INTERNATIONAL STANDARD

ISO 19142

First edition
2010-12-15

Geographic information — Web Feature Service

Information géographique — Service d'accès aux entités géographiques par le web

Reference number
ISO 19142:2010(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

Figures

Tables

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 19142 was prepared by Technical Committee ISO/TC 211, *Geographic Information/Geomatics*, in collaboration with the Open Geospatial Consortium, Invc. (OGC). The Web Feature Service (WFS) was originated within OGC.

Introduction

The Web Feature Service (WFS) represents a change in the way geographic information is created, modified and exchanged on the Internet. Rather than sharing geographic information at the file level using File Transfer Protocol (FTP), for example, the WFS offers direct fine-grained access to geographic information at the feature and feature property level. Web feature services allow clients to only retrieve or modify the data they are seeking, rather than retrieving a file that contains the data they are seeking and possibly much more. That data can then be used for a wide variety of purposes, including purposes other than their producers' intended ones.

In the taxonomy of services defined in ISO 19119, the WFS is primarily a feature access service but also includes elements of a feature type service, a coordinate conversion/transformation service and a geographic format conversion service.

Geographic information — Web Feature Service

1 Scope

This International Standard specifies the behaviour of a web feature service that provides transactions on and access to geographic features in a manner independent of the underlying data store. It specifies discovery operations, query operations, locking operations, transaction operations and operations to manage stored parameterized query expressions.

Discovery operations allow the service to be interrogated to determine its capabilities and to retrieve the application schema that defines the feature types that the service offers.

Query operations allow features or values of feature properties to be retrieved from the underlying data store based upon constraints, defined by the client, on feature properties.

Locking operations allow exclusive access to features for the purpose of modifying or deleting features.

Transaction operations allow features to be created, changed, replaced and deleted from the underlying data store.

Stored query operations allow clients to create, drop, list and describe parameterized query expressions that are stored by the server and can be repeatedly invoked using different parameter values.

NOTE This International Standard does not address the access control issues.

This International Standard defines 11 operations:

— GetCapabilities (discovery operation);

— DescribeFeatureType (discovery operation);

— GetPropertyValue (query operation);

— GetFeature (query operation);

— LockFeature (locking operation);

— GetFeatureWithLock (query and locking operation);

— Transaction (transaction operation);

— CreateStoredQuery (stored query operation);

— DropStoredQuery (stored query operation);

— ListStoredQueries (stored query operation);

— DescribeStoredQueries (stored query operation).

2 Conformance

Table 1 specifies the conformance classes defined by this International Standard and the tests specified in Annex A that shall be satisfied in order to comply with each class.

Table 1 also lists the following.

a) Which, if any, filter encoding (see ISO 19143:2010, Clause 2) conformance tests need to be satisfied with each WFS conformance class.

b) Which, if any, GML (see ISO 19136:2007) conformance tests need to be satisfied with each WFS conformance class.

Table 1 — Conformance classes

Conformance class name	Operation or behaviour	WFS conformance test	FES conformance test(s)	GML conformance test(s)
Simple WFS	The server shall implement the following operations: GetCapabilities, DescribeFeatureType, ListStoredQueries, DescribeStoredQueries, GetFeature operation with at least the StoredQuery action. One stored query, that fetches a feature using its id, shall be available, but the server may also offer additional stored queries. Additionally, the server shall conform to at least one of the HTTP GET, HTTP POST or SOAP conformance classes.	A.1.1	ISO 19143:2010, A.1	ISO 19136:2007, A.1.1, A.1.4, A.1.5, A.1.7, B.3, B.5, B.2.3
Basic WFS	The server shall implement the Simple WFS conformance class and shall additionally implement the GetFeature operation with the Query action and the GetPropertyValue operation.	A.1.2	ISO 19143:2010, A.2, A.7, A.8, A.10, A.11, A.12, A.14	ISO 19136:2007, B.4
Transactional WFS	The server shall implement the Basic WFS conformance class and shall also implement the Transaction operation.	A.1.3		
Locking WFS	The server shall implement the Transactional WFS conformance class and shall implement at least one of the GetFeatureWithLock or LockFeature operations.	A.1.4		
HTTP GET	The server shall implement the Key-value pair encoding for the operations that the server offers.	A.1.5		
HTTP POST	The server shall implement the XML encoding for the operations that the server implements.	A.1.6		
SOAP	The server shall implement XML-encoded requests and results within SOAP Envelopes.	A.1.7		
Inheritance	The server shall implement the schema-element() function for XPath expressions.	A.1.8	ISO 19143:2010, A.15	

Table 1 (*continued*)

Conformance class name	Operation or behaviour	WFS conformance test	FES conformance test(s)	GML conformance test(s)
Remote resolve	The server shall implement the ability to resolve remote resource references.	A.1.9		ISO 19136:2007, B.2.1
Response paging	The server shall implement the ability to page through the set of response features or values.	A.1.10		ISO 19136:2007, B.3
Standard joins	The server shall implement join predicates using all Filter operators except the spatial and temporal operators.	A.1.11	ISO 19143:2010, A.8, A.10	
Spatial joins	The server shall implement join predicates using spatial operators.	A.1.12	ISO 19143:2010, A.11, A.12	
Temporal joins	The server shall implement join predicates using temporal operators.	A.1.13	ISO 19143:2010, A.9, A.10	
Feature versions	The server shall implement the ability to navigate feature versions.	A.1.14	ISO 19143:2010, A.11	
Manage stored queries	The server shall implement the CreateStoredQuery and the DropStoredQuery operations.	A.1.15	ISO 19143:2010, A.1	

3 Normative references

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

ISO/TS 19103:2005, *Geographic information — Conceptual schema language*

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

ISO 19143:2010, *Geographic information — Filter encoding*

IETF RFC 2616, *Hypertext Transfer Protocol — HTTP/1.1* (June 1999)

IETF RFC 4646, *Tags for Identifying Languages* (September 2006)

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

OGC 07-092r3, *Definition identifier URNs in OGC namespace*, OGC® Best Practices (15 January 2009)

W3C SOAP, *Simple Object Access Protocol (SOAP) 1.2*, W3C Note (27 April 2007)

W3C WSDL, *Web Services Description Language (WSDL) 1.1*, W3C Note (15 March 2001)

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** (4.6)

[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** (4.17) from a **server** (4.28)

[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 (4.5) 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** (4.3) 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** (4.1) 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. The term "feature type" or "feature instance" should be used when only one is meant.

4.8
feature identifier
identifier that uniquely designates a **feature** (4.7) instance

4.9
filter expression
predicate expression encoded using XML

[ISO 19143:2010, definition 4.11]

4.10
interface
named set of **operations** (4.17) that characterize the behaviour of an entity

[ISO 19119:2005, definition 4.2]

4.11
join predicate
filter expression (4.9) that includes one or more clauses that constrain properties from two different entity types

[ISO 19143:2010, definition 4.16]

NOTE In this International Standard, the entity types will be **feature** (4.7) types.

4.12
join tuple
set of two or more object instances that satisfy a filter that includes **join predicates** (4.11)

NOTE In this International Standard, the object instances will be **feature** (4.7) instances.

4.13
local resource
resource that is under the direct control of a system

NOTE In this International Standard, the system is a web feature service and the resource is held in a data store that is directly controlled by that service.

4.14
locator attribute
attribute (4.1) whose value is a reference to a **local resource** (4.13) or **remote resource** (4.20)

NOTE In XML, this attribute is commonly called an href and contains a URI reference to the remote resource (see W3C XLink).

4.15
Multipurpose Internet Mail Extensions (MIME) type
media type and subtype of data in the body of a message that designates the native representation (canonical form) of such data

[IETF RFC 2045:1996]

4.16
namespace
<XML>
collection of names, identified by a URI reference, which are used in XML documents as **element** (4.6) names and **attribute** (4.1) names

[W3C XML Namespaces:1999]

4.17
operation
specification of a transformation or query that an object may be called to execute

[ISO 19119:2005, definition 4.3]

4.18
property
facet or attribute of an object, referenced by a name

[ISO 19143:2010, definition 4.21]

4.19
resource
asset or means that fulfils a requirement

[ISO 19115:2003, definition 4.10]

NOTE In this International Standard, the resource is a **feature** (4.7), or any identifiable component of a feature (e.g. a property of a feature).

4.20
remote resource
resource that is not under the direct control of a system

NOTE In this International Standard, the system is a web feature service. The resource is not held in any data store that is directly controlled by that service and thus cannot be directly retrieved by the service.

4.21
request
invocation of an **operation** (4.17) by a **client** (4.2)

[ISO 19128:2005, definition 4.10]

4.22
relocate
<reference> update a reference to a resource that has been moved or copied to a new location

EXAMPLE A **server** (4.28) is generating a **response** (4.24) to a GetFeature **request** (4.21), it has to copy a referenced **feature** (4.7) into the response document and the server has to "relocate" the original link contained in the referencing feature to the copy placed in the response document.

4.23
resolve
retrieval of a referenced resource and its insertion into a server-generated response document

NOTE The insertion may be accomplished by either replacing the reference inline with a copy of the resource or by relocating the reference to point to a copy of the resource that has been placed in the response document.

4.24
response
result of an **operation** (4.17) returned from a **server** (4.28) to a **client** (4.2)

[ISO 19128:2005, definition 4.11]

4.25
response model
schema (4.26) defining the properties of each **feature** (4.7) type that can appear in the **response** (4.24) to a query **operation** (4.17)

NOTE This is the schema of feature types that a **client** (4.2) can obtain using the DescribeFeatureType operation (see Clause 9).

4.26
schema
formal description of a model

[ISO 19101:2002, definition 4.25]

NOTE In general, a schema is an abstract representation of an object's characteristics and relations to other objects. An XML schema represents the relationship between the **attributes** (4.1) and **elements** (4.6) of an XML object (for example, a document or a portion of a document).

4.27
schema <XML Schema>
collection of **schema** (4.26) components within the same target **namespace** (4.16)

[ISO 19136:2007, definition 4.1.54]

EXAMPLE Schema components of W3C XML Schema are types, **elements** (4.6), **attributes** (4.1), groups, etc.

4.28
server
particular instance of a **service** (4.29)

[ISO 19128:2005, definition 4.12]

4.29
service
distinct part of the functionality that is provided by an entity through **interfaces** (4.10)

[ISO 19119:2005, definition 4.1]

4.30
service metadata
metadata describing the **operations** (4.17) and geographic information available at a **server** (4.28)

[ISO 19128:2005, defintion 4.14]

4.31
traversal
<XML>
using or following an XLink link for any purpose

[W3C XLink:2001]

4.32
tuple
ordered list of values

[ISO 19136:2007, definition 4.1.63]

NOTE In this International Standard, the ordered list will generally be a finite sequence of **features** (4.7), each of a specific feature type.

4.33
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** (4.16) is <scheme>://<authority><path>?<query>.

5 Conventions

5.1 Abbreviated terms

CGI	Common Gateway Interface
CRS	Coordinate Reference System
DCP	Distributed Computing Platform
EPSG	European Petroleum Survey Group
FES	Filter Encoding Specification
GML	Geography Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
KVP	Keyword-value pairs
MIME	Multipurpose Internet Mail Extensions
OGC	Open Geospatial Consortium
OWS	OGC Web Service
SQL	Structured Query Language
SOAP	Simple Object Access Protocol
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
VSP	Vendor Specific Parameter
WFS	Web Feature Service
WSDL	Web Services Description Language
XML	Extensible Markup Language

5.2 Use of examples

This International Standard makes extensive use of XML examples. They are meant to illustrate the various aspects of a web feature service specified in this International Standard. The bulk of the examples can be found in Annex B with some examples embedded in the body of the specification. All examples reference fictitious servers and data. Thus, this International Standard does not assert that any XML or keyword-value pair encoded examples, copied from this International Standard, will necessarily execute correctly or validate using a particular XML validation tool.

5.3 XML schemas

Throughout this International Standard, XML Schema (see W3C XML Schema Part 1 and W3C XML Schema Part 2) fragments are used to define the XML encoding of WFS operations. These fragments are gathered into a single validated schema file in Annex C.

5.4 UML Notation

5.4.1 Class diagrams

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

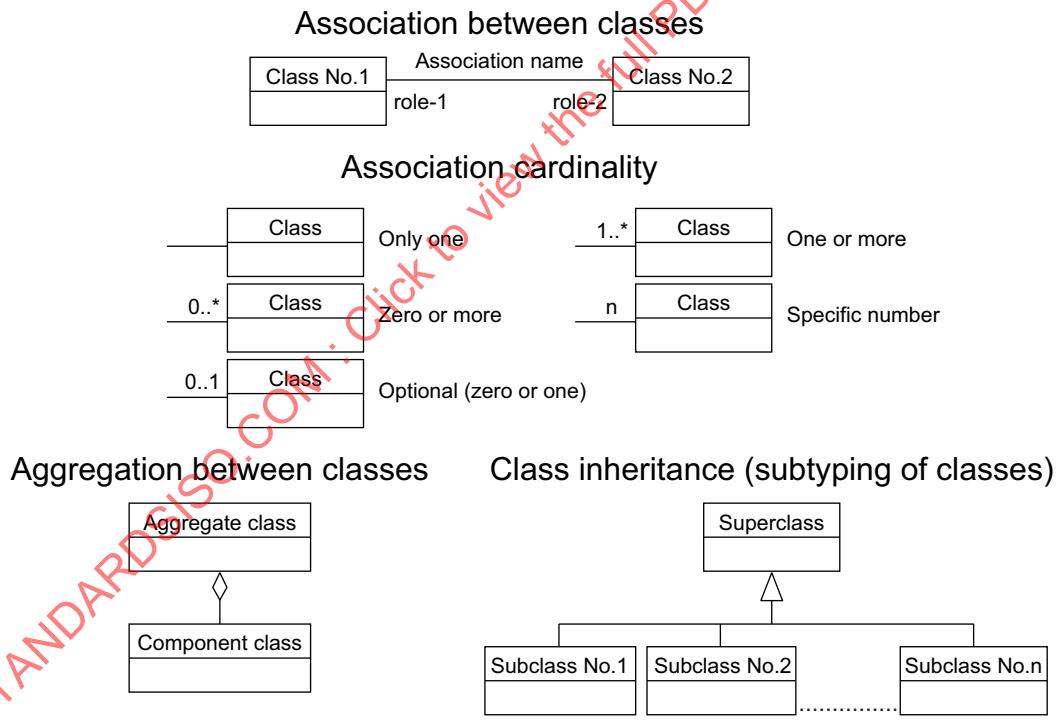

Figure 1 — UML notation in class diagrams

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

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

e) <<Type>> 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>> 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.

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

1) CharacterString – A sequence of characters;

2) LocalisedCharacterString – A CharacterString associated with a locale;

3) Boolean – A value specifying TRUE or FALSE;

4) URI – An identifier of a resource that provides more information;

5) Integer – An integer number.

5.4.2 State machine notation

The approach to dynamic modelling used is that described by the UML Reference Manual. The main technique is the state machine view. A summary of the UML notation for state diagrams is shown in Figure 2.

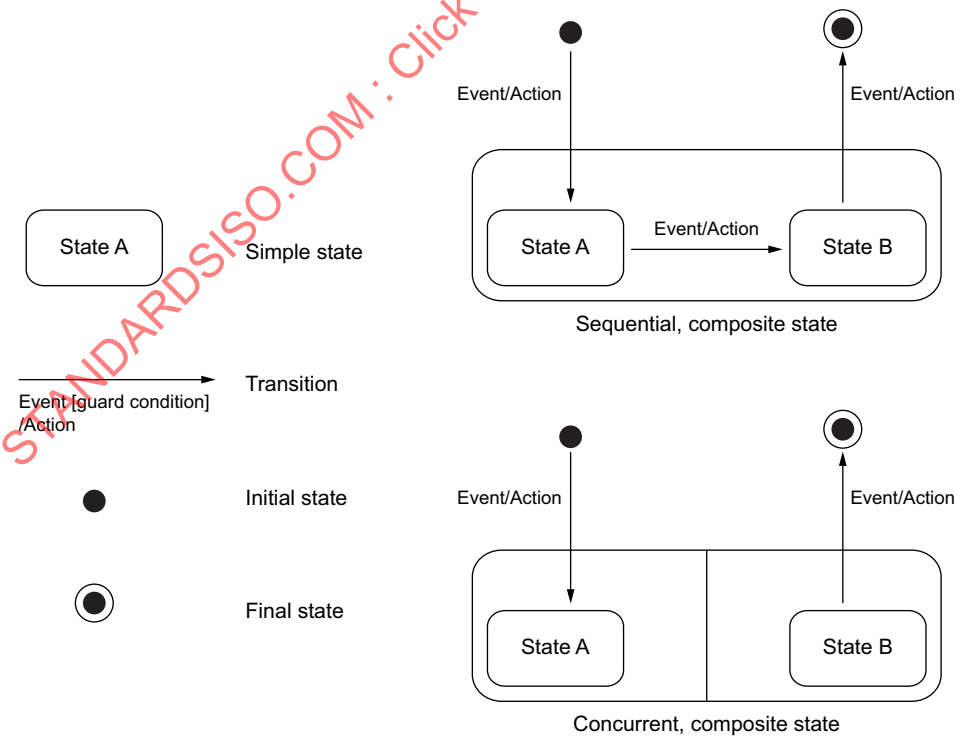

Figure 2 — Summary of UML state diagram notations

6 Basic service elements

6.1 Introduction

This clause describes aspects of a Web Feature Service's behaviour that are independent of particular operations or are common to several operations or interfaces.

6.2 Version numbering and negotiation

6.2.1 Version number form and value

Version numbers shall be encoded as described in OGC 06-121r3:2009, 7.3.1. Implementations of this International Standard shall use the value "2.0.0" as the protocol version number.

6.2.2 Appearance in service metadata and in requests

A web feature server that conforms to this International Standard shall list the version "2.0.0" in its service metadata (see 8.3). A server may support several versions whose values clients can discover according to a set of version negotiation rules (see OGC 06-121r3:2009, 7.3.2).

The version number used in requests to a server shall be equal to a version number that the server has declared it supports (except during negotiation as described in 6.2.3). If the server receives a request with a version number that it does not support, the server shall raise an InvalidParameterValue exception (see 7.5).

The version number 2.0.0 shall be specified for any request that conforms to this International Standard.

6.2.3 Version number negotiation

All WFS implementations shall support version negotiation according to the rules described in OGC 06-121r3:2009, 7.3.2.

6.2.4 Request encoding

This International Standard defines two methods of encoding WFS requests. The first uses XML as the encoding language. The second encoding uses keyword-value pairs (KVP) to encode the various parameters of a request.

EXAMPLE An example of a keyword-value pair is "REQUEST=GetCapabilities", where "REQUEST" is the keyword and "GetCapabilities" is the value.

The KVP encoding is a subset of the XML encoding since KVP encoding is not amenable for encoding certain WFS operations, such as the Transaction operation. In both cases, the response to a request or exception reporting shall be identical.

Table 2 correlates WFS operations and their encoding semantics as defined in this International Standard.

Table 2 — Operation request encoding

Operation	Request encoding
GetCapabilities	XML & KVP
DescribeFeatureType	XML & KVP
GetPropertyValue	XML & KVP
GetFeature	XML & KVP
GetFeatureWithLock	XML & KVP
LockFeature	XML & KVP
Transaction	XML
CreateStoredQuery	XML
DropStoredQuery	XML & KVP
ListStoredQueries	XML & KVP
DescribeStoredQueries	XML & KVP

6.2.5 KVP parameter encoding rules

6.2.5.1 Introduction

Aspects of KVP encoding WFS requests are described in 6.2.5.2 and 6.2.5.3.

6.2.5.2 Parameter ordering and case

Parameter names shall not be case-sensitive, but parameter values shall be case-sensitive. In this International Standard, parameter names used in KVP encoding are typically shown in uppercase for typographical clarity, not as a requirement.

Parameters in a request may be specified in any order.

A web feature service shall be prepared to encounter parameters that are not part of this International Standard. In terms of producing results according to this International Standard, a web feature service shall ignore such parameters.

6.2.5.3 Parameter lists

Parameters consisting of lists shall be encoded as described in ISO 19143:2010, 5.5.

EXAMPLE In the case where multiple join queries (see 7.9.2.5.3) are KVP-encoded, parentheses are used to isolate the various parameters so that their values align correctly. This is illustrated by the following example:

```
TYPENAMES=(ns1:F1,ns2:F2)(ns1:F1,ns1:F1)&ALIASES=(A,B)(C,D)&FILTER=(<Filter> … for A,B …
    </Filter>)(<Filter>…for C,D…</Filter>)
```

This KVP-encoded fragment encodes two join queries. The first query joins the feature types "ns1:F1" and "ns2:F2" which have been aliased to "A" and "B". The second query joins the feature types "ns1:F1" and "ns1:F1" (a self-join) which have been aliased to "C" and "D". The FILTER parameter encodes two filters, one for each query, delimited with parentheses. All the parameter values align 1:1. This fragment is equivalent to the following two individual KVP-encoded fragments:

```
TYPENAMES=ns1:F1,ns2:F2&ALIASES=A,B&FILTER=<Filter>…for A,B…</Filter>
TYPENAMES=ns1:F1,ns1:F1&ALIASES=C,D&FILTER=<Filter>…for C,D…</Filter>
```

6.3 Namespaces

Namespaces (see W3C XML Namespaces) are used to discriminate XML vocabularies from one another. For the WFS, there are four normative namespace definitions, namely:

— (http://www.opengis.net/wfs/2.0) – for the WFS interface vocabulary;

— (http://www.opengis.net/gml/3.2) – for the GML vocabulary (see ISO 19136:2007);

— (http://www.opengis.net/fes/2.0) – for the OGC Filter vocabulary (see ISO 19143:2010, 5.4);

— (http://www.opengis.net/ows/1.1) – for the OWS Common vocabulary (see OGC 06-121r3).

In addition, WFS implementations may make use of one or more GML Application Schemas and these schemas will, in turn, use one or more application namespaces (e.g. http://www.someserver.com/myns). While many of the examples in this International Standard use a single namespace, multiple namespaces may be used, as shown in 11.3.3.

For XML-encoded requests, the namespaces used in the request shall be encoded using the notation "xmlns:prefix=namespace_uri" in the root element of the request (see W3C XML Namespaces).

For KVP-encoded requests, the NAMEPSPACES parameter (see 7.6.6) shall be used to declare any namespaces used in the request.

6.4 Service bindings

The main body of this International Standard defines the encoding of WFS request and response messages, independent of any particular communication protocol. However, implementations of this International Standard shall support one of HTTP GET, HTTP POST or SOAP over HTTP POST (see Clause 2). Annex D contains a detailed discussion of these service bindings.

7 Common elements

7.1 Encoding of features

Servers that conform to this International Standard shall operate upon features encoded using GML. The version of GML that shall be supported is ISO 19136:2007. However, the operations in this International Standard are defined in a manner that allows them to work with previous and future versions of GML. Therefore, servers may implement support for additional versions of GML, other than ISO 19136:2007. Servers shall advertise all supported versions of GML in their capabilities document, using the inputFormat and outputFormat parameter domains (see Table 12).

Servers may also support additional non-GML feature encodings that shall also be listed in the server's capabilities document (see Table 12). However, this International Standard does not describe how a server would operate upon such encodings.

7.2 Resource identifiers

7.2.1 Assigning resource identifiers

Each feature instance in a WFS shall be assigned a persistent unique resource identifier which is assigned by the server when the feature is created.

This identifier shall be invariant under all WFS operations, including delete, which means that a resource identifier cannot be reused once it has been assigned.

Resource identifiers are not intended to associate WFS resources with real-world objects and the values do not have to be meaningful outside the scope of a web feature service instance.

7.2.2 Encoding resource identifiers

For features encoded using GML, the resource identifier shall be encoded in the XML attribute gml:id. This International Standard does not describe how resource identifiers are encoded for other output formats.

Within filter expressions, specific feature instances can be identified using the fes:ResourceId element. If the server supports versioning, specific versions of a feature can be referenced using the versionAction attribute contained within an fes:ResourceId element (see ISO 19143:2010, 7.11.2).

7.2.3 Version identification

If the server supports versioning of features, then the server shall maintain version information about each feature instance. This International Standard makes no assumptions about how that version information is maintained. Functions defined in the Filter Encoding Standard (see ISO 19143:2010, 7.11.2) allow version navigation based on the resource identifier.

7.3 Property references

7.3.1 XPath subset

GML allows features to have complex or aggregate non-geometric properties. A problem thus arises about how components of the complex value of such properties are referenced in the various places where value references are required (e.g. query and filter expressions). When the feature content that a WFS offers is encoded using XML, a WFS shall use XPath (see W3C XML Path Language) expressions for referencing the properties and components of the value of properties of a feature. The minimum mandatory subset of XPath that servers shall support is described in ISO 19143:2010, 7.4.4.

Supporting the schema-element() XPath function is optional. However, if the server conforms to the Inheritance conformance class (see Table 1) then the server shall also support the schema-element() function (see ISO 19143:2010, 7.4.4).

7.3.2 Accessor function

In GML, a feature property may contain its value as content encoded inline or reference its value with a simple XLink (see ISO 19136:2007, 7.2.3). This means that, in the course of evaluating an XPath expression, a server may need to resolve a resource reference. To accommodate this requirement, all WFS implementations shall provide a concrete implementation of an XPath accessor function called wfs:valueOf().

The argument to the function shall be the name of a property of a feature and the response shall be the value of the property. The value could simply be a text node or a list of element nodes that is the value of the named property.

The function shall resolve all locally referenced resources and, if the server advertises in its capabilities document that it can resolve remote references (see Table 13), the function shall resolve all remote resource references as well. In the event that the server only supports locally referenced resources, and it encounters a remotely referenced resource, the server shall raise an OptionNotSupported exception (see OGC 06-121r3:2009, Table 25).

When to use the valueOf() function in an XPath expression can be determined by inspecting the application schema of the server. If a property is declared so as to allow value references (see ISO 19136:2007, 7.2.3.3 and 7.2.3.7) then the wfs:valueOf() function should be specified in an XPath expression (see B.2.2 and B.4.5).

7.4 Predicate expression encoding

A number of operations defined in this International Standard contain predicate expressions that identify a subset of features to be operated upon. Such predicate expressions may enumerate a specific set of features to operate on, or a set of features may be defined by specifying constraints on the properties and/or components of the value of properties of a feature type.

XML-encoded predicate expressions shall be encoded using the fes:Filter element as described in ISO 19143:2010, 7.2.

KVP-encoded predicate expressions shall be encoded using the parameters describes in ISO 19143:2010, Table 2.

The specific set of predicates that a web feature service implements shall be advertised in the server's capabilities document using the filter capabilities section (see 8.3.3).

All implementations of this International Standard shall, at a minimum, implement the Query conformance class (see ISO 19143:2010, Table 1).

All implementations of this International Standard that implement the Basic WFS conformance class (see Table 1) shall also implement the Minimum Spatial Filter conformance class (see ISO 19143:2010, Table 1).

7.5 Exception reporting

In the event that a web feature service encounters an error while processing a request or receives an invalid request, it shall generate an XML document indicating that an error has occurred. The format of the XML error response is specified by, and shall validate against, the exception response schema defined in Clause 8 of the OWS Common Implementation Specification (see OGC 06-121r3:2009).

An ows:ExceptionReport element may contain one or more WFS processing exceptions specified using the ows:Exception element. The mandatory version attribute is used to indicate the version of the service exception report schema. This value shall be "2.0.0". The optional language attribute may be used to indicate the language used. The code list for the language parameter is defined in IETF RFC 4646.

Individual exception messages are contained within the ows:ExceptionText element. The mandatory code attribute shall be used to associate an exception code with the accompanying message.

The optional locator attribute may be used to indicate where an exception was encountered in the request that generated the error. Table 3 indicates what value the locator parameter should have for each exception code.

Table 3 — WFS exception codes

exceptionCode values	Meaning of code	"Locator" value	Conformance Class
CannotLockAllFeatures	A locking request with a lockAction of ALL failed to lock all the requested features.	If the operation includes the optional "handle" parameter, report its value as the value of the "location" parameter.	Locking WFS
DuplicateStoredQueryIdValue	The identifier specified for a stored query expression is a duplicate.	The "locator" parameter shall contain the value of the duplicate identifier.	Manage stored queries
DuplicateStoredQuery ParameterName	This specified name for a stored query parameter is already being used within the same stored query definition.	The "locator" parameter shall list the name of the duplicate stored query parameter.	Manage stored queries
FeaturesNotLocked	For servers that do not support automatic data locking (see 15.2.3.1), this exception indicates that a transaction operation is modifying features that have not previously been locked using a LockFeature (see Clause 12) or GetFeatureWithLock (see Clause 13) operation.	If the operation includes the optional "handle" parameter, report its value as the value of the "location" parameter.	Locking WFS
InvalidLockId	The value of the lockId parameter on a Transaction operation is invalid because it was not generated by the server.	The "locator" parameter shall contain the value of the invalid lockId.	Locking WFS
InvalidValue	A Transaction (see Clause 15) has attempted to insert or change the value of a data component in a way that violates the schema of the feature.	The "locator" parameter shall contain the name of the property being incorrectly modified.	Transactional WFS
LockHasExpired	The specified lock identifier on a Transaction or LockFeature operation has expired and is no longer valid.	The "locator" parameter shall contain the value of the expired lock identifier.	Locking WFS
OperationParsingFailed	The request is badly formed and failed to be parsed by the server.	The "locator" parameter shall contain the value of the "handle" parameter if one is available. Otherwise, the "locator" parameter shall contain the name of the badly formed operation.	All (see Table 1)
OperationProcessingFailed	An error was encountered while processing the operation.	The "locator" parameter shall contain the value of the "handle" parameter if one is available. Otherwise, the "locator" parameter shall contain the name of the operation that failed.	All (see Table 1)
ResponseCacheExpired	The response cache used to support paging has expired and the results are no longer available.	If the operation includes the optional "handle" parameter, report its value as the value of the "locator" parameter.	Response paging

Servers shall implement the generic exception codes in Table 25 of OGC 06-121r3:2009 for all conformance classes defined in this International Standard.

