2007, definition 4.10]

4.6
element
〈XML〉 basic information item of an XML document containing child elements, **attributes** and character data

[ISO 19136:2007, definition 4.1.23]

4.7
feature
abstraction of real world phenomena

[ISO 19101:2002, definition 4.11]

NOTE A feature can occur as a type or an instance. It is intended that the term “feature type” or “feature instance” be used when only one is meant.

4.8
feature identifier
identifier that uniquely designates a **feature** instance

[ISO 19142:2010, definition 4.8]

4.9
feature reference
Uniform Resource Identifier that identifies a **feature**

4.10
filter capabilities XML
metadata, encoded in XML, that describes which **predicates** defined in this International Standard a system implements

4.11
filter expression
predicate expression encoded using XML

4.12
filter expression processor
component of a system that processes a **filter expression**

4.13
function
rule that associates each **element** from a domain (source, or domain of the function) to a unique element in another domain (target, co-domain, or range)

[ISO 19107:2003, definition 4.41]

4.14
interface
named set of **operations** that characterize the behaviour of an entity

[ISO 19119:2005, definition 4.2]

4.15
literal value
constant, explicitly specified value

NOTE This contrasts with a value that is determined by resolving a chain of substitution (e.g. a variable).

4.16
join predicate
filter expression that includes one or more clauses that constrain properties from two different entity types

NOTE In this International Standard, the entity types are **resource** types.

4.17
namespace
〈XML〉 collection of names, identified by a URI reference which are used in XML documents as **element** names and **attribute** names

[W3C XML Namespaces]

4.18
operation
specification of a transformation or query that an object may be called to execute

[ISO 19119:2005, definition 4.3]

4.19
predicate
set of computational **operations** applied to a data instance which evaluate to true or false

4.20
predicate expression
formal syntax for describing a **predicate**

4.21
property
facet or **attribute** of an object referenced by a name

4.22
request
invocation of an **operation** by a **client**

[ISO 19128:2005, definition 4.10]

4.23
resource
asset or means that fulfils a requirement

[ISO 19115:2003, definition 4.10]

NOTE In this International Standard, a resource is assumed to have identity.

4.24
response
result of an **operation** returned from a **server** to a **client**

[ISO 19128:2005, definition 4.11]

4.25
service
distinct part of the functionality that is provided by an entity through **interfaces**

[ISO 19119:2005, definition 4.1]

4.26
server
particular instance of a **service**

[ISO 19128:2005, definition 4.12]

4.27
tuple
ordered list of values

[ISO 19136:2007, definition 4.1.63]

NOTE In this International Standard, the ordered list is generally a finite sequence of **resources**.

4.28
Uniform Resource Identifier
URI
unique identifier for a **resource**, structured in conformance with IETF RFC 2396

[ISO 19136:2007, definition 4.1.65]

NOTE The general syntax is <scheme>::<scheme-specified-part>. The hierarchical syntax with a namespace is <scheme>://<authority><path>?<query>.

5 Conventions

5.1 Abbreviated terms

BBOX	Bounding Box
CRS	Coordinate Reference System
EBNF	Extended Backus-Naur Form
EPSG	European Petroleum Survey Group
GML	Geography Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
KVP	Keyword-value Pair
OGC	Open Geospatial Consortium
SRS	Spatial Reference System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

URN Uniform Resource Name

UTC Coordinated Universal Time

W3C World Wide Web Consortium

WFS Web Feature Service

XML Extensible Markup Language

5.2 UML notation

5.2.1 Figure 1 describes the Unified Modelling Language (UML) notations used in this International Standard for UML class diagrams.

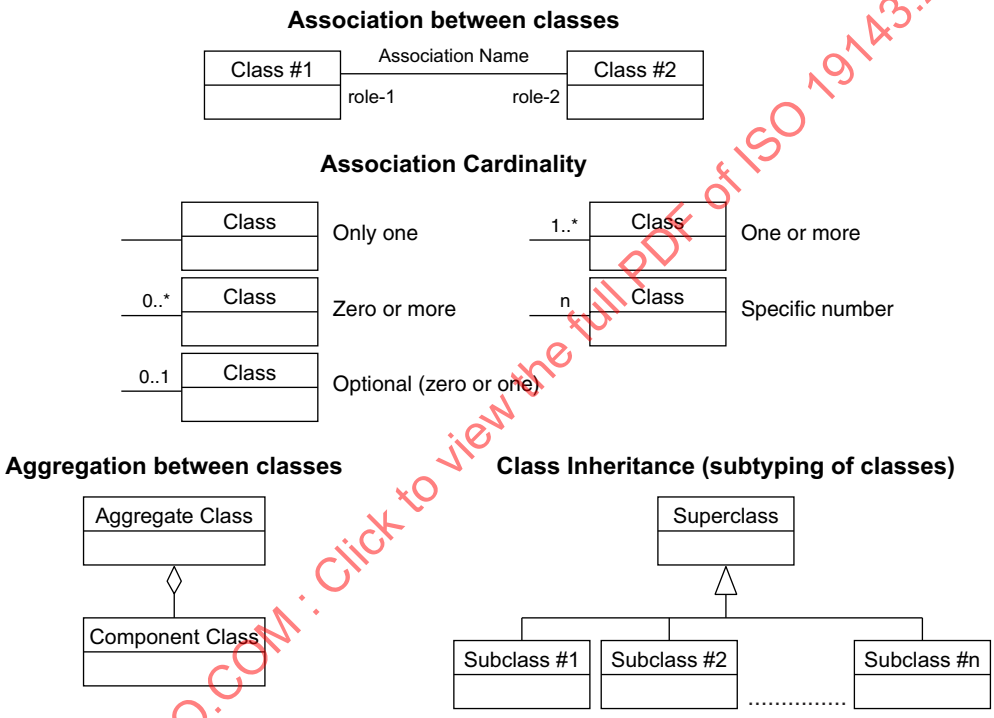

Figure 1 — UML notation in class diagrams

5.2.2 In these class diagrams, the following stereotypes of UML classes are used:

a) <<DataType>> is a descriptor of a set of values that lack identity (independent existence and the possibility of side effects). A DataType is a class with no operations, whose primary purpose is to hold the information.

b) <<Enumeration>> is a data type whose instances form a list of alternative literal values. Enumeration means a short list of well-understood potential values within a class.

c) <<CodeList>> is a flexible enumeration for expressing a long list of potential alternative values. If the list alternatives are completely known, an enumeration shall be used; if the only likely alternatives are known, a code list shall be used.

d) <<Interface>> is a definition of a set of operations that is supported by objects having this interface. An Interface class cannot contain any attributes.

e) <<Type>> is a stereotyped class used for specification of a domain of instances (objects), together with the operations applicable to the objects. A Type class may have attributes and associations.

f) <<Union>> is a list of alternate attributes where only one of those attributes may be present at any time.

See also ISO/TS 19103:2005, 6.8.2 and D.8.3.

5.2.3 In this International Standard, the following standard data types are used:

a) CharacterString is a sequence of characters;

b) LocalisedCharacterString is a CharacterString associated with a locale;

c) Boolean is a value specifying TRUE or FALSE;

d) URI is an identifier of a resource that provides more information;

e) Integer is an integer number.

5.3 Use of examples

This International Standard makes use of XML examples. They are meant to illustrate the various aspects of filters discussed in this International Standard. While every effort has been made to ensure that the examples are well formed and valid, this goal may be sacrificed for the sake of clarity. For instance, many examples are formatted in a specific way to highlight a particular aspect that would render the example invalid from the perspective of an XML validation tool. Furthermore, most examples reference fictitious servers and data.

Thus, this International Standard does not assert that any XML encoded example, copied from this International Standard, would necessarily execute correctly or validate using a particular XML validation tool.

5.4 Namespaces

Namespaces (a specified in W3C XML Namespaces) are used to discriminate XML vocabularies from one another. The following namespaces are normatively used in this International Standard:

a) (http://www.opengis.net/fes/2.0): for the Filter vocabulary;

b) (http://www.opengis.net/gml/3.2): for the GML vocabulary.

5.5 KVP-encoded parameter lists

This International Standard defines both XML and KVP encodings for query and filter expressions. Several of the parameters in the KVP-encoding consist of lists of values (see Table 2) and possibly lists of lists of values. This subclause defines how to encode lists of values as the value of a parameter.

Parameters consisting of lists shall use the comma (“,”) as the delimiter between items in the list. In addition, multiple lists may be specified as the value of a parameter by enclosing each list in parentheses; “(“, ”)”.

EXAMPLE 1 This example shows a list of items.

PARAMETER=item1,item2,item3,item4a%2Citem4b

This list consists of four values: item1, item2, item3 and the value “item4a,item4b”.

NOTE In this example, the embedded comma in the last item has been encoded as per IETF RFC 2396 in order to distinguish it from the commas used in the list of delimit list entries.

EXAMPLE 2 This example shows multiple lists of items assigned to a single parameter.

PARAMETER=(item11,item12,item13)(item21,item22,item23)

5.6 XML Schema fragments

This International Standard makes use of XML Schema (as given in W3C XML Schema Part 1 and W3C XML Schema Part 2) fragments to define the XML encoding of the components of a filter expression. These XML Schema fragments are collected into a set of consolidated schema files in Annex B.

6 Query expressions

6.1 General

A query expression (see Figure 2) is an action that performs a search over some set of resources and returns a subset of those resources. Other standards that reference this International Standard shall assert what a resource is.

EXAMPLE A WFS would assert that a resource is a feature.

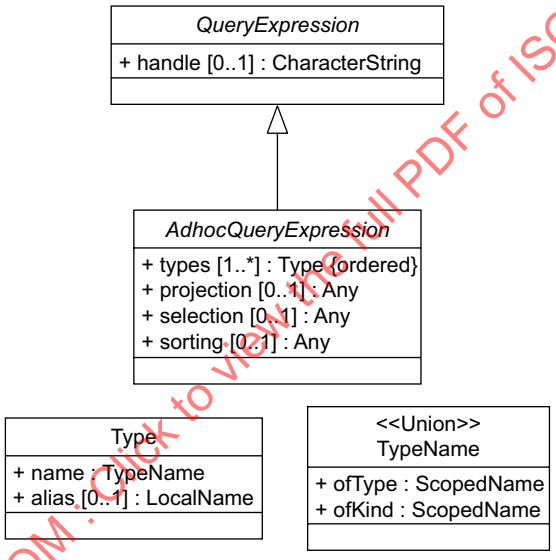

Figure 2 — Query expressions

6.2 Abstract query expressions

This International Standard defines the abstract element fes:AbstractQueryExpression as the head of a substitution group of query expressions. The element fes:AbstractQueryExpression is defined by the following XML Schema fragment:

```
<xsd:element name="AbstractQueryExpression"
             type="fes:AbstractQueryExpressionType" abstract="true"/>
<xsd:complexType name="AbstractQueryExpressionType" abstract="true">
   <xsd:attribute name="handle" type="xsd:string"/>
</xsd:complexType>
```

The fes:AbstractQueryExpression element defines the handle attribute which can be used to assign user-defined identifier to the query expression for the purpose of error handling or correlating the response to a query, from within a series of queries, with the source query expression.

International Standards that reference this International Standard shall declare the types(s) of resources that can be queried and shall derive query expressions from fes:AbstractQueryExpression.

6.3 Ad hoc query expression

6.3.1 General considerations

A fundamental type of query expression is the ad hoc query expression. It is ad hoc in the sense that the query is not known before the time it is being executed as, for example, a stored query would be.

An ad hoc query expression is a query expression that contains the names of one or more resource types to query, an optional projection clause enumerating the properties of the resource to present in the response, an option selection clause that constraints the properties of those resources types in order to define a result set and an optional sorting clause specifying the order in which the result set is presented.

This subclause defines the head of an substitution group called fes:AbstractAdhocQueryExpression from which standards that reference this International Standard can derive concrete ad hoc query expressions.

6.3.2 XML encoding

The following XML Schema fragment define the abstract element fes:AbstractAdhocQueryExpression

```
<xsd:element name="AbstractAdhocQueryExpression"
             type="fes:AbstractAdhocQueryExpressionType"
             substitutionGroup="fes:AbstractQueryExpression"
             abstract="true"/>
<xsd:complexType name="AbstractAdhocQueryExpressionType" abstract="true">
   <xsd:complexContent>
      <xsd:extension base="fes:AbstractQueryExpressionType">
         <xsd:sequence>
            <xsd:element ref="fes:AbstractProjectionClause"
                         minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="fes:AbstractSelectionClause" minOccurs="0"/>
            <xsd:element ref="fes:AbstractSortingClause" minOccurs="0"/>
         </xsd:sequence>
         <xsd:attribute name="typeNames"
                        type="fes:TypeNamesListType" use="required"/>
         <xsd:attribute name="aliases"
                        type="fes:AliasesType"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="TypeNamesListType">
    <xsd:list itemType="fes:TypeNamesType"/>
</xsd:simpleType>
<xsd:simpleType name="TypeNamesType">
    <xsd:union memberTypes="fes:SchemaElement xsd:QName"/>
</xsd:simpleType>
<xsd:simpleType name="SchemaElement">
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="schema\-element\(.+\)"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="AliasesType">
   <xsd:list itemType="xsd:NCName"/>
</xsd:simpleType>
<xsd:element name="AbstractProjectionClause" abstract="true"/>
<xsd:complexType name="AbstractProjectionClauseType" abstract="true"/>
<xsd:element name="AbstractSelectionClause" abstract="true"/>
<xsd:complexType name="AbstractSelectionClauseType" abstract="true"/>
<xsd:element name="AbstractSortingClause" abstract="true"/>
<xsd:complexType name="AbstractSortingClauseType" abstract="true"/>
```

6.3.3 KVP-encoding

Table 2 defines the KVP-encoding for an ad hoc query expression.

Table 2 — KVP-encoding for ad hoc query expression

URL Component	O/M[a]	Description
TYPENAMES	M[b]	A comma-separated list of resource types to query. Specifying more that one name indicates that a join is being performed.
ALIASES	O	A comma-separated list of aliases for the resource types listed as the value of the TYPENAMES parameter.
Projection clause		
PROPERTYNAME	O	If more that one feature type name is specified as the value of the TYPENAMES keyword, a list of parameter lists shall be specified (see 5.5.). Each sublist shall correspond 1:1 with each feature type name listed as the value of the TYPENAMES parameter.
Selection clause		
FILTER (Mutually exclusive with RESOURCEID and BBOX)	O	The value of the parameter shall be a filter expression encoded using the language specified by the FILTER_LANGUAGE parameter.
FILTER_LANGUAGE	O	Indicates the predicate language used to encode the filter expression that is the value of the FILTER parameter. The default value urn:ogc:def:queryLanguage:OGC-FES:Filter shall be used to indicate that the value of FILTER parameter is a string encoding the filter using an XML fragment as defined in this International Standard.
RESOURCEID (Mutually exclusive with FILTER and BBOX)	O	A comma-separated list of resource identifiers to retrieve from some data store.
BBOX (Mutually exclusive with FILTER and RESOURCEID)	O	A bounding rectangle, encoded as specified in OGC 06-131r3, indicating that all resources which intersect that BBOX shall be retrieved from some data store.
Sorting clause		
SORTBY	O	The SORTBY parameter is used to specify a list of value references that should be used to order (upon presentation) the set of resource instances that satisfy the query. The value of the SORTBY parameter shall have the form "PropertyName [ASC|DESC][,PropertyName [ASC|DESC],…]" where the letters ASC are used to indicate an ascending sort and the letters DESC are used to indicate a descending sort. If neither ASC nor DESC are specified, the default sort order shall be ascending. An example value might be: "SORTBY=Field1 DESC,Field2 DESC,Field3". In this case the results are sorted by Field 1 descending, Field2 descending and Field3 ascending.

[a] O = Optional, M = Mandatory.

[b] Standards that reference this International Standard may change the requirement for the TYPENAME parameter. In such cases, the referencing standard shall document whether the TYPENAME parameter is mandatory, optional or mandatory in some cases and optional in others.

6.3.3.1 Parameter discussion

6.3.3.1.1 typeNames parameter

The mandatory typeNames parameter shall be used within an ad hoc query expression to encode the names of one or more correlated resource types to be queried. Individual resource type names shall be encoded as QName (as given in W3C XML Schema Part 2).

NOTE For the KVP-encoding of the typeNames parameter, see 5.5 for encoding lists of values.

International Standards that derive query expressions from fes:AbstractQueryExpression shall declare the resource type(s) that can be queried.

6.3.3.1.2 aliases parameter

The optional aliases parameter may be used within an ad hoc query expression to specify alternate names for the resource type names specified as the value of the typeNames parameter. A resource type alias may be used anywhere; the resource type name may be used within the context of the query expression.