Multiple exceptions may be reported in a single exception report, so server implementations should endeavour to report as many exceptions as necessary to clearly describe a problem.

EXAMPLE If parsing an operation fails because the value of a parameter is invalid, the server should report an OperationParsingFailed exception and an InvalidParameterValue exception.

Annex B contains examples of exception reports.

7.6 Common request parameters

7.6.1 Introduction

Request parameters common to all or several operations defined in this International Standard are described in 7.6.2 to 7.6.6.

7.6.2 Base request type

7.6.2.1 Request semantics

The base request type (see Figure 3) is an abstract type from which all WFS operations, except the GetCapabilities operation, are subtyped.

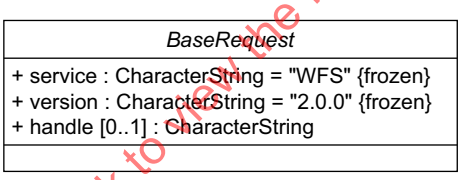

Figure 3 — BaseRequest

7.6.2.2 XML encoding

The following XML Schema fragment specifies the XML encoding of the type BaseRequest:

```
<xsd:complexType name="BaseRequestType" abstract="true">
   <xsd:attribute name="service" type="xsd:string"
                  use="required" fixed="WFS"/>
   <xsd:attribute name="version" type="xsd:string"
                  use="required" fixed="2.0.0"/>
   <xsd:attribute name="handle" type="xsd:string"/>
</xsd:complexType>
```

7.6.2.3 KVP encoding

Table 4 defines the KVP encoding of the base request type.

The value of the mandatory REQUEST keyword shall indicate which service operation is being invoked.

NOTE XML-encoded requests do not have a REQUEST parameter because the name of the root element encodes the name of the service operation being invoked.

Table 4 — KVP encoding of the base request type

URLComponent	Operation	O/M[a]	Description
SERVICE	All operations.	M	See 7.6.2.4.
VERSION[b] (All operations)	All operations except GetCapabilities.	M	See 7.6.2.5.

[a] O = Optional, M = Mandatory.

[b] VERSION is mandatory for all operations except the GetCapabilities operation.

7.6.2.4 service parameter

In XML, this parameter shall be encoded using an attribute named service (see 7.6.2.2).

In the KVP encoding, this parameter shall be encoded using the SERVICE keyword (see 7.6.2.3).

The mandatory service parameter shall be used to indicate which of the available service types, at a particular server, is being invoked. When invoking a web feature service, the value of the service parameter shall be "WFS".

7.6.2.5 version parameter

In XML, this parameter shall be encoded using an attribute named version (see 7.6.2.2).

In KVP encoding, this parameter shall be encoded using the VERSION keyword (see 7.6.2.3).

All WFS requests (except the GetCapabilities operation) shall include a parameter called version.

The version parameter shall be used to indicate to which version of the WFS specification the request encoding conforms and is used in version negotiation as described in 6.2.3. When encoding a WFS request in accordance with this International Standard, the value of the version attributed shall be fixed to 2.0.0, which corresponds to the version of this International Standard.

7.6.2.6 handle parameter

In XML, this parameter shall be encoded using an attribute named handle (see 7.6.2.2).

This parameter is not defined for KVP encoding.

The handle parameter may optionally be specified on a request.

The purpose of the optional handle parameter is to allow a client application to associate a mnemonic name with a request for error handling purposes.

If a handle is specified for an operation and an exception is encountered (see 7.5) when processing that operation, a Web Feature Service shall assign the value of the handle attribute to the locator attribute in the ows:ExceptionText element (see OGC 06-121r3:2009, Clause 8) in order to identify the operation or action that generated the exception. If a handle is not specified, the server may omit the locator attribute in the ows:ExceptionText element or may use some other means, such as line numbers, to locate the exception within the operation. The handle attribute is particularly useful when used with the Transaction operation (see Clause 15) which can contain many actions. Specifying a handle for each action allows the server to exactly locate an exception within a Transaction operation.

7.6.3 Standard presentation parameters

7.6.3.1 Parameter semantics

Standard presentation parameters (see Figure 4) are used to control how query results are presented in a response document. These parameters may appear in the GetPropertyValue (see Clause 10), GetFeature (see Clause 11) and GetFeatureWithLock (see Clause 13) operations.

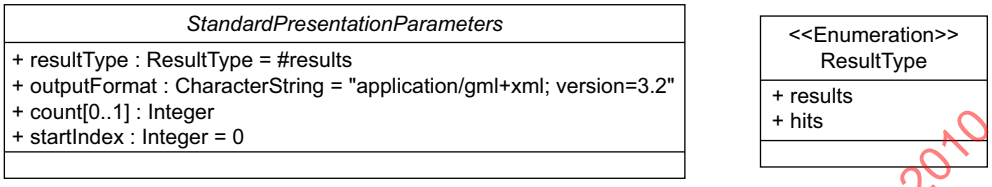

Figure 4 — StandardPresentationParameters

7.6.3.2 XML encoding

The following fragment defines the XML encoding to the standard presentation parameters:

```
<xsd:attributeGroup name="StandardPresentationParameters">
   <xsd:attribute name="startIndex"
                  type="xsd:nonNegativeInteger" default="0"/>
   <xsd:attribute name="count"
                  type="xsd:nonNegativeInteger"/>
   <xsd:attribute name="resultType"
                  type="wfs:ResultTypeType" default="results"/>
   <xsd:attribute name="outputFormat"
                  type="xsd:string" default="application/gml+xml; version=3.2"
</xsd:attributeGroup>
<xsd:simpleType name="ResultTypeType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="results"/>
      <xsd:enumeration value="hits"/>
   </xsd:restriction>
</xsd:simpleType>
```

7.6.3.3 KVP encoding

Table 5 defines the KVP encoding of the standard presentation parameters.

Table 5 — KVP encoding of standard presentation parameters

URLComponent	Operation	O/M[a]	Default	Description
STARTINDEX	GetPropertyValue, GetFeature, GetFeatureWithLock	O	1	See 7.6.3.4.
COUNT	GetPropertyValue, GetFeature, GetFeatureWithLock	O	1	See 7.6.3.5.
OUTPUTFORMAT	DescribeFeatureType, GetPropertyValue, GetFeature, GetFeatureWithLock	O	application/gml+xml; version=3.2	See 7.6.3.7.
RESULTTYPE	GetPropertyValue, GetFeature, GetFeatureWithLock	O	results	See 7.6.3.6.

[a] O = Optional, M = Mandatory.

7.6.3.4 startIndex parameter

For XML-encoded requests, this parameter shall be encoded using an attribute named startIndex (see 7.6.3.2).

For KVP-encoded requests, this parameter shall be encoded using the STARTINDEX keyword (see 7.6.3.3).

The optional startIndex parameter indicates the index within the result set from which the server shall begin presenting results in the response document.

7.6.3.5 count parameter

For XML-encoded requests, this parameter shall be encoded using an attribute named count (see 7.6.3.2).

For KVP-encoded requests, this parameter shall be encoded using the COUNT keyword (see 7.6.3.3).

The optional count parameter limits the number of explicitly requested values (i.e. features or property values) that are presented in a response document.

Only values of the types explicitly requested as the value of the typeNames parameter (see 7.9.2.4.1) shall be counted in the tally. Nested values contained within the explicitly requested value types shall not be counted.

EXAMPLE The tally of features in the following XML fragment is 4. The embedded reference to the feature with gml:id="2" and the feature with gml:id="5", which is that value abc:Prop4, are not included in the tally.

```
<?xml version="1.0"?>
<wfs:FeatureCollection
   timeStamp="2010-08-01T22:47:02"
   numberMatched="4" numberReturned="4"
   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:xlink="http://www.w3.org/1999/xlink"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.someserver.com/myns ./myns.xsd
                       http://www.opengis.net/wfs/2.0
                       http://schemas.opengis.net/wfs/2.0.0/wfs.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
      <wfs:member>
         <myns:Feature gml:id="1">
         ...
         </myns:Feature>
      </wfs:member>
      <wfs:member>
         <myns:Feature gml:id="2">
         ...
         </myns:Feature>
      </wfs:member>
      <wfs:member>
         <myns:Feature gml:id="3">
         ...
         </myns:Feature>
      </wfs:member>
      <wfs:member>
         <myns:Feature gml:id="4">
            <myns:Property1> ... <myns:Property1>
            <myns:Property2> ... <myns:Property2>
            <myns:Property3 xlink:href="#2"/>
            <myns:Property4>
               <myns:Feature gml:id="5">
               ...
               </myns:Feature>
            </myns:Property4>
         </myns:Feature>
      </wfs:member>
</wfs:FeatureCollection>
```

In the case of a join query (see 7.9.2.5.3.1), each tuple of explicitly requested value types shall be counted as one in the tally. Nested value members contained within any of the explicitly requested value types of the tuple do not count.

The count value applies to the entire result set (i.e. the result set generated by processing one or more Query actions) and the constraint shall be applied to the values in the order in which they are presented. Once the count limit is reached, request processing may terminate and the response document, containing, at most, count values, shall be presented to the client.

There is no predefined default value defined for the count parameter and the absence of the parameter shall mean that all values in the result set shall be presented to the client, subject to any server, a configured limit. If the server does have a server configured count limit, that limit shall be advertised in the server's capabilities document using the CountDefault constraint (see Table 14).

7.6.3.6 resultType parameter

For XML-encoded requests, this parameter shall be encoded using an attribute named resultType (see 7.6.3.2).

For KVP-encoded requests, this parameter shall be encoded using the RESULTTYPE keyword (see 7.6.3.3).

A WFS can respond to a query operation in one of two ways (excluding an exception response). It may either generate a complete response document containing resources that satisfy the operation, or it may simply generate an empty response container that indicates the count of the total number of resources that the operation would return. Which of these two responses a WFS generates is determined by the value of the optional resultType parameter.

The possible values for this parameter are "results" and "hits".

If the value of the resultType parameter is set to "results", the server shall generate a complete response document containing resources that satisfy the operation. The root element of the response container shall include a count of the number of resources actually presented in the response document (see 7.7.4.3). The root element of the response container shall also include a count of the total number of resources that the operations actually found, which will always be equal to or greater than the number of resources presented in the response document (see 7.7.4.2).

If the value of the resultType attribute is set to "hits", the server shall generate an empty response document containing no resource instances and the root element of the response container shall contain the count of the total number of resources that the operation found (see 7.7.4.2). The value for the number of resources presented in the response document (see 7.7.4.3) shall be set to zero.

7.6.3.7 outputFormat parameter

For XML-encoded requests, this parameter shall be encoded using an attribute named outputFormat (see 7.6.3.2).

For KVP-encoded requests, this parameter shall be encoded using the OUTPUTFORMAT keyword (see 7.6.3.3).

The optional outputFormat parameter specifies the format used to encode resources in the response to a query operation. The default value is "application/gml+xml; version=3.2", indicating that resources in the response document shall be encoded using GML (see ISO 19136).

Every WFS that conforms to this International Standard shall support this default value.

A server may advertise additional values for the outputFormat parameter in its capabilities document (see 8.3.3), indicating that multiple output formats, including previous versions of GML, are supported. However, this International Standard does not assign any specific meaning to these additional values. In cases where additional outputFormat values are specified in the server's capabilities document, this International Standard recommends that a descriptive narrative be included for each value listed.

7.6.4 Standard resolve parameters

7.6.4.1 Parameter semantics

Servers that conform to this International Standard shall implement the ability to resolve local resource references.

Servers may optionally implement the ability to, upon request, resolve remote resource references and shall advertise this ability using the ImplementsRemoteResolve constraint in their capabilities document (see Table 13).

How resource references are handled by a server is controlled by the resolve, resolveDepth and resolveTimeout parameters as described in 7.6.4.2 to 7.6.4.7.

The standard resolve parameters (see Figure 5) may appear in the GetPropertyValue (see Clause 10), GetFeature (see Clause 11) and GetFeatureWithLock (see Clause 13) operations.

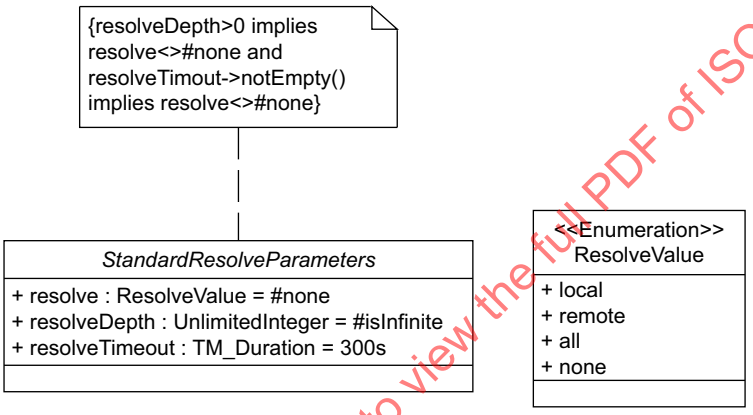

Figure 5 — StandardResolveParameters

7.6.4.2 XML encoding

The following fragment defines the XML encoding for the set of standard resolve parameters:

```
<xsd:attributeGroup name="StandardResolveParameters">
   <xsd:attribute name="resolve"
                  type="wfs:ResolveValueType" default="none"/>
   <xsd:attribute name="resolveDepth"
                  type="wfs:positiveIntegerWithStar" default="*"/>
   <xsd:attribute name="resolveTimeout"
                  type="xsd:positiveInteger" default="300"/>
</xsd:attributeGroup>
<xsd:simpleType name="ResolveValueType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="local"/>
      <xsd:enumeration value="remote"/>
      <xsd:enumeration value="all"/>
      <xsd:enumeration value="none"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="positiveIntegerWithStar">
   <xsd:union memberTypes="xsd:positiveInteger wfs:StarStringType"/>
</xsd:simpleType>
<xsd:simpleType name="StarStringType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="*"/>
   </xsd:restriction>
</xsd:simpleType>
```

7.6.4.3 KVP encoding

Table 6 defines the KVP encoding of the standard resolve parameters.

Table 6 — KVP encoding of standard resolve parameters

URLComponent	Operation	O/M[a]	Default	Description
RESOLVE	GetPropertyValue, GetFeature, GetFeatureWithLock	O	None	See 7.6.4.4.
RESOLVEDEPTH	GetPropertyValue, GetFeature, GetFeatureWithLock	O	*	See 7.6.4.5. RESOLVE parameter shall have a value other than “none”.
RESOLVETIMEOUT	GetPropertyValue, GetFeature, GetFeatureWithLock	O	Server Specific (see ResolveTimeoutDefault, Table 14)	See 7.6.4.6. RESOLVE parameter shall have a value other than “none”.

[a] O = Optional, M = Mandatory.

7.6.4.4 resolve parameter

For XML-encoded requests, this parameter shall be encoded using an attribute named resolve (see 7.6.4.2).

For KVP-encoded requests, this parameter shall be encoded using the RESOLVE keyword (see 7.6.4.3).

The optional resolve parameter controls whether and which (i.e. local or remote) resource references are resolved by an operation.

The value domain of the resolve parameter is: "local", "remote", "all" or "none". Depending on which capabilities a server implements, it may support some or all of this value domain. Which resolve parameter values a server supports shall be advertised in the server's capabilities document (see 8.3.3).

A resolve parameter value of "local" means that an operation shall only resolve local references.

A resolve parameter value of "remote" means that an operation shall only resolve remote resource references.

A resolve parameter value of "all" means that an operation shall resolve all resource references.

A resolve parameter value of "none" means that an operation shall not resolve any resource references. This is also the default value if the resolve parameter is not specified.

7.6.4.5 resolveDepth parameter

For XML-encoded requests, this parameter shall be encoded using an attribute named resolveDepth (see 7.6.4.2).

For KVP-encoded requests, this parameter shall be encoded using the RESOLVEDEPTH keyword (see 7.6.4.3).

The optional resolveDepth parameter indicates the depth to which nested resource references shall be resolved in a response document.

The range of valid values for this parameter consists of non-negative integers plus "*".

A value specified for the resolveDepth parameter shall only be used if the resolve parameter (see 7.6.4.4) is specified on the operation or action where the resolveDepth parameter appears and the value of the resolve parameter is not set to "none".

If a value is not specified for the resolve parameter or if the value of the resolve parameter is set to "none", the server shall ignore any value specified for the resolveDepth parameter.

If the value of the resolveDepth parameter is specified as "0", a server shall not resolve any resource references.

If the value of the resolveDepth parameter is specified as "1", a server shall resolve the immediate resource references and include their value in the response document. However, if those resolved resources contain any nested resource references, those nested references shall not be resolved.

If the value of the resolveDepth parameter is specified as "*", a server shall resolve all immediate resource references are all nested resource references.

Having resolved references to the specified resolveDepth, a server shall arrange that any deeper nested references are relocated to point to their intended resources.

EXAMPLE 1 This may require that references to local resources within the data store of a WFS be converted to remote reference back to the same server in a response document.

In the event that a circular reference is detected by the server before reaching the specified resolveDepth, the server shall abort further nested resource resolution and arrange for the last reference in the chain to reference the appropriate, already resolved, resource.

EXAMPLE 2 Consider the following chain of resource references: A -> B -> C -> D -> E -> B. If the resolveDepth value is set to 8, a server would resolve all resources to E (i.e. 4 levels) and then arrange for resource E to point to the already resolved resource B. At that point, the server shall cease resource resolution since the remaining resources, to a depth of 8 levels, have already been resolved due to the circular reference.

7.6.4.6 resolveTimeout parameter

For XML-encoded requests, the resolveTimeout parameter shall be encoded using an attribute named resolveTimeout (see 7.6.4.2).

For KVP-encoded requests, the resolveTimeout parameter shall be encoded using the RESOLVETIMEOUT keyword (see 7.6.4.3).

The optional resolveTimeout parameter controls how long a server shall wait to receive a response when resolving resource references.

The resolveTimeout parameter is of type xsd:positiveInteger and specifies the expiry time, in seconds. If the resolveTimeout parameter is not specified, the server wait time is implementation-dependent and shall be advertised in the server's capabilities document using the ResolveTimeoutDefault constraint (see Table 14).

A value specified for the resolveTimeout parameter shall only be used if the resolve parameter is specified on the request element that includes the resolveTimeout parameter and the value of the resolve parameter is not set to "none".

If the value of the resolve parameter is set to "none", the server shall ignore any value specified for the resolveTimeout parameter.

7.6.4.7 Unresolvable references

In the event that a server cannot resolve a resource reference, the server shall simply report the original unresolved URI in the response. This is not considered an exception.

7.6.5 Standard input parameters

7.6.5.1 Parameter semantics

Standard input parameters (see Figure 6) are a set of parameters used to assert the encoding of resources upon input and the CRS of any geometric values those resources might contain.

These parameters can be specified on the Insert (see 15.2.4), Update (see 15.2.5) and Replace (see 15.2.6) actions of the Transaction operation (see Clause 15).

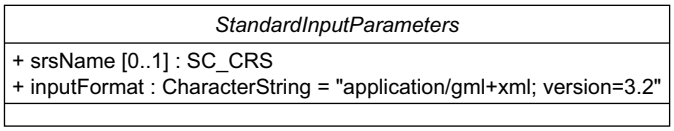

Figure 6 — StandardInputParameters

7.6.5.2 XML encoding

The following fragment defines the XML encoding for the standard input parameters.

```
<xsd:attributeGroup name="StandardInputParameters">
   <xsd:attribute name="inputFormat" type="xsd:string"
      default="application/gml+xml; version=3.2"/>
   <xsd:attribute name="srsName" type="xsd:anyURI"/>
</xsd:attributeGroup>
```

7.6.5.3 KVP encoding

The standard input parameters are not defined for KVP-encoded requests because a KVP encoding is not defined for the Transaction operation (see Clause 15).

7.6.5.4 inputFormat parameter

For XML-encoded requests, the inputFormat parameter shall be encoded using an attribute named inputFormat (see 7.6.5.2).

The inputFormat parameter is not defined for KVP-encoded requests.

The inputFormat parameter may be used to assert the feature encoding used to express features upon input using the Insert action of the Transaction operation (see 15.2.4.1) or when features are updated using the Update (see 15.2.5.1) or Replace (see 15.2.6.1) actions.

For servers that implement the Transactional WFS conformance class (see Table 1), the default value of the inputFormat attribute shall be "application/gml+xml; version=3.2", indicating that input features are encoded using GML (see ISO 19136).

A server may advertise additional values for the inputFormat parameter in its capabilities document (see 8.3.3), indicating that multiple input formats, including previous versions of GML, are supported. However, this International Standard does not assign any specific meaning to these additional values. In cases where additional inputFormat values are specified in the server's capabilities document, this International Standard recommends that a descriptive narrative be included for each value listed.

7.6.5.5 srsName parameter

For XML-encoded requests, the srsName parameter shall be encoded using an attribute named srsName.

The standard input parameter, srsName, is not defined for KVP-encoded requests.

The srsName parameter may be specified as a parameter of the Transaction operation (see Clause 15) or as a parameter on the transaction actions Insert (see 15.2.4.1), Update (see 15.2.5.1) or Replace (see 15.2.6.1).

If specified as a parameter of the Transaction operation, the value of the srsName attribute shall assert the default CRS used to encode feature geometries within the Transaction (see 15.2.2).

If an srsName value is specified as a parameter of the Transaction actions Insert, Update or Replace, its value shall supersede any CRS value specified using the srsName parameter on the Transaction operation.

The srsName (see ISO 19136:2007, 10.1.3.1) parameter may also be used to assert the CRS of an individual geometry and supersedes any previous CRS assertions.

If specified, the value of the srsName parameter shall be equivalent to the default CRS value specified using the wfs:DefaultCRS element in the capabilities document (see 8.3.3) or any of the wfs:OtherCRS values (see Table 11) for the relevant feature type.

If the specified CRS is not supported for the specified feature type, the WFS shall raise an InvalidParameterValue exception (see Table 3).

If the srsName parameter is not specified, the WFS shall interpret this to mean that geometries are encoded in the default CRS as specified using the wfs:DefaultCRS element in the capabilities document (see 8.3.3).

7.6.6 Additional common keywords for KVP-encoded requests

Table 7 defines additional keywords for KVP-encoded WFS requests.

Table 7 — Additional common keywords for KVP-encoded WFS requests

URLComponent	Operation	O/M[a]	Description
NAMESPACES	All operations	O	Used to specify namespaces and their prefixes. The format shall be *xmlns(prefix,escaped_url)* where *escaped_url* is defined in OGC 06-121r3:2009, 11.3. If the prefix is not specified then the default namespace shall be assumed. More that one namespace may be bound by specifying a comma-separated list of *xmlns()* values. This parameter is not defined for XML-encoded requests because XML has another mechanism for asserting namespaces (see 6.3).
VSPs		O	A server may implement additional KVP parameters that are not part of this International Standard. These are known as VSPs. VSPs allow vendors to specify additional parameters that will enhance the results of requests. A server shall produce valid results even if the VSPs are missing or malformed, or if VSPs are supplied that are not known to the server. Unknown VSPs shall be ignored. A server may choose not to advertise some or all of its VSPs. If VSPs are included in the Capabilities XML (see 8.3), the ows:ExtendedCapabilities element (see OGC 06-121r3:2009, 7.4.6) shall be extended accordingly. Additional schema documents may be imported containing the extension(s) of the ows:ExtendedCapabilities element. Any advertised VSP shall include or reference additional metadata describing its meaning (see 8.4). WFS implementers should choose VSP names with care, to avoid clashes with WFS parameters defined in this International Standard.

[a] O = Optional, M = Mandatory.

7.7 Standard response parameters

7.7.1 Parameter semantics

Standard response parameters (see Figure 7) are response parameters used in the definition of the response collection for the GetPropertyValue (see Clause 10), GetFeature (see Clause 11) and GetFeatureWithLock (see Clause 13) operations.

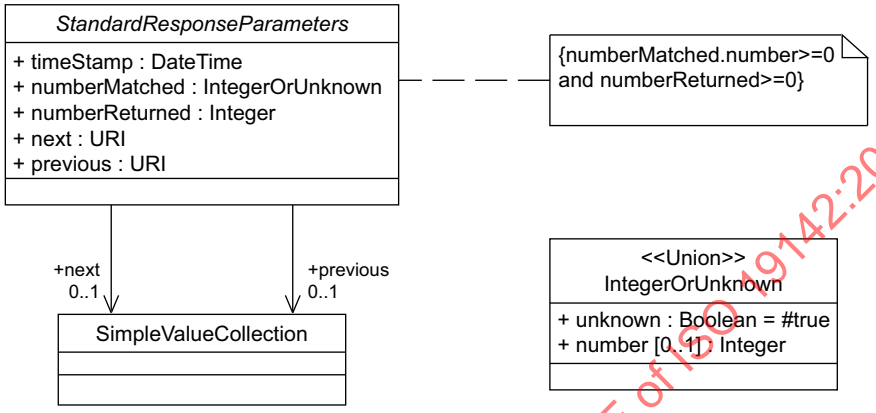

Figure 7 — StandardResponseParameters

7.7.2 XML encoding

The following XML Schema fragment defines the XML encoding of the standard response parameters:

```
<xsd:attributeGroup name="StandardResponseParameters">
   <xsd:attribute name="timeStamp" type="xsd:dateTime" use="required"/>
   <xsd:attribute name="numberMatched" type="xsd:nonNegativeIntegerOrUnknown" use="required"/>
   <xsd:attribute name="numberReturned" type="xsd:nonNegativeInteger" use="required"/>
   <xsd:attribute name="next" type="xsd:anyURI"/>
   <xsd:attribute name="previous" type="xsd:anyURI"/>
</xsd:attributeGroup>
<xsd:simpleType name="nonNegativeIntegerOrUnknown">
   <xsd:union>
      <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:enumeration value="unknown"/>
         </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType>
         <xsd:restriction base="xsd:nonNegativeInteger"/>
      </xsd:simpleType>
   </xsd:union>
</xsd:simpleType>
```

7.7.3 KVP encoding

Only an XML encoding is defined for WFS response messages.

7.7.4 Parameter discussion

7.7.4.1 timeStamp parameter

The timeStamp parameter shall be used by a WFS to indicate the time and date when a response collection was generated.

7.7.4.2 numberMatched parameter

The numberMatched parameter shall be used in a response document to report the total number of features or values, of the types requested in an operation, that are in the result set. This value does not necessarily have to match the number of features or values actually presented in the response document (see 7.7.4.3). If a server is unable to report the total number of matched features or values then it shall use the value "unknown" to indicate this.

By setting the value of the resultType parameter (see 7.6.3.6) to "hits", a count of the total number of features or values that an operation would return can be obtained without having to incur the cost of transmitting the result set. For servers that support response paging (see 7.7.4.4), the value of the next parameter shall be set to then fetch the first subset of response features or values.

7.7.4.3 numberReturned parameter

The numberReturned parameter shall be used to indicate a count of the number of features or values that are presented within a response document.

NOTE See 7.6.3.4 for value-counting rules.

If a value is specified for the count parameter (see 7.6.3.5), the value of the numberReturned parameter shall be equal to or less than the specified value of the count parameter.

If the value of the count parameter is not explicitly specified in the operation, a default value may be configured and advertised in the server's capabilities document (see 8.3.3) and shall have the same effect on numberReturned as if it were specified in the operation.

If the count parameter is not specified, either explicitly or through a server configuration, the value of the numberReturned response parameter shall be equal to the value of the numberMatched response parameter.

7.7.4.4 Response paging

7.7.4.4.1 Introduction

Response paging is the ability of a client to scroll through a set of response features or values, N features or values one at a time, much like one scrolls through the response from a search engine one page at a time.

Servers that support response paging shall advertise this fact in their capabilities document using the ImplementsResultPaging constraint (see Table 13).

Response paging is accomplished using the previous and next parameters defined on the response collections (see 10.3.1 and 11.3.1).

In order for paging to be triggered, either the count parameter (see 7.6.3.5) shall be set on the request or the server shall implement a default value for this parameter that shall be advertised in the server's capabilities document (see Table 14).

The value of the previous or next attribute shall be server-generated URIs that retrieve the corresponding set or results. The specific format of these URIs is implementation dependent, as are the details of how or if the server caches the results of an operation in order to be able to present them to the client one subset at a time.

Upon resolving the previous or next URIs, servers shall either generate a valid response collection containing the next or previous set of features or values (or join tuples in the case of a join), or an exception message indicating that the result set is no longer available (perhaps because the client waited too long before following the previous or next links and the results have, in the meantime, been purged from the server's cache). In this case, the server shall generate a ResponseCacheExpired exception (see Table 3).

Servers shall advertise how long result sets are cached for the purpose of response paging using the ResponseCacheTimeout constraint in their capabilities document (see Table 14).

EXAMPLE To illustrate the use of these parameters, consider the following GetFeature (see Clause 11) request:

```
<GetFeature service="WFS" version="2.0.0" count="100"
xmlns="http://www.opengis.net/wfs/2.0"
xmlns:cw="http://www.someserver.com/cw"
xmlns:fes="http://www.opengis.net/ogc/1.1"
xmlns:gml="http://www.opengis.net/gml/3.2">
   <Query typeNames="cw:MyFeatureType"/>
</GetFeature>
```

The sequence of interactions with the server proceeds as follows.

a) A client sends the request to a server that supports paging.

b) The server responds with a wfs:FeatureCollection element containing the first 100 records in the result set. The *next* attribute is set so that the client can retrieve the next 100 features, but the previous attribute is not set since this is the first set of features in the response set.

c) The client traverses the *next* URI.

d) The server responds with another wfs:FeatureCollection element containing the next 100 features in the result set. In this case, the server sets both the *previous* and *next* attribute values, so that the client can retrieve the previous 100 features or the next 100 features in the result set of the GetFeature request originally posted in a).

e) The client continues to traverse each *next* URI, until a <wfs:FeatureCollection> response is received without the *next* attribute set. This indicates that the last set of features, from the original request posted in a), has been retrieved.

f) Similarly, a client can traverse the *previous* URI until a <wfs:FeatureCollection> response is received that does have the *previous* attribute set, indicating that the first set of 100 features has been retrieved.

7.7.4.4.2 Transactional consistency of response paging

7.7.4.4.2.1 Declaring transactional consistency

Servers, in their capabilities document, shall advertise whether they support transactional consistency for response paging using the PagingIsTransactionSafe constraint (see Table 14).

7.7.4.4.2.2 Response paging with transactional consistency

Servers that declare transactional consistency shall maintain transactional consistency between paging iterations. Thus, the view of the data that a client sees while paging through the response set shall be consistent with respect to the time that the originating request was executed.

7.7.4.4.2.3 Response paging without transactional consistency

Servers that do not declare transactional consistency shall not be required to maintain transactional consistency or state between paging iterations. Thus, it is possible for new features to be added, updated or removed from the complete result set between iterations. As a consequence, it is possible for a result set element to be skipped or duplicated between iterations.

7.8 Use of the schemaLocation attribute

Any GML document generated by a WFS shall reference an appropriate GML application schema document so that the output can be validated. This shall be accomplished using the schemaLocation attribute, as defined in the XML Schema specification (see W3C XML Schema Part 1).

EXAMPLE The following XML fragment shows the use of the schemaLocation attribute on the root element indicating the location of an XML Schema document that can be used for validation:

```
<?xml version="1.0" ?>
<wfs:FeatureCollection
   xmlns="http://www.someserver.com/myns"
   xmlns:myns="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.someserver.com/wfs.cgi?request=DescribeFeatureType&amp;typenames=TreesA_1M,RoadL_1M"> …
```

In this example, the service uses a DescribeFeatureType operation (see Clause 9) to call back to itself to reference a schema document where the TreesA_1M and RoadL_1M elements are declared.

7.9 Query expressions

7.9.1 Introduction

A query expression is an action that directs a server to search its data store for resources that satisfy some filter expression encoded within the query. Query expressions search for resources of a single type or they can join two or more resource types and thus search for tuples of resources.

In this International Standard, the queryable resources are features (see 7.1 and ISO 19143:2010, 6.3.3.1.1).

A query expression resolves to a set of features or feature tuples that satisfy its filter expression. The result set is, in turn, operated upon by some operation defined in this International Standard.

EXAMPLE The GetFeature operation (see Clause 11) uses a query expression to identify a subset of features to present to a client in a response document. Similarly, the LockFeature operation (see Clause 12) uses a query expression to identify a subset of features to lock.

This International Standard defines two types of query expressions: ad hoc query expressions and stored query expressions.

Ad hoc query expressions are encoded in XML using the element wfs:Query. They are called "ad hoc" because the query is not stored by the server and is only known at runtime.

Stored query expressions are encoded in XML using the wfs:StoredQuery element. A stored query expression is a query that has been previously saved in the server's data store and can be executed at any time using the query's identifier.

Both query expression types are derived from the abstract query expression elements defined in Clause 6 of ISO 19143:2010.

7.9.2 Ad hoc query expression

7.9.2.1 Request semantics

An ad hoc query expression may be used in a GetPropertyValue (see Clause 10), GetFeature (see Clause 11), GetFeatureWithLock (see Clause 13) or LockFeature (see Clause 12) operation to identify the set of features to be operated upon.

As shown in Figure 8, an ad hoc query expression contains a typeNames parameter, projection clause, a selection clause and a sorting clause.

The mandatory typeNames parameter lists the name of one or more feature types to query.

The optional projection clause identifies a subset of optional feature properties that shall be presented in the result set. The projection clause for XML-encoded ad hoc query expressions may also be used to control, on a per-property basis, how resource references (see 7.6.4) are resolved in the response document.

NOTE How resource references are resolved in a response document can only be controlled at the operation level for KVP-encoded requests. This is accomplished using the RESOLVE, RESOLVEDEPTH and RESOLVETIMEOUT keywords (see 7.6.4.3). This is in contrast to XML-encoded requests where resource resolution can be controlled at the operation and property levels.

The optional selection clause specifies criteria that conditionally select features from a server's data store.

The optional sorting clause specifies how the features in the response document should be ordered.

An ad hoc query also allows a CRS to be asserted and used when presenting feature geometries in a response document.

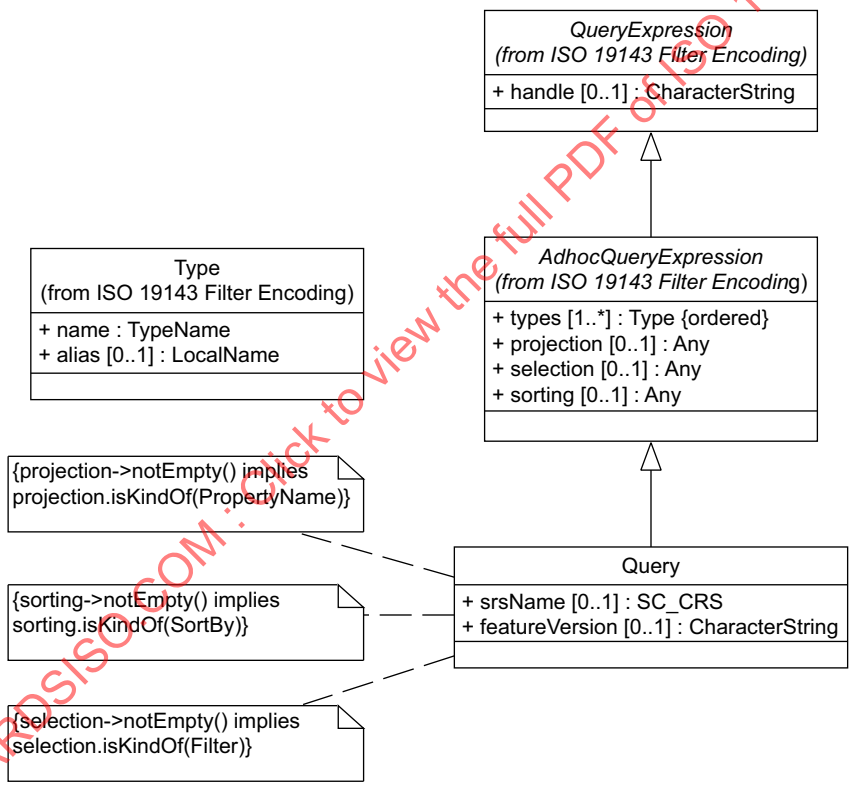

Figure 8 — Ad hoc query expression

7.9.2.2 XML encoding

The following XML Schema fragment defines the XML encoding for an ad hoc query expression:

```
<xsd:element name="Query" type="wfs:QueryType"
             substitutionGroup="fes:AbstractAdhocQueryExpression"/>
<xsd:complexType name="QueryType">
   <xsd:complexContent>
      <xsd:extension base="fes:AbstractAdhocQueryExpressionType">
         <xsd:attribute name="srsName" type="xsd:anyURI"/>
         <xsd:attribute name="featureVersion" type="xsd:string"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

The wfs:Query element includes the typeNames parameter inherited from the fes:AbstractAdhocQueryExpressionType type (see ISO 19143:2010, 6.3.2).

7.9.2.3 KVP encoding

Table 8 defines the KVP encoding for ad hoc query expressions.

Table 8 includes parameters from 7.9.2.4 and ISO 19143:2010, Table 2, which are necessary to KVP-encode an ad hoc query expression.

Table 8 — Keywords for ad hoc query KVP encoding

URL component	O/M[a]	Description
TYPENAMES	M[b]	See 7.9.2.4.1.
ALIASES	O	See 7.9.2.4.3.
SRSNAME	O	See 7.9.2.4.4.
Projection clause	O	See Table 9.
FILTER	O	See ISO 19143:2010, 6.3.3.
FILTER_LANGUAGE	O	See ISO 19143:2010, 6.3.3.
RESOURCEID	O	See ISO 19143:2010, 6.3.3.
BBOX	O	See OGC 06-121r3.
SORTBY	O	See ISO 19143:2010, Clause 8. The SORTBY parameter is used to specify a list of property names whose values should be used to order (upon presentation) the set of feature instances that satisfy the query. The value of the SORTBY parameter shall have the form *"PropertyName [ASC|DESC][,PropertyName [AASC|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, Field 2 descending and Field 3 ascending.

[a] O = Optional, M = Mandatory.

[b] The TYPENAMES parameter is mandatory in all cases, except when the RESOURCEID parameter is specified (see 7.9.2.4.1).

If multiple KVP-encoded query expressions appear in a WFS request, the parameter values for each query expression shall be isolated from one another using parentheses (see 6.2.5.3). If an optional parameter is specified for one query expression, then a value shall be specified for that parameter for all query expressions encoded in the request.

Only one of a set of mutually exclusive parameters shall be specified in a KVP-encoded request that encodes multiple query expressions; for the chosen parameter, a value shall be specified for each encoded query expression.

EXAMPLE If a KVP-encoded request contains multiple query expressions and one of those query expressions uses the FILTER keyword to encode the predicate, then all the query expressions in the request must use the FILTER keyword. You cannot, for example, have a KVP-encoded request with multiple query expressions where one query uses the FILTER keyword to encode the predicate and another query expression uses the RESOURCEID keyword.

If used in a request, the BBOX keyword shall only encode a single bounding box (see OGC 06-121r3:2009, 10.2.3). However, that bounding box predicate shall apply to all query expressions encoded in the request (see B.8.5.4). This is in contrast to the FILTER and RESOURCEID keywords which can encode multiple predicates corresponding to each query expression encoded in a KVP-encoded request (see B.8.4.15).

As is the case for XML-encoded requests, multiple KVP-encoded query expressions encoded in a single WFS operation shall be considered independent of each other.

7.9.2.4 Parameter discussion

7.9.2.4.1 typeNames parameter

For XML-encoded ad hoc query expressions, the typeNames parameter shall be encoded using the typeNames attribute on the wfs:Query element. This attribute is inherited from the abstract element:

```
fes:AbstractAdhocQueryExpressionType (see ISO 19143, 6.3.2).
```

For KVP-encoded ad hoc query expressions, the typeNames parameter shall be encoded using the TYPENAMES keyword (see Table 8). The TYPENAMES parameter is mandatory in all cases, except when the RESOURCEID parameter is specified. In this case, the TYPENAMES parameter may be omitted because each feature instance can be identified by its resource identifier alone (see 7.2). If both the TYPENAMES and RESOURCEID parameters are specified then all the feature instances identified by the RESOURCEID parameter shall be of the type specified by the TYPENAMES parameter; otherwise the server shall raise an InvalidParameterValue exception (see OGC 06-121r3:2009, Table 25) where the "locator" attribute (see OGC 06-121r3:2009, 8.4) value shall be set to "RESOURCEID".

The typeNames parameter (see ISO 19143:2010, 6.3.3.1.1) shall be used within an ad hoc query expression to encode the names of one or more correlated feature types to be queried. Individual feature type names shall be encoded as QName (see W3C XML Schema Part 2).

The value of each QName listed as a value of the typeNames parameter shall match one of the feature type names advertised in the server's capabilities document (see 8.3.3).

7.9.2.4.2 schema-element() function

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

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

7.9.2.4.3 aliases parameter

For XML-encoded ad hoc query expressions, the aliases parameter shall be encoded using the aliases attribute on the wfs:Query element. The aliases attribute is inherited from the abstract element:

```
fes:AbstractAdhocQueryExpressionType (see ISO 19143, 6.3.2).
```

For KVP-encoded ad hoc query expressions, the aliases parameter shall be encoded using the ALIASES keyword.

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

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

EXAMPLE 1

TYPENAMES=(ns1:FeatureType1,ns2:FeatureType2)(ns2:FeatureType2,ns2:FeatureType3)&ALIASES=(A,B)(C,D)

This KVP-encoded example encodes two ad hoc query expressions, each performing a join. The first expression is joining the feature types ns1:FeatureType1 and ns2:FeatureType2 which are aliased to A and B. The second expression is joining the feature type ns2:FeatureType2 and ns2:FeatureType3 which are aliased to C and D.

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

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

Aliases are typically used in query expressions that perform a join operation (see 7.9.2.5.3.1), to support self-join, that is a join of one feature type back to itself.

EXAMPLE 2

typeNames="myns:Feat1 myns:Feat1" aliases="a b"

This XML-encoded example shows the encoding of the typeName parameter. The first feature type, myns:Feat1, is aliased to the name "a" and the second feature type, myns:Feat1, is aliased to the name "b". Thus, properties from the first instance of myns:Feat1 can be referenced in a request as "/a/property_name", and properties from the second instance of myns:Feat1 can be referenced in a request as "/b/property_name", where the token "property_name" is used as a place holder for the name of any property of the feature myns:Feat1.

7.9.2.4.4 srsName parameter

The optional srsName attribute may be used to assert a specific WFS-supported CRS transformation to be applied to the geometries of the features returned in a response document.

The value of the srsName parameter may be the wfs:DefaultCRS or any of the wfs:OtherCRS values listed for the feature type in a server's capabilities document (see 8.3.3). If no srsName value is supplied, then the feature geometries shall be encoded in the response document using the advertised wfs:DefaultCRS value.

This attribute has no meaning for feature types with no spatial properties and shall be ignored.

Servers that advertise more than one wfs:OtherCRS value in their capabilities document (see 8.3.3) shall be able to transform between the CRS used to store features and any CRS requested using the srsName attribute.

Servers that implement this International Standard shall be able to process srsName attribute values using the following format model:

`urn:ogc:def:objectType:authority:version:`*`<EPSG code>`* (see OGC 07-092r3)

In this format model, objectType shall have the value of "crs", authority shall have the value "crs" and the value *<EPSG Code>* is a placeholder for the actual EPSG code value.

EXAMPLE srsName="urn:ogc:def:crs:EPSG::26986".

7.9.2.4.5 Projection clause

7.9.2.4.5.1 Request semantics

Every feature representation generated by a WFS shall include all the mandatory properties for the feature type according to the schema description (see Clause 9), and then may include a selection of the other properties, according to the schema description.

The projection clause enumerates which of the non-mandatory properties of a feature shall be included in the response to a query.

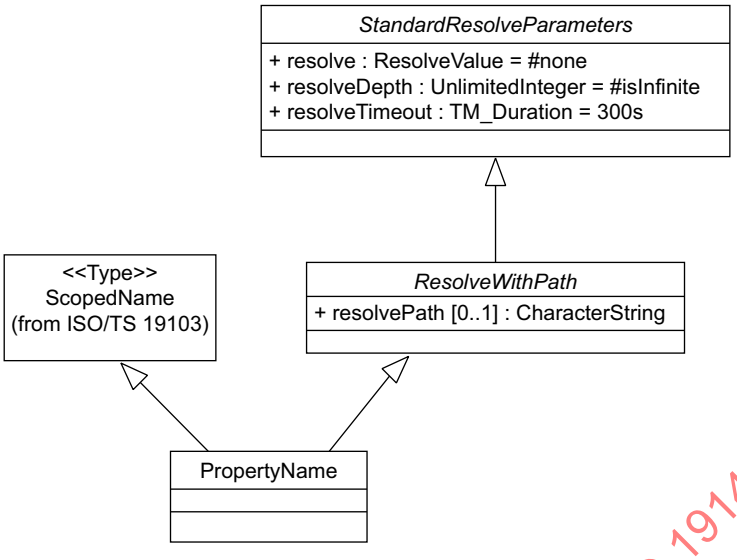

Figure 9 — Query projection clause

NOTE There is typically some flexibility in the structure of the description of a feature type of interest, especially concerning the optional or mandatory nature of each property. The W3C XML Schema (see W3C XML Schema Part 1) language, used to define GML application schemas, uses certain notations to indicate whether property elements are mandatory or optional or how many occurrences are valid. Thus, the GML representation of a feature may be schema-valid with only a subset of the possible properties present. A WFS client should be prepared to deal with a situation where it receives more property values than it requests using a projection clause.

7.9.2.4.5.2 XML encoding

The following XML-schema fragment defined the XML encoding for the wfs:PropertyName element:

```
<xsd:element name="PropertyName"
             substitutionGroup="fes:AbstractProjectionClause">
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="xsd:QName">
            <xsd:attributeGroup ref="wfs:StandardResolveParameters"/>
            <xsd:attribute name="resolvePath" type="xsd:string"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
```

7.9.2.4.5.3 KVP encoding

Table 9 defines the KVP encoding of a projection clause.

Table 9 — KVP encoding of projection clause

URL component	O/M[a]	Description
PROPERTYNAME	O	A list of non-mandatory properties to include in the response. If more that one feature type name is specified as the value of the TYPENAMES keyword (in a non-join query), a corresponding list of parameter lists shall be specified (see 6.2.5.3). Each sublist shall correspond 1:1 with each feature type name listed as the value of the TYPENAMES parameter.
StandardResolveParameters		See Table 6.

[a] O = Optional, M = Mandatory.

7.9.2.4.6 Parameter discussion

7.9.2.4.6.1 PropertyName parameter

For XML-encoded requests, the projection clause shall be encoded using one or more wfs:PropertyName elements. The value of each wfs:PropertyName element is a QName whose value shall match the name of one of the property names of one of the feature types listed in the typeNames attribute of the parent wfs:Query element in the GML representation of the relevant feature.

For KVP-encoded requests, the projection clause shall be encoded using the keyword PROPERTYNAME (see Table 9). The value of the PROPERTYNAME keyword is a list or multiple lists of a QName whose values shall match the name of one of the property names of one of the feature types listed as the value of the TYPENAMES keyword. Multiple lists of property names corresponding to multiple query expressions shall be isolated from one another by enclosing them within parentheses (see 6.2.5.3).

7.9.2.4.6.2 Standard resolve parameters

For XML-encoded requests, the standard resolve parameters shall be encoded using the attributes named resolve, resolveDepth and resolveTimeout (see 7.6.4.2) on the wfs:PropertyName element. For XML-encoded requests, the standard resolve parameters on the projection clause control, on a per-property basis, how resource references are to be resolved within a response document. A value for these parameters specified on the wfs:PropertyName element shall supersede a value, if specified, on any enclosing parent element.

For KVP-encoded requests, the standard resolve parameters shall be encoded using the keywords RESOLVE, RESOLVEDEPTH and RESOLVETIMEOUT (see 7.6.4.3). For KVP-encoded requests, how resource references are resolved in a response document cannot be controlled on a pre-property basis and so no equivalent KVP encoding for the standard resolve parameters is specified at this level. Instead, the keywords RESOLVE, RESOLVEDEPTH and RESOLVETIMEOUT may be used in KVP-encoded requests to control how resource references are resolved in a response document for all feature properties

7.9.2.4.7 Resolution path

For XML-encoded requests, the resolvePath parameter shall be encoded as an attribute named resolvePath.

The resolvePath parameter is not defined for KVP-encoded requests.

The resolvePath parameter modifies the behaviour of the resolve parameter. The normal behaviour of the resolve parameter, when its value is set to local, remote or all, is to resolve all resource references to the depth specified by the resolveDepth parameter (see 7.6.4.5).

The resolvePath parameter, however, shall trigger resource resolution in the response document only along a certain property path.

A value specified for the resolvePath parameter shall only be used if the value of the nearest (in scope) resolve parameter is not set to "none". If the value of the nearest resolve attribute is set to "none", any value specified for the resolvePath parameter shall be ignored.

The value of the resolvePath parameter is a path expression but its specific encoding depends on how features are encoded.

For GML, which is the canonical feature encoding in this International Standard, the value of the resolutionPath parameter shall be an XPath (see W3C XML Path Language) expression that results in an object element. Value resolution shall stop at these instances and the value of the resolveDepth parameter shall be ignored.

EXAMPLE 1 The following is an example of the resolvePath attribute used in a query on the feature type Parcel:

Feature Instance examples:

```
<Parcel gml:id="DEXXXX00000000">
   <registerEntry>
      <LandRegisterEntry gml:id="DEXXXX00000001">
         <relatedTo xlink:href="#DEXXXX00000002"/>
      </LandRegisterEntry>
   </registerEntry>
</Parcel>
<LandRegisterEntry gml:id="DEXXXX00000002">
   <sequenceNumber>1</sequenceNumber>
</LandRegisterEntry>
```

Query example:

```
<wfs:Query typeNames="Parcel">
   <wfs:PropertyName resolve="all"
   resolvePath="valueOf(relatedTo)">valueOf(registerEntry)</wfs:PropertyName>
   <fes:Filter>
      <!-- some filter expression -->
   </fes:Filter>
</wfs:Query>
```

Specifying the resolvePath attribute has the effect that, besides all the Parcel features meeting the criteria, all their associated LandRegisterEntry features and the LandRegisterEntry features associated with them along the “relatedTo” property are returned, as well.

If the schema-element() function is supported by a server in XPath expressions, it shall also be supported in resolvePath parameter. If used, it shall match not only the element name that is the parameter of the schema-element() function, but also all elements directly or indirectly in its substitution group.

EXAMPLE 2 schema-element(gml:AbstractFeature) would match all features.

7.9.2.5 Selection clause

7.9.2.5.1 XML Encoding

For XML-encoded requests, the selection clause shall be encoded using the fes:Filter element (see ISO 19143:2010, Clause 7).

7.9.2.5.2 KVP encoding

For KVP-encoded requests, the selection clause shall be encoded using one of the keywords FILTER, RESOURCEID or BBOX (see ISO 19143:2010, Table 2).

7.9.2.5.3 Join processing

7.9.2.5.3.1 Join queries

A WFS may optionally support join queries.

A join operation query finds tuples (i.e. pairs, triples, etc.) of features, among a list of feature types, that satisfy some join condition specified using a filter expression (see ISO 19143:2010, Clause 7). If the join condition is satisfied then that tuple of features shall be in the result set of the query expression.

A join operation query shall be encoded by

a) listing the feature types to join using the typeNames parameter (see 7.9.2.4.1), and

b) specifying a join predicate between the feature types using a filter expression (see ISO 19143:2010, Clause 7).

Servers that implement join queries shall implement an inner join, meaning that only feature tuples that match the join conditions shall be returned in the result set.

EXAMPLE 1 The following query expression uses a join to find the spouse of the person whose Identifier is "12345".

```
<wfs:Query typeNames="myns:Person myns:Person" aliases="a b">
   <fes:Filter>
      <fes:And>
         <fes:PropertyIsEqualTo>
            <fes:ValueReference>a/Identifier</fes:ValueReference>
            <fes:Literal>12345</fes:Literal>
         </fes:PropertyIsEqualTo>
         <fes:PropertyIsEqualTo>
            <fes:ValueReference>a/spouse<fes:ValueReference>
            <fes:ValueReference>b/Identifier</fes:ValueReference>
         </fes:PropertyIsEqualTo>
      </fes:And>
   </fes:Filter>
</wfs:Query>
```

In this example, a join predicate between "a/spouse" and "b/Identifier" is used to locate the spouse of the person whose identifier is "12345". This is also an example of a self-join since the myns:Person feature type is being joined to itself in order to identify the spouse.

EXAMPLE 2 The following query expression uses a spatial join to find all park features that contain lakes:

```
<wfs:Query typeNames="myns:Parks myns:Lakes">
   <fes:Filter>
      <fes:Contains>
         <fes:ValueReference>ns1:Parks/geometry</fes:PropertyName>
         <fes:ValueReference>ns1:Lakes/geometry</fes:PropertyName>
      </fes:Contains>
   </fes:Filter>
</wfs:Query>
```

The list of feature types to join is specified using the typeNames attribute (i.e. *typeName="myns:Parks myns:Lakes"*) on the wfs:Query element. The join predicate is specified using the fes:Filter element and finds all pairs of *ns1:Park* and *ns1:Lake* features whose geometries satisfy the spatial operator fes:Contains. The response to a request with a join query is described in 11.3.3.6.

Join queries are categorized as standard, spatial and temporal joins based on the operators that are used in the join predicates.

If all the Filter operators, except the spatial and temporal operators, may be used in a join predicate, the server implements standard join queries as in Example 1.

If spatial operators may be used in join predicates, the server implements spatial join queries as in Example 2.

If temporal operators may be used in join predicates, the server implements temporal join queries.

If a web feature service does support joins, it shall advertise this fact using the ImplementsStandardJoins, ImplementsSpatialJoins and ImplementsTemporalJoins constraints in its capabilities document (see 8.3.3).

7.9.2.5.3.2 Shared properties

In response to a join query that contains a projection clause that specifies the name of a property that is shared among two or more joined feature types, as is possible in GML (see ISO 19136:2007), that shared property shall appear in all features of the response tuple that contain that property.

EXAMPLE Consider two feature types ns1:A and ns2:B that share a property element called ns3:C. If the projection clause of an ad hoc join query on ns1:A and ns2:B specifies the common property ns3:C, how does a server know if the client means ns3:C from feature type ns1:A or ns3:C from feature type ns2:B? The answer is that the server does not know but also that it does not need to know since ns3:C should appear in both the ns1:A and ns2:B features in the response document.

7.9.2.5.3.3 Use of the schema-element() function in joins

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

In the event that a schema-element() function is specified in a join operation, the WFS shall raise an OperationNotSupported exception (see Table 3) where the "locator" attribute (see OGC 06-121r3:2009, 8.4) value shall be set to "schema-element()".

7.9.2.5.3.4 Coordinate reference system handling

When comparing two geometries having different srsName values in a predicate, the server shall either convert one of the geometries to the CRS of the other geometry or the server shall convert both geometries to a third common CRS before performing the comparison.

In the event that either or both geometries in a predicate do not have a CRS associated with them, the server shall assume that the geometry is in the default CRS asserted for its feature type in the server's capabilities document. Comparisons of the two geometries can then proceed as described in the previous sentence.

7.9.2.5.3.5 Units of measure handling

This International Standard does not define any support for handling conversions between unit of measure.

7.9.2.5.4 Sorting clause

7.9.2.5.4.1 Request semantics

The sorting clause of an ad hoc query expression shall be specified using a SortBy value (see Figure 10 and ISO 19143:2010, Clause 8).

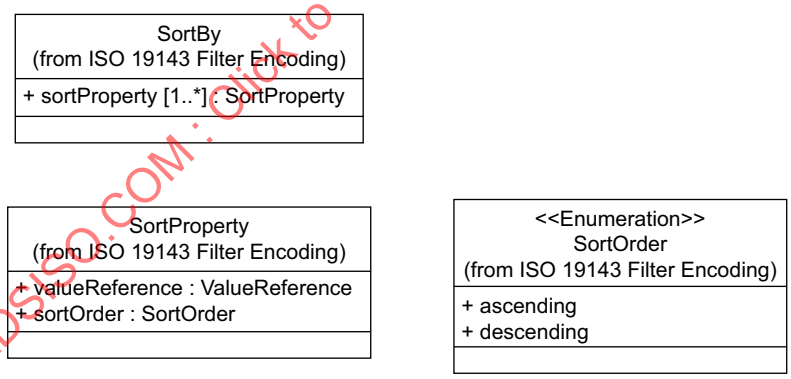

Figure 10 — Query sorting clause

7.9.2.5.4.2 XML encoding

In XML, the sorting clause of an ad hoc query expression shall be encoded using the fes:SortBy element (see ISO 19143:2010, Clause 8).

An ad hoc query expression shall have a minimum of zero and a maximum of one fes:SortBy child elements.

7.9.2.5.4.3 KVP encoding

For KVP-encoded requests, the sorting clause shall be encoded using the keyword SORTBY (see ISO 19143:2010, Table 2).

7.9.2.5.4.4 Sort processing

A web feature service that receives an ad hoc query expression without a sorting clause shall generate a response document in which features are presented in whatever order the server chooses. However, to comply with this International Standard, servers shall ensure that whatever order is presented when an ad hoc query, not containing a sort clause, is first executed, is preserved across subsequent executions of the same ad hoc query expression on the same set of features.

EXAMPLE A server may choose to sort the features by their gml:id if the client has not specified a specific sorting clause. Subsequent invocations of the same query expression on the same set of data should result in a response document that presents the features in the same order.

In the event that a web feature service receives a request containing an ad hoc query with a sorting clause, the service shall respond by presenting features in the response document in the requested order.

Sorting is only supported within the scope of a single ad hoc query expression. Global sorting over all features in a result set, generated by multiple ad hoc query expressions being encoded in the request, shall not be supported. The trivial exception to this statement, of course, is the case where the request contains a single query expression.

When processing a query with a sorting clause, the sort shall be executed in the context of all the data that matches the request parameters before the response is possibly reduced to a set within the limits of the count attribute (see 7.6.3.5).

7.9.3 Stored query expression

7.9.3.1 Request semantics

A stored query expression may be used in a GetPropertyValue (see Clause 10), GetFeature (see Clause 11), GetFeatureWithLock (see Clause 13) or LockFeature (see Clause 12) operation to identify a set of features to be operated upon.

A stored query expression (see Figure 11) is a persistent, parameterized, identifiable query expression. A stored query can be repeatedly invoked using its identifier with different values bound to its parameters each time.

All servers shall implement the ability to list, describe and execute stored queries.

All server implementations shall offer a stored query that fetches features based on their identifier. Additional stored queries that are packaged with the server may also be offered.

Clause 14 describes a set of operations for managing stored query expressions.

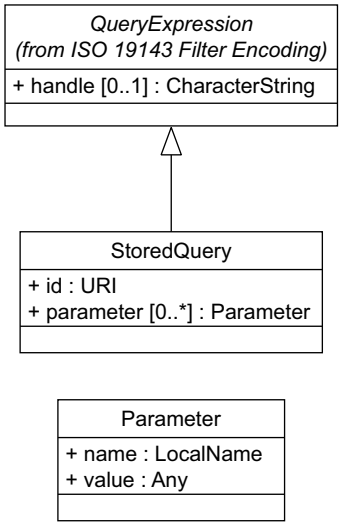

Figure 11 — StoredQuery

7.9.3.2 XML encoding

The following XML Schema fragment defines the XML encoding of a stored query expression:

```
<xsd:element name="StoredQuery" type="wfs:StoredQueryType"
             substitutionGroup="fes:AbstractQueryExpression"/>
<xsd:complexType name="StoredQueryType">
   <xsd:complexContent>
      <xsd:extension base="fes:AbstractQueryExpressionType">
         <xsd:sequence>
            <xsd:element name="Parameter" type="wfs:ParameterType"
                         minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="id" type="xsd:anyURI" use="required"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ParameterType" mixed="true">
   <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="1"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="xsd:string" use="required"/>
</xsd:complexType>
```

The abstract type fes:AbstractQueryExpressionType is described in ISO 19143:2010, 6.2.

7.9.3.3 KVP encoding

Table 10 defines the KVP-encoding for a stored query expression.

Table 10 — Keywords for Stored query KVP encoding

URL component	O/M[a]	Description
STOREDQUERY_ID	M	The identifier of the stored query to invoke.
storedquery_parameter=value	O	Each parameter of a stored query shall be encoded in KVP as a KVP. Stored query parameters shall not have names that conflict with any WFS parameter name.

[a] O = Optional, M = Mandatory.

Unlike ad hoc queries, where more than one query can be encoded in a single KVP-encoded request, only one stored query shall be invoked per KVP-encoded request.

7.9.3.4 Stored query identifier

For XML-encoded requests, the stored query identifier shall be encoded using the id attribute on the wfs:StoredQuery element.

For KVP-encoded requests, the stored query identifier shall be encoded using the STOREDQUERY_ID keyword.

Each stored query shall be assigned an identifier unique to the server that can be used to invoke the query.

7.9.3.5 Stored query parameters

For XML-encoded requests, each stored query parameter shall be encoded using a wfs:Parameter element. The name of the parameter shall be encoded as the value of the name attribute on the wfs:Parameter element. The value of the parameter shall be encoded as the content of the wfs:Parameter element.

For KVP-encoded requests, stored query parameters shall be encoded as keyword-value pairs. The parameter name shall be encoded as the keyword and its value shall be encoded as the value of the keyword-value pair.

EXAMPLE This example does a GetFeature request that invokes the GetTreesByArea stored query with the "AREA" parameter.