The number of list elements in the value of the aliases parameter shall match the number of corresponding resource type names in the value of the typeNames parameter and shall be correlated 1:1.

EXAMPLE 1 < … typeNames="ns1:ResourceType1, ns2:ResourceType2" aliases="A B" …>

This example encodes an ad hoc query expression fragment that queries the resource types ns1:ResourceType1 and ns2:ResourceType2 which are aliased to A and B. Thus, the tokens A or B can be used within the filter expression of the query expression as alternate names for the resource types ns1:ResourceType1 and ns2:ResourceType2.

Each alias specified in the value of aliases attribute shall be unique within the context of a single query expression.

If the aliases attribute is used, an alias shall be specified for each resource type name listed as the value of typeNames attribute.

Aliases are typically used in query expressions that perform a join operation to support self-joins. That is a join of one resource type back to itself.

EXAMPLE 2 typeNames="myns:ResType1 myns:ResType1" aliases="a b"

In this example, the first resource type, myns:ResType1, is aliased to the name "a" and the second resource type, myns:Feat1, is aliased to the name "b". Thus properties from the first instance of myns:ResType1 can be referenced in a request as "/a/myns:property_name" and properties from the second instance of myns:ResType2 can be referenced in a request as "/b/myns:property_name" where the token "myns:property_name" is used as a place holder for the name of any property of the resource type myns:ResType1.

6.3.3.1.3 Projection clause

A projection clause encodes a list of optional resource properties that shall be available in a query response.

For XML-encoded requests, specifications which implement the projection clause of an ad hoc query expression shall define a concrete element derived from fes:AbstractAdhocProjectionClause.

For KVP-encoded requests, the PROPERTYNAME keyword shall be used to encode the projection clause. The value of the PROPERTYNAME keyword shall be a comma-separated list of property names.

6.3.3.1.4 Selection clause

The selection clause defines a set of query predicates that shall be applied to a dataset in order to define a subset of data to be operated upon.

Services that implement this International Standard shall use the fes:Filter element, which is substitutable for fes:AbstractSelectionClause, to encode the selection clause of a query expression.

For XML-encoded requests, the selection clauses shall be encoded using the fes:Filter element.

For KVP-encoded requests, the select clause shall be encoded using the keywords FILTER, FILTER_LANGUAGE, RESOURCEID, BBOX (see Table 2). The parameters FILTER, RESOURCEID and BBOX are mutually exclusive. In the event that a selection clause specifies more than one of these parameters, an OperationNotSupported (as given in OGC 06-121r3, Table 25) exception shall be raised.

6.3.3.1.5 Join queries

A join query finds tuples (i.e. pairs, triples, etc.) of resources, among a list of resource types, that satisfy a filter expression which includes join predicates. If the filter expression is satisfied, that tuple of resources is considered to be in the result set of the query expression.

A join query is encoded by:

a) listing the resource types to join using the typeNames parameter (see 6.3.3.1.1);

b) specifying join predicates in the selection clause that reference properties of the resource types listed as the values of the typeNames parameter (see 6.3.3.1.1).

Services that implement join queries shall implement an inner join meaning that only resource tuples which match the join conditions shall be returned in the result set.

6.3.3.1.6 schema-element() function

If the list of values for the typeNames parameters contains a single QName then the schema-element() function can be used to trigger a sequence of queries on the specified resource type and any resource type whose object elements are in the substitution group of the specified resource type.

EXAMPLE typeNames="schema-element(ns1:Vehicles)" might, along with ns1:Vehicle, query the resource types ns1:Cars, ns1:Boats, etc …

The schema-element() function shall not be used if a join operation (see 6.3.3.1.5) is being performed.

6.3.3.1.7 Sorting clause

A sorting clause can be used to assert the order in which resources shall appear in response to an ad hoc query expression.

For XML-encoded requests, standards that reference this International Standard shall use the fes:SortBy element (see Clause 9), which is substitutable for fes:AbstractSortingClause, to encode the sorting clause of an ad hoc query expression.

For KVP-encoded requests, the keyword SORTBY shall be used to encode a sorting clause (see Table 2).

7 Filter

7.1 General considerations

A filter (see Figure 3) is used to identify a subset of resources from a collection of resources whose property values satisfy a set of logically connected predicates. If the property values of a resource satisfy all the predicates in a filter then that resource is considered to be part of the resulting subset.

This clause defines the XML encoding of a filter as a set of predicate expressions, contained within the root element fes:Filter, encoded using the elements defined herein.

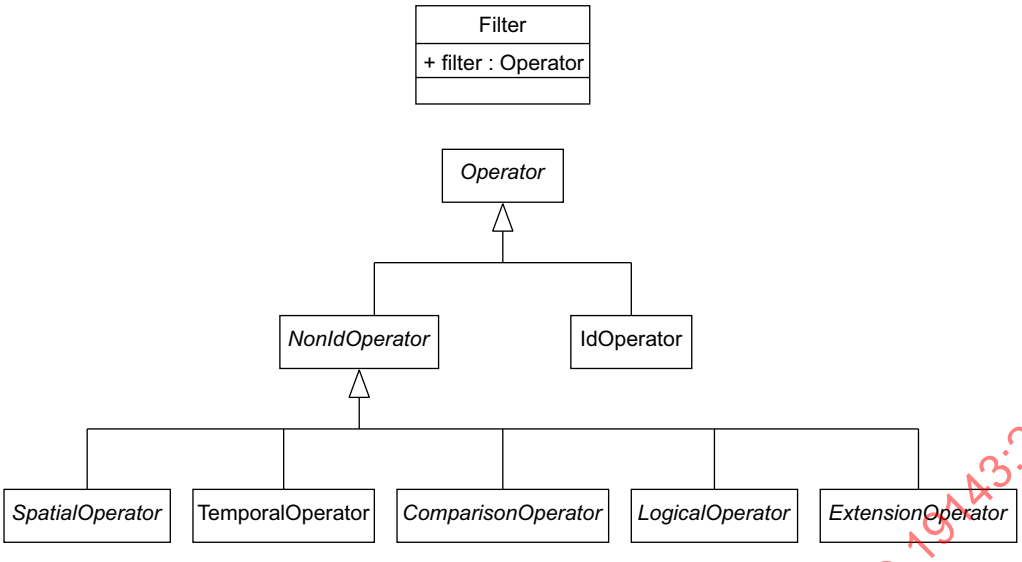

Figure 3 — Filter

7.2 Encoding

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

```
<xsd:element name="Filter"
             type="fes:FilterType"
             substitutionGroup="fes:AbstractSelectionClause"/>
<xsd:complexType name="FilterType">
   <xsd:complexContent>
      <xsd:extension base="fes:AbstractSelectionClauseType">
         <xsd:sequence>
            <xsd:group ref="fes:FilterPredicates"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:group name="FilterPredicates">
  <xsd:choice>
      <xsd:element ref="fes:comparisonOps"/>
      <xsd:element ref="fes:spatialOps"/>
      <xsd:element ref="fes:temporalOps"/>
      <xsd:element ref="fes:logicOps"/>
      <xsd:element ref="fes:extensionOps"/>
      <xsd:element ref="fes:Function"/>
      <xsd:element ref="fes:_Id" maxOccurs="unbounded"/>
   </xsd:choice>
</xsd:group>

<xsd:element name="extensionOps"
             type="fes:ExtensionOpsType"
             abstract="true"/>
<xsd:complexType name="ExtensionOpsType" abstract="true"/>
```

The elements contained within the fes:Filter element are discussed in detail in subsequent clauses.

7.3 Expressions

7.3.1 General considerations

An expression (see Figure 4) is a combination of one or more symbols that form part of a predicate. In this International Standard, valid symbols shall be encoded using the XML elements defined in this International Standard. Expressions are encoded by nesting these elements to form XML fragments that validate against the schemas in Annex C.

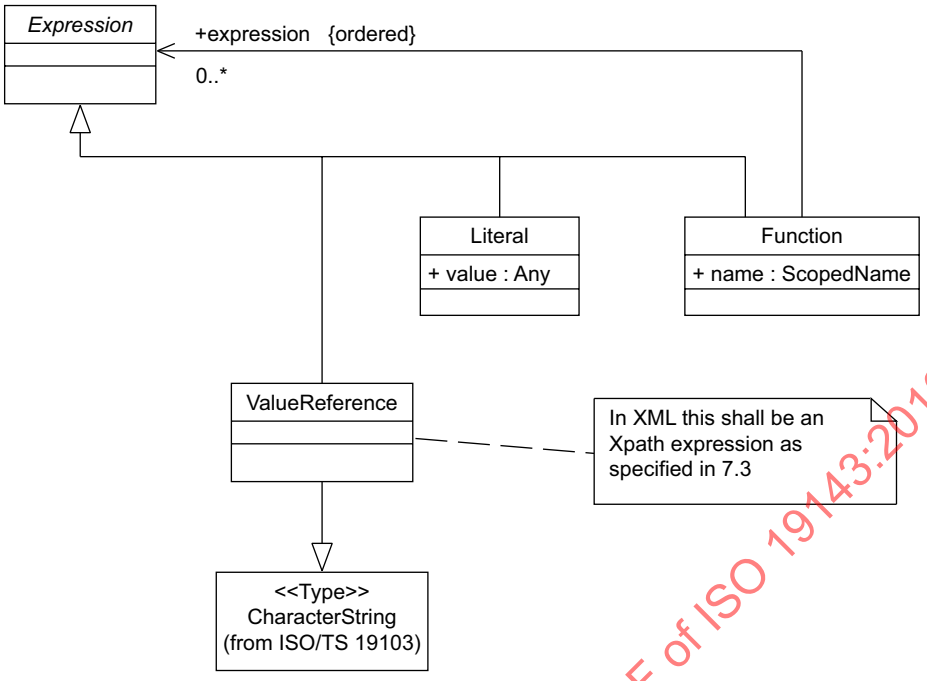

Figure 4 — Expression

7.3.2 Encoding

An expression can be formed using the following XML elements:

- `fes:ValueReference`
- `fes:Literal`
- `fes:Function`

These elements all belong to the substitution group expression and 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 **fes:expression** element is an abstract element 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 fes:expression element is:

```
<xsd:element name="expression" abstract="true"/>
```

7.4 Value references

7.4.1 General considerations

A value reference is a string that represents a value that is to be evaluated by a predicate. The string can, for example, be the name of a property of a resource or a path expression that represents some value that is part of the property of a resource. At runtime, a predicate is evaluated by replacing the value reference by the value it refers to and then executing whatever test is encoded by the predicate.

7.4.2 Encoding

The following fragment defines the XML encoding for the fes:ValueReference element:

```
<xsd:element name="ValueReference" type="xsd:string" substitutionGroup="fes:expression"/>
```

7.4.3 Property names in GML

The fes:ValueReference element can used to specify the name of any property of an object whose value shall be tested by a predicate in a filter expression. For services that implement this International Standard and use GML (see ISO 19136), property names shall be encoded using XML elements as described in the Extensible Markup Language (XML) 1.0 (see W3C XML) specification. In addition, GML property names may be qualified with a namespace, in which case the name shall conform to the Namespaces in XML (as given in W3C XML Namespaces) specification. The following definition is taken from Clauses 2 and 3 of W3C XML Namespaces:

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 given in Annex B of Namespaces in XML (W3C XML Namespaces).

EXAMPLE 1 Examples of valid property names are: age, temperature, _KHz, INWATERA_1M.WKB_GEOM

EXAMPLE 2 Examples of invalid property names are: +Domain, 123_SomeName

7.4.4 XPath expressions

In cases where the data model of the service that implements this International Standard is represented as XML, as is the case with ISO 19142 where GML (see ISO 19136) is used, value references can refer to parts of a complex property and shall be encoded using the XML Path Language (given in W3C XML Path Language).

The XML Path Language (as given in W3C XML Path Language) specification is a language for addressing parts of an XML document, or in the case of this International Standard, for referencing XML elements and attributes that represent the properties of an object encoded in XML.

This International Standard 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, services that implement this specification and require the use of XPath, shall support a subset of the XPath language. The following set of rules defines this subset of the XPath language.

a) The abbreviated form of the child and attribute axis specifier (see W3C XML Path Language) shall be supported.

b) The context node shall be the resource element, except in the case of a join operation, in which case the context node shall be the parent of the resource element.

c) Each step in the path may include an XPath predicate.

d) At least the following predicate expression items shall be supported:

   1) a positive non-zero integer may be used to indicate which child of the context node should be selected (i.e. an index). This allows ordered properties with repeatable values to be specifically referenced;

   2) an equality predicate for the form “=value” may be used to indicate which child of the context node should be selected based on its value. This allows properties with repeatable value to be specifically referenced by value;

3) equality tests of the form “child=value” may be used to identify a specific object property by constraining the child elements of the property. Equality tests can be logically combined using the “and” or “or” operators.

NOTE Consider the following example:

```
<Building>
  <name>City hall</name>
  <addresses>
   <Address>
     <city>Bonn</city>
     <street>Oxfordstrasse</street>
     <number>1</number>
   </Address>
   <Address>
     <city>Bonn</city>
     <street>Breitestrasse</street>
     <number>5</number>
   </Address>
 </addresses>
</Building>


<Filter>
  <PropertyIsEqualTo>
   <ValueReference>addresses/Address/city</ValueReference>
   <Literal>Bonn</Literal>
  </PropertyIsEqualTo>
</Filter>
```

In this example “City hall” has two address values. The following XPath expression can be used to reference the number of the first Address value:

```
addresses/Address[street="Oxfordstrasse"]/number
```

e) The last step of the XPath expression shall be a resource property or sub-component of a resource property.

f) The function schema-element() may be supported.

Other standards that reference this International Standard shall declare whether implementing the schema-element() function is mandatory or optional.

EXAMPLE 1 The WFS standard (see ISO 19142) declares that implementing the schema-element() function is optional and defines a conformance class to test whether a WFS implementation supports that method or not.

Other standards that reference this International Standard may extend this XPath subset as required.

EXAMPLE 2 The WFS standard (see ISO 19142) extends this subset by allowing the use of an accessor function called wfs:valueOf().

Annex D defines the subset of the XPath grammar for the path expressions used in this International Standard. It follows the EBNF (see ISO/IEC 14977) notation defined in the XPath 2.0 specification, Appendix A “XPath grammar” (see http://www.w3.org/TR/xpath20/#nt-bnf).

7.5 Literals

7.5.1 General considerations

This subclause defines how the XML encoding of a filter expression defined in this International Standard encodes literal values. A literal value is any part of a statement or expression which should be used as provided.

7.5.2 Encoding

The following XML Schema fragment defines the fes:Literal element:

```
<xsd:element name="Literal" type="fes:LiteralType"
   substitutionGroup="fes:expression"/>
<xsd:complexType name="LiteralType" mixed="true">
   <xsd:sequence>
      <xsd:any minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="type" type="xsd:QName"/>
</xsd:complexType>
```

The fes:Literal element is used to encode any explicitly stated value. If the literal value is a geometric value, the value shall be encoded following the rules of GML (defined in ISO 19136).

Literals can, optionally, be typed using the type attribute. The value of the attribute type is the name of type from some type system.

EXAMPLE The following XML fragment: <Literal type="xs:date">1963-10-13</Literal> encodes a date value. The type of the value is xs:date as defined in (see W3C XML Schema Part 2).

7.6 Functions

7.6.1 General considerations

This section defines the encoding of single valued functions using the fes:Function element. A function is a named procedure that performs a distinct computation. A function can accept zero or more arguments as input and generates a single result.

Functions may be used to extend the filter syntax with additional operators that can be used in filter expressions.

If a standard that references this International Standard extends the filter syntax using the fes:Function element, any additional functions shall be documented in the referencing standard. An implementation of the referencing standard shall, in its filter capabilities (see 7.14.6), declare these additional functions.

7.6.2 Encoding

The following XML Schema fragment declares the fes:Function element:

```
<xsd:element name="Function" type="fes:FunctionType"
   substitutionGroup="fes:expression"/>
<xsd:complexType name="FunctionType">
   <xsd:sequence>
      <xsd:element ref="fes:expression"
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="xsd:string" use="required"/>
</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 fes:Function element. The arguments themselves are in-turn expressions (see 7.3) and shall appear in the order in which they are defined in the filter capabilities document (see 7.14.6).

Functions that can be used as filter operators and can thus be combined using logical operations (see 7.10) shall return a boolean result.

EXAMPLE The following XML fragment uses a function to invoke an operator named “ClassifiedAs” to find roads classified as major highways within some area of interest. The function accepts two arguments; the name of a classification scheme, and the name of a node within that classification scheme.