```
http://www.someserver.com/wfs.cgi?request=GetFeature&storedquery_id=urn-
        x:wfs:StoredQueryId:SomeCompanyName:GetTreesByArea&AREA=10000
```

A server shall ensure that stored query parameter names do not collide with WFS KVP keywords.

Stored query parameters are referenced by name and may thus appear in any order when encoding a stored query invocation.

7.9.3.6 GetFeatureById stored query

All servers shall implement a stored query that accepts a single argument, named "id" of type xsd:string, and returns a single feature whose identifier is equal to the specified value of the id argument.

This stored query shall have the identifier: urn:ogc:def:query:OGC-WFS::GetFeatureById.

EXAMPLE The following URL invokes the mandatory GetFeatureById stored query to retrieve and present a single feature from the server's data store:

```
http://www.someserver.com/wfs.cgi?SERVICE=WFS&
                                  VERSION=2.0.0&
                                  REQUEST=GetFeature&
                                  STOREDQUERY_ID=urn:ogc:def:query:OGC-WFS::GetFeatureById&
                                  ID=INWATERA_1M.1013
```


8 GetCapabilities operation

8.1 Introduction

The GetCapabilities operation generates a service metadata document describing a WFS service provided by a server.

All web feature services shall implement the KVP encoding of the GetCapabilities operation.

A web feature service may optionally implement the XML encoding of the GetCapabilities operation.

8.2 Request

8.2.1 Request semantics

Figure 12 describes the schema of a GetCapabilities request.

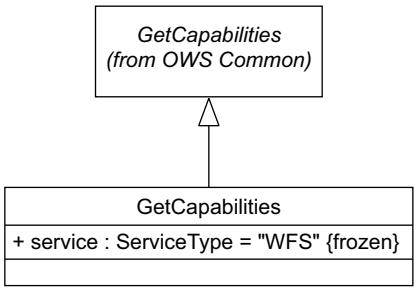

Figure 12 — GetCapabilities request

8.2.2 XML encoding

The following XML Schema fragment defines the XML encoding of the GetCapabilities request:

```
<xsd:element name="GetCapabilities" type="wfs:GetCapabilitiesType"/>
<xsd:complexType name="GetCapabilitiesType">
   <xsd:complexContent>
      <xsd:extension base="ows:GetCapabilitiesType">
         <xsd:attribute name="service" type="ows:ServiceType" use="required" fixed="WFS"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

The base type, ows:GetCapabilitiesType, is defined in the OWS Common Implementation Specification (see OGC 06-121r3:2009, 7.2.4).

8.2.3 KVP encoding

The KVP encoding of the GetCapabilities request shall be as specified in OGC 06-121r3:2009, 7.2.2.

8.3 Response

8.3.1 Response semantics

Figure 13 describes the schema of a GetCapabilities response.

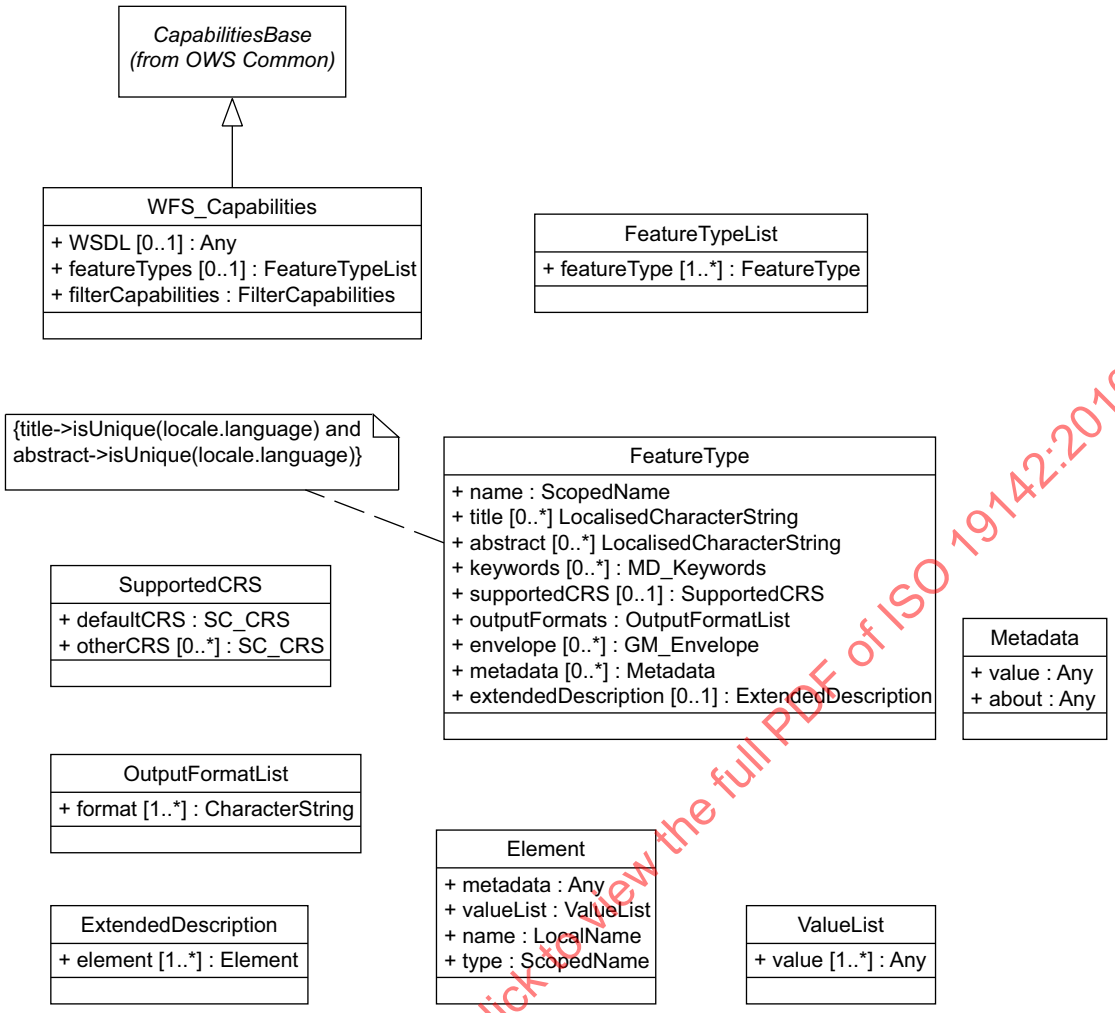

Figure 13 — GetCapabilities response

8.3.2 XML encoding

The root element of the response to a GetCapabilities request is the wfs:WFS_Capabilities element which is declared by the following XML Schema fragment:

```
<xsd:element name="WFS_Capabilities" type="wfs:WFS_CapabilitiesType"/>
<xsd:complexType name="WFS_CapabilitiesType">
   <xsd:complexContent>
      <xsd:extension base="ows:CapabilitiesBaseType">
         <xsd:sequence>
            <xsd:element name="WSDL" minOccurs="0">
               <xsd:complexType>
                  <xsd:complexContent>
                     <xsd:restriction base="xsd:anyType">
                        <xsd:attributeGroup ref="xlink:simpleLink"/>
                     </xsd:restriction>
                  </xsd:complexContent>
               </xsd:complexType>
            </xsd:element>
            <xsd:element ref="wfs:FeatureTypeList" minOccurs="0"/>
            <xsd:element ref="fes:Filter_Capabilities" minOccurs="0"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

The base type, ows:CapabilitiesBaseType, is defined in the OWS Common Implementation Specification (see OGC 06-121r3:2009, 7.2.4).

The elements ows:ServiceIdentification, ows:ServiceProvider and ows:OperationMetadata are inherited from the base type ows:CapabilitiesBaseType.

8.3.3 Capabilities document

In addition to the sections defined in OGC 06-121r3:2009, 7.4, the capabilities response document shall contain the following sections:

1) WSDL section (optional)

This section allows a server to reference an optional WSDL document that describes the operations that the service offers (see Annex E). This section may be included in addition to the OperationsMetadata section for tools that know how to use the WSDL.

2) FeatureType list section (mandatory)

This section defines the list of feature types that are offered by a web feature service. Lightweight metadata is provided about each feature type as described in Table 11.

3) Filter capabilities section (mandatory)

The schema of the Filter Capabilities Section is defined in ISO 19143:2010, 7.13, and is used to advertise the expressions that may be used to form query predicates.

NOTE In the schema, the wfs:FeatureTypeList and fes:Filter_Capabilities elements are specified as optional (i.e. minOccurs="0"). This is done in order to support the Sections parameter (see OGC 06-121r3:2009, 7.3.3) of the GetCapabilities request which allows abbreviated service metadata documents to be requested. However, when the full service metadata is generated, it shall contain a FeatureType list section and a Filter capabilities section.

8.3.4 FeatureTypeList section

The wfs:FeatureTypeList element shall contain a list of feature types, subtyped from gml:AbstractFeatureType, that a WFS offers.

The following XML Schema fragment defines the wfs:FeatureTypeList element:

```
<xsd:element name="FeatureTypeList" type="wfs:FeatureTypeListType"/>
<xsd:complexType name="FeatureTypeListType">
   <xsd:sequence>
      <xsd:element name="FeatureType" type="wfs:FeatureTypeType"
         maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="FeatureTypeType">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:QName"/>
      <xsd:element ref="wfs:Title" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wfs:Abstract" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="ows:Keywords" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:choice>
         <xsd:sequence>
            <xsd:element name="DefaultCRS" type="xsd:anyURI"/>
            <xsd:element name="OtherCRS" type="xsd:anyURI"
                         minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:element name="NoCRS">
            <xsd:complexType/>
         </xsd:element>
      </xsd:choice>
      <xsd:element name="OutputFormats" type="wfs:OutputFormatListType"
                   minOccurs="0"/>
      <xsd:element ref="ows:WGS84BoundingBox"
```

```
                  minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="MetadataURL" type="wfs:MetadataURLType"
                  minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="ExtendedDescription"
                  type="wfs:ExtendedDescriptionType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="OutputFormatListType">
   <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="Format" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="MetadataURLType">
   <xsd:attributeGroup ref="xlink:simpleLink"/>
   <xsd:attribute name="about" type="xsd:anyURI"/>
</xsd:complexType>
<xsd:complexType name="ExtendedDescriptionType">
   <xsd:sequence>
      <xsd:element ref="wfs:Element" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name="Element" type="wfs:ElementType"/>
<xsd:complexType name="ElementType">
   <xsd:sequence>
      <xsd:element ref="ows:Metadata"/>
      <xsd:element ref="wfs:ValueList"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="type" type="xsd:QName" use="required"/>
</xsd:complexType>
<xsd:element name="ValueList" type="wfs:ValueListType"/>
<xsd:complexType name="ValueListType">
   <xsd:sequence maxOccurs="unbounded">
      <xsd:element ref="wfs:Value"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name="Value" type="xsd:anyType"/>
```

The wfs:FeatureTypeList element shall contain one wfs:FeatureType element for each feature type that the service offers. The element contains metadata about the feature type.

Table 11 lists the elements that are used to describe each feature type listed within the wfs:FeatureTypeList element.

Table 11 — Elements to describe feature types

Element name	Description
Name	The namespace-qualified name of the feature type. This element is mandatory.
Title	An unordered list of zero or more human-readable titles that briefly identify this feature type in menus. The xml:lang attribute may be used to specify a language for the title. If more than one wfs:Title element is listed, each title shall have a different value for the xml:lang attribute.
Abstract	An unordered list of zero or more wfs:Abstract elements. Each wfs:Abstract element is a descriptive narrative for more information about the feature type. The xml:lang attribute may be used to specify a language for the abstract. If more than one wfs:Abstract element is listed, each abstract shall have a different value for the xml:lang attribute.
Keywords	The ows:Keywords element contains short words to aid catalogue searching.
DefaultCRS	The wfs:DefaultCRS element indicates which coordinate reference system shall be used by a WFS to express the state of a spatial feature, if not otherwise explicitly identified within a query or transaction request. For example, if a GetFeature request specifies no CRS value for the wfs:Query srsName attribute, any spatial properties of feature data satisfying the request shall be expressed using the wfs:DefaultCRS value. The CRS shall be encoded using the URL format defined in "Definition Identifier URNs in the OGC namespace" (see OGC 07-092r2). The wfs:DefaultCRS shall not necessarily be the internal storage CRS used for the feature data, and therefore should not be interpreted as such. If the wfs:DefaultCRS is different from the internal storage CRS, then the WFS shall support a transformation between the wfs:DefaultCRS and the internal storage CRS. The effects of such a transformation shall be considered when determining and declaring the guaranteed data accuracy.
OtherCRS	The wfs:OtherCRS element shall be used to indicate other supported CRSs within transaction and query requests. A 'supported CRS' means that the WFS supports the transformation of spatial properties between the wfs:OtherCRS and the internal storage CRS. The effects of such a transformation shall be considered when determining and declaring the guaranteed data accuracy.
NoCRS	The wfs:NoCRS element shall be used for feature types that have no spatial properties, and therefore no CRS whatsoever. It is not a requirement for Features and FeatureCollections to have spatial properties. The wfs:NoCRS element shall never imply, and therefore cannot be used for, semantics of "Unknown CRS". This element is used as an identifying label only, and therefore has no element or attribute content.
OutputFormats	The wfs:OutputFormats element shall be a list of MIME types indicating the output formats that may be generated for a feature type. If this optional element is not specified, then all the result formats listed for the GetFeature operation are assumed to be supported.
WGS84BoundingBox	The ows:WGS84BoundingBox element may be used to indicate the edges of an enclosing rectangle in decimal degrees of latitude and longitude in WGS84. Its purpose is to facilitate geographic searches by indicating where instances of the particular feature type exist. Since multiple ows:WGS84BoundingBox elements may be specified, a WFS may indicate where various clusters of data exist. This knowledge aids client applications by letting them know where they should query in order to have a high probability of finding feature data.
MetadataURL	A WFS may use zero or more wfs:MetadataURL elements to offer detailed metadata about the data in a particular feature type. The xlink:href element shall be used to reference any metadata. The optional about attribute may be used to reference the aspect of the element which includes this wfs:MetadataURL element that this metadata provides more information about.
ExtendedDescription	A WFS may add elements to the description of a feature type, without having to redefine the capabilities schema, using the wfs:ExtendedDescription element. The wfs:ExtendedDescription element contains one or more wfs:Element elements. The wfs:Element element includes a name attribute, a type attribute and contains a value list enumerating one or more values for the named extended descriptive element. The name attribute is used to designate the name of the extended descriptive element. The type attribute is used to designate a type for the values in the value list of the extended descriptive element. The type shall be taken from the list of built-in types defined by XML Schema (see W3C XML Schema Part 2). The wfs:Element element also includes an ows:Metadata element that shall be used to reference metadata describing the wfs:Element. The wfs:ExtendedDescription element is intended to be used by communities of interest to customize the description of a feature type for specific purposes, or by vendors wishing to add vendor-specific descriptive information to the description of a feature type in a capabilities document. In all cases, clients shall be able to safely ignore all of the extended descriptive elements. Each extended description wfs:Element added to the description of a feature type shall be accompanied by an ows:Metadata element offering descriptive metadata about the added element.

8.3.5 Parameter domains and constraints

8.3.5.1 Introduction

The ows:Parameter and ows:Constraint elements are defined in the OWS Common Implementation Specification (see OGC 06-121r3:2009, Table 13) and allow valid domain values and constraints to be defined globally for all operations or locally for specific operations that a web feature service offers.

8.3.5.2 Parameter domains

Table 12 defines the parameter domains that may be defined in the capabilities document of a web feature service.

Table 12 — Parameter domains for WFS operations

Operation name	Parameter name	Expected value type	Description/Possible values
All operations (except GetCapabilities)	version	string	Shall include the value "2.0.0". May include the value "1.1.0", "1.0.0" or other vendor-specific version number.
GetFeature GetFeatureWithLock Transaction	srsName	string URI or MIME type	List of CRSs that the WFS is capable of handling.
GetCapabilities	AcceptVersions	string	Shall include the value "2.0.0". May include the values "1.1.0", "1.0.0" or other vendor-specific version.
GetCapabilities	AcceptFormats	MIME type	Shall include the value "text/xml". May include other MIME types such as "text/html" or "text/plain" or any other vendor-supported MIME type the server is capable of generating.
GetCapabilities	Sections	string	Zero or more of the following list of values: "ServiceIdentification", "ServiceProvider", "OperationsMetadata", "FeatureTypeList", "Filter_Capabilities".
DescribeFeatureType	outputFormat	string or MIME type	Shall include the value "application/gml+xml; version=3.2". May include any other string or MIME type that the server supports, including previous version of GML.
GetPropertyValue	outputFormat	string	Shall include the value "application/gml+xml; version=3.2". May include any other string or MIME type that the server supports, including previous versions of GML.
GetPropertyValue	resolve	string	Shall include the values “none” and “local”. May also include the values “remote” and “all”, indicating that the server can resolve remote resource references.
GetFeature	outputFormat	string or MIME type	Shall include the value "application/gml+xml; version=3.2". May include any other string or MIME type that the server supports, including previous versions of GML.
GetFeature	resolve	string	Shall include the values “none” and “local”. May also include the values “remote” and “all”, indicating that the server can resolve remote resource references.

Table 12 (*continued*)

Operation name	Parameter name	Expected value type	Description/Possible values
GetFeatureWithLock	outputFormat	string or MIME type	Shall include the values "application/gml+xml; version=3.2". May include any other string or MIME type that the server supports, including previous version of GML.
GetFeatureWithLock	resolve	string	Shall include the values “none” and “local”. May also include the values “remote” and “all”, indicating that the server can resolve remote resource references.
CreateStoredQuery	language	anyURI	Shall include the value "urn-x:wfs:StoredQueryLanguage:WFS_QueryExpression". May also include other values indicating that other languages are supported. This International Standard does not assign any meaning to or describe any additional values that might be listed.
Transaction	inputFormat	string or MIME type	Shall include the value "application/gml+xml; version=3.2". May include any other string or MIME type that the server supports, including previous version of GML.
Transaction	vendorId	string	Any string that is used as a vendor identifier for the wfs:Native element.

Servers shall, either locally for each operation or globally for the entire service, populate the parameter domains in Table 12 for all the operations they claim to support in their capabilities document.

In general, the domain of a parameter is specific to a web feature service implementation.

EXAMPLE 1 The allowed values for the srsName parameter are dictated by the particular transformations that a WFS supports.

In some cases, however, the domain of a parameter is defined by this International Standard.

EXAMPLE 2 The domain of the resolve parameter (see 7.6.4) is defined in this International Standard as consisting of the values "local", "remote", "all" or "none".

In such cases, a web feature service may only restrict the domain. If the full domain is supported, then the parameters domain shall not be listed in the capabilities document.

8.3.5.3 Service and operation constraints

All the service constraints listed in Table 13 shall be specified in the capabilities document (see 8.3.3) of a server with the appropriate value set to indicate whether the server conforms to the corresponding conformance class or not.

Table 13 — Service constraints

Constraint name	Possible values and/or value types	Description
ImplementsBasicWFS	Boolean value; either "TRUE" or "FALSE"	Indicates that the server implements the Basic WFS conformance class.
ImplementsTransactionalWFS	Boolean value; either "TRUE" or "FALSE"	Indicates that the server implements the Transactional WFS conformance class.
ImplementsLockingWFS	Boolean value; either "TRUE" or "FALSE"	Indicates that the server implements the Locking conformance class.
KVPEncoding	Boolean value; either "TRUE" or "FALSE"	Indicates that the server implements the HTTP GET conformance class.
XMLEncoding	Boolean value; either "TRUE" or "FALSE"	Indicates that the server implements the HTTP POST conformance class.
SOAPEncoding	Boolean value; either "TRUE" or "FALSE"	Indicates that the server implements the SOAP conformance class.
ImplementsInheritance	Boolean value; either "TRUE" or "FALSE"	Indicates that the server implements the Inheritance conformance class.
ImplementsRemoteResolve	Boolean value; either "TRUE" or "FALSE"	Indicates that the operation to which the constraint is specified can resolve references that are remote to the server. If the constraint is not specified then the value of FALSE shall be assumed.
ImplementsResultPaging	Boolean value; either "TRUE" or "FALSE"	Indicates that the server supports the ability to page through a result set *count* features one at a time.
ImplementsStandardJoins	Boolean value; either "TRUE" or "FALSE"	Indicates that the server implements the Standard Joins conformance class.
ImplementsSpatialJoins	Boolean value; either "TRUE" or "FALSE"	Indicates that the server implements the Spatial Joins conformance class.
ImplementsTemporalJoins	Boolean value; either "TRUE" or "FALSE"	Indicates that the server implements the Temporal Joins conformance class.
ImplementsFeatureVersioning	Boolean value; either "TRUE" or "FALSE"	Indicates that the server implements the Feature Versions conformance class.
ManageStoredQueries	Boolean value; either "TRUE" or "FALSE"	Indicates that the server implements the Manage Stored Queries conformance class.

A server may optionally specify one or more of the constraints defined in Table 14 in its capabilities document (see 8.3.3).

Table 14 — Operation Constraints

Constraint Name	Possible values and/or value types	Default value	WFS operation	Description
AutomaticDataLocking	Boolean value; either "TRUE" or "FALSE"	FALSE	Transaction	Indicates that the transaction operation automatically locks data in order to maintain consistency, thus alleviating the client from having to use the LockFeature or GetFeatureWithLock operations to lock the features to be modified.
PreservesSiblingOrder	Boolean value; either "TRUE" or "FALSE"	FALSE	Transaction	Specifies whether the server preserves sibling order for properties with cardinality greater than 1. If the value is true, the server shall preserve sibling order. Otherwise, sibling order is not guaranteed to be preserved.
PagingIsTransactionSafe	Boolean value; either "TRUE" or "FALSE"	FALSE	GetFeature GetFeatureWithLock GetPropertyValue	Specifies whether the server maintains transactional consistency between paging iterations.
CountDefault	Integer value greater than or equal to zero		GetFeature GetFeatureWithLock GetPropertyValue	Specifies the default value for the count parameter. If the constraint is not specified, and Response Paging is neither supported nor triggered by the request, the entire result shall be returned in one response. Servers are strongly encouraged to advertise a value for CountDefault as a means of self-defence, so that a request may not clog a server.
ResolveTimeoutDefault	Integer value greater than zero		GetFeature GetFeatureWithLock GetPropertyValue	Defines the maximum number of seconds a server shall wait before receiving a response while resolving resource references. If the constraint is not specified, the server shall wait indefinitely in order to resolve a remote reference.
SortLevelLimit	Integer value greater than zero		GetFeature GetFeatureWithLock	The SortLevelLimit constraint defines the maximum number of properties that may be simultaneously sorted. In the event that a request contains too many fes:SortProperty elements for a particular service (i.e. exceeds the SortLevelLimit constraint), the service shall respond with an exception as specified in 7.5. If the constraint is not specified then there is no limit to the number of sort properties that may be specified.
ResolveLocalScope	Integer value greater than zero OR the character "*"	*	GetFeature GetFeatureWithLock GetPropertyValue	Defines the minimum and maximum number of levels, when resolving references to resources that are part of the server's local data store. The value "*" means to as many as all levels. If the constraint is not specified, the default value of "*" shall be assumed.

Table 14 (*continued*)

Constraint Name	Possible values and/or value types	Default value	WFS operation	Description
ResolveRemoteScope	Integer value greater than zero OR the character "*"	*	GetFeature GetFeatureWithLock GetPropertyValue	Defines the minimum and maximum number of levels, when resolving remote references. The value "*" means to as many as all levels. If the constraint is not specified, the default value of "*" shall be assumed.
ResponseCacheTimeout	Integer value greater than zero		GetFeature GetFeatureWithLock GetPropertyValue	Defines the length of time, in seconds, that responses shall be cached to support paging (see 7.7.4.4). If the constraint is not specified then the response cache never times out.
QueryExpressions	QName; one of wfs:Query of wfs:StoredQuery		GetFeature GetFeatureWithLock GetPropertyValue LockFeature	The names of the supported query expression elements.
NOTE The constraints may be specified on the indicated operation. If more that one operation is listed, the constraint may be specified on each operation individually (perhaps with different values for each operation) or at the service level, indicating that the constraint applies for all listed operations.				

8.4 Extension points

Extension points allow servers to advertise additional values for existing parameters, dynamically add elements to the metadata describing feature types, or execute operations not described in this International Standard. This International Standard contains the following extension points:

— DescribeFeatureType/@outputFormat (see 9.2.4.2)

— GetFeature/@outputFormat (see 11.2.4.1)

— fes:AbstractQueryExpression (see 10.2.4.4, 11.2.4.3 and 12.2.4.1)

— GetFeatureWithLock/@outputFormat (see 13.1)

— Transaction/Insert/@inputFormat (see 15.2.4.2)

— Transaction/Update/@inputFormat (see 15.2.5.2.4)

— Transaction/Replace/@inputFormat (see 15.2.6.2.2)

— wfs:ExtendedDescription element (see Table 11)

— ows:ExtendedCapabilities (see OGC 06-121r3:2009, 7.4.6)

— wfs:Native (see 15.2.8)

— wfs:AbstractTransactionAction (see 15.2.2)

This International Standard assigns no meaning or behaviour to any value advertised for these extension points in the server's capabilities document, except as described herein. However, if additional values or behaviours are advertised in a server's capabilities document, they shall be accompanied by additional metadata describing their meaning.

Metadata may be associated with each wfs:Element element contained within the wfs:ExtendedDescription element using the ows:Metadata element (see OGC 06-121r3:2009, Table 32).

The OGC Web Services Common Specification (see OGC 06-121r3:2009, 7.4.6) contains a complete description of how to associate metadata with parameter values or operations contained within a capabilities document. The following examples illustrate the usage.

EXAMPLE 1 The following XML fragment illustrates how to reference metadata about additional GetFeature output format values.

```
<Operation name="GetFeature">
   <DCP>
      <HTTP>
         <Get xlink:href="http://www.someserver.com/wfs.cgi?"/>
      </HTTP>
   </DCP>
   <Parameter name="outputFormat">
      <AllowedValues>
         <Value>application/gml+xml; version=3.2</Value>
         <Value>application/gml+xml; version=3.1</Value>
         <Value>application/gml+xml; version=2.1</Value>
      </AllowedValues>
   </Parameter>
   <Metadata about="application/gml+xml; version=3.1"
             xlink:href="http://portal.opengeospatial.org/files/?artifact_id=4700"/>
   <Metadata about="application/gml+xml; version=2.1"
             xlink:href="http://portal.opengeospatial.org/files/?artifact_id=11339"/>
</Operation>
```

EXAMPLE 2 The following XML fragment illustrates how to reference metadata about vendor-specific operations executed using the wfs:Native element.