```
<fes:Filter>
   <fes:And>
      <fes:Function name="ClassifiedAs">
         <fes:Literal>RoadTaxonomy</fes:Literal>
         <fes:Literal>Major Highway</fes:Literal>
      </fes:Function>
      <fes:BBOX>
         <fes:ValueReference>/RS1/geometry</fes:ValueReference>
         <gml:Envelope srsName="urn:ogc:def:crs:EPSG::1234">
            <gml:lowerCorner>10 10</gml:lowerCorner>
            <gml:upperCorner>20 20</gml:upperCorner>
         </gml:Envelope>
      </fes:BBOX>
   </fes:And>
</fes:Filter>
```

7.7 Comparison operators

7.7.1 General considerations

A comparison operator (see Figure 5) 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.

A service that implements this International Standard shall, in its filter capabilities (see 7.14.3), declare which comparison operators it supports.

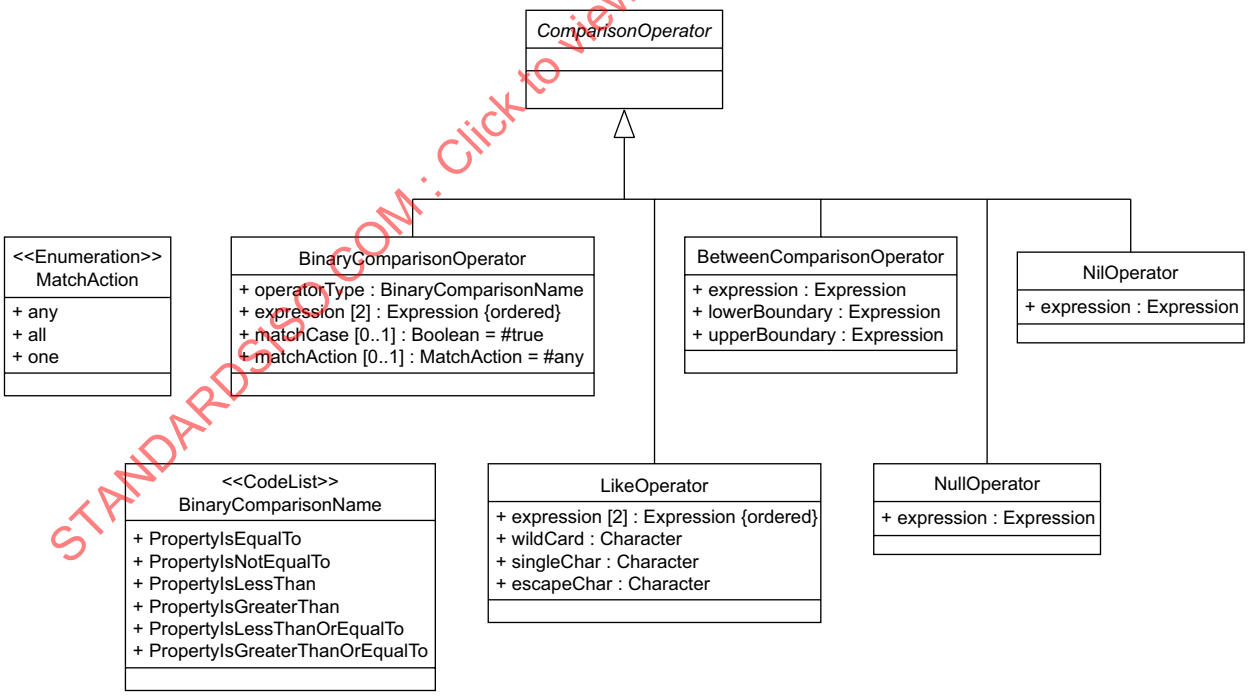

Figure 5 — ComparisonOperator

7.7.2 Encoding

The following XML Schema fragment defines the XML encoding for comparison operators:

```
<xsd:element name="comparisonOps"
             type="fes:ComparisonOpsType"
             abstract="true"/>
<xsd:complexType name="ComparisonOpsType" abstract="true"/>
<xsd:element name="PropertyIsEqualTo"
             type="fes:BinaryComparisonOpType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsNotEqualTo"
             type="fes:BinaryComparisonOpType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsLessThan"
             type="fes:BinaryComparisonOpType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsGreaterThan"
             type="fes:BinaryComparisonOpType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsLessThanOrEqualTo"
             type="fes:BinaryComparisonOpType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsGreaterThanOrEqualTo"
             type="fes:BinaryComparisonOpType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsLike"
             type="fes:PropertyIsLikeType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsNull"
             type="fes:PropertyIsNullType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsNil"
             type="fes:PropertyIsNilType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsBetween"
             type="fes:PropertyIsBetweenType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:complexType name="BinaryComparisonOpType">
   <xsd:complexContent>
      <xsd:extension base="fes:ComparisonOpsType">
         <xsd:sequence>
            <xsd:element ref="fes:expression" minOccurs="2" maxOccurs="2"/>
         </xsd:sequence>
         <xsd:attribute name="matchCase" type="xsd:boolean"
                        use="optional" default="true"/>
         <xsd:attribute name="matchAction" type="fes:MatchActionType"
                        use="optional" default="Any"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="MatchActionType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="All"/>
      <xsd:enumeration value="Any"/>
      <xsd:enumeration value="One"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="PropertyIsLikeType">
   <xsd:complexContent>
      <xsd:extension base="fes:ComparisonOpsType">
         <xsd:sequence>
            <xsd:element ref="fes:expression" minOccurs="2" maxOccurs="2"/>
         </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="fes:ComparisonOpsType">
         <xsd:sequence>
            <xsd:element ref="fes:expression" minOccurs="0"/>
```

```
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <xsd:complexType name="PropertyIsNilType">
      <xsd:complexContent>
         <xsd:extension base="fes:ComparisonOpsType">
            <xsd:sequence>
               <xsd:element ref="fes:expression" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="nilReason" type="xsd:string"/>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <xsd:complexType name="PropertyIsBetweenType">
      <xsd:complexContent>
         <xsd:extension base="fes:ComparisonOpsType">
            <xsd:sequence>
               <xsd:element ref="fes:expression"/>
               <xsd:element name="LowerBoundary" type="fes:LowerBoundaryType"/>
               <xsd:element name="UpperBoundary" type="fes:UpperBoundaryType"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <xsd:complexType name="LowerBoundaryType">
      <xsd:choice>
         <xsd:element ref="fes:expression"/>
      </xsd:choice>
   </xsd:complexType>
   <xsd:complexType name="UpperBoundaryType">
      <xsd:sequence>
         <xsd:element ref="fes:expression"/>
      </xsd:sequence>
   </xsd:complexType>
```

7.7.3 Prameter discussion

7.7.3.1 Binary comparisons

This International Standard defines a standard set of comparison operators (=,<,>,>=,<=,<>): equal to, less than, greater than, less than or equal to, greater than or equal to and not equal to. These comparison operators are encoded using the complex type BinaryComparisonOpType.

7.7.3.2 matchCase parameter

The matchCase attribute, which is of type Boolean, shall be used to specify how a filter expression processor should perform string comparisons. A value of true means that string comparisons shall match case. This shall be the default value. The value false means that string comparisons are performed caselessly.

7.7.3.3 matchAction parameter

The matchAction attribute can be used to specify how the comparison predicate shall be evaluated for a collection of values (e.g. in XML, properties having maxOccurs > 1) and not including some additional context to identify a specific value from the collection to be tested. Possible values for the attribute are: *All*, *Any* or *One*. A value of *All* means that all values in the collection shall satisfy the predicate. A value of *Any* means that any of the value in the collection can satisfy the predicate. Finally, a value of *One* means that only one of the values in the collection shall satisfy the predicate.

If the value of the matchAction attribute is *One*, additional context (e.g. XPath index) can be included to indicate which value in the collection should satisfy the predicate.

EXAMPLE The following example illustrates the use of the matchAction attribute. Consider the following XML fragment, which is an instance of a GML (see ISO 19136) feature:

```
<ex:Building gml:id="b123">
 <gml:name>175 Fifth Ave.</gml:name>
 <gml:name>Flatiron</gml:name>
 <gml:name>Acme Building</gml:name>
 <!-- ... -->
</ex:Building>
```

and consider the following filter expression:

```
<fes:Filter>
  <fes:PropertyIsEqualTo matchAction="...">
   <fes:ValueReference>gml:name</fes:ValueReference>
   <fes:Literal>Flatiron</fes:Literal>
  </fes:PropertyIsEqualTo>
</fes:Filter>
```

If the value of the matchAction attribute is set to *Any*, this predicate will evaluate to true since there is at least one gml:name value that satisfied the predicate. If the value of the matchAction attribute is *All*, this predicate will evaluate to false since not all gml:name values are Flatiron. Finally, if the matchAction attribute is set to *One* then the expression will evaluate to true since only one gml:name value is Flatiron.

If the value of the matchAction attribute is Any or All, the ValueReference XPath expression shall not include an index predicate. If the matchAction attribute is One an XPath index predicate may be specified and the predicate shall only evaluate to true if not only one value matches the predicate but the specific value indicates by the index matches the value.

7.7.3.4 PropertyIsLike operator

The PropertyIsLike element is intended to encode a character string comparison operator with pattern matching. A combination of regular characters, the wildCard character, the singleChar character, and the escapeChar character define the pattern. 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.

7.7.3.5 PropertyIsNull operator

The PropertyIsNull operator tests the specified property to see if it exists in the resource being evaluated. This corresponds to checking whether the property exists in the real-world.

7.7.3.6 PropertyIsNil operator

The PropertyIsNil operator tests the content of the specified property and evaluates if it is nil. The operator can also evaluate the nil reason using the nilReason parameter. The implied operator for evaluating the nil reason is “equals”.

7.7.3.7 PropertyIsBetween operator

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

7.8 Spatial operators

7.8.1 General considerations

A spatial operator (see Figure 6) shall determine whether its geometric arguments satisfy the stated spatial relationship. The operator shall evaluate to true if the spatial relationship is satisfied. Otherwise, the operator shall evaluate to false.

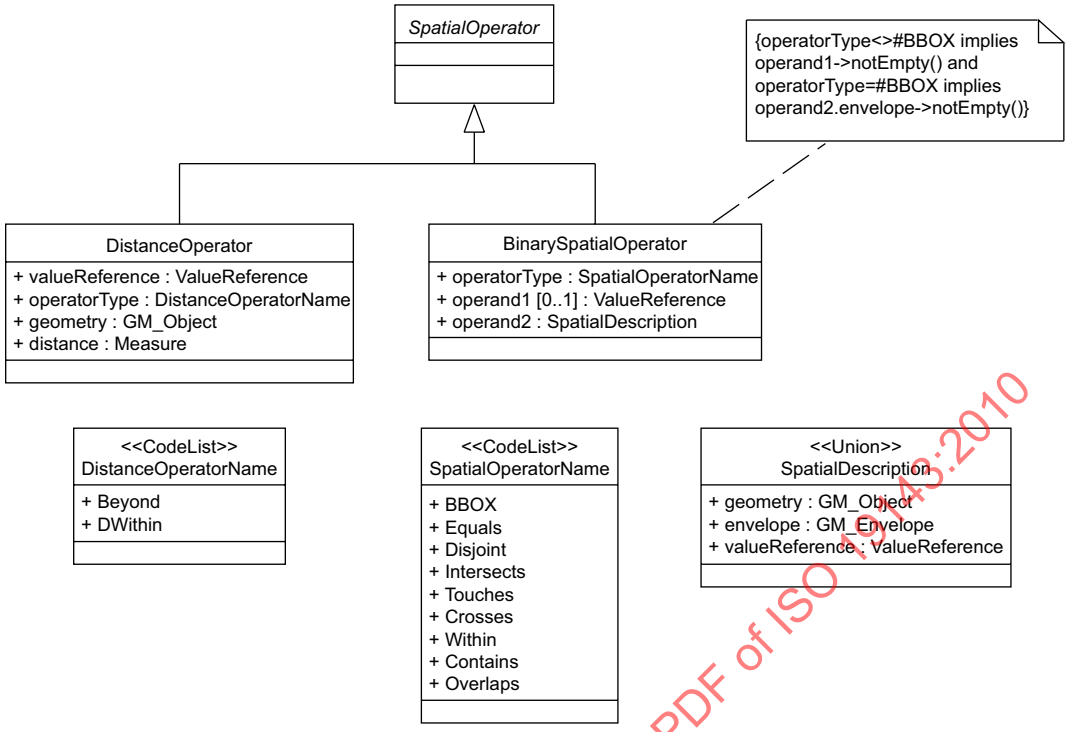

Figure 6 — SpatialOperator

Table 3 maps the spatial operators described in this International Standard to the set of spatial operators defined in ISO 19125-1.

Table 3 — Mapping of ISO 19143 spatial operators to ISO 19125-1 spatial operators

ISO 19143 spatial operator	ISO 19125-1 spatial operator
Equals	Equals
Disjoin	Disjoint
Touches	Touches
Within[a]	Within
Overlaps	Overlaps
Crosses	Crosses
Intersects	Intersects
Contains[a]	Contains
DWithin	N/A
Beyond	N/A
BBOX	equivalent to NOT Disjoint with gml:Envelope
[a] (A Within B) implies that (B Contains A) (see ISO 19125-1:2004, 6.1.14.3)	

A service that implements this International Standard shall, in its filter capabilities (see 7.14.4), declare which spatial operators it supports.

7.8.2 Encoding

The following XML Schema fragment defines the XML encoding for spatial operators:

```
<xsd:element name="spatialOps" type="fes:SpatialOpsType" abstract="true"/>
<xsd:complexType name="SpatialOpsType" abstract="true"/>
<xsd:element name="Equals"
             type="fes:BinarySpatialOpType"
             substitutionGroup="fes:spatialOps"/>
<xsd:element name="Disjoint"
             type="fes:BinarySpatialOpType"
             substitutionGroup="fes:spatialOps"/>
<xsd:element name="Touches"
             type="fes:BinarySpatialOpType"
             substitutionGroup="fes:spatialOps"/>
<xsd:element name="Within"
             type="fes:BinarySpatialOpType"
             substitutionGroup="fes:spatialOps"/>
<xsd:element name="Overlaps"
             type="fes:BinarySpatialOpType"
             substitutionGroup="fes:spatialOps"/>
<xsd:element name="Crosses"
             type="fes:BinarySpatialOpType"
             substitutionGroup="fes:spatialOps"/>
<xsd:element name="Intersects"
             type="fes:BinarySpatialOpType"
             substitutionGroup="fes:spatialOps"/>
<xsd:element name="Contains"
             type="fes:BinarySpatialOpType"
             substitutionGroup="fes:spatialOps"/>
<xsd:element name="DWithin"
             type="fes:DistanceBufferType"
             substitutionGroup="fes:spatialOps"/>
<xsd:element name="Beyond"
             type="fes:DistanceBufferType"
             substitutionGroup="fes:spatialOps"/>
<xsd:element name="BBOX"
             type="fes:BBOXType"
             substitutionGroup="fes:spatialOps"/>
<xsd:complexType name="BinarySpatialOpType">
   <xsd:complexContent>
      <xsd:extension base="fes:SpatialOpsType">
         <xsd:sequence>
            <xsd:element ref="fes:ValueReference"/>
            <xsd:choice>
               <xsd:element ref="fes:expression"/>
               <xsd:any namespace="##other"/>
            </xsd:choice>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BBOXType">
   <xsd:complexContent>
      <xsd:extension base="fes:SpatialOpsType">
         <xsd:sequence>
            <xsd:element ref="fes:expression" minOccurs="0"/>
            <xsd:any namespace="##other"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DistanceBufferType">
   <xsd:complexContent>
      <xsd:extension base="fes:SpatialOpsType">
         <xsd:sequence>
            <xsd:element ref="fes:expression" minOccurs="0"/>
            <xsd:any namespace="##other"/>
            <xsd:element name="Distance" type="fes:MeasureType"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="MeasureType">
   <xsd:simpleContent>
```

```
      <xsd:extension base="xsd:double">
         <xsd:attribute name="uom" type="fes:UomIdentifier" use="required"/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="UomIdentifier">
   <xsd:union memberTypes="fes:UomSymbol fes:UomURI"/>
</xsd:simpleType>
<xsd:simpleType name="UomSymbol">
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="[^: \n\r\t]+"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="UomURI">
   <xsd:restriction base="xsd:anyURI">
      <xsd:pattern value="([a-zA-Z][a-zA-Z0-9\-\+\.]*:|\.\./|\./|#).*"/>
   </xsd:restriction>
</xsd:simpleType>
```

Spatial operators shall be used to test whether the value of a geometric property, referenced using the name of the property, and a literal geometric value or the value of another geometric property satisfy the spatial relationship implied by the operator.

EXAMPLE The fes:Overlaps operator evaluates whether the value of the specified geometric property and the specified literal geometric value or value of another geometric property spatially overlap.

Literal geometric values shall be expressed using GML (as defined in ISO 19136).

NOTE Although the canonical version of GML supported by this International Standard is GML 3.2 (see ISO 19136), the filter schemas have been crafted in such a way as to allow valid filter expressions to be encoded that use other versions of GML (see C.5, Example 11).

7.8.3 Operator semantics

7.8.3.1 Spatial operators

The semantics of the other operators Equals, Disjoint, Touches, Within, Overlaps, Crosses, Intersects, and Contains are defined in ISO 19125-1:2004, 6.1.14.

7.8.3.2 BBOX operator

The fes: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 <fes:Not><fes:Disjoint> … </fes:Disjoint></fes:Not> meaning that the fes:BBOX operator shall identify all geometries that spatially interact with the box. If the optional fes:PropertyName element is not specified, the calling service shall apply the BBOX operator to all the spatial properties of the resource.

7.8.3.3 DWithin and Beyond operators

The spatial operators DWithin and Beyond shall test whether the value of a geometric property is within or beyond a specified distance of the specified literal geometric value. Distance values shall be expressed using the fes:Distance element. The content of the fes:Distance element shall represent the magnitude of the distance and the uom attribute shall be used to specify the units of measure. Units of measure can be expressed as symbols or as a URI that links to a definition of a unit of measure that may not have a conventional symbol or when it is desired to indicate a precise or variant definition.

EXAMPLE The following XML fragment:

```
<Distance unit="m">10</Distance>
```

encodes a distance value of 10 m.

7.8.4 Coordinate reference system handling

In filter expressions, geometric values shall be encoded using GML, as given in ISO 19136. In GML, the optional srsName attribute is used to specify the coordinate reference system for the coordinates of geometry. Attention is drawn to the fact that specifications that reference this International Standard shall specify how to handle the following cases that can arise where two expressions that resolve to geometric values are compared:

a) the two geometry values have difference srsName values;

b) one or both of the geometry values do not have an srsName value specified.

Although this International Standard does not mandate what actions to take to resolve these situations, it is clear that some strategies for resolving these issues can require coordinate transformations to be applied to one or both of the geometries.

7.9 Temporal operators

7.9.1 General considerations

A temporal operator (see Figure 7) determines whether its time arguments satisfy the stated temporal relationship. The operator evaluates to true if the temporal relationship is satisfied. Otherwise, the operator evaluates to false.

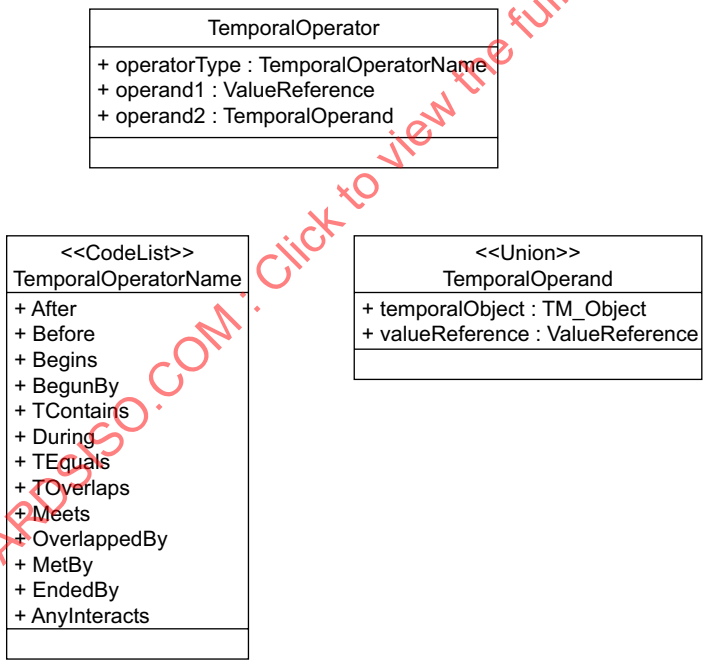

Figure 7 — TemporalOperator

Table 4 maps the temporal operators described in this International Standard to the set of temporal operators defined in ISO 19108.

Table 4 — Mapping of ISO 19143 temporal operators to ISO 19108 temporal operators

ISO 19143 temporal operator	ISO 19108 temporal operator[a]
After	After
Before	Before
Begins	Begins
BegunBy	Begun
TContains	Contains
During	During
TEquals	Equals
TOverlaps	Overlaps
Meets	Meets
OverlappedBy	OverlappedBy
MetBy	MetBy
EndedBy	EndedBy
AnyInteracts	N/A

[a] The values in this column are enumerates of TM_RelativePosition used in the relativePosition operation from ISO 19108.

A service that implements this International Standard shall, in its filter capabilities (see 7.14.5), declare which temporal operators it supports.

7.9.2 Encoding

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

```
<xsd:element name="temporalOps" type="fes:TemporalOpsType" abstract="true"/>
<xsd:complexType name="TemporalOpsType" abstract="true"/>
<xsd:element name="After"
             type="fes:BinaryTemporalOpType"
             substitutionGroup="fes:temporalOps"/>
<xsd:element name="Before"
             type="fes:BinaryTemporalOpType"
             substitutionGroup="fes:temporalOps"/>
<xsd:element name="Begins"
             type="fes:BinaryTemporalOpType"
             substitutionGroup="fes:temporalOps"/>
<xsd:element name="BegunBy"
             type="fes:BinaryTemporalOpType"
             substitutionGroup="fes:temporalOps"/>
<xsd:element name="TContains"
             type="fes:BinaryTemporalOpType"
             substitutionGroup="fes:temporalOps"/>
<xsd:element name="During"
             type="fes:BinaryTemporalOpType"
             substitutionGroup="fes:temporalOps"/>
<xsd:element name="EndedBy"
             type="fes:BinaryTemporalOpType"
             substitutionGroup="fes:temporalOps"/>
<xsd:element name="Ends"
             type="fes:BinaryTemporalOpType"
             substitutionGroup="fes:temporalOps"/>
<xsd:element name="TEquals"
             type="fes:BinaryTemporalOpType"
             substitutionGroup="fes:temporalOps"/>
<xsd:element name="Meets"
             type="fes:BinaryTemporalOpType"
             substitutionGroup="fes:temporalOps"/>
<xsd:element name="MetBy"
             type="fes:BinaryTemporalOpType"
```

```
                substitutionGroup="fes:temporalOps"/>
<xsd:element name="TOverlaps"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
<xsd:element name="OverlappedBy"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
<xsd:element name="AnyInteracts"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
<xsd:complexType name="BinaryTemporalOpType">
   <xsd:complexContent>
      <xsd:extension base="fes:TemporalOpsType">
         <xsd:sequence>
            <xsd:element ref="fes:ValueReference"/>
            <xsd:choice>
               <xsd:element ref="fes:expression"/>
               <xsd:any namespace="##other"/>
            </xsd:choice>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

As defined in this International Standard, temporal operators are used to test whether a value reference and a filter expression, that evaluates to a temporal value, satisfy the temporal relationship implied by the operator.

EXAMPLE The fes:TOverlaps operator evaluates whether the value of the specified temporal property and the specified literal temporal value temporally overlap.

Literal temporal values are expressed using GML (as defined in ISO 19136).

The semantics of all the temporal operators except fes:AnyInteracts are defined in ISO 19108:2002, 5.2.3.5.

Applicable to TM_Period only, the temporal operator fes:AnyInteracts is a shortcut operator semantically equivalent to NOT (Before OR Meets OR MetBy OR After).

If any input value of TM_TemporalPosition is indeterminate, an exception shall be raised.

7.9.3 Time zone handling

When temporal operators are used to compare time instants or periods, the time offset for local time zone shall be handled as described in XML Schema Part 2, 3.2.7: Datatypes (see W3C XML Schema Part 2).

7.10 Logical operators

7.10.1 General considerations

A logical operator (see Figure 8) 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.

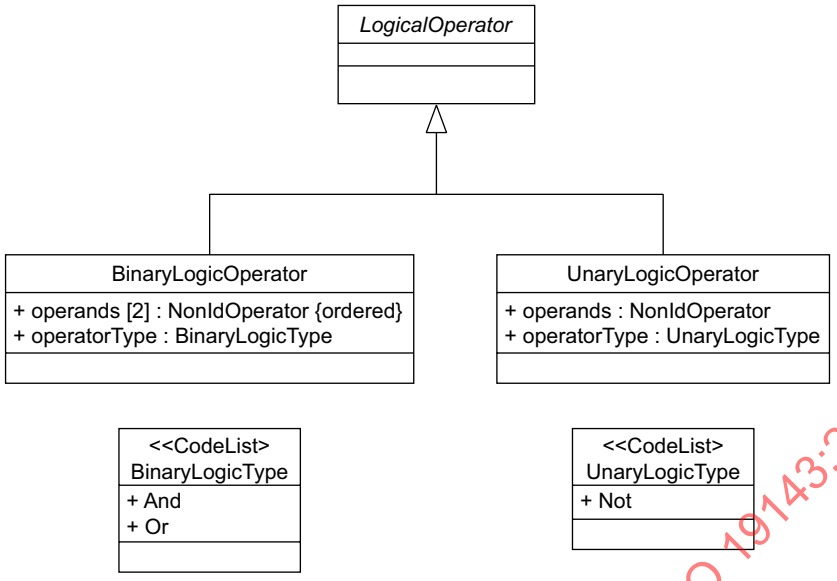

Figure 8 — LogicalOperator

7.10.2 Encoding

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

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

The fes:And, fes:Or and fes:Not elements can be used to combine elements, that are substitutable for the abstract fes:expression element, to form more complex compound expressions.

7.11 Object identifiers

7.11.1 General considerations

An object identifier is meant to represent a unique identifier for an instance of a resource within the context of the web service that is serving the resource. This International Standard defines the abstract element fes:AbstractId (see Figure 9) as the head of an XML substitution group that can be used to define a resource identifier element that can be used as a predicate within a filter expressions for specific resource types.

This International Standard defines the element fes:ResourceId (see Figure 9) that can be used as a predicate to identify any identifiable resource within a filter expression.

Specifications that reference this International Standard can also define their own element for identifying resources, if the fes:ResourceId element is deemed unsuitable, and add it to the fes:AbstractId substitution group.

Which resource identifier elements can be used predicates in a filter expression shall be advertized in the filter capabilities document (see 7.14.2).

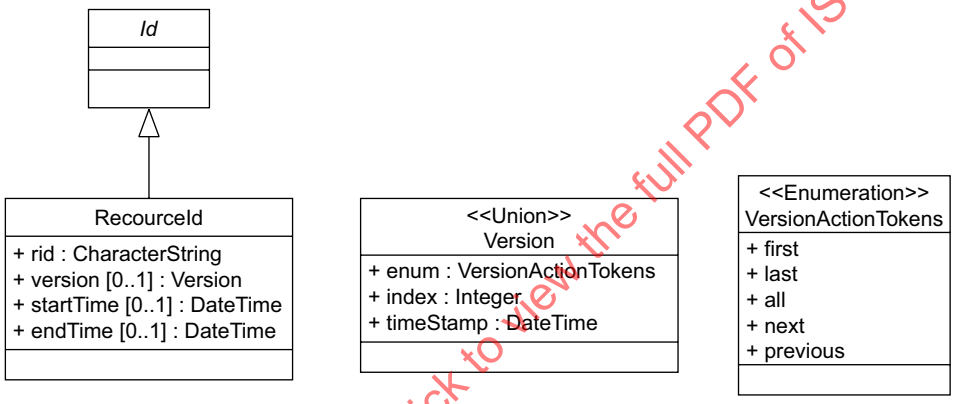

Figure 9 — ResourceId

7.11.2 Encoding

The following XML schema fragment declares the abstract element fes:AbstractId as well as the concrete element fes:ResourceId:

```
<xsd:element name="_Id" type="fes:AbstractIdType" abstract="true"/>
<xsd:complexType name="AbstractIdType" abstract="true"/>
<xsd:element name="ResourceId"
             type="fes:ResourceIdType"
             substitutionGroup="fes:_Id"/>
<xsd:complexType name="ResourceIdType">
   <xsd:complexContent>
      <xsd:extension base="fes:AbstractIdType">
         <xsd:attribute name="rid" type="xsd:string" use="required"/>
         <xsd:attribute name="previousRid" type="xsd:string"/>
         <xsd:attribute name="version" type="fes:VersionType"/>
         <xsd:attribute name="startDate" type="xsd:dateTime"/>
         <xsd:attribute name="endDate" type="xsd:dateTime"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="VersionType">
   <xsd:union memberTypes="fes:VersionActionTokens
                           xsd:positiveInteger
                           xsd:dateTime"/>
</xsd:simpleType>
```

```
<xsd:simpleType name="VersionActionTokens">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="FIRST"/>
      <xsd:enumeration value="LAST"/>
      <xsd:enumeration value="PREVIOUS"/>
      <xsd:enumeration value="NEXT"/>
      <xsd:enumeration value="ALL"/>
   </xsd:restriction>
</xsd:simpleType>
```

Within filter expressions, specific resource instances can be identified using the fes:ResourceId element.

The rid attribute specifies the id of the resource that shall be selected by the predicate.

The previousRid attribute may be used, in implementations that support versioning, to report the previous identifier of a resource.

For attributes version, startTime and endTime are used to navigate versions of a resource if an implementation that references this International Standard supports versioning.

If an implementation that references this International Standard does not support versioning, any value specified for these attributes shall be ignored and the predicate shall always select the single version that is available.

If an implementation that references this International Standard supports versioning, the rid shall be a system generated hash containing a logical resource identifier and a version number. The specific details of the hash are implementation dependant and shall be opaque to a client.

The version attribute may then be used to navigate the various versions of a resource.

The version attribute may be an integer N indicating that the Nth version of the resource shall be selected. The first version of a resource shall be numbered 1. If N exceeds the number of versions available, the latest version of the resource shall be selected.

The version attribute may also be date indicating that the version of the resource closest to the specified date shall be selected.

The versionAction attribute may also be the strings FIRST, LATEST, PREVIOUS, NEXT and ALL. The token FIRST shall select the first version of a resource. The token LATEST shall select the most recent version of a resource. The PREVIOUS and NEXT tokens shall select the previous or next version of a resource relative to the version specified using the rid attribute. The token ALL shall select all available version of a resource.

The attributes startTime and endTime may be used to specify a predicate that selects all versions of a resource between the specified start date and end date. The startTime and endTime attributes shall always be specified together. If the startTime and endTime are specified, the version attribute shall not be specified.

7.12 Extensions

7.12.1 General considerations

Standards that reference this International Standard may extend the filter syntax by:

a) adding functions using the fes:Function element (see 7.6);

b) adding new operators by defining new elements in the filter syntax.

Each of these methods can be used to extend the filter syntax, as described in 7.12.2 and 7.12.3.

Ad hoc extensions to the filter syntax are strongly discouraged because such extensions are not interoperable. Instead, extensions should be made within the context of a standard that references this International Standard.

7.12.2 Extending filter using the fes:Function element

Filter may be extended by adding new functions to the syntax that are invoked using the fes:Function element (see 7.6). Any functions added to the filter syntax shall be listed in the filter capabilities document using the fes:Function element (see 7.14.6). It is strongly recommended that additional functions be fully documented in the filter capabilities document using comments or the ows:Metadata element (see 7.14.6).

7.12.3 Extending filter by adding new elements

The filter syntax may also be extended by adding new elements that represent new operators. This may be accomplished by adding new elements that are substitutable for one of fes:comparisonOps, fes:spatialOps, fes:temporalOps, or fes:extensionOps (see 7.2).

Any new operators that are added to the filter syntax shall be defined in a namespace other than the filter namespace (see 5.4).

Any new operators that are substitutable for fes:comparisonOps, fes:spatialOps or fes:temporalOps shall be listed in the capabilities document using the pattern "extension:*Name_of_new_Operator*" (see 7.7, 7.8, 7.9) where the token "*Name_of_new_Operator*" is a placeholder for the name of the new operator.

Any new operators that are substitutable for fes:extensionOps shall be listed in the fes:Extended_Capabilities section of the filter capabilities document (see 7.14.7).