```
<Operation name="Transaction">
   <DCP>
      <HTTP>
         <Get xlink:href="http://www.someotherserver.com/transform?"/>
      </HTTP>
   </DCP>
   <Parameter name="Native">
      <AllowedValues>
         <Value>ALTER SESSION ENABLE PARALLEL DML</Value>
         <Value>MySecretAction</Value>
      </AllowedValues>
   </Parameter>
   <Metadata about="ALTER SESSION ENABLE PARALLEL DML"
   xlink:href="http://download-
     uk.oracle.com/docs/cd/A97630_01/server.920/a96540/statements_22a.htm#2053493"/>
   <Metadata about="MySecretAction" xlink:href=http://www.yas.com/MySecretActionDescription.html/>
</Operation>
```

EXAMPLE 3 The following XML fragment illustrates how a vendor-specific operation can be advertised in a server's capabilities document using the ows:ExtendedCapabilities element.

```
<ExtendedCapabilities>
   <Operation name="MySecretOperation">
      <DCP>
         <HTTP>
            <Get xlink:href="http://www.someserver.com/wfs.cgi?"/>
         </HTTP>
      </DCP>
      <Metadata about="MySecretOperation"
                xlink:href="http://www.myserver.com/Operations/MySecretOperation.html"/>
   </Operation>
</ExtendedCapabilities>
```

8.5 Exceptions

In the event that a web feature service encounters an error parsing a GetCapabilities request, it shall raise an OperationParsingFailed exception as described in 7.5.

In the event that a web feature service encounters an error processing a GetCapabilities request, it shall raise an OperationProcessingFailed exception as described in 7.5.

9 DescribeFeatureType operation

9.1 Introduction

The DescribeFeatureType operation returns a schema description of feature types offered by a WFS instance. The schema descriptions define how a WFS expects feature instances to be encoded on input (via Insert, Update and Replace actions) and how feature instances shall be encoded on output (in response to a GetPropertyValue, GetFeature or GetFeatureWithLock operation).

9.2 Request

9.2.1 Request semantics

Figure 14 describes the schema of a DescribeFeatureType request.

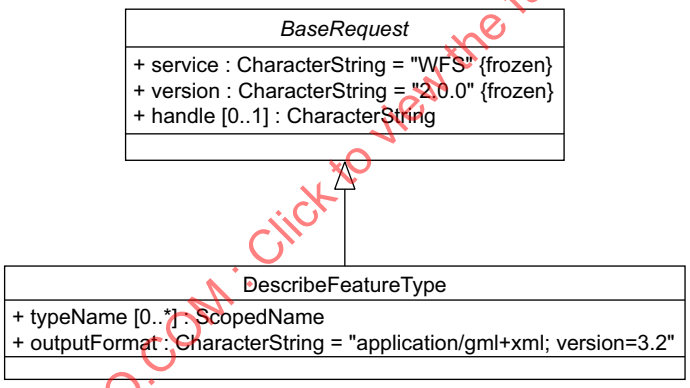

Figure 14 — DescribeFeatureType request

9.2.2 XML Encoding

The following XML Schema fragment defines the XML encoding of a DescribeFeatureType request:

```
<xsd:element name="DescribeFeatureType" type="wfs:DescribeFeatureTypeType"/>
<xsd:complexType name="DescribeFeatureTypeType">
   <xsd:complexContent>
      <xsd:extension base="wfs:BaseRequestType">
         <xsd:sequence>
            <xsd:element name="TypeName" type="xsd:QName"
                         minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="outputFormat" type="xsd:string"
                        default="application/gml+xml; version=3.2"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

9.2.3 KVP Encoding

Table 15 defines the KVP encoding for the DescribeFeatureType request.

Table 15 — DescribeFeatureType KVP encoding

URL component	O/M[a]	Description
Common Keywords (REQUEST=DescribeFeatureType)		See Table 7. (Only keywords for all operations or the DescribeFeatureType operation.)
TYPENAME	O	A comma-separated list of feature types to describe. If no value is specified, the complete application schema offered by the server shall be described.
OUTPUTFORMAT	O	Shall support the value "application/gml+xml; version=3.2", indicating that a GML (see ISO 19136:2007) application schema shall be generated. A server may support other values to which this International Standard does not assign any meaning.

[a] O = Optional, M = Mandatory.

9.2.4 Parameter discussion

9.2.4.1 typeNames parameter

For XML-encoded requests, the typeNames parameter shall be encoded using the typeNames attribute on the wfs:DescribeFeatureType element.

For KVP-encoded requests, the typeNames parameter shall be encoded using the TYPENAMES keyword.

The typeNames parameter encodes the names of one or more feature types that shall be described by the DescribeFeatureType operation.

The set of permissible values for the typeNames parameter shall be the set of feature type names listed in a server's capabilities document (see 8.3.3).

If the typeName parameter is omitted, the complete application schemas supported by the server shall be returned in response.

9.2.4.2 outputFormat parameter

For XML-encoded requests, the outputFormat parameter shall be encoded using an attribute of the same name on the wfs:DescribeFeatureType element.

For KVP-encoded requests, the outputFormat parameter shall be encoded using the keyword OUTPUTFORMAT (see 7.6.3.3).

The outputFormat parameter is used to indicate the schema description language that shall be used to describe feature types.

The default value of "application/gml+xml; version=3.2" shall be used to indicate that the feature types shall be described using a GML (see ISO 19136:2007) application schema. Every WFS that conforms to this International Standard shall support this default value.

A server may advertise additional values for the outputFormat attribute in its capabilities document (see Table 12), indicating that multiple output formats, including previous versions of GML, are supported. However, this International Standard only assigns meaning to the value "application/gml+xml; version=3.2".

If any such additional outputFormat values are listed in the capabilities document, this International Standard recommends that additional metadata (see OGC 06-121r3:2009, Table 32) describing the meaning of the additional values be included in the capabilities document (see 8.3.3).

NOTE Since previous OGC versions of this International Standard (see OGC 02-058 and OGC 04-094) are bound to specific versions of GML, a server can support previous versions of GML by advertising in its capabilities document (see Table 12) that it supports previous OGC versions of this International Standard.

9.3 Response

9.3.1 Introduction

In response to a DescribeFeatureType request, where the value of the outputFormat parameter has been set to "application/gml+xml; version=3.2", a WFS implementation shall return a complete and valid GML (see ISO 19136:2007) application schema that contains the definition of the feature types listed in the request. The document(s) returned by the DescribeFeatureType request may be used to validate feature instances generated by the WFS in the form of feature collections on output or feature instances specified as input for transaction operations.

9.3.2 Supporting multiple namespaces

An XML Schema document can only declare elements in a single namespace (see W3C XML Schema Part 2). In the event that a DescribeFeatureType operation requests that feature types in multiple namespaces be described, the server shall generate the complete schema for one of the requested namespaces and import the remaining namespaces. This International Standard does not mandate which namespace's schema should be generated and which namespaces should be imported.

EXAMPLE Consider the following request to describe feature types in multiple namespaces:

```
<?xml version="1.0" ?>
<DescribeFeatureType
   version="2.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:ns01="http://www.server01.com/ns01"
   xmlns:ns02="http://www.server02.com/ns02"
   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">
   <TypeName>ns01:TreesA_1M</TypeName>
   <TypeName>ns02:RoadL_1M</TypeName>
</DescribeFeatureType>
```

A WFS might generate the following response to this request:

```
<?xml version="1.0" ?>
<xsd:schema
   targetNamespace="http://www.server01.com/ns01"
   xmlns:ns01="http://www.server01.com/ns01"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified">

   <import namespace="http://www.server02.com/ns02"
           schemaLocation="http://www.yourserver.com/wfs.cgi?
           REQUEST=DescribeFeatureType&amp;TYPENAMES=ns02:RoadL_1M"/>

   <xsd:element name="TREESA_1M" type="ns01:TREESA_1MType" substitutionGroup="gml:_Feature"/>
   <xsd:complexType name="TREESA_1MType">
      <xsd:complexContent>
         <xsd:extension base="gml:AbstractFeatureType">
            <!-- list property declarations for feature type TREESA_1M -->
```

```
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <!-- other declarations that are part of this schema -->
</xsd:schema>
```

9.4 Exceptions

In the event that a web feature service encounters an error parsing a DescribeFeatureType request, it shall raise an OperationParsingFailed exception as described in 7.5.

In the event that a web feature service encounters an error processing a DescribeFeatureType request, it shall raise an OperationProcessingFailed exception as described in 7.5.

10 GetPropertyValue operation

10.1 Introduction

The GetPropertyValue operation allows the value of a feature property or part of the value of a complex feature property to be retrieved from the data store for a set of features identified using a query expression (see 7.9).

10.2 Request

10.2.1 Request semantics

Figure 15 describes the schema of a GetPropertyValue request.

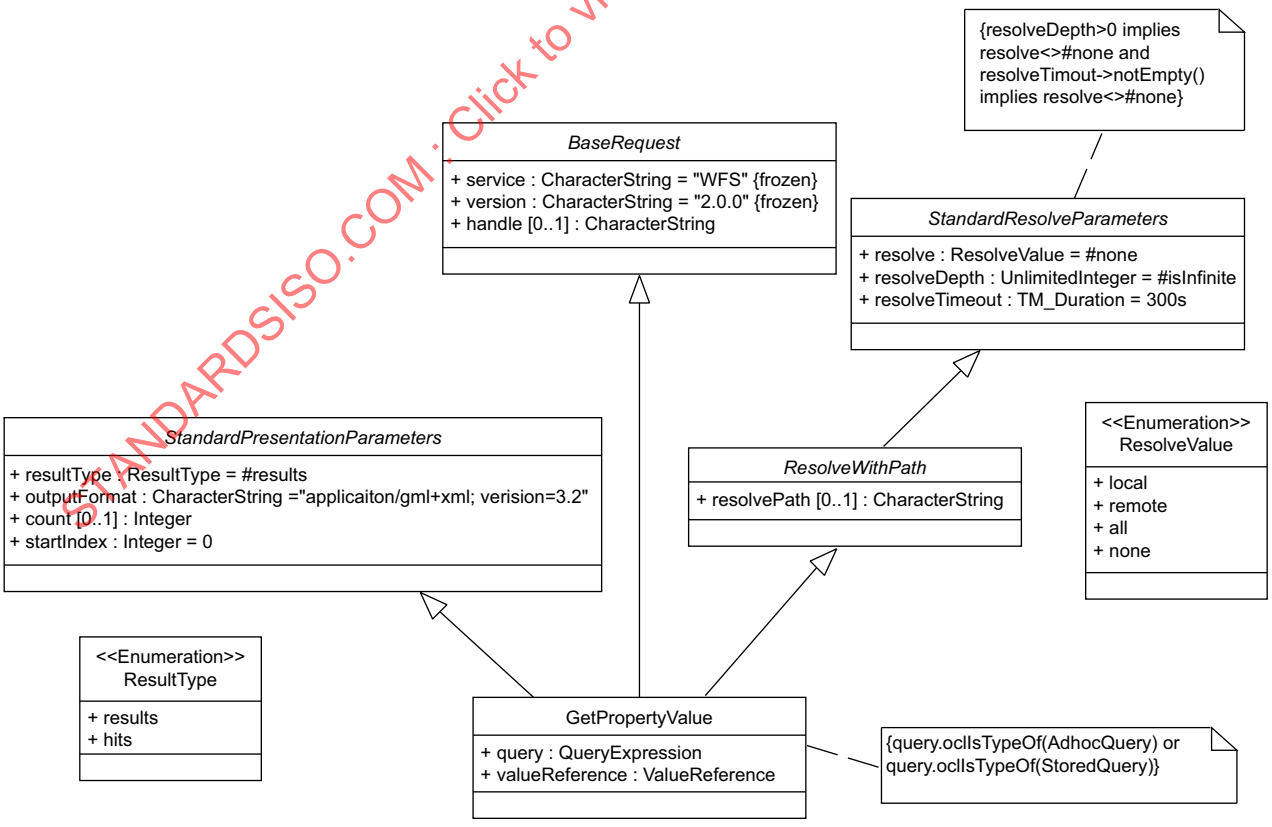

Figure 15 — GetPropertyValue request

10.2.2 XML Encoding

The following XML fragment defines the XML encoding for the GetPropertyValue operation:

```
<xsd:element name="GetPropertyValue" type="wfs:GetPropertyValueType"/>
<xsd:complexType name="GetPropertyValueType">
   <xsd:complexContent>
      <xsd:extension base="wfs:BaseRequestType">
         <xsd:sequence>
            <xsd:element ref="fes:AbstractQueryExpression"/>
         </xsd:sequence>
         <xsd:attribute name="valueReference" type="xsd:string" use="required"/>
         <xsd:attribute name="resolvePath" type="xsd:string"/>
         <xsd:attributeGroup ref="wfs:StandardPresentationParameters"/>
         <xsd:attributeGroup ref="wfs:StandardResolveParameters"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

10.2.3 KVP Encoding

Table 16 defines the KVP encoding for the GetPropertyValue operation.

Table 16 — Keywords for GetPropertyValue KVP encoding

URL component	O/M[a]	Description
Common Keywords (REQUEST=GetPropertyValue)		See Table 7. (Only keywords listed for all operations or the GetPropertyValue operation.)
Adhoc Query Keywords (Mutually exclusive with Stored Query Keywords)		See Table 8.
Stored Query Keywords (Mutually exclusive with Adhoc Query Keywords)		See Table 10.
VALUEREFERENCE	M	See 10.2.4.3.
RESOLVEPATH	O	See 7.9.2.4.7.
[a] O = Optional, M = Mandatory.		

10.2.4 Parameter discussion

10.2.4.1 Standard presentation parameters

See 7.6.3.

10.2.4.2 Standard resolve parameters

See 7.6.4.

10.2.4.3 valueReference parameter

The valueReference parameter is an XPath expression (see ISO 19143:2010, 7.4.4) that identifies a node, or child node of a property node of a feature whose value shall be retrieved from a server's data store and reported in the response document.

The response shall be a text node or a list of element nodes that is the value of the node pointed to by the valueReference parameter.

In the case where the value is a reference to a remote resource, the valueOf() accessor function may be used to resolve the remote value.

NOTE There is a difference between how the valueReference parameter with the valueOf() XPath operator (see 7.3.2) behaves and what the standard resolve parameters control. The valueReference is used to express queries to determine the features/values in the result set, while the standard resolve parameters (see 7.6.4) are used to determine how property values inside the selected features/values are encoded and presented in response documents.

EXAMPLE Consider the following XML fragment encoding and instances of the feature type myns:Person:

```
<myns:Person  gml:id="p4467"
   <gml:identifier
        codespace=http://www.canadaSIN.com>424679360</gml:identifier>
   <myns:lastName>Smith</myns:lastName>
   <myns:firstName>Mary</myns:firstName>
   <myns:age>18</myns:age>
   <myns:sex>Female</myns:sex>
   <myns:spouse xlink:href="#p4456"/>
   <myns:location xlink:href = "#p101" />
   <myns:mailAddress>
      <myns:Address gml:id="a201">
         <myns:streetName>Main St.</myns:streetName>
         <myns:streetNumber>4</myns:streetNumber>
         <myns:city>SomeCity</myns:city>
         <myns:province>Someprovince</myns:province>
         <myns:postalCode>X1X 1X1</myns:postalCode>
         <myns:country>Canada</myns:country>
      </myns:Address>
   </myns:mailAddress>
   <myns:phone>416-123-4567</myns:phone>
   <myns:phone>416-890-1234</myns:phone>
   <myns:livesIn xlink:href="#h32"/>
   <myns:isDriving xlink:href="r1432"/>
</myns:Person>
```

A value reference of *valueReference="lastName"* would present the last name of the person, Smith in this example, in the response document.

A value reference of *valueReference="myns:mailAddress/myns:Address/myns:city"* would present the name of the city in which Mary Smith lives (SomeCity in this example).

A value reference of valueReference="valueOf(myns:location)" would present the location of Mary Smith in the response document by resolving the reference to the value with id "#p101".

10.2.4.4 AbstractQueryExpression parameter

The GetPropertyValue request operates on a set of features identified using a query expression.

This International Standard defines the elements wfs:Query (see 7.9.2.2) and wfs:StoredQuery (see 7.9.3.2) that may be used as query expressions in a GetPropertyValue operation and which are substitutable for a fes:AbstractQueryExpression (see ISO 19143:2010, 6.2).

Other elements may be defined as being substitutable for fes:AbstractQueryExpression. However, this International Standard only assigns meaning to the wfs:Query and wfs:StoredQuery elements.

The KVP encoding of an ad hoc query expression is defined in 7.9.2.3.

The KVP encoding of a stored query expression is defined in 7.9.3.3.

KVP-encoded GetPropertyValue requests shall only contain query expressions of one type: either an ad hoc query expression or a stored query expression.

10.3 Response

10.3.1 Response semantics

Figure 16 describes the schema of a GetPropertyValue response.

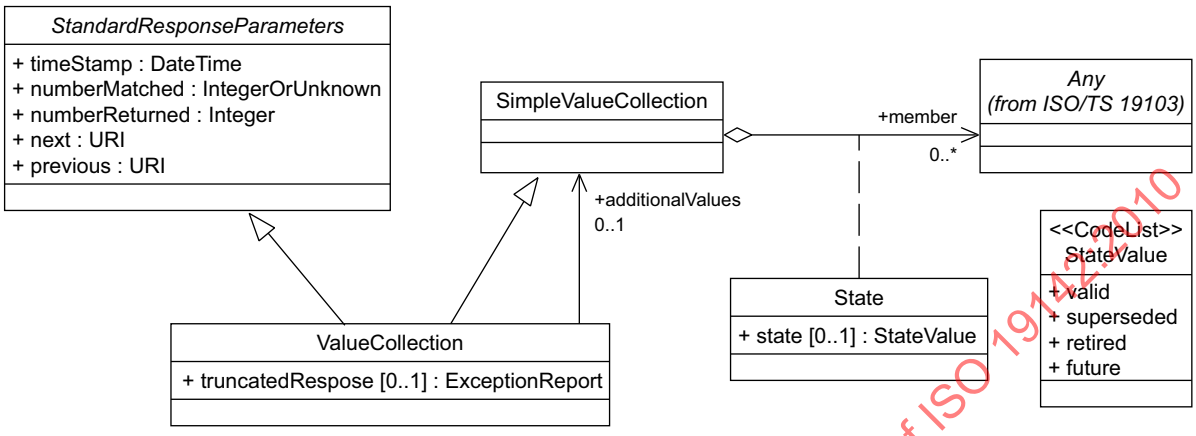

Figure 16 — GetPropertyValue response

10.3.2 XML encoding

The following XML Schema fragment defines the response to a GetPropertyValue operation:

```
<xsd:element name="ValueCollection" type="wfs:ValueCollectionType"/>
<xsd:complexType name="ValueCollectionType">
   <xsd:sequence>
      <xsd:element ref="wfs:member" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wfs:additionalValues" minOccurs="0"/>
      <xsd:element ref="wfs:truncatedResponse" minOccurs="0"/>
   </xsd:sequence>
   <xsd:attributeGroup ref="wfs:StandardResponseParameters"/>
</xsd:complexType>
<xsd:element name="member" type="wfs:MemberPropertyType"/>
<xsd:complexType name="MemberPropertyType" mixed="true">
   <xsd:choice minOccurs="0">
      <xsd:any processContents="lax" namespace="##other"/>
      <xsd:element ref="wfs:Tuple"/>
      <xsd:element ref="wfs:SimpleFeatureCollection"/>
   </xsd:choice>
   <xsd:attribute name="state" type="wfs:StateValueType"/>
   <xsd:attributeGroup ref="xlink:simpleLink"/>
</xsd:complexType>
<xsd:element name="Tuple" type="wfs:TupleType"/>
<xsd:complexType name="TupleType">
   <xsd:sequence>
      <xsd:element ref="wfs:member" minOccurs="2" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name="additionalValues">
   <xsd:complexType>
      <xsd:choice>
         <xsd:element ref="wfs:ValueCollection"/>
         <xsd:element ref="wfs:SimpleFeatureCollection"/>
      </xsd:choice>
   </xsd:complexType>
</xsd:element>
<xsd:element name="truncatedResponse">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="ows:ExceptionReport"/>
      </xsd:sequence>
```

```
      </xsd:complexType>
   </xsd:element>
   <xsd:simpleType name="StateValueType">
      <xsd:union>
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="valid"/>
               <xsd:enumeration value="superseded"/>
               <xsd:enumeration value="retired"/>
               <xsd:enumeration value="future"/>
            </xsd:restriction>
         </xsd:simpleType>
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:pattern value="other:\w{2,}"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:union>
   </xsd:simpleType>
```

The wfs:ValueCollection element contains zero or more wfs:member elements that contain values from the set of values identified using the query expression in the GetPropertyValue operation. Values may be encoded inline, within the wfs:member element or referenced via the xlink:href attribute. The xlink:href attribute is declared as part of the xlink:simpleLink attribute group on the wfs:member element.

If additional values need to be included in the response document in order to satisfy the resolution of referenced resources (see 7.6.4), these objects shall be placed within the wfs:additionalValues element, and resource references in the returned values shall be relocated to point to these local copies of the referenced resources. As specified in 7.6.4.5, the server shall also arrange that all unresolved nested resource references are appropriately relocated to point to the intended resources.

If a server encounters an exception after beginning the transmission of a response document to the client, the server shall terminate the response document by including the wfs:truncatedResponse element that shall contain an ows:ExceptionReport element reporting the exception (see 7.5). If the server successfully transmits the entire response document, the wfs:truncatedResponse element shall not be included in the response.

10.3.3 State parameter

Servers that do not support the Feature versions conformance class (see Table 1) shall not use the state parameter.

Servers that do support the Feature versions conformance class (see Table 1) shall use the state parameter to report the state of a versioned value in a response document. The value domain of the state parameter is the string "retired", “superseded”, “valid”, "future" and any other string prefixed with the token "other:". The token "retired" shall be used to indicate that the value has been deleted from the underlying data store; “superseded” shall indicate that the particular version has been superseded by a newer version in the underlying data store; “valid” shall indicate that the version is the current version in the underlying data store; "future" shall indicate a value which is not yet valid. This may be for legal reasons, it may be used to distribute a proposed value, etc.

Any other string prefixed with the token "other:" may be used to report a server-specific versioned state. This International Standard does not assign any meaning to values prefixed with the token "other:".

10.3.4 Standard response parameters

For a discussion of the standard response parameters, see 7.7.

10.4 Exceptions

In the event that a web feature service encounters an error parsing a GetPropertyValue request, it shall raise an OperationParsingFailed exception as described in 7.5.

In the event that a web feature service encounters an error processing a GetPropertyValue request, it shall raise an OperationProcessingFailed exception as described in 7.5.

11 GetFeature operation

11.1 Introduction

The GetFeature operation returns a selection of features from a data store. A WFS processes a GetFeature request and returns a response document to the client that contains zero or more feature instances that satisfy the query expressions specified in the request.

The canonical representation of features uses GML (see ISO 19136:2007), and the form of the GetFeature request is modelled after this representation of the result set. For this reason, it is necessary to have a clear understanding of how the General Feature Model (see ISO 19109) is mapped into the GML representation. The reference description of GML is given by ISO 19136:2007 but the salient aspects are summarized here.

In GML, a feature is represented as an XML element. The name of the feature element indicates the Feature Type, conventionally given in UpperCamelCase (see OGC 06-121r3:2009, 11.6.2), such as *xmml:BoreHole* or *myns:SecondaryCollege*.

The content of a feature element is a set of elements that describes the feature in terms of a set of properties. Each child element of the feature element is a property. The name of the property indicates what the property means, conventionally given in lowerCamelCase (see OGC 06-121r3:2009, 11.6.2), such as *gml:boundedBy* or *xmml:collarLocation*.

The value of a property is given inline by the content of the property element, or by reference as the value of a resource identified in a link carried as an XML attribute of the property element. If the inline form is used, then the content may be a literal (a number, text, etc.), or may be structured using XML elements, but no assumptions can be made about the structure of the value of a property. In some cases, the value of a property of feature may be another feature, for example a *myns:School* feature may have a property *myns:frontsOn*, whose value is a *myns:Road*, which will itself have further properties, etc.

11.2 Request

11.2.1 Request semantics

Figure 17 describes the schema of a GetFeature request.

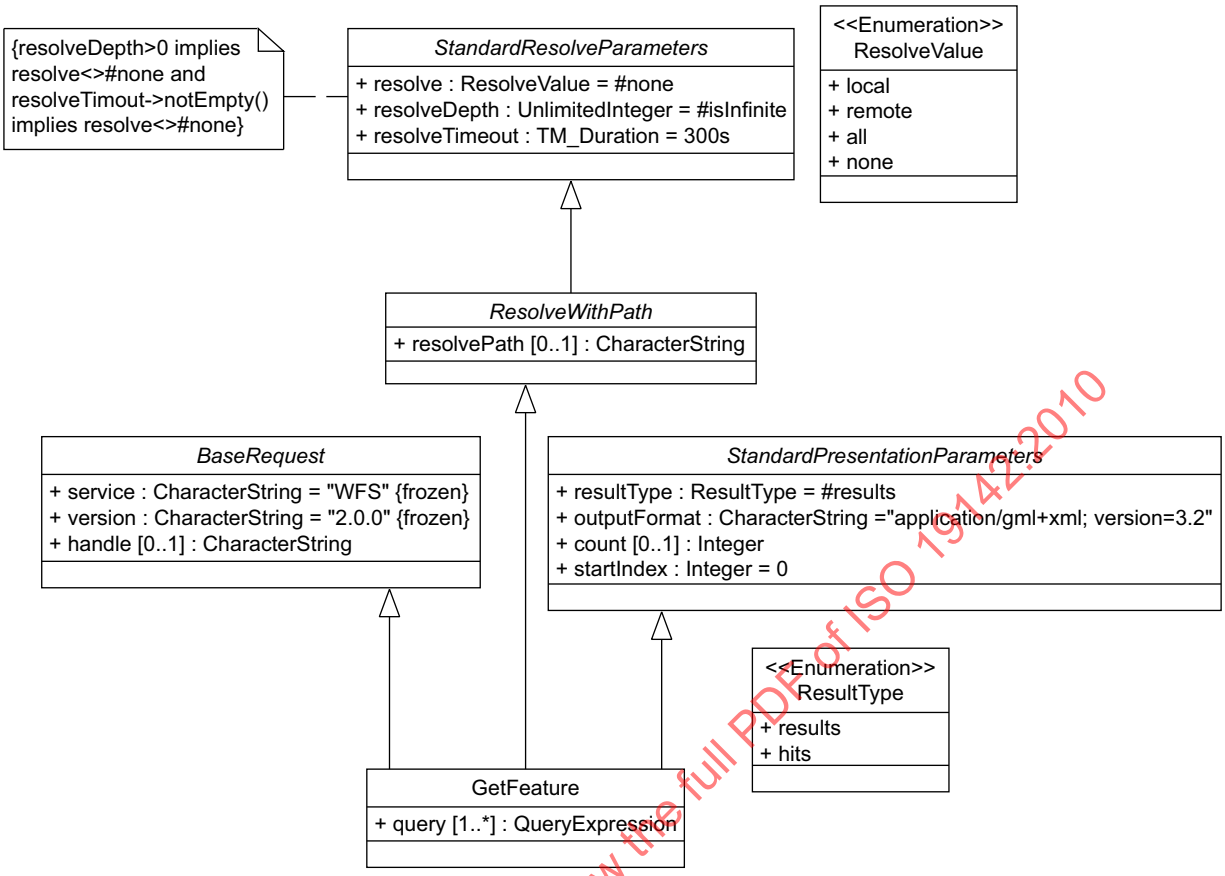

Figure 17 — GetFeature request

11.2.2 XML encoding

The XML encoding of a GetFeature request is defined by the following XML Schema fragment:

```
<xsd:element name="GetFeature" type="wfs:GetFeatureType"/>
<xsd:complexType name="GetFeatureType">
   <xsd:complexContent>
      <xsd:extension base="wfs:BaseRequestType">
         <xsd:sequence>
            <xsd:element ref="fes:AbstractQueryExpression"
                         maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attributeGroup ref="wfs:StandardPresentationParameters"/>
         <xsd:attributeGroup ref="wfs:StandardResolveParameters"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

A GetFeature request contains one or more query expressions. A query expression identifies a set of feature instances that shall be presented to a client in the response document. Query expressions in a GetFeature request shall be independent of each other and may be executed in any order. The response collections, however, shall be presented in the order in which the query expressions are encountered in a GetFeature request.

11.2.3 KVP encoding

Table 17 defines the KVP encoding for a GetFeature request.

Table 17 — Keywords for GetFeature KVP encoding

URL component	Description
Common Keywords (REQUEST=GetFeature)	See Table 7 for additional parameters that may be used in a KVP-encoded GetFeature request.
Standard Presentation Parameters	See Table 5.
Standard Resolve Parameters	See Table 6.
Adhoc Query Keywords (Mutually exclusive with Stored Query Keywords)	See Table 8.
Stored Query Keywords (Mutually exclusive with Adhoc Query Keywords)	See Table 10.

11.2.4 Parameter discussions

11.2.4.1 Standard presentation parameters

See 7.6.3.

11.2.4.2 Standard resolve parameters

See 7.6.4.

11.2.4.3 AbstractQueryExpression parameter

The GetFeature request operates on a set of features identified using one or more query expressions.