EXAMPLE The following XML Schema illustrates how to add a new comparison operator named "myops:PropertyExists" to the filter syntax.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
   targetNamespace="http://www.someserver.com/myops/1.0"
   xmlns:myops="http://www.someserver.com/myops/1.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   version="2.0.0">

   <xsd:import namespace="http://www.opengis.net/fes/2.0"
   schemaLocation="http://schemas.opengis.net/filter/2.0.0/filterAll.xsd"/>

   <xsd:element name="PropertyExists"
                type="myops:PropertyExistsType"
                substitutionGroup="fes:comparisonOps"/>
   <xsd:complexType name="PropertyExistsType">
      <xsd:complexContent>
         <xsd:extension base="fes:ComparisonOpsType">
            <xsd:sequence>
               <xsd:element ref="fes:ValueReference"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:schema>
```

The following XML fragment illustrates how the myops:PropertyExists operator might be used:

```
<?xml version="1.0"?>
<fes:Filter
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:myops="http://www.someserver.com/myops/1.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/fes/2.0
                    http://schemas.opengis.net/filter/2.0.0/filterAll.xsd
                    http://www.someserver.com/myops/1.0 ./filter23.xsd">
   <fes:And>
      <myops:PropertyExists>
         <fes:ValueReference>Person/age</fes:ValueReference>
      </myops:PropertyExists>
      <fes:PropertyIsBetween>
```

```
            <fes:ValueReference>Person/age</fes:ValueReference>
            <fes:LowerBoundary>
               <fes:Literal>18</fes:Literal>
            </fes:LowerBoundary>
            <fes:UpperBoundary>
               <fes:Literal>200</fes:Literal>
            </fes:UpperBoundary>
         </fes:PropertyIsBetween>
      </fes:And>
</fes:Filter>
```

7.13 Filter capabilities

The filterCapabilities.xsd schema defines metadata that is used to describe the specific elements that a particular implementation of this International Standard supports. A client application can inspect the filter capabilities metadata (Figures 10 to 17) and be able to determine which operators and types a filter expression processor supports.

The filter capabilities metadata can be embedded in a larger metadata document describing a system of which a filter expression processor is one component. For example a web feature service (see ISO 19142) that uses the XML encoding of a filter expression would include a filter capabilities fragment in its capabilities document to advertize what filter capabilities the service supports.

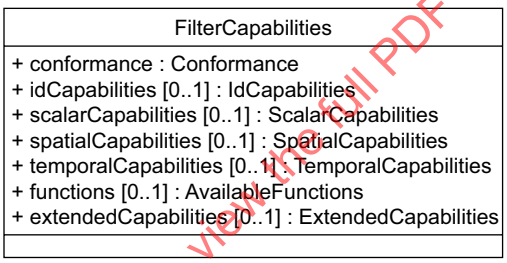

Figure 10 — FilterCapabilities

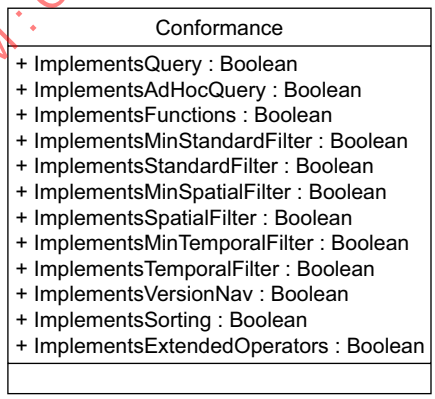

Figure 11 — Conformance

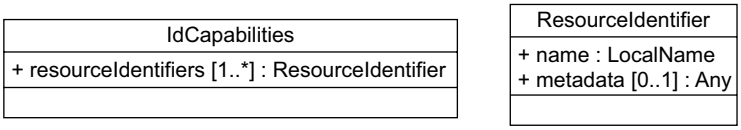

Figure 12 — IdCapabilities

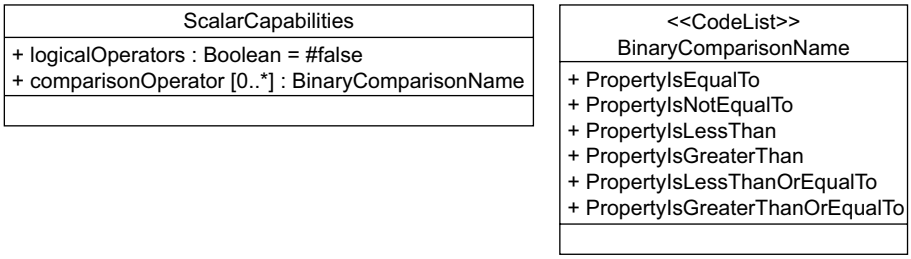

Figure 13 — ScalarCapabilities

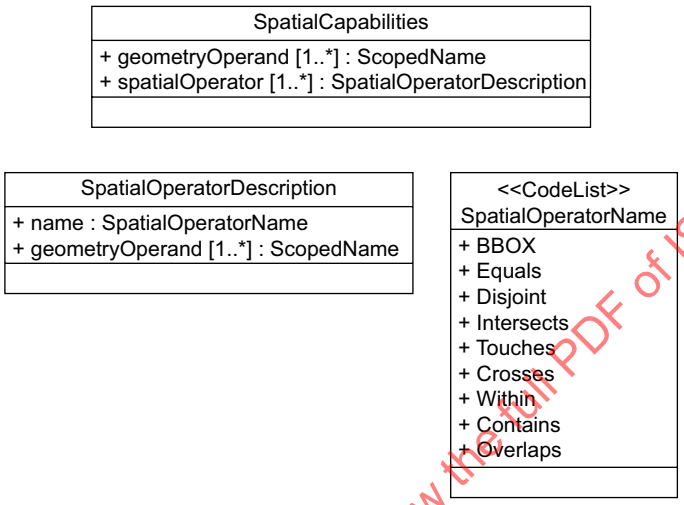

Figure 14 — SpatialCapabilities

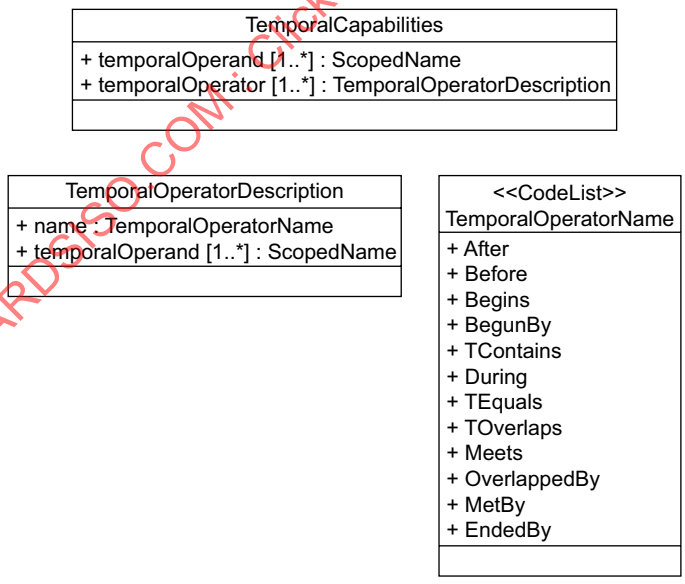

Figure 15 — TemporalCapabilities

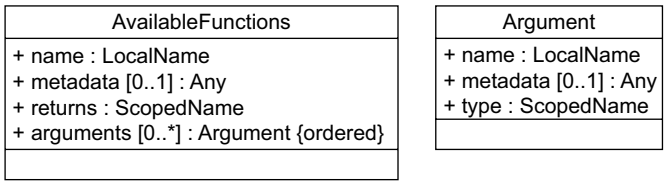

Figure 16 — AvailableFunction

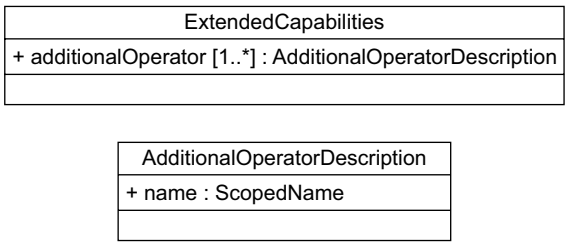

Figure 17 — ExtendedCapabilities

7.14 Encoding

7.14.1 Capability categories

Filter capabilities are divided into five categories: id capabilities, scalar capabilities, spatial capabilities, temporal capabilities and the ability to test the schema of a resource for the existence or absence of a named property. 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="Conformance"
                   Type="fes:ConformanceType"/>
      <xsd:element name="Id_Capabilities"
                   type="fes:Id_CapabilitiesType"
                   minOccurs="0"/>
      <xsd:element name="Scalar_Capabilities"
                   type="fes:Scalar_CapabilitiesType"
                   minOccurs="0"/>
      <xsd:element name="Spatial_Capabilities"
                   type="fes:Spatial_CapabilitiesType"
                   minOccurs="0"/>
      <xsd:element name="Temporal_Capabilities"
                   type="fes:Temporal_CapabilitiesType"
                   minOccurs="0"/>
      <xsd:element name="Functions"
                   type="fes:AvailableFunctionsType"
                   minOccurs="0"/>
      <xsd:element name="Extended_Capabilities"
                   type="fes:Extended_CapabilitiesType"
                   minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
</xsd:element>
```

7.14.2 Conformance clause

The conformance clauses/subclauses shall be used to declare which conformance classes a particular implementation of this International Standard implements.

The following XML Schema fragment declares the schema of the fes:Conformance clause:

```
<xsd:complexType name="ConformanceType">
   <xsd:sequence>
      <xsd:element name="Constraint" type="ows:DomainType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

It is a list of ows:Constraint elements whose value is either “TRUE” or “FALSE”.

Table 5 — Names of conformance class constraints

Conformance class name (see Table 1)	Constraint name
Query	ImplementsQuery
Ad hoc query	ImplementsAdHocQuery
Functions	ImplementsFunctions
Resource Identification	ImplementsResourceId
Minimum Standard Filter	ImplementsMinStandardFilter
Standard Filter	ImplementsStandardFilter
Minimum Spatial Filter	ImplementsMinSpatialFilter
Spatial Filter	ImplementsSpatialFilter
Minimum Temporal Filter	ImplementsMinTemporalFilter
Temporal Filter	ImplementsTemporalFilter
Version navigation	ImplementsVersionNav
Sorting	ImplementsSorting
Extended Operators	ImplementsExtendedOperators
Minimum XPath	ImplementsMinimumXPath
Schema Element Function	ImplementsSchemaElementFunc

Implementations of this International Standard shall include each of the constraints listed in Table 5 settings its value to “TRUE” to indicate that the implementation implements the corresponding conformance class or “FALSE” to indicate that the implementation does not implement the conformance class.

EXAMPLE The following XML fragment illustrates a conformance section:

```
<fes:Conformance>
   <fes:Constraint name="ImplementsQuery">
      <ows:NoValues/>
      <ows:DefaultValue>TRUE</ows:DefaultValue>
   </fes:Constraint>
   <fes:Constraint name="ImplementsAdHocQuery">
      <ows:NoValues/>
      <ows:DefaultValue>TRUE</ows:DefaultValue>
   </fes:Constraint>
   <fes:Constraint name="ImplementsMinStandardFilter">
      <ows:NoValues/>
      <ows:DefaultValue>TRUE</ows:DefaultValue>
   </fes:Constraint>
   <fes:Constraint name="ImplementsStandardFilter">
      <ows:NoValues/>
      <ows:DefaultValue>FALSE</ows:DefaultValue>
   </fes:Constraint>
   <fes:Constraint name="ImplementsMinSpatialFilter">
      <ows:NoValues/>
      <ows:DefaultValue>TRUE</ows:DefaultValue>
   </fes:Constraint>
   <fes:Constraint name="ImplementsSpatialFilter">
      <ows:NoValues/>
      <ows:DefaultValue>FALSE</ows:DefaultValue>
   </fes:Constraint>
   <fes:Constraint name="ImplementsMinTemporalFilter">
      <ows:NoValues/>
      <ows:DefaultValue>FALSE</ows:DefaultValue>
```

```
      </fes:Constraint>
      <fes:Constraint name="ImplementsVersionNav">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </fes:Constraint>
      <fes:Constraint name="ImplementsSorting">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </fes:Constraint>
   </fes:Conformance>
```

7.14.3 Id capabilities

This International Standard defines the element fes:ResourceId as a generic element for referencing resources by id. Implementations that reference this International Standard, however, may define their own elements for resource identifications. The resource identifiers section of the filter capabilities document allows such implementations to declare the name or names of the elements it uses for resource identifiers.

The following XML Schema fragment declares the schema of the fes:ResourceIdentifiers section:

```
<xsd:complexType name="Id_CapabilitiesType">
   <xsd:sequence>
      <xsd:element name="ResourceIdentifier"
                   type="fes:ResourceIdentifierType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ResourceIdentifierType">
   <xsd:sequence>
      <xsd:element ref="ows:Metadata" minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="xsd:QName" use="required"/>
</xsd:complexType>
```

It is a list of element names that represent the resource identifier elements that the service supports. These element names shall be considered synonyms.

EXAMPLE A catalogue that implements this International Standard may include the following Id_Capabilities element within its filter capabilities document:

```
<fes:Id_Capabilities>
 <fes:Id_Element>cat:RecordId</fes:Id_Element>
 <fes:Id_Element>fes:ResourceId</fes:Id_Element>
</fes:Id_capabilities>
```

indicating that the service can accept the cat:RecordId or fes:ResourceId element as predicates in a filter expression.

7.14.4 Scalar capabilities

The fes:Scalar_Capabilities element can be specified by a service to advertise which logical, comparison and arithmetic operators the service supports. If the fes:Scalar_Capabilities element is not specified then a client shall assume that the service does not support any logical or comparison operators and does not implement any additional functions.

Scalar capabilities include the ability to process logical expressions and comparisons. The following XML Schema defines how scalar capabilities are encoded:

```
<xsd:complexType name="Scalar_CapabilitiesType">
   <xsd:sequence>
      <xsd:element ref="fes:LogicalOperators" minOccurs="0"/>
      <xsd:element name="ComparisonOperators"
                   type="fes:ComparisonOperatorsType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ComparisonOperatorsType">
   <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="ComparisonOperator"
                   type="fes:ComparisonOperatorType"/>
   </xsd:sequence>
</xsd:complexType>
```

The fes:LogicalOperators element is used to indicate that the filter can process *And*, *Or* and *Not* operators. The XML encoding for the fes:LogicalOperators element is declared by the following XML schema fragment:

```
<xsd:element name="LogicalOperators">
   <xsd:complexType/>
</xsd:element>
```

The fes:ComparisonOperators element is used to indicate which comparison operators are supported by a service. The XML encoding of the fes:ComparisonOperators element is declared by the following XML Schema fragment:

```
<xsd:complexType name="ComparisonOperatorsType">
      <xsd:sequence maxOccurs="unbounded">
         <xsd:element name="ComparisonOperator"
                      type="fes:ComparisonOperatorType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ComparisonOperatorType">
      <xsd:attribute name="name"
                     type="fes:ComparisonOperatorNameType" use="required"/>
   </xsd:complexType>
   <xsd:simpleType name="ComparisonOperatorNameType">
      <xsd:union>
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="PropertyIsEqualTo"/>
               <xsd:enumeration value="PropertyIsNotEqualTo"/>
               <xsd:enumeration value="PropertyIsLessThan"/>
               <xsd:enumeration value="PropertyIsGreaterThan"/>
               <xsd:enumeration value="PropertyIsLessThanOrEqualTo"/>
               <xsd:enumeration value="PropertyIsGreaterThanOrEqualTo"/>
               <xsd:enumeration value="PropertyIsLike"/>
               <xsd:enumeration value="PropertyIsNull"/>
               <xsd:enumeration value="PropertyIsNil"/>
               <xsd:enumeration value="PropertyIsBetween"/>
            </xsd:restriction>
         </xsd:simpleType>
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:pattern value="extension:\w{2,}"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:union>
   </xsd:simpleType>
```

The pattern “extension:*OperatorName*” shall be used to list any additional comparison operators that may be added to the filter syntax (see 7.12). The token “*OperatorName*” is a placeholder for the actual name of the additional operator.

7.14.5 Spatial capabilities

The fes:Spatial_Capabilities element can be specified by a service to advertise which spatial operators and geometric operands the service supports. If the fes:Spatial_Capabilities element is not specified, a client shall assume that the service does not support any spatial operators.

A service that supports spatial filtering shall include a spatial capabilities section in its capabilities document. 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 declared in this International Standard: Equals, Disjoint, Touches, Within, Overlaps, Crosses, Intersects, Contains, DWithin and Beyond. Spatial capabilities are encoded according to the following XML Schema fragments:

```
   <xsd:complexType name="Spatial_CapabilitiesType">
      <xsd:sequence>
         <xsd:element name="GeometryOperands"
                      type="fes:GeometryOperandsType"/>
         <xsd:element name="SpatialOperators"
                      type="fes:SpatialOperatorsType"/>
      </xsd:sequence>
```

```
</xsd:complexType>
<xsd:complexType name="GeometryOperandsType">
   <xsd:sequence>
      <xsd:element name="GeometryOperand" maxOccurs="unbounded">
         <xsd:complexType>
            <xsd:attribute name="name" type="xsd:QName" use="required"/>
         </xsd:complexType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SpatialOperatorsType">
   <xsd:sequence>
      <xsd:element name="SpatialOperator"
                   type="fes:SpatialOperatorType"
                   maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SpatialOperatorType">
   <xsd:sequence>
      <xsd:element name="GeometryOperands"
                   type="fes:GeometryOperandsType"
                   minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="fes:SpatialOperatorNameType"/>
</xsd:complexType>
<xsd:simpleType name="SpatialOperatorNameType">
   <xsd:union>
      <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:enumeration value="BBOX"/>
            <xsd:enumeration value="Equals"/>
            <xsd:enumeration value="Disjoint"/>
            <xsd:enumeration value="Intersects"/>
            <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:enumeration value="DWithin"/>
         </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:pattern value="extension:\w{2,}"/>
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:union>
</xsd:simpleType>
```

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

The pattern "extension:*OperatorName*" shall be used to list any additional spatial operators that may be added to the filter syntax (see 7.12). The token "*OperatorName*" is a placeholder for the actual name of the additional operator.

7.14.6 Temporal capabilities

The fes:Temporal_Capabilities element can be specified by a service to advertise which temporal operators and temporal operands the service supports. If the fes:Temporal_Capabilities element is not specified then a client shall assume that the service does not support any temporal operators.

A service that supports temporal filtering shall include a temporal capabilities section in its capabilities document. Temporal capabilities include the ability to filter temporal data of specified temporal types based on

the spatial operators declared in this International Standard: After, Before, Begins, BegunBy, TContains, During, TEquals, TOverlaps, Meets, OverlappedBy, MetBy and EndedBy.

Temporal capabilities are encoded according to the following XML Schema fragments:

```
<xsd:complexType name="Temporal_CapabilitiesType">
   <xsd:sequence>
      <xsd:element name="TemporalOperands"
                   type="fes:TemporalOperandsType"/>
      <xsd:element name="TemporalOperators"
                   type="fes:TemporalOperatorsType"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TemporalOperandsType">
   <xsd:sequence>
      <xsd:element name="TemporalOperand" maxOccurs="unbounded">
         <xsd:complexType>
            <xsd:attribute name="name" type="xsd:QName" use="required"/>
         </xsd:complexType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TemporalOperatorsType">
   <xsd:sequence>
      <xsd:element name="TemporalOperator"
                   type="fes:TemporalOperatorType"
                   maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TemporalOperatorType">
   <xsd:sequence>
      <xsd:element name="TemporalOperands"
                   type="fes:TemporalOperandsType"
                   minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="name"
                  type="fes:TemporalOperatorNameType" use="required"/>
</xsd:complexType>
<xsd:simpleType name="TemporalOperatorNameType">
   <xsd:union>
      <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:enumeration value="After"/>
            <xsd:enumeration value="Before"/>
            <xsd:enumeration value="Begins"/>
            <xsd:enumeration value="BegunBy"/>
            <xsd:enumeration value="TContains"/>
            <xsd:enumeration value="During"/>
            <xsd:enumeration value="TEquals"/>
            <xsd:enumeration value="TOverlaps"/>
            <xsd:enumeration value="Meets"/>
            <xsd:enumeration value="OverlappedBy"/>
            <xsd:enumeration value="MetBy"/>
            <xsd:enumeration value="Ends"/>
            <xsd:enumeration value="EndedBy"/>
         </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:pattern value="extension:\w{2,}"/>
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:union>
</xsd:simpleType>
```

An implementation of this International Standard shall list the temporal operators and temporal operand types that it supports as the content of the fes:TemporalCapabilities element of its filter capabilities document. Temporal operands are listed using the fes:TemporalOperands element. Temporal operands can be defined globally (as the first child of the fes:Temporal_Capabilities element) indicating that all temporal operators know how to process the specified operands or locally for each temporal operator (as the content of the fes:TemporalOperator element) indicating that the specific operator knows how to process the specified operands.

The pattern “extension:*OperatorName*” shall be used to list any additional temporal operators that may be added to the filter syntax (see 7.12). The token “*OperatorName*” is a placeholder for the actual name of the additional operator.

7.14.7 Functions

The fes:Functions element shall be used in a filter capabilities document to enumerate the function that may be used in filter expressions.

The description of each function shall include a declaration of the return type, the list of arguments with optional names and expected types. The ows:Metadata element may be used to reference detailed metadata about the function or its arguments. The following fragment defines the schema of fes:Functions element.

```
<xsd:complexType name="AvailableFunctionsType">
   <xsd:sequence>
      <xsd:element name="Function"
                   type="fes:AvailableFunctionType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="AvailableFunctionType">
   <xsd:sequence>
      <xsd:element ref="ows:Metadata" minOccurs="0"/>
      <xsd:element name="Returns" type="xsd:QName"/>
      <xsd:element name="Arguments"
                   type="fes:ArgumentsType" minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:complexType name="ArgumentsType">
   <xsd:sequence>
      <xsd:element name="Argument"
                   type="fes:ArgumentType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ArgumentType">
   <xsd:sequence>
      <xsd:element ref="ows:Metadata" minOccurs="0"/>
      <xsd:element name="Type" type="xsd:QName"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="xsd:string" use="required"/>
</xsd:complexType>
```

7.14.8 Extended capabilities

The fes:Extended_Capabilities element may be specified within a service's filter capabilities document to advertise any additional operators that standards that reference this International Standard might add to the filter syntax.

The following XML Schema fragment defines the schema of the fes:Extended_Capabilities element:

```
<xsd:complexType name="Extended_CapabilitiesType">
   <xsd:sequence>
      <xsd:element name="AdditionalOperators"
                   type="fes:AdditionalOperatorsType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="AdditionalOperatorsType">
   <xsd:sequence>
      <xsd:element name="Operator"
                   type="fes:ExtensionOperatorType"
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ExtensionOperatorType">
   <xsd:attribute name="name" type="xsd:QName" use="required"/>
</xsd:complexType>
```

Any additional operators added to the filter syntax shall be advertised in the extended capabilities section of the filter capabilities document. This extended capabilities section contains a list of additional operator names. How new operators can be added to the filter syntax is described in 7.12.3.

8 Sorting

8.1 General considerations

The fes:SortBy element (see Figure 18) is an optional part of a service-specific query and is included in the normative set of Filter schema files (i.e. “sort.xsd”).

The fes:SortBy element is used to specify property names whose values shall be used to order (upon presentation) the set of resources that satisfy a filter expression.

The fes:SortBy element is provided as an entry point to invoke the sorting mechanism of the filter expression processor. The sorting mechanism is not specified in this International Standard. The only requirement is that the sort sequence shall be consistent, given the same data set and sort request, between consecutive invocations of the sort.

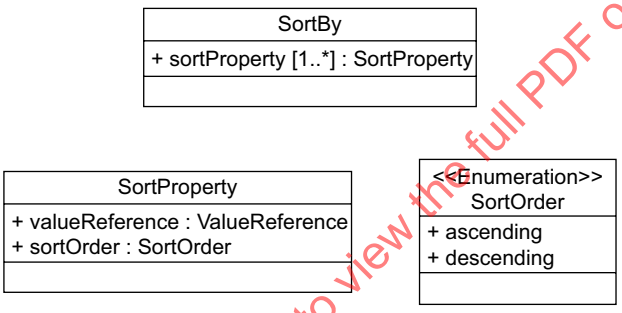

Figure 18 — SortBy

8.2 Encoding

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

```
<xsd:element name="SortBy"
             type="fes:SortByType"/>
<xsd:complexType name="SortByType">
   <xsd:sequence>
      <xsd:element name="SortProperty"
                   type="fes:SortPropertyType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SortPropertyType">
   <xsd:sequence>
      <xsd:element ref="fes:ValueReference"/>
      <xsd:element name="SortOrder" type="fes:SortOrderType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="SortOrderType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="DESC"/>
      <xsd:enumeration value="ASC"/>
   </xsd:restriction>
</xsd:simpleType>
```

The fes:SortBy element shall have a minimum of one and an unbounded number of fes:SortProperty child elements. In the event that multiple fes:SortProperty elements exist, the sequence of the fes:SortProperty elements shall determine the order of ordering. The fes:SortProperty element shall have a single fes:PropertyName child element and an optional fes:SortOrder child element. The fes:SortOrder element

shall have a value of either “ASC” or “DESC”. The value “ASC” shall indicate that an ascending sort, in the collation sequence of the underlying data store, be performed. The value “DESC” shall indicate that a descending sort, in the collation sequence of the filter expression processor, be performed. In the event that fes:SortOrder is not included within the fes:SortProperty element of a request, the service shall assume a sort order of ASC.

8.3 Exceptions

In the event that the fes:ValueReference element contains a reference to a value of an unknown resource type, the service shall raise an InvalidParameterValue (as given in OGC 06-121r3, Table 25) exception.

In the event that the fes:ValueReference element contains a reference to a value of a known resource type but that value if not defined for the resource type being queried, the service shall raise an InvalidParameterValue (as given in OGC 06-121r3, Table 25) exception.

In the event that multiple fes:SortProperty elements are specified whose fes:ValueReference elements reference the same value, the service shall raise a DuplicateSortKey exception with the exception locator (as given in OGC 06-121r3, 8.4) indicating the duplicate value reference.

Annex A
(normative)

Conformance testing

A.1 Test cases for query

a) Test purpose: verify the correct use of query elements.

b) Test method: verify that a concrete element is materialized that is substitutable for fes:AbstractQueryElement. Use that element to submit query requests and check that the query operation works according to its description.

c) Reference: 6.2.

d) Test type: basic test.

A.2 Test cases for ad hoc query

a) Test purpose: verify the correct use of XML-encoded and KVP-encoded ad hoc query elements.

b) Test method: verify that concrete elements are materialized that are substitutable for fes:AbstractAdhocQuery, fes:AbstractSelectionClause, fes:AbstractProjectionClause and fes:AbstractSortingClause. Use those concrete elements to formulate and submit ad hoc query requests and check that that ad hoc query requests work according to their description. Verify that the server implements KVP-encoded ad hoc queries. Submit KVP-encoded ad hoc query requests and check that the ad hoc query operations work according to their description.

c) Reference: 6.3.2, 6.3.3.

d) Test type: basic test.

A.3 Test cases for functions

a) Test purpose: verify that additional functions are defined and verify their correct use.

b) Test method: verify that one or more functions are listed in the filter capabilities document. Submit requests and check that the additional filter functions operate according to their description.

c) Reference: 7.6, 7.14.7.

d) Test type: basic Test.

A.4 Test cases for resource identification

a) Test purpose: verify the correct use of the resource identification operator.

b) Test method: submit requests that query specific instances of resource and check that the ResourceId operator operates according to its description.

c) Reference: 7.11.

d) Test type: basic test.

A.5 Test cases for minimum standard filter

a) Test purpose: verify the correct use of comparison and logical operators.

b) Test method: verify that the operators PropertyIsEqualTo, PropertyIsNotEqualTo, PropertyIsLessThan, PropertyIsGreaterThan, PropertyIsLessThanOrEqualTo, PropertyIsGreaterThanOrEqualTo are listed in the filter capabilities document. Verify that all the logical operators are listed in the filter capabilities document. Submit requests and check that the implemented comparison and logical operators work according to their description.

c) Reference: 7.7.3, 7.7.3.2, 7.7.3.3, 7.10, 7.14.4.

d) Test type: basic test.

A.6 Test cases for standard filter

a) Test purpose: verify the correct use of additional comparison operators.

b) Test method: verify that the Minimum Standard Filter conformance class (see A.4) is satisfied. Verify that the operators PropertyIsNull, PropertyIsNil, PropertyIsLike and PropertyIsBetween are listed in the filter capabilities document. Submit requests and check that the operators work according to their description.

c) Reference: 7.7.3.4, 7.7.3.5, 7.7.3.6, 7.7.3.7, 7.14.4.

d) Test type: basic test.

A.7 Test cases for minimum spatial filter

a) Test purpose: verify the correct use of the BBOX operator.

b) Test method: verify that the BBOX operator is listed in the filter capabilities document. Submit requests and check that the BBOX operator works according to its description.

c) Reference: 7.8.3.2.

d) Test type: basic test.

A.8 Test cases for spatial filter

a) Test purpose: verify the correct use of additional spatial operators.

b) Test method: verify that the Minimum Spatial Filter conformance class is satisfied. Verify that one or more additional spatial operators are listed in the filter capabilities document. Submit requests and check that the additional spatial operators work according to their description.

c) Reference: 7.8.2, 7.8.3, 7.8.3.3, 7.8.4, 7.14.5.

d) Test type: basic test.

A.9 Test cases for minimum temporal filter

a) Test purpose: verify the correct use of the During operator.

b) Test method: verify that the During operator is listed in the filter capabilities document. Submit requests and check that the During operator works according to its description.

c) Reference: 7.9.2, 7.9.3, 7.14.6.

d) Test type: basic test.

A.10 Test cases for temporal filter

a) Test purpose: verify the correct use of additional temporal operators.

b) Test method: verify that the Minimum Temporal Filter conformance class is satisfied. Verify that one or more additional temporal operators are listed in the filter capabilities document. Submit requests and check that the additional temporal operators work according to their description.

c) Reference: 7.9.2, 7.9.3, 7.14.6.

d) Test type: basic test.

A.11 Test cases for version navigation

a) Test purpose: verify the ability to navigate feature versions using the fes:ResourceId version navigation capability.

b) Test method: submit requests and check that the operations work according to their description.

c) Reference: 7.11.

d) Test type: basic test.

A.12 Test cases for sorting

a) Test purpose: verify the correct use of sorting.

b) Test method: submit requests and check that the operations work according to their description.

c) Reference: ISO 19143:2010, Clause 8.

d) Test type: basic test.

A.13 Test cases for extended operators

a) Test purpose: verify the correct use of extended operators.

b) Test method: verify that a list of extended operators in presented in the filter capabilities document. Submit requests to check that the extended operators work according to their description.

c) Reference: 7.12.3,7.14.8.

d) Test type: basic test.

A.14 Test cases for XPath

a) Test purpose: verify that the correct use of the minimum set of XPath capabilities.

b) Test method: submit requests to check that XPath processing operates according to its description.

c) Reference: 7.4.4.

d) Test type: basic test.

A.15 Test cases for schema-element() function

a) Test purpose: verify the correct use of the schema-element() XPath function.

b) Test method: verify that the Minimum XPath conformance class is satisfied. Submit requests that use the schema-element() function and check that the function operates according to its description.

c) Reference: 7.4.4, A.14.

d) Test type: basic test.

Annex B
(informative)

Filter schema definitions

B.1 General considerations

The XML schema fragments presented in this International Standard are gathered in this annex and factored into files that may be used to validate XML-encoded query and filter expressions.

B.2 Schema files

B.2.1 expr.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
   targetNamespace="http://www.opengis.net/fes/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   version="2.0.0">

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

   <xsd:element name="ValueReference" type="xsd:string"
                substitutionGroup="fes:expression"/>

   <xsd:element name="Function" type="fes:FunctionType"
                substitutionGroup="fes:expression"/>
   <xsd:complexType name="FunctionType">
      <xsd:sequence>
         <xsd:element ref="fes:expression"
                      minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string" use="required"/>
   </xsd:complexType>

   <xsd:element name="Literal" type="fes:LiteralType"
                substitutionGroup="fes:expression"/>
   <xsd:complexType name="LiteralType" mixed="true">
      <xsd:sequence>
         <xsd:any minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="type" type="xsd:QName"/>
   </xsd:complexType>
</xsd:schema>
```

B.2.2 filter.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
   targetNamespace="http://www.opengis.net/fes/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   version="2.0.0">


   <xsd:include schemaLocation="expr.xsd"/>
   <xsd:include schemaLocation="query.xsd"/>
   <xsd:include schemaLocation="filterCapabilities.xsd"/>
```

```
<xsd:element name="Filter"
             type="fes:FilterType"
             substitutionGroup="fes:AbstractSelectionClause"/>
<xsd:complexType name="FilterType">
   <xsd:complexContent>
      <xsd:extension base="fes:AbstractSelectionClauseType">
         <xsd:sequence>
            <xsd:group ref="fes:FilterPredicates"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<!-- ================================================================== -->
<!-- FILTER PREDICATES                                                  -->
<!-- ================================================================== -->
<xsd:group name="FilterPredicates">
  <xsd:choice>
      <xsd:element ref="fes:comparisonOps"/>
      <xsd:element ref="fes:spatialOps"/>
      <xsd:element ref="fes:temporalOps"/>
      <xsd:element ref="fes:logicOps"/>
      <xsd:element ref="fes:extensionOps"/>
      <xsd:element ref="fes:Function"/>
      <xsd:element ref="fes:_Id" maxOccurs="unbounded"/>
   </xsd:choice>
</xsd:group>