For XML-encoded requests, this International Standard defines the elements wfs:Query (see 7.9.2.2) and wfs:StoredQuery (see 7.9.3.2) that may be used as query expressions in a GetFeature request and which are substitutable for fes:AbstractQueryExpression (see ISO 19143:2010, 6.2). Multiple wfs:Query and/or wfs:StoredQuery elements may be encoded into a single GetFeature request.

Other elements may be defined as being substitutable for fes:AbstractQueryExpression. However, this International Standard only assigns meaning to the wfs:Query and wfs:StoredQuery elements.

The KVP encoding of an ad hoc query expression is defined in 7.9.2.3. Multiple ad hoc query expressions may be encoded into a KVP-encoded request.

The KVP encoding of a stored query expression is defined in 7.9.3.3. Only a single stored query expression may be encoded into a KVP-encoded request.

KVP-encoded requests shall only contain query expressions of one type: either ad hoc query expression(s) or a stored query expression.

11.3 Response

11.3.1 Response semantics

Figure 18 describes the schema of a GetFeature response.

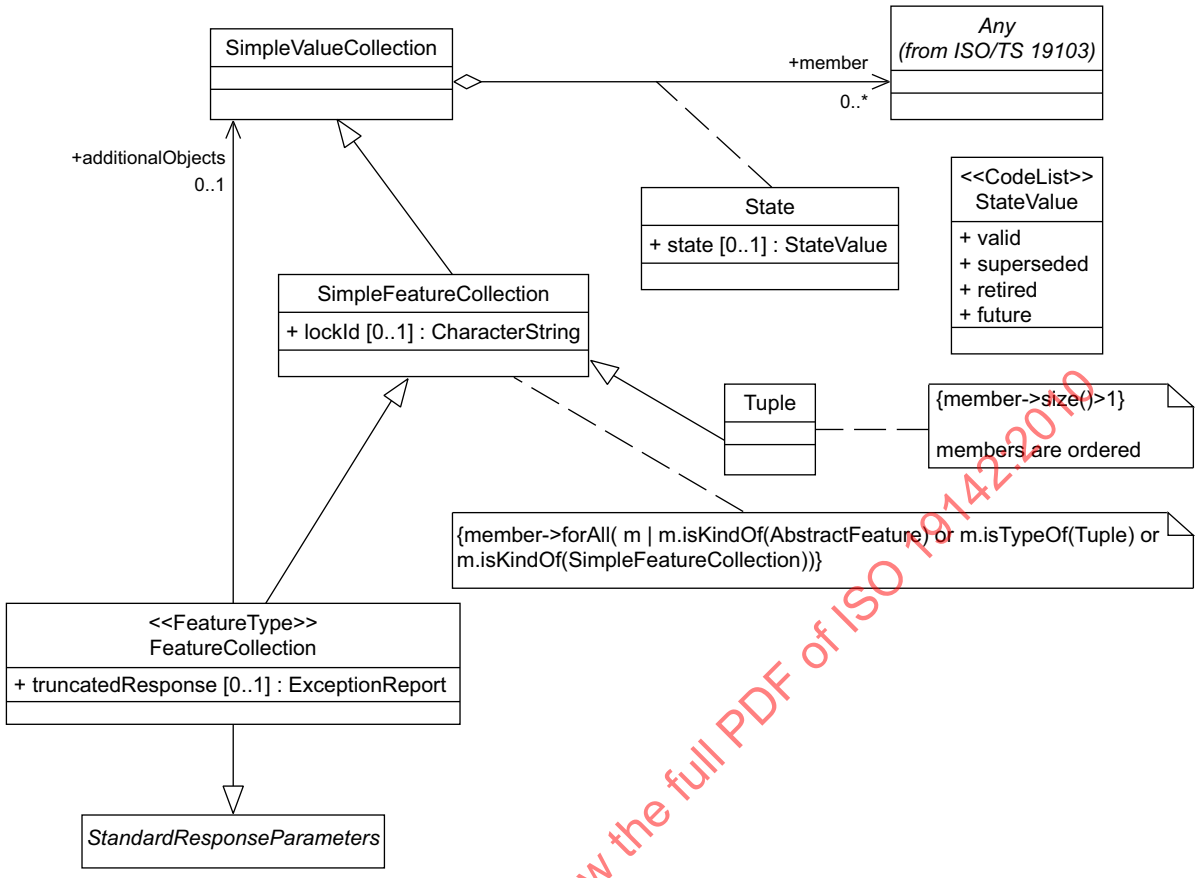

Figure 18 — GetFeature response

11.3.2 XML encoding

The response to a GetFeature request is an XML document with a root element, wfs:FeatureCollection, described by the following XML Schema fragment:

```
<xsd:element name="FeatureCollection"
             type="wfs:FeatureCollectionType"
             substitutionGroup="wfs:SimpleFeatureCollection"/>
<xsd:complexType name="FeatureCollectionType">
   <xsd:complexContent>
      <xsd:extension base="wfs:SimpleFeatureCollectionType">
         <xsd:sequence>
            <xsd:element ref="wfs:additionalObjects" minOccurs="0"/>
            <xsd:element ref="wfs:truncatedResponse" minOccurs="0"/>
         </xsd:sequence>
         <xsd:attributeGroup ref="wfs:StandardResponseParameters"/>
         <xsd:attribute name="lockId" type="xsd:string"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:element name="additionalObjects">
   <xsd:complexType>
      <xsd:choice>
         <xsd:element ref="wfs:ValueCollection"/>
         <xsd:element ref="wfs:SimpleFeatureCollection"/>
      </xsd:choice>
   </xsd:complexType>
</xsd:element>
<xsd:element name="SimpleFeatureCollection"
             type="wfs:SimpleFeatureCollectionType"/>
<xsd:complexType name="SimpleFeatureCollectionType">
   <xsd:sequence>
```

```
        <xsd:element ref="wfs:boundedBy" minOccurs="0"/>
        <xsd:element ref="wfs:member" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:element name="boundedBy" type="wfs:EnvelopePropertyType"/>
<xsd:complexType name="EnvelopePropertyType">
    <xsd:sequence>
        <xsd:any namespace="##other"/>
    </xsd:sequence>
</xsd:complexType>
```

11.3.3 Parameter discussions

11.3.3.1 lockId parameter

The lockId parameter shall not be populated in response to the GetFeature operation.

See 13.3.2 for a discussion of the lockId parameter.

11.3.3.2 state parameter

See the XML fragment in 10.3.2 for the declaration of the wfs:member element which contains the state attribute.

Servers that do not support the Feature versions conformance class (see Table 1) shall not use the state parameter.

Servers that do support the Feature versions conformance class shall use the state parameter to report the state of a versioned feature in a response document. The value domain of the state parameter is the string "retired", “superseded”, “valid” and any other string prefixed with the token "other:". The token "retired" shall be used to indicate that the feature has been deleted from the underlying data store; “superseded” shall indicate that the particular version of a feature has been superseded by a newer version in the underlying data store; “valid” shall indicate that the feature version is the current version in the underlying data store; "future" shall indicate a value which is not yet valid. This may be for legal reasons, it may be used to distribute a proposed value, etc.

Any other string prefixed with the token "other:" may be used to report a server-specific versioned feature state. This International Standard does not assign any meaning to values prefixed with the token "other:".

11.3.3.3 Standard response parameters

See 7.7 for a description of the standard response parameters.

11.3.3.4 Single query response

If a GetFeature operation contains a single query expression, a server shall respond with a wfs:FeatureCollection element containing zero or more wfs:member elements, each containing or referencing a feature in the response set of a GetFeature operation.

11.3.3.5 Multiple query response

If a GetFeature operation contains multiple query expressions, a server shall respond with a wfs:FeatureCollection element containing one or more wfs:member elements, each containing a wfs:FeatureCollection element containing the response to one of the query expressions in the request.

Component feature collections may be encoded inline in the response document or referenced or both.

EXAMPLE 1 The following fragment illustrates a response to GetFeature operations that contains multiple query expressions where the component collections are encoded inline.

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection
   timeStamp="2008-08-01T22:47:02"
   numberMatched="349" numberReturned="300"
   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.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:member>
      <wfs:FeatureCollection
         timeStamp="2008-08-01T22:47:02"
         numberMatched="15" numberReturned="15">
         ...
      </wfs:FeatureCollection>
   </wfs:member>
   <wfs:member>
      <wfs:FeatureCollection
         timeStamp="2008-08-01T22:47:04"
         numberMatched="257" numberReturned="257">
         ...
      </wfs:FeatureCollection>
   </wfs:member>
   <wfs:member>
      <wfs:FeatureCollection
         timeStamp="2008-08-01T22:47:06"
         numberMatched="77" numberReturned="28">
         ...
      </wfs:FeatureCollection>
   </wfs:member>
</wfs:FeatureCollection>
```

EXAMPLE 2 The following fragment illustrates a response to GetFeature operations that contains multiple query expressions where the component collections are referenced.

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection
   timeStamp="2008-08-01T22:47:02"
   numberMatched="349" numberReturned="300"
   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.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:member xlink:href="..."/>
   ...
   <wfs:member xlink:href="..."/>
</wfs:FeatureCollection>
```


Feature members that satisfy more than one query expression in a GetFeature operation shall appear in one collection and be referenced by all others.

The order of the collections in the response document shall be the same as the order of the query expressions in the GetFeature request, although the queries may be processed in any order the server desires.

11.3.3.6 Join query response

If GetFeature operations contain a query expression that is performing a join, a server shall use the wfs:Tuple element (see 10.3.2) to report all tuples of features that satisfy the join predicate.

EXAMPLE 1 The following XML fragment shows the response generated by a query expression joining three feature types *ns1:FeatureTypeOne, ns1:FeatureTypeTwo and ns2:FeatureTypeThree*:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection
   timeStamp="2008-08-01T22:47:02"
   numberMatched="200"
   numberReturned="200"
   xmlns:ns1="http://www.someserver.com/ns1"
   xmlns:ns2="http://www.someserver.com/ns2"
   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.someserver.com/ns1 ./ns1.xsd
                       http://www.xomeserver.com/ns2 ./ns2.xsd
                       http://www.opengis.net/wfs/2.0
                       http://schemas.opengis.net/wfs/2.0.0/wfs.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:boundedBy>
      <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
         <gml:lowerCorner>-32.7638053894043 104.1429748535156</gml:lowerCorner>
         <gml:upperCorner>0 133.3553924560547</gml:upperCorner>
      </gml:Envelope>
   </wfs:boundedBy>
   <wfs:member>
      <wfs:Tuple>
         <wfs:member>
            <ns1:FeatureTypeOne>...</ns1:FeatureTypeOne>
         </wfs:member>
         <wfs:member>
            <ns1:FeatureTypeTwo>...</ns1:FeatureTypeTwo>
         </wfs:member>
         <wfs:member>
            <ns2:FeatureTypeThree>...</ns2:FeatureTypeThree>
         </wfs:member>
      </wfs:Tuple>
   </wfs:member>
   <wfs:member>
      <wfs:Tuple>
         <wfs:member>
            <ns1:FeatureTypeOne>...</ns1:FeatureTypeOne>
         </wfs:member>
         <wfs:member>
            <ns1:FeatureTypeTwo>...</ns1:FeatureTypeTwo>
         </wfs:member>
         <wfs:member>
            <ns1:FeatureTypeThree>...</ns1:FeatureTypeThree>
         </wfs:member>
      </wfs:Tuple>
   </wfs:member>
   <wfs:member>
      <wfs:Tuple>
         <wfs:member xlink:href="..."/>
         <wfs:member xlink:href="..."/>
         <wfs:member>
            <ns2:FeatureTypeThree>...</ns2:FeatureTypeThree>
         </wfs:member>
      </wfs:Tuple>
   </wfs:member>
   ...
</wfs:FeatureCollection>
```

GetFeature operations that contain a mix of join and non-join query expressions can contain component feature collections containing a mixture of wfs:member and wfs:Tuple elements.

EXAMPLE 2 The following fragment shows a response document for a GetFeature operation that contains a mix of join and non-join query expressions.

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection
   timeStamp="2008-08-01T22:47:02"
   numberMatched="349"
   numberReturned="300"
   xmlns="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.someserver.com/myns ./RoadRail.xsd
                       http://www.opengis.net/wfs/2.0
                       http://schemas.opengis.net/wfs/2.0.0/wfs.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:boundedBy>
      <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
         <gml:lowerCorner>-32.7638053894043 104.1429748535156</gml:lowerCorner>
         <gml:upperCorner>0 133.3553924560547</gml:upperCorner>
      </gml:Envelope>
   </wfs:boundedBy>
   <wfs:member>
      <wfs:FeatureCollection
         timeStamp="2008-08-01T22:47:02"
         numberMatched="15"
         numberReturned="15">
         <wfs:member>...</wfs:member>
         <wfs:member>...</wfs:member>
         ...
      </wfs:FeatureCollection>
   </wfs:member>
   <wfs:member>
      <wfs:FeatureCollection
         timeStamp="2008-08-01T22:47:04"
         numberMatched="257"
         numberReturned="257">
         <wfs:member>
            <wfs:Tuple>
               <wfs:member>...</wfs:member>
               <wfs:member>...</wfs:member>
               ...
            </wfs:Tuple>
         </wfs:member>
         <wfs:member>
            <wfs:Tuple>
               <wfs:member>...</wfs:member>
               <wfs:member>...</wfs:member>
               ...
            </wfs:Tuple>
         </wfs:member>
         ...
      </wfs:FeatureCollection>
   </wfs:member>
   <wfs:member>
      <wfs:FeatureCollection
         timeStamp="2008-08-01T22:47:06"
         numberMatched="57"
         numberReturned="28">
         <wfs:member>...</wfs:member>
         <wfs:member>...</wfs:member>
         ...
      </wfs:FeatureCollection>
   </wfs:member>
</wfs:FeatureCollection>
```


11.3.4 Additional objects

If additional objects need to be included in the response document in order to satisfy the resolution of referenced resources (see 7.6.4), these objects shall be placed within the wfs:additionalObjects element, and resource references in the returned features shall be relocated to point to these local copies of the referenced resources. As specified in 7.6.4.5, the server shall also arrange that all unresolved nested resource references are appropriately relocated to point to the intended resources.

11.3.5 GetFeatureById response

Executing a GetFeature operation containing the GetFeatureById stored query (see 7.9.3.6) shall generate a response composed of a single feature encoded according to the values of the outputFormat parameter (see 7.6.3.7). The response shall only contain the feature XML without any of the normal containing elements generated as a result of executing a GetFeature operation (see 11.3.3.4, 11.3.3.5 and 11.3.3.6).

11.4 Exceptions

In the event that a web feature service encounters an error parsing a GetFeature request, it shall raise an OperationParsingFailed exception as described in 7.5.

In the event that a web feature service encounters an error processing a GetFeature request, it shall raise an OperationProcessingFailed exception as described in 7.5.

12 LockFeature operation

12.1 Introduction

Web connections are inherently stateless. As a consequence of this, the semantics of serializable transactions are not preserved. To understand the issue, consider an update operation.

The client fetches a feature instance. The feature is then modified on the client side, and submitted back to the WFS via a Transaction request for update. Serializability is lost since there is nothing to guarantee that while the feature was being modified on the client side, another client did not come along and update the same feature in the database.

One way to ensure serializability is to require that access to data be done in a mutually exclusive manner; that is, while one transaction accesses a data item, no other transaction may modify the same data item. This may be accomplished by using locks that control access to the data.

The purpose of the LockFeature operation is to expose *a long-term feature locking* mechanism to ensure consistency. The lock is considered to be long-term because network latency would make feature locks last relatively longer than native commercial database locks.

If a server implements the LockFeature operation, this fact shall be advertised in the server's capabilities document (see 8.3.3).

12.2 Request

12.2.1 Request semantics

Figure 19 describes the schema of a LockFeature request.

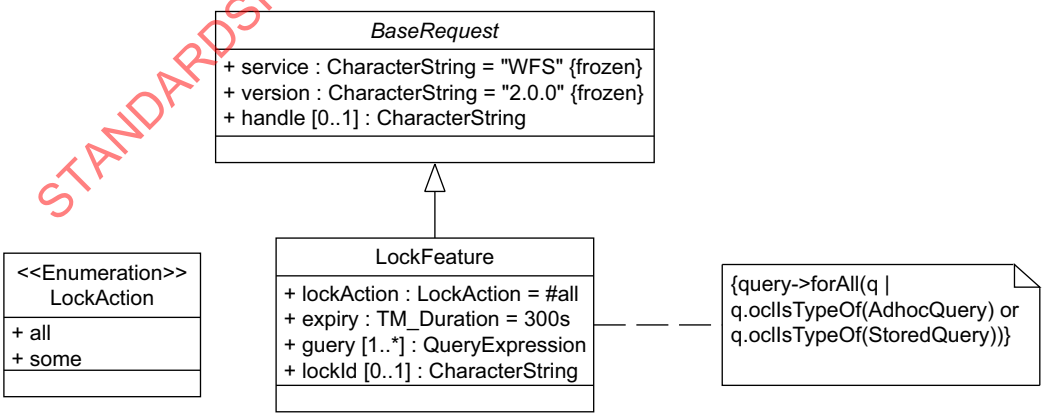

Figure 19 — LockFeature request

12.2.2 XML encoding

The following XML Schema fragment defines the XML encoding of a LockFeature request:

```
<xsd:element name="LockFeature" type="wfs:LockFeatureType"/>
<xsd:complexType name="LockFeatureType">
   <xsd:complexContent>
      <xsd:extension base="wfs:BaseRequestType">
         <xsd:sequence>
            <xsd:AbstractQueryExpression maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="lockId" type="xsd:string"/>
         <xsd:attribute name="expiry" type="xsd:positiveInteger" default="300"/>
         <xsd:attribute name="lockAction" type="wfs:AllSomeType" default="ALL"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="AllSomeType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="ALL"/>
      <xsd:enumeration value="SOME"/>
   </xsd:restriction>
</xsd:simpleType>
```

12.2.3 KVP encoding

Table 18 defines the KVP encoding for the LockFeature operation.

Table 18 — Keywords for LockFeature KVP encoding

URL component	O/M[a]	Default	Description
Common Keywords (REQUEST=LockFeature)			See Table 7. (Only keywords for all operations or the LockFeature operation.)
Adhoc Query Keywords (Mutually exclusive with Stored Query Keywords and LOCKID keyword)			See Table 8.
Stored Query Keywords (Mutually exclusive with Adhoc Query Keywords and LOCKID keyword)			See Table 10.
LOCKID	O		Specify an existing lock identifier for the purpose of resetting the lock expiry.
EXPIRY	O	300	The number of seconds the lock shall persist before being cleared if no Transaction request is received to release the locks.
LOCKACTION	O	ALL	Specify how the lock shall be acquired. ALL indicates that all features shall be locked in order for the operation to succeed, otherwise the operation shall raise a CannotLockAllFeatures exception. SOME indicates that the server shall lock as many features as possible; the operation shall always succeed although the returned lockId may not lock anything.

[a] O = Optional, M = Mandatory.

12.2.4 Parameter discussions

12.2.4.1 AbstractQueryExpression parameter

The LockFeature request operates on a set of features identified using one or more query expressions.

For XML-encoded requests, this International Standard defines the elements wfs:Query (see 7.9.2.2) and wfs:StoredQuery (see 7.9.3.2) that may be used as query expressions in a LockFeature request and which are substitutable for fes:AbstractQueryExpression (see ISO 19143:2010, 6.2).

Other elements may be defined as being substitutable for fes:AbstractQueryExpression. However, this International Standard only assigns meaning to the wfs:Query and wfs:StoredQuery elements.

The KVP encoding of an ad hoc query expression is defined in 7.9.2.3. Multiple ad hoc query expressions may be encoded into a KVP-encoded request.

The KVP encoding of a stored query expression is defined in 7.9.3.3. Only a single stored query expression may be encoded into a KVP-encoded request.

KVP-encoded requests shall only contain query expressions of one type: either ad hoc query expression(s) or a stored query expression.

12.2.4.2 lockId parameter

A lockId parameter specified within a LockFeature operation shall be used to reset the expiry time of an existing lock.

EXAMPLE 1 The following XML-encoded request resets the lock expiry of the specified lock to 600 s:

```
<wfs:LockFeature lockId="1013" expiry="600"/>
```

Such a LockFeature operation shall be executed before the specified lock has expired; otherwise the server shall raise a LockHasExpired exception (see Table 3).

The expiry shall be reset relative to the time that the original lock was acquired.

EXAMPLE 2 If a lock was originally acquired for 300 s, and after 50 s the lock expiry is reset to 600 s, the lock shall remain valid for an additional 550 s.

12.2.4.3 Lock expiry parameter

The expiry parameter may be used to set a limit on how long a WFS shall hold a lock on a feature in the event that a transaction is never issued that would release the lock.

The expiry limit is specified in number of seconds.

The expiry timer shall be started once the entire lock request has been processed and the lock response has been completely transmitted to the client.

Once the specified time period has elapsed, a WFS shall release the lock if it exists. Any further transactions issued against that lock using the same lock identifier shall fail and the service shall raise an InvalidParameterValue exception as described in 7.5.

If a value is not specified for the expiry parameter, the default expiry period shall be 300 s.

The expiry time of an existing lock can be extended as described in 12.2.4.2.

12.2.4.4 lockAction parameter

The optional lockAction parameter controls how feature locks are acquired.

A lock action of ALL indicates that the WFS shall acquire a lock on all requested feature instances. If all requested feature instances cannot be locked, then the operation shall fail, the service shall raise a CannotLockAllFeatures exception as specified in 7.5, and no feature instances shall remain locked.

A lock action of SOME indicates that the WFS shall acquire a lock on as many of the requested feature instances as it can.

The default lock action shall be ALL.

Figure 20 presents a state machine for the LockFeature operation.

12.2.5 State machine for WFS locking

This subclause defines the state machine for the lock state for a service that provides the WFS interface. The state diagram shows the allowed transitions between the states. All other state transitions are disallowed and are considered errors if exhibited by a service.

A service shall support the ability to lock more than one set of features where each feature may only be locked by one lock. Each of the locks is independent when viewed from the service defined by the WFS specification.

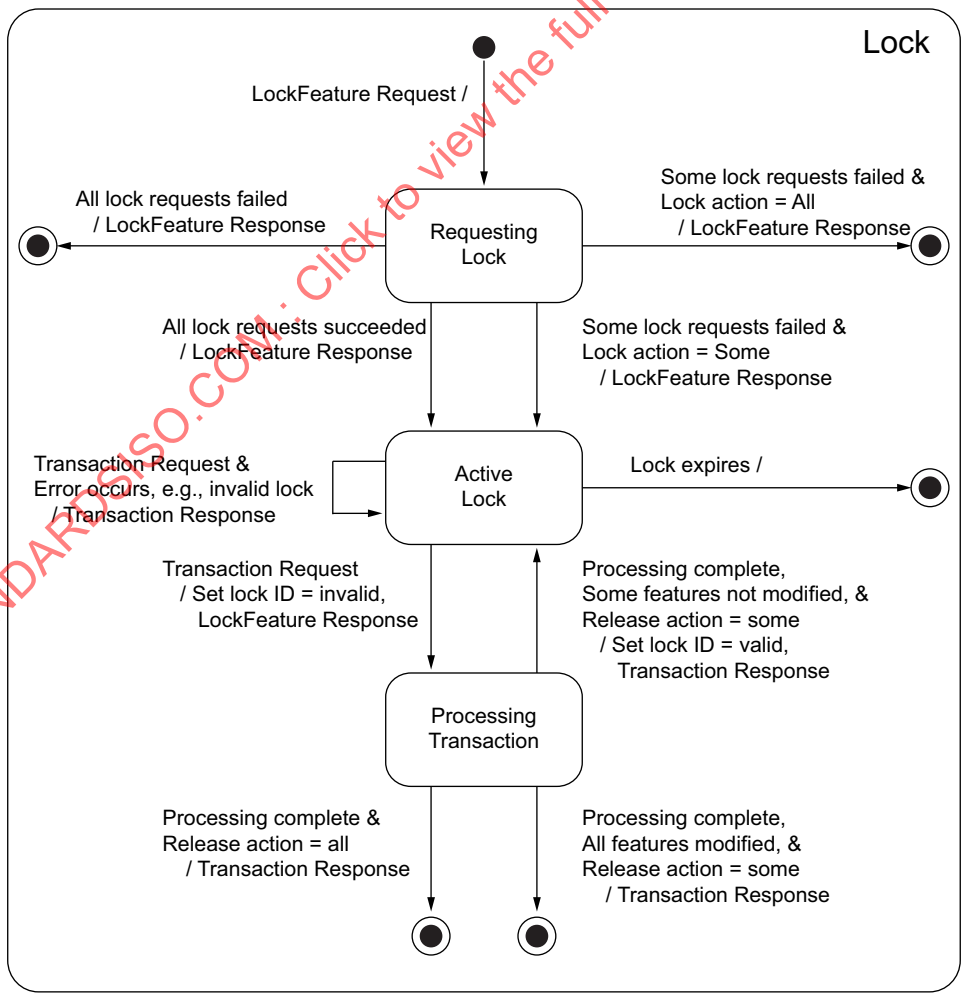

Figure 20 — State diagram for a WFS lock

12.3 Response

12.3.1 Response semantics

Figure 21 describes the schema of a LockFeature response.

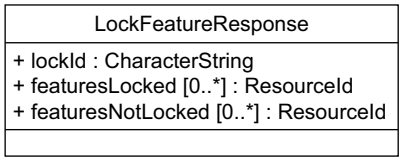

Figure 21 — LockFeature response

12.3.2 XML encoding

The following XML Schema fragment defines the XML encoding of a LockFeature response:

```
<xsd:element name="LockFeatureResponse" type="wfs:LockFeatureResponseType"/>
<xsd:complexType name="LockFeatureResponseType">
   <xsd:sequence>
      <xsd:element name="FeaturesLocked"
                   type="wfs:FeaturesLockedType" minOccurs="0"/>
      <xsd:element name="FeaturesNotLocked"
                   type="wfs:FeaturesNotLockedType" minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="lockId" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="FeaturesLockedType">
  <xsd:sequence maxOccurs="unbounded">
    <xsd:element ref="fes:ResourceId"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="FeaturesNotLockedType">
  <xsd:sequence maxOccurs="unbounded">
    <xsd:element ref="fes:ResourceId"/>
  </xsd:sequence>
</xsd:complexType>
```

In response to a LockFeature request, a web feature service shall generate a wfs:LockFeatureResponse element. This document shall contain a lock identifier that a client application may use in subsequent WFS operations to operate upon the set of locked feature instances. The response may also contain the optional elements wfs:FeaturesLocked and wfs:FeaturesNotLocked, depending on the value of the lockAction attribute (see 13.2.4.2).

If the lock action is specified as ALL and all identified feature instances are successfully locked, a WFS shall respond with a wfs:WFS_LockFeatureResponse element that contains the wfs:FeaturesLocked element and no wfs:FeatureNotLocked element (since either all identified feature instances may be locked or none at all). If some or all feature instances cannot be locked, a WFS shall respond with an exception, indicating that the lock request failed because some or all feature instances are locked by other clients.

If the lock action is specified as SOME, then the wfs:WFS_LockFeatureResponse element shall contain the <wfs:FeaturesLocked> and wfs:FeatureNotLocked elements. The wfs:FeaturesLocked element shall list the feature identifiers of all the feature instances that were locked by the LockFeature request. The wfs:FeaturesNotLocked element shall contain a list of feature identifiers for the feature instances that could not be locked by the web feature service (possibly because they were already locked by someone else). If the lock request results in no features being locked, then a WFS shall respond with a <wfs:LockFeatureResponse> document that contains a value for the lockId attribute but that contains neither a wfs:FeaturesLocked element nor a wfs:FeatureNotLocked element (in other words, an empty response). After the request is completed, the lockId shall be immediately released, since no resources were locked. If

the same lockId is used in a subsequent transaction, an InvalidLockId exception shall be raised, as described in 7.5, since that lockId value no longer exists.

No assumption is made about the format of the lock identifier. The only requirement is that it can be expressed in the character set of the transaction request.

12.4 Exceptions

If a WFS does not implement the LockFeature operation then it shall generate an OperationNotSupported exception, indicating that the operation is not supported, if such a request is encountered.

In the event that a web feature service does support the LockFeature operation and encounters an error parsing the request, it shall raise an OperationParsingFailed exception as described in 7.5.

In the event that a web feature service does support the LockFeature operation and encounters an error processing the request, it shall raise an OperationProcessingFailed exception as described in 7.5.

13 GetFeatureWithLock operation

13.1 Introduction

The GetFeatureWithLock operation is functionally similar to the GetFeature operation (see Clause 11) except that, in response to a GetFeatureWithLock operation, a WFS shall not only generate a response document similar to that of the GetFeature operation, but shall also lock the features in the result set, presumably to update the features in a subsequent Transaction operation (see Clause 15).

13.2 Request

13.2.1 Request semantics

Figure 22 describes the schema of a GetFeatureWithLock request.

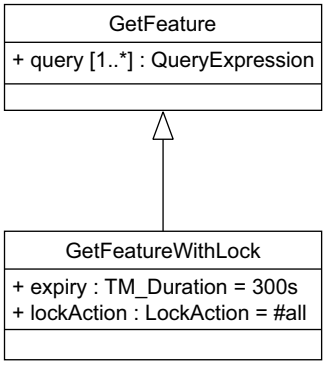

Figure 22 — GetFeatureWithLock request

13.2.2 XML encoding

The following XML Schema fragment declares the XML encoding for the GetFeatureWithLock operation:

```
<xsd:element name="GetFeatureWithLock" type="wfs:GetFeatureWithLockType"/>
<xsd:complexType name="GetFeatureWithLockType">
   <xsd:complexContent>
      <xsd:extension base="wfs:GetFeatureType">
```

```
            <xsd:attribute name="expiry" type="xsd:positiveInteger" default="300"/>
            <xsd:attribute name="lockAction" type="wfs:AllSomeType" default="ALL"/>
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>
```

13.2.3 KVP encoding

The KVP encoding of the GetFeatureWithLock operation is similar to that of the GetFeature operation (see Table 17). Table 19 lists the additional KVP parameters that may be used with the GetFeatureWithLock operation.