<!-- ================================================================== -->
<!-- COMPARISON OPERATORS                                               -->
<!-- ================================================================== -->
<xsd:element name="comparisonOps"
             type="fes:ComparisonOpsType"
             abstract="true"/>
<xsd:complexType name="ComparisonOpsType" abstract="true"/>
<xsd:element name="PropertyIsEqualTo"
             type="fes:BinaryComparisonOpType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsNotEqualTo"
             type="fes:BinaryComparisonOpType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsLessThan"
             type="fes:BinaryComparisonOpType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsGreaterThan"
             type="fes:BinaryComparisonOpType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsLessThanOrEqualTo"
             type="fes:BinaryComparisonOpType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsGreaterThanOrEqualTo"
             type="fes:BinaryComparisonOpType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsLike"
             type="fes:PropertyIsLikeType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsNull"
             type="fes:PropertyIsNullType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsNil"
             type="fes:PropertyIsNilType"
             substitutionGroup="fes:comparisonOps"/>
<xsd:element name="PropertyIsBetween"
             type="fes:PropertyIsBetweenType"
             substitutionGroup="fes:comparisonOps"/>

<!-- ================================================================== -->
<!-- SPATIAL OPERATORS                                                  -->
<!-- ================================================================== -->
<xsd:element name="spatialOps" type="fes:SpatialOpsType" abstract="true"/>
<xsd:complexType name="SpatialOpsType" abstract="true"/>
<xsd:element name="Equals"
             type="fes:BinarySpatialOpType"
             substitutionGroup="fes:spatialOps"/>
<xsd:element name="Disjoint"
             type="fes:BinarySpatialOpType"
```

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

    <!-- ============================================================== -->
    <!-- TEMPORAL OPERATORS                                             -->
    <!-- ============================================================== -->
    <xsd:element name="temporalOps" type="fes:TemporalOpsType" abstract="true"/>
    <xsd:complexType name="TemporalOpsType" abstract="true"/>
    <xsd:element name="After"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
    <xsd:element name="Before"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
    <xsd:element name="Begins"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
    <xsd:element name="BegunBy"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
    <xsd:element name="TContains"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
    <xsd:element name="During"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
    <xsd:element name="EndedBy"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
    <xsd:element name="Ends"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
    <xsd:element name="TEquals"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
    <xsd:element name="Meets"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
    <xsd:element name="MetBy"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
    <xsd:element name="TOverlaps"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
    <xsd:element name="OverlappedBy"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
    <xsd:element name="AnyInteracts"
                type="fes:BinaryTemporalOpType"
                substitutionGroup="fes:temporalOps"/>
```

```
<!-- ========================================================================= -->
<!-- LOGICAL OPERATORS                                                         -->
<!-- ========================================================================= -->
<xsd:element name="logicOps" type="fes:LogicOpsType" abstract="true"/>
<xsd:complexType name="LogicOpsType" abstract="true"/>
<xsd:element name="And"
             type="fes:BinaryLogicOpType"
             substitutionGroup="fes:logicOps"/>
<xsd:element name="Or"
             type="fes:BinaryLogicOpType"
             substitutionGroup="fes:logicOps"/>
<xsd:element name="Not"
             type="fes:UnaryLogicOpType"
             substitutionGroup="fes:logicOps"/>

<!-- ========================================================================= -->
<!-- EXTENSION OPERATORS                                                       -->
<!-- ========================================================================= -->
<xsd:element name="extensionOps"
             type="fes:ExtensionOpsType"
             abstract="true"/>
<xsd:complexType name="ExtensionOpsType" abstract="true"/>

<!-- ========================================================================= -->
<!-- OBJECT/RECORDS IDENTIFIERS                                                -->
<!-- ========================================================================= -->
<xsd:element name="_Id" type="fes:AbstractIdType" abstract="true"/>
<xsd:complexType name="AbstractIdType" abstract="true"/>

<!-- ========================================================================= -->
<!-- CONCRETE OBJECT IDENTIFIERS                                               -->
<!-- ========================================================================= -->
<xsd:element name="ResourceId"
             type="fes:ResourceIdType"
             substitutionGroup="fes:_Id"/>
<xsd:complexType name="ResourceIdType">
   <xsd:complexContent>
      <xsd:extension base="fes:AbstractIdType">
         <xsd:attribute name="rid" type="xsd:string" use="required"/>
         <xsd:attribute name="previousRid" type="xsd:string"/>
         <xsd:attribute name="version" type="fes:VersionType"/>
         <xsd:attribute name="startDate" type="xsd:dateTime"/>
         <xsd:attribute name="endDate" type="xsd:dateTime"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="VersionType">
   <xsd:union memberTypes="fes:VersionActionTokens
                           xsd:positiveInteger
                           xsd:dateTime">
      <xsd:annotation>
         <xsd:documentation>
            Allows navigation of versioned resources.
            Can be the tokens FIRST,LAST,PREVIOUS,NEXT,ALL relative
            to the specified rid. It is up to the WFS to arrange
            for the version id to be hashed into the RID.
            Can be an integer indicating which version of a
            feature to fetch. 1=first, 2=second, etc...
            Can be a
         </xsd:documentation>
      </xsd:annotation>
   </xsd:union>
</xsd:simpleType>
<xsd:simpleType name="VersionActionTokens">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="FIRST"/>
      <xsd:enumeration value="LAST"/>
      <xsd:enumeration value="PREVIOUS"/>
      <xsd:enumeration value="NEXT"/>
      <xsd:enumeration value="ALL"/>
   </xsd:restriction>
</xsd:simpleType>
```

```
<!-- ========================================================================= -->
<!-- TYPE DECLARATIONS                                                         -->
<!-- ========================================================================= -->
<xsd:complexType name="BinaryComparisonOpType">
   <xsd:complexContent>
      <xsd:extension base="fes:ComparisonOpsType">
         <xsd:sequence>
            <xsd:element ref="fes:expression" minOccurs="2" maxOccurs="2"/>
         </xsd:sequence>
         <xsd:attribute name="matchCase" type="xsd:boolean"
                        use="optional" default="true"/>
         <xsd:attribute name="matchAction" type="fes:MatchActionType"
                        use="optional" default="Any"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="MatchActionType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="All"/>
      <xsd:enumeration value="Any"/>
      <xsd:enumeration value="One"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="PropertyIsLikeType">
   <xsd:complexContent>
      <xsd:extension base="fes:ComparisonOpsType">
         <xsd:sequence>
            <xsd:element ref="fes:expression" minOccurs="2" maxOccurs="2"/>
         </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="fes:ComparisonOpsType">
         <xsd:sequence>
            <xsd:element ref="fes:expression" minOccurs="0"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="PropertyIsNilType">
   <xsd:complexContent>
      <xsd:extension base="fes:ComparisonOpsType">
         <xsd:sequence>
            <xsd:element ref="fes:expression" minOccurs="0"/>
         </xsd:sequence>
         <xsd:attribute name="nilReason" type="xsd:string"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="PropertyIsBetweenType">
   <xsd:complexContent>
      <xsd:extension base="fes:ComparisonOpsType">
         <xsd:sequence>
            <xsd:element ref="fes:expression"/>
            <xsd:element name="LowerBoundary" type="fes:LowerBoundaryType"/>
            <xsd:element name="UpperBoundary" type="fes:UpperBoundaryType"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="LowerBoundaryType">
   <xsd:choice>
      <xsd:element ref="fes:expression"/>
   </xsd:choice>
</xsd:complexType>
<xsd:complexType name="UpperBoundaryType">
   <xsd:sequence>
      <xsd:element ref="fes:expression"/>
   </xsd:sequence>
</xsd:complexType>
```

```
<xsd:complexType name="BinarySpatialOpType">
   <xsd:complexContent>
      <xsd:extension base="fes:SpatialOpsType">
         <xsd:sequence>
            <xsd:element ref="fes:ValueReference"/>
            <xsd:choice>
               <xsd:element ref="fes:expression"/>
               <xsd:any namespace="##other"/>
            </xsd:choice>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BinaryTemporalOpType">
   <xsd:complexContent>
      <xsd:extension base="fes:TemporalOpsType">
         <xsd:sequence>
            <xsd:element ref="fes:ValueReference"/>
            <xsd:choice>
               <xsd:element ref="fes:expression"/>
               <xsd:any namespace="##other"/>
            </xsd:choice>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BBOXType">
   <xsd:complexContent>
      <xsd:extension base="fes:SpatialOpsType">
         <xsd:sequence>
            <xsd:element ref="fes:expression" minOccurs="0"/>
            <xsd:any namespace="##other"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DistanceBufferType">
   <xsd:complexContent>
      <xsd:extension base="fes:SpatialOpsType">
         <xsd:sequence>
            <xsd:element ref="fes:expression" minOccurs="0"/>
            <xsd:any namespace="##other"/>
            <xsd:element name="Distance" type="fes:MeasureType"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BinaryLogicOpType">
   <xsd:complexContent>
      <xsd:extension base="fes:LogicOpsType">
         <xsd:choice minOccurs="2" maxOccurs="unbounded">
            <xsd:group ref="fes:FilterPredicates"/>
         </xsd:choice>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="UnaryLogicOpType">
   <xsd:complexContent>
      <xsd:extension base="fes:LogicOpsType">
         <xsd:sequence>
            <xsd:choice>
               <xsd:group ref="fes:FilterPredicates"/>
            </xsd:choice>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="MeasureType">
   <xsd:simpleContent>
      <xsd:extension base="xsd:double">
         <xsd:attribute name="uom" type="fes:UomIdentifier" use="required"/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="UomIdentifier">
   <xsd:union memberTypes="fes:UomSymbol fes:UomURI"/>
</xsd:simpleType>
<xsd:simpleType name="UomSymbol">
   </xsd:annotation>
```

```
      <xsd:restriction base="xsd:string">
         <xsd:pattern value="[^: \n\r\t]+"/>
      </xsd:restriction>
   </xsd:simpleType>
   <xsd:simpleType name="UomURI">
      </xsd:annotation>
      <xsd:restriction base="xsd:anyURI">
         <xsd:pattern value="([a-zA-Z][a-zA-Z0-9\-\+\.]*:|\.\./|\./|#).*"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:schema>
```

B.2.3 query.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
   targetNamespace="http://www.opengis.net/fes/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   version="2.0.0">

   <xsd:element name="AbstractQueryExpression"
                type="fes:AbstractQueryExpressionType" abstract="true"/>
   <xsd:complexType name="AbstractQueryExpressionType" abstract="true">
      <xsd:attribute name="handle" type="xsd:string"/>
   </xsd:complexType>

   <xsd:element name="AbstractAdhocQueryExpression"
                type="fes:AbstractAdhocQueryExpressionType"
                substitutionGroup="fes:AbstractQueryExpression"
                abstract="true"/>
   <xsd:complexType name="AbstractAdhocQueryExpressionType" abstract="true">
      <xsd:complexContent>
         <xsd:extension base="fes:AbstractQueryExpressionType">
            <xsd:sequence>
               <xsd:element ref="fes:AbstractProjectionClause"
                            minOccurs="0" maxOccurs="unbounded"/>
               <xsd:element ref="fes:AbstractSelectionClause" minOccurs="0"/>
               <xsd:element ref="fes:AbstractSortingClause" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="typeNames"
                           type="fes:TypeNamesListType" use="required"/>
            <xsd:attribute name="aliases"
                           type="fes:AliasesType"/>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

   <xsd:simpleType name="TypeNamesListType">
       <xsd:list itemType="fes:TypeNamesType"/>
   </xsd:simpleType>
   <xsd:simpleType name="TypeNamesType">
       <xsd:union memberTypes="fes:SchemaElement xsd:QName"/>
   </xsd:simpleType>
   <xsd:simpleType name="SchemaElement">
      <xsd:restriction base="xsd:string">
         <xsd:pattern value="schema\-element\(.+\)"/>
      </xsd:restriction>
   </xsd:simpleType>
   <xsd:simpleType name="AliasesType">
      <xsd:list itemType="xsd:NCName"/>
   </xsd:simpleType>

   <xsd:element name="AbstractProjectionClause" abstract="true"/>
   <xsd:complexType name="AbstractProjectionClauseType" abstract="true"/>

   <xsd:element name="AbstractSelectionClause" abstract="true"/>
   <xsd:complexType name="AbstractSelectionClauseType" abstract="true"/>

   <xsd:element name="AbstractSortingClause" abstract="true"/>
   <xsd:complexType name="AbstractSortingClauseType" abstract="true"/>

</xsd:schema>
```

B.2.4 sort.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
   targetNamespace="http://www.opengis.net/fes/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   version="2.0.0">

   <xsd:include schemaLocation="query.xsd"/>
   <xsd:include schemaLocation="expr.xsd"/>

   <!-- ============================================= -->
   <!-- SORTBY EXPRESSION                             -->
   <!-- ============================================= -->
   <xsd:element name="SortBy"
                type="fes:SortByType"
                substitutionGroup="fes:AbstractSortingClause"/>

   <!-- ============================================= -->
   <!-- COMPLEX TYPES                                 -->
   <!-- ============================================= -->
   <xsd:complexType name="SortByType">
      <xsd:sequence>
         <xsd:element name="SortProperty"
                      type="fes:SortPropertyType" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="SortPropertyType">
      <xsd:sequence>
         <xsd:element ref="fes:ValueReference"/>
         <xsd:element name="SortOrder" type="fes:SortOrderType" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:simpleType name="SortOrderType">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="DESC"/>
         <xsd:enumeration value="ASC"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:schema>
```

B.2.5 filterCapabilities.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
   targetNamespace="http://www.opengis.net/fes/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:ows="http://www.opengis.net/ows/1.1"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xml="http://www.w3.org/XML/1998/namespace"
   elementFormDefault="qualified"
   version="2.0.0">

   <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
               schemaLocation="http://www.w3.org/2001/xml.xsd"/>

   <xsd:import namespace="http://www.opengis.net/ows/1.1"
               schemaLocation="../../ows/1.1.0/owsAll.xsd"/>

   <xsd:element name="Filter_Capabilities">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="Conformance"
                         type="fes:ConformanceType"/>
            <xsd:element name="Id_Capabilities"
                         type="fes:Id_CapabilitiesType"/>
            <xsd:element name="Scalar_Capabilities"
                         type="fes:Scalar_CapabilitiesType"
                         minOccurs="0"/>
            <xsd:element name="Spatial_Capabilities"
                         type="fes:Spatial_CapabilitiesType"
                         minOccurs="0"/>
```

```
            <xsd:element name="Temporal_Capabilities"
                         type="fes:Temporal_CapabilitiesType"
                         minOccurs="0"/>
            <xsd:element name="Functions"
                         type="fes:AvailableFunctionsType" minOccurs="0"/>
            <xsd:element name="Extended_Capabilities"
                         type="fes:Extended_CapabilitiesType"
                         minOccurs="0"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>

   <!-- CONFORMANCE -->
   <xsd:complexType name="ConformanceType">
      <xsd:sequence>
         <xsd:element name="Constraint" type="ows:DomainType" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>

   <!-- RESOURCE IDENTIFIERS -->
   <xsd:complexType name="Id_CapabilitiesType">
      <xsd:sequence>
         <xsd:element name="ResourceIdentifier"
                      type="fes:ResourceIdentifierType" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ResourceIdentifierType">
      <xsd:sequence>
         <xsd:element ref="ows:Metadata" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:QName" use="required"/>
   </xsd:complexType>

   <!-- SCALAR CAPABILITIES -->
   <xsd:complexType name="Scalar_CapabilitiesType">
      <xsd:sequence>
         <xsd:element ref="fes:LogicalOperators" minOccurs="0"/>
         <xsd:element name="ComparisonOperators"
                      type="fes:ComparisonOperatorsType" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:element name="LogicalOperators">
      <xsd:complexType/>
   </xsd:element>
   <xsd:complexType name="ComparisonOperatorsType">
      <xsd:sequence maxOccurs="unbounded">
         <xsd:element name="ComparisonOperator"
                      type="fes:ComparisonOperatorType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ComparisonOperatorType">
      <xsd:attribute name="name"
                     type="fes:ComparisonOperatorNameType" use="required"/>
   </xsd:complexType>
   <xsd:simpleType name="ComparisonOperatorNameType">
      <xsd:union>
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="PropertyIsEqualTo"/>
               <xsd:enumeration value="PropertyIsNotEqualTo"/>
               <xsd:enumeration value="PropertyIsLessThan"/>
               <xsd:enumeration value="PropertyIsGreaterThan"/>
               <xsd:enumeration value="PropertyIsLessThanOrEqualTo"/>
               <xsd:enumeration value="PropertyIsGreaterThanOrEqualTo"/>
               <xsd:enumeration value="PropertyIsLike"/>
               <xsd:enumeration value="PropertyIsNull"/>
               <xsd:enumeration value="PropertyIsNil"/>
               <xsd:enumeration value="PropertyIsBetween"/>
            </xsd:restriction>
         </xsd:simpleType>
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:pattern value="extension:\w{2,}"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:union>
```

```
</xsd:simpleType>
<xsd:complexType name="AvailableFunctionsType">
   <xsd:sequence>
      <xsd:element name="Function"
                   type="fes:AvailableFunctionType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="AvailableFunctionType">
   <xsd:sequence>
      <xsd:element ref="ows:Metadata" minOccurs="0"/>
      <xsd:element name="Returns" type="xsd:QName"/>
      <xsd:element name="Arguments"
                   type="fes:ArgumentsType" minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:complexType name="ArgumentsType">
   <xsd:sequence>
      <xsd:element name="Argument"
                   type="fes:ArgumentType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ArgumentType">
   <xsd:sequence>
      <xsd:element ref="ows:Metadata" minOccurs="0"/>
      <xsd:element name="Type" type="xsd:QName"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="xsd:string" use="required"/>
</xsd:complexType>

<!-- SPATIAL CAPABILITIES -->
<xsd:complexType name="Spatial_CapabilitiesType">
   <xsd:sequence>
      <xsd:element name="GeometryOperands"
                   type="fes:GeometryOperandsType"/>
      <xsd:element name="SpatialOperators"
                   type="fes:SpatialOperatorsType"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="GeometryOperandsType">
   <xsd:sequence>
      <xsd:element name="GeometryOperand" maxOccurs="unbounded">
         <xsd:complexType>
            <xsd:attribute name="name" type="xsd:QName" use="required"/>
         </xsd:complexType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SpatialOperatorsType">
   <xsd:sequence>
      <xsd:element name="SpatialOperator"
                   type="fes:SpatialOperatorType"
                   maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SpatialOperatorType">
   <xsd:sequence>
      <xsd:element name="GeometryOperands"
                   type="fes:GeometryOperandsType"
                   minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="fes:SpatialOperatorNameType"/>
</xsd:complexType>
<xsd:simpleType name="SpatialOperatorNameType">
   <xsd:union>
      <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:enumeration value="BBOX"/>
            <xsd:enumeration value="Equals"/>
            <xsd:enumeration value="Disjoint"/>
            <xsd:enumeration value="Intersects"/>
            <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:enumeration value="DWithin"/>
         </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:pattern value="extension:\w{2,}"/>
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:union>
</xsd:simpleType>

<!-- TEMPORAL CAPABILITIES -->
<xsd:complexType name="Temporal_CapabilitiesType">
   <xsd:sequence>
      <xsd:element name="TemporalOperands"
                   type="fes:TemporalOperandsType"/>
      <xsd:element name="TemporalOperators"
                   type="fes:TemporalOperatorsType"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TemporalOperandsType">
   <xsd:sequence>
      <xsd:element name="TemporalOperand" maxOccurs="unbounded">
         <xsd:complexType>
            <xsd:attribute name="name" type="xsd:QName" use="required"/>
         </xsd:complexType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TemporalOperatorsType">
   <xsd:sequence>
      <xsd:element name="TemporalOperator"
                   type="fes:TemporalOperatorType"
                   maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TemporalOperatorType">
   <xsd:sequence>
      <xsd:element name="TemporalOperands"
                   type="fes:TemporalOperandsType"
                   minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="name"
                  type="fes:TemporalOperatorNameType" use="required"/>
</xsd:complexType>
<xsd:simpleType name="TemporalOperatorNameType">
   <xsd:union>
      <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:enumeration value="After"/>
            <xsd:enumeration value="Before"/>
            <xsd:enumeration value="Begins"/>
            <xsd:enumeration value="BegunBy"/>
            <xsd:enumeration value="TContains"/>
            <xsd:enumeration value="During"/>
            <xsd:enumeration value="TEquals"/>
            <xsd:enumeration value="TOverlaps"/>
            <xsd:enumeration value="Meets"/>
            <xsd:enumeration value="OverlappedBy"/>
            <xsd:enumeration value="MetBy"/>
            <xsd:enumeration value="Ends"/>
            <xsd:enumeration value="EndedBy"/>
         </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:pattern value="extension:\w{2,}"/>
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:union>
</xsd:simpleType>
```

```
   <!-- EXTENSION CAPABILITIES -->
   <xsd:complexType name="Extended_CapabilitiesType">
      <xsd:sequence>
         <xsd:element name="AdditionalOperators"
                      type="fes:AdditionalOperatorsType" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="AdditionalOperatorsType">
      <xsd:sequence>
         <xsd:element name="Operator"
                      type="fes:ExtensionOperatorType"
                      minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ExtensionOperatorType">
      <xsd:attribute name="name" type="xsd:QName" use="required"/>
   </xsd:complexType>

</xsd:schema>
```

B.2.6 filterAll.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
   targetNamespace="http://www.opengis.net/fes/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   version="2.0.0">

   <xsd:include schemaLocation="query.xsd"/>
   <xsd:include schemaLocation="filter.xsd"/>
   <xsd:include schemaLocation="sort.xsd"/>
</xsd:schema>
```

Annex C
(informative)

Examples

C.1 General considerations

This annex contains a number of examples of filters. Since filters 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.

C.2 XPath example

To practically illustrate the use of XPath expressions for referencing the XML elements and attributes within the description of an XML encoded object consider the fictitious 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/3.2"
  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/3.2"
      schemaLocation="http://schemas.opengis.net/gml/3.2.1/gml.xsd"/>

  <complexType name="PersonPropertyType">
   <sequence>
     <element ref="myns:Person"/>
   </sequence>
   <attributeGroup ref="gml:AssociationAttributeGroup"/>
  </complexType>
  <element name="Person" type="myns:PersonType"
      substitutionGroup="gml:AbstractFeature"/>
  <complexType name="PersonType">
   <complexContent>
     <extension base="gml:AbstractFeatureType">
      <sequence>
        <element name="insuranceNumber" type="xsd:string"/>
        <element name="lastName">
         <simpleType>
           <restriction base="string">
            <maxLength value="30"/>
           </restriction>
         </simpleType>
        </element>
        <element name="firstName">
         <simpleType>
           <restriction base="string">
            <maxLength value="10"/>
           </restriction>
         </simpleType>
        </element>
        <element name="age" type="gml:MeasureType" nillable="true"/>
        <element name="sex" type="string"/>
        <element name="spouse" type="myns:PersonPropertyType"
            minOccurs="0"/>
        <element name="location"
            type="gml:PointPropertyType" minOccurs="0"/>
```

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

Example instances of the feature type “Person” might be:

```
<Person
  xmlns="http://www.someserver.com/myns"
  xmlns:gml="http://www.opengis.net/gml/3.2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.someserver.com/myns
            http://www.pvretano.com/Test/Person.xsd"
  gml:id="P1">
  <insuranceNumber>345678345</insuranceNumber>
  <lastName>Smith</lastName>
  <firstName>John</firstName>
  <age uom="years">35</age>
  <sex>male</sex>
```

```
  <spouce xlink:href="P2"/>
  <location>
   <gml:Point gml:id="PT1" srsName="urn:ogc:crs:EPSG::4326">
     <gml:pos>10 10</gml:pos>
   </gml:Point>
  </location>
  <mailAddress>
   <Address>
     <streetName>Main</streetName>
     <streetNumber>10</streetNumber>
     <city>SomeTown</city>
     <province>Ontario</province>
     <postalCode>M1R1K9</postalCode>
     <country>Canada</country>
   </Address>
  </mailAddress>
  <phone>4161234567</phone>
  <phone>4168901234</phone>
</Person>

<Person
  xmlns="http://www.someserver.com/myns"
  xmlns:gml="http://www.opengis.net/gml/3.2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.someserver.com/myns
            http://www.pvretano.com/Test/Person.xsd"
  gml:id="P2">
  <insuranceNumber>111222333</insuranceNumber>
  <lastName>Smith</lastName>
  <firstName>Jane</firstName>
  <age uom="years">32</age>
  <sex>female</sex>
  <spouce xlink:href="#P1"/>
  <location>
   <gml:Point gml:id="PT1" srsName="urn:ogc:crs:EPSG::4326">
     <gml:pos>15 31</gml:pos>
   </gml:Point>
  </location>
  <mailAddress>
   <Address>
     <streetName>Main</streetName>
     <streetNumber>10</streetNumber>
     <city>SomeTown</city>
     <province>Ontario</province>
     <postalCode>M1R1K9</postalCode>
     <country>Canada</country>
   </Address>
  </mailAddress>
  <phone>4161234567</phone>
  <phone>4165678901</phone>
</Person>>
```

Using XPath (as defined in W3C XML Path Language) expressions, each XML element within the description of a *Person* feature that is the root element of an XML document can be referenced as shown in Table C.1 (omitting the namespace qualifiers for the sake of clarity).

Table C.1 — XPath expressions and property values for Person example

XPath Expression in Person element context	Alternate XPath Expression in XML document context	Example Property Value
insuranceNumber	Person/insuranceNumber	111222333
lastName	Person/lastName	Smith
firstName	Person/firstName	Jane
age	Person/age	32
sex	Person/sex	female
spouse	Person/spouse	#P1
location	Person/location	<gml:Point> <gml:pos>15 31</gml:pos> </gml:Point>
mailAddress/Address/streetNumber	Person/mailAddress/Address/streetNumber	10
mailAddress/Address/streetName	Person/mailAddress/Address/streetName	Main St.
mailAddress/Address/city	Person/mailAddress/Address/city	SomeTown
mailAddress/Address/province	Person/mailAddress/Address/province	Ontario
mailAddress/Address/postalCode	Person/mailAddress/Address/postalCode	M1R1K9
mailAddress/Address/country	Person/mailAddress/Address/country	Canada
phone[1]	Person/phone[1]	416-123-4567
phone[2]	Person/phone[2]	416-567-8901

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 can 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.

C.3 XPath predicate example

The purpose of this example is to illustrate the use of XPath predicates to identify specific data elements to test if a Filter expression.

Consider the following sample dataset:

```
<?xml version="1.0" ?>
<wfs:FeatureCollection
  timeStamp="2008-09-07T19:00:00"
  numberReturned="2"
  numberMatched="unknown"
  xmlns="http://www.someserver.com/myns"
  xmlns:myns="http://www.someserver.com/myns"
  xmlns:wfs="http://www.opengis.net/wfs/2.0"
  xmlns:gml="http://www.opengis.net/gml/3.2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wfs/2.0
```

```
                    http://schemas.opengis.net/wfs/2.0.0/wfs.xsd
                    http://www.someserver.com/myns
                    http://www.someserver.com/schemas/BuildingSchema.xsd">

  <wfs:member>
   <Building gml:id="B1">
     <name>City hall</name>
     <addresses>
      <Address>
        <city>Bonn</city>
        <street>Oxfordstrasse</street>
        <number>1</number>
      </Address>
      <Address>
        <city>Bonn</city>
        <street>Breitestrasse</street>
        <number>5</number>
      </Address>
     </addresses>
   </Building>
  </wfs:member>
</wfs:FeatureCollection>
```

The following filter expression can be used to select all Building features in “Oxfordstrasse 1”:

```
<fes:Filter>
  <fes:PropertyIsEqualTo>
  <fes:ValueReference>/Building/addresses/Address[street='Oxfordstrasse']/number</fes:ValueReference>
   <fes:Literal>1</fes:Literal>
  </fes:PropertyIsEqualTo>
</fes:Filter>
```

C.4 XPath schema-element() example

The schema-element() function is a built-in XPath function (see W3C XML Path Language) that test an element to see if it is the same substitution group as the function argument.

Consider the examples XPath expression:

```
    /Person/mailAddress/USAddress/streetName
```

Person is the feature type name, *mailAddress* is a property of the type Person, USAddress is the property type of the *mailAddress*, *streetName* is a property of the *Address* type. Now consider the XPath expression:

```
/Person/mailAddress/schema-element(Address)/streetName
```

In this case, the WFS shall consider the property *streetName* of all subtypes of *Address*, i.e. USAddress, EUAddress, AUAddress.

C.5 Filter examples

EXAMPLE 1

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

```
<fes:Filter
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/fes/2.0
   http://schemas.opengis.net/filter/2.0.0/filterAll.xsd">
  <fes:PropertyIsEqualTo>
    <fes:ValueReference>SomeProperty</fes:ValueReference>
    <fes:Literal>100</fes:Literal>
  </fes:PropertyIsEqualTo>
</fes: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.

```
<fes:Filter
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/fes/2.0
   http://schemas.opengis.net/filter/2.0.0/filterAll.xsd">
  <fes:PropertyIsLessThan>
    <fes:ValueReference>DEPTH</fes:ValueReference>
    <fes:Literal>30</fes:Literal>
  </fes:PropertyIsLessThan>
</fes:Filter>
```

EXAMPLE 3

This example encodes a simple spatial filter. In this case, one is 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.

```
<fes:Filter
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/fes/2.0
   http://schemas.opengis.net/filter/2.0.0/filterAll.xsd
   http://www.opengis.net/gml/3.2
   http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <fes:Not>
      <fes:Disjoint>
         <fes:ValueReference>Geometry</fes:ValueReference>
         <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
            <gml:lowerCorner>13.0983 31.5899</gml:lowerCorner>
            <gml:upperCorner>35.5472 42.8143</gml:upperCorner>
         </gml:Envelope>
      </fes:Disjoint>
   </fes:Not>
</fes:Filter>
```

An alternative encoding of this filter could have used to fes:BBOX element:

```
<fes:Filter
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/fes/2.0
   http://schemas.opengis.net/filter/2.0.0/filterAll.xsd
   http://www.opengis.net/gml/3.2
   http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <fes:BBOX>
      <fes:ValueReference>Geometry</fes:ValueReference>
      <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
         <gml:lowerCorner>13.0983 31.5899</gml:lowerCorner>
         <gml:upperCorner>35.5472 42.8143</gml:upperCorner>
      </gml:Envelope>
   </fes:BBOX>
</fes:Filter>
```

EXAMPLE 4

In this example, Examples 2 and 3 are combined 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 m.

```
<fes:Filter
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/fes/2.0
   http://schemas.opengis.net/filter/2.0.0/filterAll.xsd
   http://www.opengis.net/gml/3.2
   http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <fes:And>
```

```
      <fes:PropertyIsLessThan>
         <fes:ValueReference>DEPTH</fes:ValueReference>
         <fes:Literal>30</fes:Literal>
      </fes:PropertyIsLessThan>
      <fes:Not>
         <fes:Disjoint>
            <fes:ValueReference>Geometry</fes:ValueReference>
               <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
                  <gml:lowerCorner>13.0983 31.5899</gml:lowerCorner>
                  <gml:upperCorner>35.5472 42.8143</gml:upperCorner>
               </gml:Envelope>
         </fes:Disjoint>
      </fes:Not>
   </fes:And>
</fes:Filter>
```

EXAMPLE 5

A fes: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 fes:Filter element.

A filter applied to a GML version 3 data store:

```
<fes:Filter
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/fes/2.0
   http://schemas.opengis.net/filter/2.0.0/filterAll.xsd">
   <fes:ResourceId rid="TREESA_1M.1234"/>
   <fes:ResourceId rid="TREESA_1M.5678"/>
   <fes:ResourceId rid="TREESA_1M.9012"/>
   <fes:ResourceId rid="INWATERA_1M.3456"/>
   <fes:ResourceId rid="INWATERA_1M.7890"/>
   <fes:ResourceId rid="BUILTUPA_1M.4321"/>
</fes:Filter>
```

EXAMPLE 6

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

```
<fes:Filter
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/fes/2.0
   http://schemas.opengis.net/filter/2.0.0/filterAll.xsd">
   <fes:PropertyIsEqualTo>
      <fes:Function name="SIN">
         <fes:ValueReference>DISPERSION_ANGLE</fes:ValueReference>
      </fes:Function>
      <fes:Literal>1</fes:Literal>
   </fes:PropertyIsEqualTo>
</fes:Filter>
```

EXAMPLE 7

This example assumes that the server advertises support for a function called “Add” in its filter capabilities document. The example encodes a filter that includes an arithmetic expression. This filter is equivalent to the expression PROPA = PROPB + 100.

```
<fes:Filter
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/fes/2.0
   http://schemas.opengis.net/filter/2.0.0/filterAll.xsd">
   <fes:PropertyIsEqualTo>
      <fes:ValueReference>PROPA</fes:ValueReference>
      <fes:Function name="Add">
         <fes:ValueReference>PROPB</fes:ValueReference>
         <fes:Literal>100</fes:Literal>
      </fes:Function>
   </fes:PropertyIsEqualTo>
</fes:Filter>
```