Table 19 — Additional keywords for GetFeatureWithLock KVP encoding

URL component	O/M[a]	Default	Description
EXPIRY	O	300	This parameter may only be specified if the request is GetFeatureWithLock. Its value is a positive integer. The value is specified in seconds and indicates the length of time a lock will be held on the features in the result set. If the parameter is not specified then the locks will be held for the default expiry period of 300 s.
LOCKACTION	O	ALL	Specify how the lock shall be acquired. ALL indicates that all features shall be locked in order for the operation to succeed, otherwise the operation shall raise a CannotLockAllFeatures exception. SOME indicates that the server shall lock as many features as possible; the operation shall always succeed although the returned lockId may not lock anything.

[a] O = Optional, M = Mandatory.

13.2.4 Parameter discussion

13.2.4.1 expiry parameter

The expiry parameter is used to set a time limit for how long the WFS shall maintain the locks, in the event that a Transaction request is never received to release them. The default lock duration is 300 s or 5 min.

13.2.4.2 lockAction parameter

The lockAction parameter controls how a WFS attempts to obtain locks on features in the result set of the operation.

A value of ALL (the default value) indicates that the WFS shall attempt to lock all features in the result set. In the event that this is not possible, because some of the features are already locked, the WFS shall raise a CannotLockAllFeatures exception as described in 7.5. If all features are successfully locked, the WFS shall report to the client a lock identifier (using the lockId attribute on the wfs:FeatureCollection element) that can be used in a subsequent Transaction operation to operate on the locked features and release the locks (see 15.2.3.1.2).

A value of SOME indicates that the WFS shall lock as many feature in the result set as possible. The response document shall only contain those features that were successfully locked and the WFS shall report to the client a lock identifier (using the lockId attribute on the wfs:FeatureCollection element) that can be used in a subsequent Transaction operation to operate on the locked features and release the locks (see 15.2.3.1.2).

13.2.4.3 resultType parameter

The resultType parameter is described in 7.6.3.6.

The only valid value for the resultType attribute for a GetFeatureWithLock request shall be "results". If a client specifies a value of "hits", the server shall raise an InvalidParameterValue exception (see 7.5).

13.3 Response

13.3.1 Introduction

The response to a GetFeatureWithLock operation is similar to the response to a GetFeature operation. The only difference is that the response to a GetFeatureWithLock operation includes a value for the lockId parameter.

13.3.2 lockId parameter

For a GetFeatureWithLock request, a WFS shall generate a response document that includes the lock identifier used to lock features in the result set. The lock identifier shall be encoded in the response using the lockId attribute that is declared on the wfs:FeatureCollection element (see 11.3.2).

EXAMPLE The following XML fragment illustrates how to include the lockId attribute in the response to a GetFeatureWithLock operation:

```
<wfs:FeatureCollection lockId="00A01"… >
…
</wfs:FeatureCollection>
```

The ellipses are meant to represent all the other components included in the GetFeatureWithLock response which are identical to the components included in the GetFeature response (see 11.3).

13.4 Exceptions

If a WFS does not implement the GetFeatureWithLock operation then it shall generate an OperationNotSupported exception, indicating that the operation is not supported, if such a request is encountered.

In the event that a web feature service does support the GetFeatureWithLock operation and encounters an error parsing the request, it shall raise an OperationParsingFailed exception as described in 7.5.

In the event that a web feature service does support the GetFeatureWithLock operation and encounters an error processing the request, it shall raise an OperationProcessingFailed exception as described in 7.5.

14 Stored query management

14.1 Introduction

This clause describes the operations ListStoredQueries, DescribeStoredQueries, CreateStoredQuery and DropStoredQuery.

All servers shall support the ListStoredQueries and DescribeStoredQueries operations and shall implement the GetFeatureById stored query (see Table 1 and 7.9.3.6).

Servers that implement the Manage stored queries conformance class (see Table 1) shall declare this in their capabilities document using the ManageStoredQueries constraint (see Table 13) and shall implement the CreateStoredQuery and DropStoredQuery operations.

14.2 Defining stored queries

14.2.1 XML encoding

The following XML Schema fragment defines the XML encoding for describing or defining a stored query expression:

```
<xsd:complexType name="StoredQueryDescriptionType">
   <xsd:sequence>
      <xsd:element ref="wfs:Title" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wfs:Abstract" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="ows:Metadata" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="Parameter"
                   type="wfs:ParameterExpressionType"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="QueryExpressionText"
                   type="wfs:QueryExpressionTextType"
                   minOccurs="1" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="id" type="xsd:anyURI" use="required"/>
</xsd:complexType>
<xsd:complexType name="ParameterExpressionType">
   <xsd:sequence>
      <xsd:element ref="wfs:Title" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wfs:Abstract" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="ows:Metadata" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="xsd:string" use="required"/>
   <xsd:attribute name="type" type="xsd:QName" use="required"/>
</xsd:complexType>
<xsd:complexType name="QueryExpressionTextType" mixed="true">
   <xsd:choice>
      <xsd:any namespace="##other" processContents="skip"
               minOccurs="0" maxOccurs="unbounded"/>
      <xsd:any namespace="##targetNamespace" processContents="skip"
               minOccurs="0" maxOccurs="unbounded"/>
   </xsd:choice>
   <xsd:attribute name="returnFeatureTypes"
                  type="wfs:ReturnFeatureTypesListType" use="required"/>
   <xsd:attribute name="language" type="xsd:anyURI" use="required"/>
   <xsd:attribute name="isPrivate" type="xsd:boolean" default="false"/>
</xsd:complexType>
<xsd:simpleType name="ReturnFeatureTypesListType">
   <xsd:list itemType="xsd:QName"/>
</xsd:simpleType>
```

The type wfs:StoredQueryDescriptionType is used to declare the element wfs:StoredQueryDescription that provides a description of a stored query in response to a DescribeStoredQueries operation (see 14.4) and the element wfs:StoredQueryDefinition that is used by the CreateStoredQuery operation (see 14.5) to define a stored query.

The description or definition of a stored query contains metadata describing the stored query, a list of zero or more arguments that the stored query accepts and one or more component query expressions that are executed when the stored query is invoked.

14.2.2 Parameter discussion

14.2.2.1 Title parameter

The Title parameter may be used to assign a human-readable name to a stored query. Multiple titles can be specified in one or more languages. If more than one wfs:Title element is specified, the values of the xml:lang attribute for each shall be different.

The language of a Title string shall be declared using a two-character language code as specified in IETF RFC 4646. The default language value is "en".

14.2.2.2 Abstract parameter

The Abstract parameter may be used to assign a human-readable descriptive to a stored query. Multiple abstracts can be specified in one or more languages. If more than one wfs:Abstract element is specified, the values of the xml:lang attribute for each shall be different.

The language of an Abstract string shall be declared using a two-character language code as specified in IETF RFC 4646. The default language value is "en".

14.2.2.3 Metadata parameter

The Metadata parameter may be used to encode inline or reference more detailed metadata about a stored query. The ows:Metadata element is described in OGC 06-121r3.

14.2.2.4 Parameter parameter

14.2.2.4.1 Introduction

A stored query may have zero or more arguments. Each argument of a stored query shall be enumerated in the definition of the stored query using the wfs:Parameter element.

14.2.2.4.2 Title, Abstract and Metadata parameters

See 14.2.2.1, 14.2.2.2 and 14.2.2.3 for a description of the Title, Abstract and Metadata parameters.

14.2.2.4.3 name parameter

The name of each argument of a stored query shall be encoded using the name attribute on the wfs:Parameter element.

14.2.2.4.4 type parameter

The type of each argument of a stored query shall be encoded using the type attribute on the wfs:Parameter element. The value of the type attribute shall be a QName identifying a type that describes the content model of the corresponding argument.

EXAMPLE 1 type="xsd:double" indicates that the type of the stored query parameter is a double.

EXAMPLE 2 type="gml:PolygonPropertyType" indicates that the type of the stored query parameter is a GML polygon.

14.2.2.5 QueryExpressionText parameter

14.2.2.5.1 Introduction

The wfs:QueryExpressionText element shall be used to enumerate one or more component query expressions that a stored query executes when invoked.

Each wfs:QueryExpressionText element may contain a wfs:Query (see 7.9.2.2) element, or a wfs:StoredQuery (see 7.9.3.2) element or some other implementation-specific content for specifying a component query expression in another implementation language (see 14.2.2.5.3).

Within the text of a component query expression, the notation "${*argument_name*}" shall be used to denote where the values of a stored query's arguments shall be substituted at runtime. The "${*argument_name*}" notation can appear multiple times if the argument value needs to be substituted at a number of places. The token *argument_name* is a placeholder for the name of an argument of the stored query.

EXAMPLE The following stored query expression finds all the features of types FeatureType1, FeatureType2 and FeatureType3 that lie within a bounding box specified by the client when invoking the "Features In Polygon" stored query. The notation "${AreaOfInterest}" is used to indicate where the value of the AreaOfInterest argument shall be substituted at runtime. This notation appears in three places in this example.

```
<?xml version="1.0"?>
<wfs:CreateStoredQuery
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.org/fes/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:myns="http://www.someserver.com/myns"
   xsi:schemaLocation="http://www.opengis.net/wfs/2.0
                       http://schemas.opengis.net/wfs/2.0/wfs.xsd"
   service="WFS"
   version="2.0.0">
   <wfs:StoredQueryDefinition id="urn:StoredQueries:FeaturesInPolygon">
      <wfs:Title>Features In Polygon</wfs:Title>
      <wfs:Abstract>Find all the features in a Polygon.</wfs:Abstract>
      <wfs:Parameter name="AreaOfInterest" type="gml:PolygonPropertyType"/>
      <wfs:QueryExpressionText
           returnFeatureTypes="myns:Parks myns:Lakes myns:Rivers"
           language="urn:ogc:def:queryLanguage:OGC-WFS::WFS_QueryExpression"
           isPrivate="false">
         <wfs:Query typeNames="myns:Parks">
            <fes:Filter>
               <fes:Within>
                  <fes:ValueReference>geometry</fes:ValueReference>
                  ${AreaOfInterest}
               </fes:Within>
            </fes:Filter>
         </wfs:Query>
         <wfs:Query typeNames="myns:Lakes">
            <fes:Filter>
               <fes:Within>
                  <fes:ValueReference>region</fes:ValueReference>
                  ${AreaOfInterest}
               </fes:Within>
            </fes:Filter>
         </wfs:Query>
         <wfs:Query typeNames="myns:Rivers">
            <fes:Filter>
               <fes:Within>
                  <fes:ValueReference>region</fes:ValueReference>
                  ${AreaOfInterest}
               </fes:Within>
            </fes:Filter>
         </wfs:Query>
      </wfs:QueryExpressionText>
   </wfs:StoredQueryDefinition>
</wfs:CreateStoredQuery>
```

14.2.2.5.2 Declaring the returned feature types

The returnFeatureTypes attribute on the wfs:QueryExpressionText element shall be used to specify the feature type(s) that each component query expression returns.

If more than one return feature type is listed, this indicates that the corresponding component query expression returns a join tuple composed of the listed feature types.

The returned feature type names shall be from the list of feature type names that a server advertises in its capabilities document (see 8.3.3).

14.2.2.5.3 Implementation language

The language attribute of the wfs:QueryExpressionText element shall be used to specify the implementation language of a component query expression.

Servers that conform to this International Standard shall support the value "urn:ogc:def:queryLanguage:OGC-WFS::WFSQueryExpression", indicating that the component query expression is specified using the wfs:Query (see 7.9.2.2) or wfs:StoredQuery (see 7.9.3.2) elements.

Other implementation languages may be supported but this International Standard does not assign any meaning to these other values. If a server supports other implementation languages for stored queries, it shall advertise these language parameter values in its capabilities document (see Table 12).

NOTE The specification of a component query expression can be expressed in any language that the server declares that it supports in its capabilities document (see 8.3.3). This can include languages such as SQL, XQuery, XPath, and SPARQL. In fact, if the server supports it, queries can even be expressed in code such as Java.

14.2.2.5.4 IsPrivate parameter

The implementation text of a component query expression, specified using the wfs:QueryExpressionText element, may be private or public. Private means that the implementation text is visible only to the creator of the stored query. Public means that the implementation text is visible to everyone.

This characteristic is set when the stored query is created and determines whether the implementation text of the component query expression is presented in a response to a DescribeStoredQueries operation (see 14.4).

If the value of the IsPrivate parameter is "true", the server shall not present the implementation text of a component query expression in response to a DescribeStoredQueries operation.

If the value of the IsPrivate parameter is "false", the server shall present the implementation text of a component query expression in response to DescribeStoredQueries operation.

14.2.2.6 id parameter

The id parameter shall be used to assign a unique identifier that can be used to repeatedly invoke a stored query.

14.3 ListStoredQueries operation

14.3.1 Request semantics

The ListStoredQueries operation (see Figure 23) lists the stored queries available at a server.

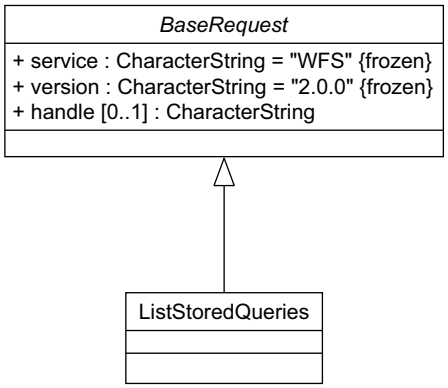

Figure 23 — ListStoredQueries request

14.3.2 XML encoding

The following XML Schema fragment defines the XML encoding for the ListStoredQueries operation:

```
<xsd:element name="ListStoredQueries" type="wfs:ListStoredQueriesType"/>
 <xsd:complexType name="ListStoredQueriesType">
    <xsd:complexContent>
       <xsd:extension base="wfs:BaseRequestType"/>
    </xsd:complexContent>
 </xsd:complexType>
```

14.3.3 KVP encoding

Table 20 defines the KVP encoding for the ListStoredQueries operation.

Table 20 — Keywords for ListStoredQueries KVP encoding

URL component	Description
Common Keywords (REQUEST= ListStoredQueries)	See Table 7. (Only keywords for all operations or the ListStoredQueries operation.)

14.3.4 Response

Figure 24 describes the response to a ListStoredQueries operation.

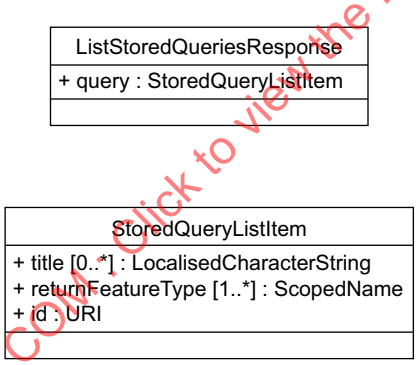

Figure 24 — ListStoredQueriesResponse

The following XML Schema fragment defines the response to a ListStoredQueries operation:

```
<xsd:element name="ListStoredQueriesResponse"
             type="wfs:ListStoredQueriesResponseType"/>
<xsd:complexType name="ListStoredQueriesResponseType">
   <xsd:sequence>
      <xsd:element name="StoredQuery" type="wfs:StoredQueryListItemType"
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="StoredQueryListItemType">
   <xsd:sequence>
      <xsd:element ref="wfs:Title" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="ReturnFeatureType"
                   type="xsd:QName" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="id" type="xsd:anyURI" use="required"/>
</xsd:complexType>
```

The response to a ListStoredQueries operation is the root element wfs:ListStoredQueriesResponse element containing one or more wfs:StoredQuery elements, each describing a stored query that a server offers.

The components wfs:Title, wfs:ReturnFeatureType and the attribute id are described in 14.2.

14.3.5 Exceptions

In the event that a server encounters an error parsing a ListStoredQueries operation, it shall raise an OperationParsingFailed exception as described in 7.5.

In the event that a server encounters an error processing a ListStoredQueries operation, it shall raise an OperationProcessingFailed exception as described in 7.5.

14.4 DescribeStoredQueries operations

14.4.1 Request semantics

The DescribeStoredQueries operation (see Figure 25) provides detailed metadata about each stored query expression that a server offers.

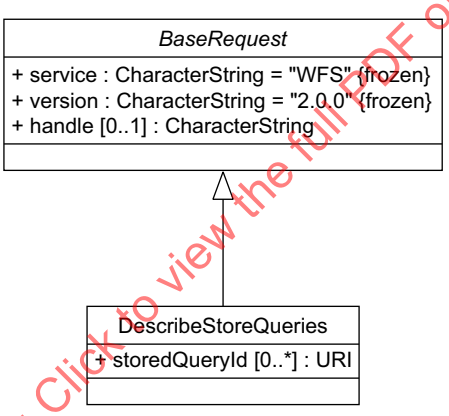

Figure 25 — DescribeStoredQueries request

14.4.2 XML encoding

The following XML Schema fragment defines the XML encoding for the DescribeStoredQueries operation:

```
<xsd:element name="DescribeStoredQueries"
             type="wfs:DescribeStoredQueriesType"/>
<xsd:complexType name="DescribeStoredQueriesType">
   <xsd:complexContent>
      <xsd:extension base="wfs:BaseRequestType">
         <xsd:sequence>
            <xsd:element name="StoredQueryId" type="xsd:anyURI"
                         minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

The wfs:DescribeStoredQueries element contains zero or more wfs:StoredQueryId elements that enumerate the identifiers of the stored queries to describe.

If no wfs:StoredQueryId elements are specified then all stored queries offered by a server shall be described.

14.4.3 KVP encoding

Table 21 defines the KVP encoding for the DescribeStoredQueries operation.

Table 21 — Keywords for DescribeStoredQueries KVP encoding

URL component	O/M[a]	Description
Common Keywords (REQUEST=DescribeStoredQueries)		See Table 7 (Only keywords for all operations or the DescribeStoredQueries operation.)
STOREDQUERY_ID	O	A comma-separated list of stored query identifiers to describe. If the keyword is not specified then all stored queries offered by a server shall be described.

[a] O = Optional, M = Mandatory.

14.4.4 Response

14.4.4.1 Response semantics

Figure 26 describes the response to a DescribeStoredQueries operation.

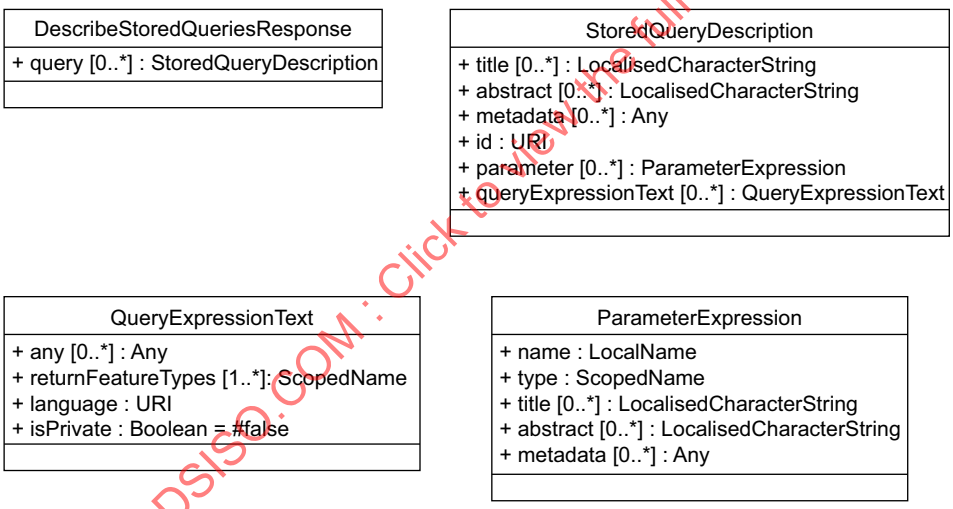

Figure 26 — DescribeStoredQueriesResponse

14.4.4.2 XML encoding

The following XML Schema fragment defines the response to a DescribeStoredQueries operation:

```
<xsd:element name="DescribeStoredQueriesResponse"
             type="wfs:DescribeStoredQueriesResponseType"/>
<xsd:complexType name="DescribeStoredQueriesResponseType">
   <xsd:sequence>
      <xsd:element name="StoredQueryDescription"
                   type="wfs:StoredQueryDescriptionType"
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

The response contains one or more wfs:StoredQueryDescription elements, each describing a stored query.

See 14.2 for a description of the wfs:StoredQueryDescriptionType type.

14.5 CreateStoredQuery operation

14.5.1 Request semantics

A stored query may be created using the CreateStoredQuery operation (see Figure 27).

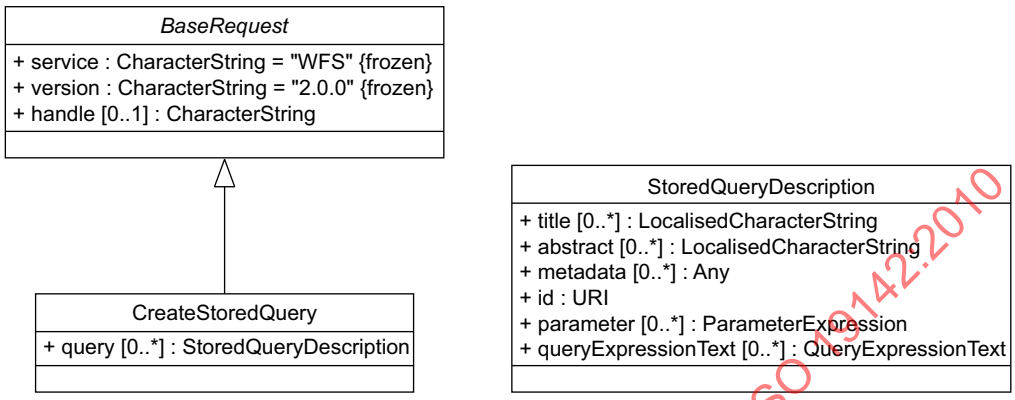

Figure 27 — CreateStoredQuery request

Not all stored queries that a server offers need to be created using the CreateStoredQuery operation. Servers can be pre-configured to offer any number of stored queries implemented in any number of ways. It is anticipated that profiles of this International Standard may define sets of stored queries that compliant WFSs must offer.

NOTE It is further anticipated that, in many cases, for the sake of efficiency and performance, pre-configured stored queries would be implemented as executable code. Such implementations would also allow for complex query logic to be implemented and hidden behind a simple stored query interface.

14.5.2 XML encoding

The following XML Schema fragment defines the XML encoding for the CreateStoredQuery operation.

```
<xsd:element name="CreateStoredQuery"
             type="wfs:CreateStoredQueryType"/>
<xsd:complexType name="CreateStoredQueryType">
   <xsd:complexContent>
      <xsd:extension base="wfs:BaseRequestType">
         <xsd:sequence>
            <xsd:element name="StoredQueryDefinition"
                         type="wfs:StoredQueryDescriptionType"
                         minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

14.5.3 KVP encoding

No KVP encoding is defined for the CreateStoredQuery operation.

14.5.4 Parameter discussions

The StoredQueryDefinition parameter, encoded using the wfs:StoredQueryDefinition element, shall contain the definition of a stored query. Multiple stored queries may be created in a single CreateStoredQuery request.

See 14.2 for a description of the wfs:StoredQueryDescriptionType type.

14.5.5 Response

Figure 28 describes the response to a CreateStoredQuery operation.

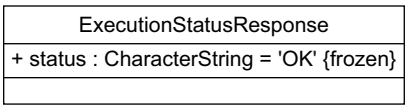

Figure 28 — CreateStoredQuery response

The following XML Schema fragment defines the response to a CreateStoredQuery operation:

```
<xsd:element name="CreateStoredQueryResponse"
             type="wfs:CreateStoredQueryResponseType"/>
<xsd:complexType name="ExecutionStatusType">
   <xsd:attribute name="status" type="xsd:string" fixed="OK"/>
</xsd:complexType>
<xsd:complexType name="CreateStoredQueryResponseType">
   <xsd:complexContent>
      <xsd:extension base="wfs:ExecutionStatusType"/>
   </xsd:complexContent>
</xsd:complexType>
```

The root element of the response is wfs:CreateStoredQuery that contains a single attribute named "status".

The only valid value for the status attribute is "OK", indicating that the stored query was successfully processed and stored by the server. Otherwise, the server shall generate an exception (see 7.5).

14.6 DropStoredQuery operations

14.6.1 Request semantics

The DropStoredQuery operation (see Figure 29) allows previously created stored queries to be dropped from the system.

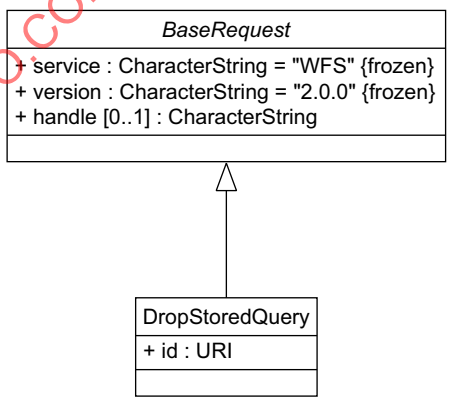

Figure 29 — DropStoredQuery request

14.6.2 XML encoding

The following XML Schema fragment defines the XML encoding for the DropStoredQuery operation:

```
<xsd:element name="DropStoredQuery">
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:extension base="wfs:BaseRequestType">
            <xsd:attribute name="id" type="xsd:anyURI" use="required"/>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>
```

The request simply accepts the identifier of the stored query to drop. The stored query identifier shall be encoded in XML using the id attribute on the wfs:DropStoredQuery element.

14.6.3 KVP encoding

Table 22 defines the KVP encoding for the DropStoredQuery operation.

Table 22 — Keywords for DropStoredQuery KVP encoding

URL component	O/M[a]	Description
Common Keywords (REQUEST= DescribeStoredQueries)		See Table 7 (Only keywords for all operations or the DescribeStoredQueries operation.)
STOREDQUERY_ID	M	A comma-separated list of stored query identifiers to describe. If the keyword is not specified then all stored queries offered by a server shall be described.
[a] O = Optional, M = Mandatory.		

14.6.4 Response

The following XML Schema fragment defines the response to a DropStoredQuery operation:

```
<xsd:element name="DropStoredQueryResponse" type="wfs:ExecutionStatusType"/>
```

The type wfs:ExecutionStatusType is described in 14.5.5.

14.7 Exceptions

If a WFS does not implement CreateStoredQuery or DropStoredQuery operations then it shall generate an OperationNotSupported exception, indicating that the operation is not supported, if any of these requests are encountered.

In the event that a web feature service does support these operations and encounters an error parsing the request, it shall raise an OperationParsingFailed exception as described in 7.5.

In the event that a web feature service does support these operations and encounters an error processing the request, it shall raise an OperationProcessingFailed exception as described in 7.5.

15 Transaction operation

15.1 Introduction

The optional Transaction operation is used to describe data transformation operations to be applied to feature instances under the control of a web feature service. Using the Transaction operation, clients can create, modify, replace and delete features in the web feature service's data store. GML (see ISO 19136:2007) shall be used as the canonical representation of features and it is the responsibility of particular WFS implementations to convert this canonical GML representation into their internal representation used in the data store (which could also be GML, in which case no conversion would be necessary).

When the transaction has been completed, a web feature service shall generate an XML response document, as described in 15.3, indicating the completion status of the operation.

Web feature services that support the optional Transaction operation shall advertise this fact in their capabilities document as described in 8.3.

15.2 Request

15.2.1 Request semantics

Figure 30 describes the schema of a Transaction request.

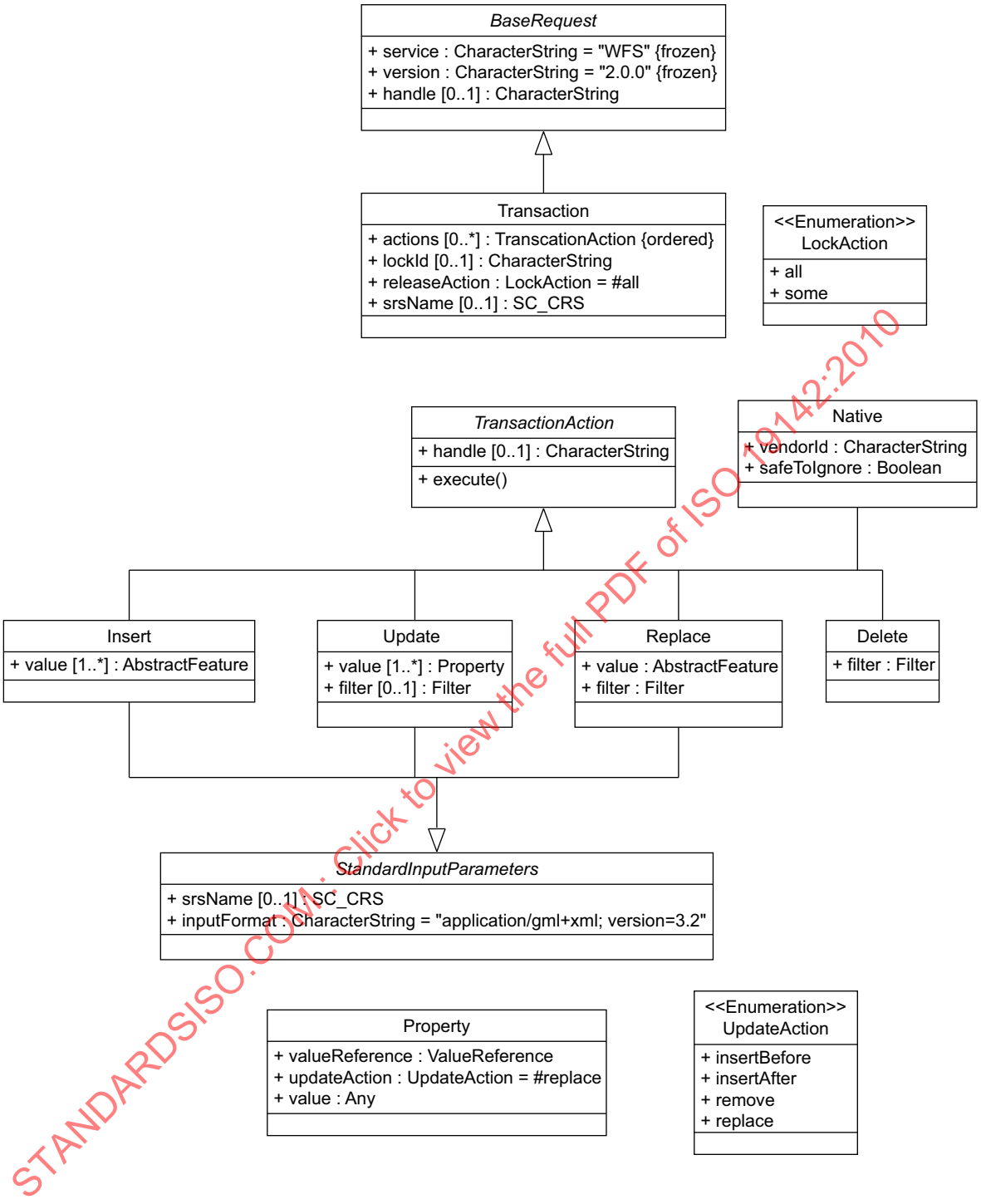

Figure 30 — Transaction request

15.2.2 XML encoding

The XML encoding of a Transaction request is defined by the following XML Schema fragment:

```
<xsd:element name="Transaction" type="wfs:TransactionType"/>
<xsd:complexType name="TransactionType">
   <xsd:complexContent>
      <xsd:extension base="wfs:BaseRequestType">
         <xsd:sequence>
            <xsd:sequence minOccurs="0" maxOccurs="unbounded">
               <xsd:element ref="wfs:AbstractTransactionAction"/>
            </xsd:sequence>
```

```
            </xsd:sequence>
            <xsd:attribute name="lockId" type="xsd:string"/>
            <xsd:attribute name="releaseAction" type="wfs:AllSomeType" default="ALL"/>
            <xsd:attribute name="srsName" type="xsd:string"/>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <xsd:element name="AbstractTransactionAction" type="wfs:AbstractTransactionActionType"
       abstract="true"/>
   <xsd:complexType name="AbstractTransactionActionType" abstract="true">
      <xsd:attribute name="handle" type="xsd:string"/>
   </xsd:complexType>
```

A wfs:Transaction element shall contain zero or more elements that are substitutable for the wfs:AbstractTransactionAction element.

The wfs:AbstractTransactionAction element contains the attribute handle that may be used to tag each transaction action with a client-specified mnemonic name for the purpose of error handling (see 7.6.2.6).

This International Standard defines the elements wfs:Insert, wfs:Update, wfs:Replace and wfs:Delete as substitutable for the wfs:AbstractTransactionAction element. These elements may be used within the wfs:Transaction element to encode operations to create, modify, replace and destroy feature instances.

A WFS shall process wfs:Insert, wfs:Update, wfs:Replace and wfs:Delete elements in the order in which they are presented in the transaction request. Subsequent update and delete actions, in a transaction request, may operate on feature instances created by previous insert actions in the same transaction request.

The wfs:Transaction element may also contain zero or more wfs:Native elements or elements substitutable for wfs:Native that describe vendor-specific actions (see 8.4).

An empty wfs:Transaction element is a valid request and can be used to release any locked features by specifying the lock identifier to be released as the content of the lockId attribute (see 12.2.4.2) and setting the releaseAction to ALL.

15.2.3 Parameter discussions

15.2.3.1 Locking

15.2.3.1.1 Declaring support for locking

Servers shall declare in their capabilities document (see 8.3.3) if they support automatic data locking using the AutomaticDataLocking constraint (see Table 14).

Servers that support automatic data locking shall not require a client to lock features before operating upon them using a Transaction operation. Such servers may optionally implement the LockFeature (see Clause 12) or GetFeatureWithLock (see Clause 13) operations and shall not require the lockId attribute to be specified in a transaction operation, unless the features were in fact previously locked using the LockFeature (see Clause 12) or GetFeatureWithLock (see Clause 13) operations.

Servers that do not support automatic data locking shall implement one or more of the LockFeature and GetFeatureWithLock operations and shall require a client to lock features prior to operating upon them. Failure to do so shall result in the server raising a FeaturesNotLocked exception (see 7.5).

15.2.3.1.2 lockId parameter

The lockId parameter shall be encoded using an attribute named lockId on the wfs:Transaction element.

The content of the lockId parameter shall be the lock identifier obtained from a previous LockFeature (see Clause 12) or GetFeatureWithLock (see Clause 13) operation.

If a WFS is presented with a Transaction request that updates, replaces or deletes locked features and the Transaction request does not contain a lockId attribute, the service shall raise a MissingParameterValue exception (see 7.5).

If a WFS is presented with a Transaction request that updates, replaces or deletes locked features and the value of the lockId attribute does not match the lock identifier value for the affected features, the service shall raise an InvalidParameterValue exception (see 7.5).

Insert actions within a Transaction request shall not be affected by the lockId attribute. If a Transaction request only contains insert actions, the lockId attribute shall not be specified.

A Transaction request does not need to affect all features locked using a particular lock identifier value.

15.2.3.2 releaseAction parameter

The releaseAction parameter, encoded as the attribute releaseAction of the wfs:Transaction element, controls how locked features are treated when a transaction request is completed.

A value of ALL indicates that the locks on all feature instances locked using the specified lockId value shall be released when the transaction completes, regardless of whether or not a particular feature instance in the locked set was actually operated upon. This is the default action if no value is specified for the releaseAction attribute.

A value of SOME indicates that only the locks on feature instances modified by the transaction shall be released. The other, unmodified, locked feature instances shall remain locked using the same lock identifier so that subsequent transactions may operate on those feature instances. In the event that the releaseAction attribute is set to the value SOME, the expiry timer shall be reset to zero after each transaction, unless all feature instances in the locked set have been operated upon. For example, if a client application locks 20 feature instances and then submits a transaction request that only operates on 10 of those locked feature instances, a releaseAction of SOME would mean that the 10 remaining unaltered feature instances shall remain locked when the transaction terminates. Subsequent transaction operations may then be submitted by the client application, using the same lock identifier to modify the remaining 10 feature instances.

15.2.3.3 srsName parameter

The srsName parameter shall be encoded as the attribute srsName on the wfs:Transaction element.

See 7.6.5.5 for a description of the srsName parameter.

15.2.4 Insert action

15.2.4.1 XML encoding

The following XML Schema fragment declares the wfs:Insert element:

```
<xsd:element name="Insert" type="wfs:InsertType"
    substitutionGroup="wfs:AbstractTransactionAction"/>
<xsd:complexType name="InsertType">
   <xsd:complexContent>
      <xsd:extension base="wfs:AbstractTransactionActionType">
         <xsd:sequence>
            <xsd:any namespace="##other" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attributeGroup ref="wfs:StandardInputParameters"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

The wfs:Insert element is used to create new feature instances in a web feature service's data store. By default, the initial state of a feature to be created is expressed using GML (see ISO 19136:2007) and shall validate relative to a GML application schema generated by the DescribeFeatureType operation

(see Clause 9). Multiple wfs:Insert elements may be enclosed in a single Transaction request and multiple feature instances may be created using a single wfs:Insert element.

15.2.4.2 Standard input parameters

See 7.6.5 for a description of the standard input parameters.

15.2.5 Update action

15.2.5.1 XML encoding

The following XML Schema fragment declares the wfs:Update element:

```
<xsd:element name="Update" type="wfs:UpdateType"
     substitutionGroup="wfs:AbstractTransactionAction"/>
<xsd:complexType name="UpdateType">
   <xsd:complexContent>
      <xsd:extension base="wfs:AbstractTransactionActionType">
         <xsd:sequence>
            <xsd:element ref="wfs:Property" maxOccurs="unbounded"/>
            <xsd:element ref="fes:Filter" minOccurs="0"/>
         </xsd:sequence>
         <xsd:attribute name="typeName" type="xsd:QName" use="required"/>
         <xsd:attributeGroup ref="wfs:StandardInputParameters"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Property" type="wfs:PropertyType"/>
<xsd:complexType name="PropertyType">
   <xsd:sequence>
      <xsd:element name="ValueReference">
         <xsd:complexType>
            <xsd:simpleContent>
               <xsd:extension base="xsd:string">
                  <xsd:attribute name="action" type="wfs:UpdateActionType" default="replace"/>
               </xsd:extension>
            </xsd:simpleContent>
         </xsd:complexType>
      </xsd:element>
      <xsd:element name="Value" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="UpdateActionType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="insertBefore"/>
      <xsd:enumeration value="insertAfter"/>
      <xsd:enumeration value="remove"/>
      <xsd:enumeration value="replace"/>
   </xsd:restriction>
</xsd:simpleType>
```

The wfs:Update element contains one or more wfs:Property elements that describe a value change to be made to one of the properties of the explicitly named feature type. Multiple wfs:Update elements may be contained in a single wfs:Transaction request, so that multiple changes can be made to features of the same or different types.

15.2.5.2 Parameter discussions

15.2.5.2.1 Property element

The wfs:Property element contains two child elements: the wfs:ValueReference element and the optional wfs:Value element.

The wfs:Value element shall either contain the replacement value for the node pointed to by the wfs:ValueReference element or shall have no content indicating that the referenced node is to be assigned a

NULL value. If, according to the schema of the feature type, the referenced node is not allowed to have a NULL value, the server shall raise an InvalidValue exception (see 7.5).

The wfs:ValueReference element shall contain a path expression that points to a feature property or a child node of a feature property to be modified. The first step of the path expression shall be a valid property name of the feature type specified using the typeName attribute on the wfs:Update element. Subsequent steps shall conform to the XPath encoding rules described in ISO 19143:2010, 7.4.4.

The action attribute on the wfs:ValueReference element controls how the value of the referenced node is updated.

If the action attribute has a value of “insertBefore”, the content of the wfs:Value element shall be inserted into a newly created node before the one pointed to by the content of the wfs:ValueReference element.

If the action attribute has a value of “insertAfter”, the content of the wfs:Value element shall be inserted into a newly created node after the one pointed to by the content of the wfs:ValueReference element.

If the action attribute has a value of “remove”, the node pointed to by the content of the wfs:ValueReference element shall be removed. In this case, the wfs:Value element may be omitted. If, however, it is specified, it shall be ignored when the action attribute is set to “remove”.

If the action attribute has a value of “replace”, the content of wfs:Value element shall replace the current content of the node pointed to by the content of the wfs:ValueReference element.

The value “replace” is the default value of the action attribute.

In all cases, if the specified action results in invalidating the feature instance according to the feature type schema obtained using a DescribeFeatureType operation (see Clause 9), the server shall raise an InvalidValue exception (see 7.5).

15.2.5.2.2 Filter element

A filter expression, encoded using the fes:Filter element (see ISO 19143:2010, Clause 7), can limit the scope of an update action to an enumerated set of features or to a set of features identified using spatial and non-spatial constraints.

In the case where the fes:Filter element does not identify any feature instances to work on, the update action shall have no effect. This is not an exception condition and no exception shall be raised by the WFS.

The full definition of the fes:Filter element is described in Clause 7 of ISO 19143:2010.

15.2.5.2.3 typeName attribute

The value of the typeName attribute shall identify a feature type to be updated. Its value shall be one of the feature types listed in the feature type list found in the web feature service's capabilities document (see 8.3.3).

15.2.5.2.4 Standard input parameters

See 7.6.5 for a description of the standard input parameters.

15.2.6 Replace action

15.2.6.1 XML encoding

The following XML Schema fragment declares the wfs:Replace element:

```
<xsd:element name="Replace" type="wfs:ReplaceType"
     substitutionGroup="wfs:AbstractTransactionAction"/>
<xsd:complexType name="ReplaceType">
   <xsd:complexContent>
      <xsd:extension base="wfs:AbstractTransactionActionType">
         <xsd:sequence>
            <xsd:any namespace="##other"/>
            <xsd:element ref="fes:Filter"/>
         </xsd:sequence>
         <xsd:attributeGroup ref="wfs:StandardInputParameters"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

Unlike the update action which modifies individual feature properties, the replace action replaces an existing feature instance with the one specified in the action. If the server supports versioning, this has the effect of creating a new latest version.

15.2.6.2 Parameter discussion

15.2.6.2.1 Filter expression

A mandatory filter expression (see ISO 19143:2010, Clause 7) shall be used to identify which feature or features shall be replaced with the instance specified within the wfs:Replace element.

15.2.6.2.2 Standard input parameters

See 7.6.5 for a description of the standard input parameters.

15.2.7 Delete action

15.2.7.1 XML encoding

The following XML Schema fragment declares the wfs:Delete element:

```
<xsd:element name="Delete" type="wfs:DeleteType"
     substitutionGroup="wfs:AbstractTransactionAction"/>
<xsd:complexType name="DeleteType">
   <xsd:complexContent>
      <xsd:extension base="wfs:AbstractTransactionActionType">
         <xsd:sequence>
            <xsd:element ref="fes:Filter"/>
         </xsd:sequence>
         <xsd:attribute name="typeName" type="xsd:QName" use="required"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

The wfs:Delete element is used to encode a delete request that removes one or more feature instances, of a specified feature type, from being queryable to a client application using the GetFeature (see Clause 11), GetFeatureWithLock (see Clause 13) and GetPropertyValue (see Clause 10) operations. This may mean that the features are deleted from the data store or that the features are retired, if the data store is a versioning data store.

15.2.7.2 Parameter discussions

15.2.7.2.1 typeName attribute

The value of the typeName attribute shall identify a feature type to be deleted. The value of the typeName parameter shall match one of the feature type names advertised in the server's capabilities document (see 8.3.3).

15.2.7.2.2 Filter expression

The scope of a delete operation shall be constrained by using the fes:Filter element as described in Clause 7 of ISO 19143:2010. In the event that the fes:Filter element does not identify any feature instances to delete, the delete action will simply have no effect. This is not an exception condition.

15.2.8 Native action

It is clear that an open interface may only support a certain common set of capabilities. The wfs:Native element is intended to allow access to vendor-specific capabilities of any particular WFS or data store.

The wfs:Native element is defined by the following XML Schema fragment:

```
<xsd:element name="Native" type="wfs:NativeType"
     substitutionGroup="wfs:AbstractTransactionAction"/>
<xsd:complexType name="NativeType" mixed="true">
   <xsd:complexContent>
      <xsd:extension base="wfs:AbstractTransactionActionType">
         <xsd:sequence>
            <xsd:any processContents="lax" namespace="##other" minOccurs="0"/>
         </xsd:sequence>
         <xsd:attribute name="vendorId" type="xsd:string" use="required"/>
         <xsd:attribute name="safeToIgnore" type="xsd:boolean" use="required"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

The wfs:Native element simply contains the vendor-specific command or operation.

The vendorId attribute shall be used to identify the vendor that recognizes the command or operation enclosed by the wfs:Native element. The attribute is provided as a means of allowing a web feature service to determine if it can deal with the command or not. Valid vendorId values shall be advertised, using a parameter constraint named "vendorId", in the service's capabilities document (see 8.3.3).

The safeToIgnore attribute is used to guide the behaviour of a web feature service when the Native action is not recognized. The safeToIgnore attribute has two possible values: *True* or *False*. The values have the following meanings:

safeToIgnore=False

A value of *False* indicates that the wfs:Native element cannot be ignored and the operation that the element is associated with will fail if the web feature service cannot deal with it.

safeToIgnore=True

A value of *True* indicates that the wfs:Native element may be safely ignored.

EXAMPLE This example illustrates the use of the wfs:Native element to enable a special feature of a SQL-based relational database. In this instance, the element indicates that this is an Oracle command and that the command may be safely ignored.

```
<Native vendorId="Oracle" safeToIgnore="True">
ALTER SESSION ENABLE PARALLEL DML
</Native>
```

15.3 Response

15.3.1 Response semantics

In response to a Transaction request, a web feature service shall generate an XML document indicating the termination status of the transaction. In addition, if the Transaction request includes wfs:Insert elements, then the web feature service shall report the feature identifiers of all newly created features.

Figure 31 describes the response to a Transaction operation.

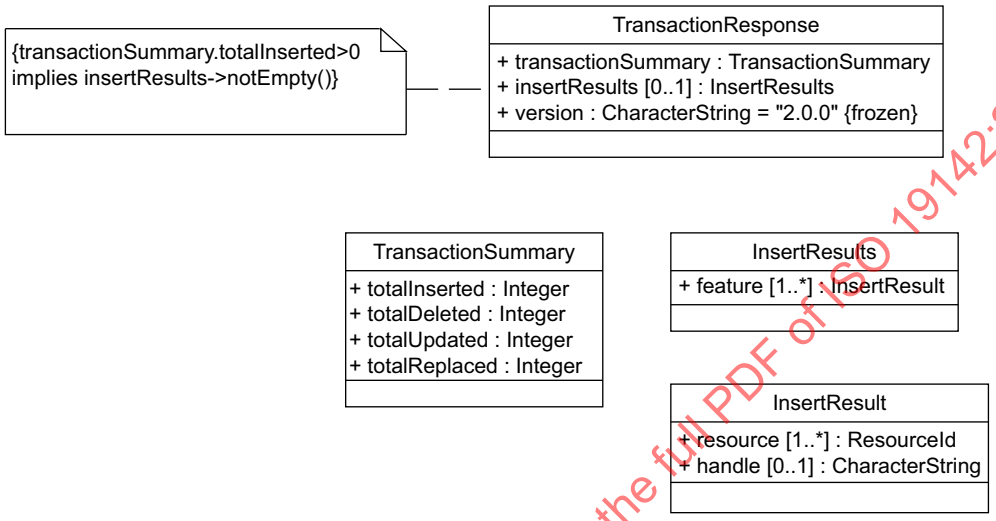

Figure 31 — Transaction response

15.3.2 TransactionResponse element

The root element of the response to a Transaction request is called wfs:TransactionResponse.

The following XML Schema fragment declares the wfs:TransactionResponse element:

```
<xsd:element name="TransactionResponse"
             type="wfs:TransactionResponseType"/>
<xsd:complexType name="TransactionResponseType">
   <xsd:sequence>
      <xsd:element name="TransactionSummary"
                   type="wfs:TransactionSummaryType"/>
      <xsd:element name="InsertResults"
                   type="wfs:ActionResultsType" minOccurs="0"/>
      <xsd:element name="UpdateResults"
                   type="wfs:ActionResultsType" minOccurs="0"/>
      <xsd:element name="ReplaceResults"
                   type="wfs:ActionResultsType" minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="version" type="xsd:string" use="required" fixed="2.0.0"/>
</xsd:complexType>
```

The wfs:TransactionResponse element contains a summary of the actions performed by the transaction and optionally a list of identifiers of any new features or feature versions created by the transaction.

15.3.3 TransactionSummary element

The following XML Schema fragment declares the wfs:TransactionSummary element:

```
<xsd:complexType name="TransactionSummaryType">
   <xsd:sequence>
      <xsd:element name="totalInserted" type="xsd:nonNegativeInteger" minOccurs="0"/>
      <xsd:element name="totalUpdated"  type="xsd:nonNegativeInteger" minOccurs="0"/>
      <xsd:element name="totalReplaced" type="xsd:nonNegativeInteger" minOccurs="0"/>
      <xsd:element name="totalDeleted"  type="xsd:nonNegativeInteger" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

The wfs:TransactionSummary element contains a count of the number of features created (i.e. wfs:totalInserted), modified (i.e. wfs:totalUpdated), replaced (i.e. wfs:totalReplaced) or deleted (i.e. wfs:totalDeleted) by the actions in a Transaction request.

If the transaction does not contain any Insert actions (see 15.2.4), the wfs:totalInserted element shall be omitted.

If the transaction does not contain any Update actions (see 15.2.5), the wfs:totalUpdated element shall be omitted.

If the transaction does not contain any Replace actions (see 15.2.6), the wfs:totalReplaced element shall be omitted.

If the transaction does not contain any Delete actions (see 15.2.7), the wfs:totalDeleted element shall be omitted.

Only features of the types listed in the web feature service's capabilities document shall be counted when generating this summary.

15.3.4 InsertResults element

The following XML Schema fragment declares the wfs:ActionResultsType type, that is the type of the wfs:InsertResults, wfs:UpdateResults and wfs:ReplaceResults elements:

```
<xsd:complexType name="ActionResultsType">
   <xsd:sequence>
      <xsd:element name="Feature"
                   type="wfs:CreatedOrModifiedFeatureType"
                   maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="CreatedOrModifiedFeatureType">
   <xsd:sequence maxOccurs="unbounded">
      <xsd:element ref="fes:ResourceId"/>
   </xsd:sequence>
   <xsd:attribute name="handle" type="xsd:string"/>
</xsd:complexType>
```

If a transaction request contains insert, update and/or replace actions then the wfs:InsertResults, wfs:UpdateResults and/or wfs:ReplaceResults elements shall be specified in the transaction response.

The wfs:InsertResults element contains one or more wfs:Feature elements that indicate the feature identifiers of newly created feature instances. One wfs:Feature element shall be reported for each newly created feature instance.

In addition, the wfs:Feature elements shall be presented in the order in which the insert actions were encountered in the wfs:Transaction element.

Additionally, wfs:Feature elements may be correlated to insert actions using the handle attribute. If a value was specified for the handle attribute on a wfs:Insert element that created a feature, the same handle value may be specified as the value of the handle attribute on the wfs:Feature element to indicate which insert action generated which feature instance.

15.3.5 UpdateResults element

If a transaction request contains update actions and the server supports versioning, then the wfs:UpdateResults element shall be specified in the transaction response to report which features have been changed and what their new identifiers are.

The wfs:UpdateResults element contains one or more wfs:Feature elements that indicate the new feature identifiers of the updated feature instances. One wfs:Feature element shall be reported for each updated feature instance. Each wfs:Feature element shall contain a fes:ResourceId element (see ISO 19143:2010, 7.11) that shall include the rid attribute whose value is the feature identifier of the newly created version of the feature and the oldRid attribute whose value shall be the identifier of the previous version.

In addition, the wfs:Feature elements shall be presented in the order in which the update actions were encountered in the wfs:Transaction element.

Additionally, wfs:Feature elements may be correlated to update actions using the handle attribute. If a value was specified for the handle attribute on a wfs:Update element that updated a feature, the same handle value may be specified as the value of the handle attribute on the wfs:Feature element to indicate which update action changed which feature.

15.3.6 ReplaceResults element

If a transaction request contains replace actions and the server supports versioning, then the wfs:ReplaceResults element shall be specified in the transaction response to report which features have been replaced and what their new identifiers are.

The wfs:ReplaceResults element contains one or more wfs:Feature elements that indicate the new feature identifiers of the replaced feature instances. One wfs:Feature element shall be reported for each replaced feature instance. Each wfs:Feature element shall contain a fes:ResourceId element (see ISO 19143:2010, 7.11) that shall include the rid attribute whose value is the feature identifier of the newly created version of the feature and the oldRid attribute whose value shall be the identifier of the previous version.

In addition, the wfs:Feature elements shall be presented in the order in which the replace actions were encountered in the wfs:Transaction element.

Additionally, wfs:Feature elements may be correlated to replace actions using the handle attribute. If a value was specified for the handle attribute on a wfsReplace element that replaced a feature, the same handle value may be specified as the value of the handle attribute on the wfs:Feature element to indicate which replace action replaced which feature instance.

15.4 Exceptions

In the event that a web feature service encounters an error parsing a Transaction request, it shall raise an OperationParsingFailed exception as described in 7.5.

If a web feature service encounters an error while processing a particular action contained in a Transaction request, then the service shall raise an OperationProcessingFailed exception as described in 7.5.

Annex A
(normative)

Conformance testing

A.1 Conformance classes

A.1.1 Simple WFS

a) Test purpose: Verify that the server implements the Simple WFS conformance class.

b) Test method: Submit requests to the server and verify the following: the capabilities document that the server generates includes the operations GetCapabilities, DescribeFeatureType, ListStoredQueries, DescribeStoredQueries and GetFeature operation with the stored query GetFeatureById; verify that the response to a DescribeFeatureType request is always a complete application schema; verify A.2.22.4; verify that the service supports at least one service binding (see Annex D). Verify the following list of conformance tests: A.2.2, A.2.3, A.2.4, A.2.5, A.2.6.1 and/or A.2.6.2, A.2.7.1 and/or A.2.7.2, A.2.8.1, A.2.9, A.2.14, A.2.15, A.2.16, A.2.17, A.2.19.1, A.2.19.2.1, A.2.21, A.2.22.4. Verify that conformance test ISO 19143:2010, A.1 is satisfied. Verify that conformance tests ISO 19136:2007, A.1.1, A.1.4, A.1.5, A.1.7, B.3, B.5 and B.2.3 are satisfied.

c) References: 7.9.3.6, Clauses 8, 9, 10 and 11 (wfs:StoredQuery action only), 14.3, 14.4

d) Test type: Capability

A.1.2 Basic WFS

a) Test purpose: Verify that the server implements the Basic WFS conformance class.

b) Test method: Verify that the server implements the Simple WFS conformance class. Verify in the capabilities document that the server includes the operations GetPropertyValue and GetFeature. Verify the operation of the GetPropertyValue and GetFeature operation with the stored query action. Verify that the server implements at least the Minimum Spatial Filter conformance class for ISO 19143. Verify the following list of conformance tests: A.2.2, A.2.7, A.2.8.1, A.2.11.2, A.2.12, A.2.13, A.2.19, A.2.20.1, A.2.20.2, A.2.22. For test A.2.23, verify that the value of the constraint ImplementsBasicWFS is set to TRUE. Verify that conformance tests ISO 19143:2010, A.2, A.7, A.8, A.9, A.11, A.12 and A.14 are satisfied. Verify that conformance test ISO 19136:2007, B.4 is satisfied.

c) References: Clauses 10 and 11 (wfs:Query and wfs:StoredQuery actions)

d) Test type: Capability

A.1.3 Transactional WFS

a) Test purpose: Verify that the server implements the Transactional WFS conformance class.

b) Test method: Verify that the server implements the Basic WFS conformance class. Verify in the capabilities document that the server includes the Transaction operation. Verify the operation of the Transaction operation. Verify the following list of conformance tests: A.2.2, A.2.8.2, A.2.10, A.2.11.1, A.2.18. For test A.2.23, verify that the value of the constraint ImplementsTransactionalWFS is set to TRUE.

c) Reference: Clause 15

d) Test type: Capability

A.1.4 Locking WFS

a) Test purpose: Verify that the server implements the Locking WFS conformance class.

b) Test method: Verify that the server implements the Transactional WFS conformance class. Verify in the capabilities document that the server includes the LockFeature or GetFeatureWithLock or both operations. Verify the operation of the listed operations. Verify the following list of conformance tests: A.1.3. For test A.2.23, verify that the value of the constraint ImplementsLockingWFS is set to TRUE.

c) References: Clauses 12 and 13

d) Test type: Capability

A.1.5 HTTP GET

a) Test purpose: Verify that the server implements the HTTP GET conformance class.

b) Test method: Verify that the server implement the KVP encoding for all the operations listed in the OperationsMetadata section of the capabilities document and all the operations that have a KVP encoding defined in this International Standard. Verify the following list of conformance tests: A.2.6.1, A.2.7.1, A.2.13.2. For test A.2.23, verify that the value of the constraint KVPEncoding is set to TRUE.

c) Reference: Annex D

d) Test type: Capability

A.1.6 HTTP POST

a) Test purpose: Verify that the server implements the HTTP POST conformance class.

b) Test method: Verify that the server implements the XML encoding for all the operations listed in the OperationsMetadata section of the capabilities document. Verify the following list of conformance tests: A.2.6.2, A.2.7.2, A.2.13.1. For test A.2.23, verify that the value of the constraint XMLEncoding is set to TRUE.

c) Reference: Annex D

d) Test type: Capability

A.1.7 SOAP

a) Test purpose: Verify that the server implements the SOAP conformance class.

b) Test method: Verify that the server implements the XML encoding for all the operations listed in the OperationsMetadata section of the capabilities document. Verify that the server supports the SOAP service binding. Verify the following list of conformance tests: A.1.2. For test A.2.23, verify that the value of the constraint SOAPEncoding is set to TRUE.

c) Reference: Annex D

d) Test type: Capability

A.1.8 Inheritance

a) Test purpose: Verify that the server implements the Inheritance conformance class.

b) Test method: Verify that the server implements the schema-element() XPath function. For test A.2.23, verify that the value of the constraint ImplementsInheritance is set to TRUE. Verify that conformance test ISO 19143:2010, A.15 is satisfied.

c) Reference: 7.9.2.4.2, A.2.22.1.2

d) Test type: Capability

A.1.9 Remote resolve

a) Test purpose: Verify that the server has the ability to resolve remote references.

b) Test method: See A.2.17.2.3. For test A.2.23, verify that the value of the constraint ImplementsRemoteResolve is set to TRUE. Verify that conformance test ISO 19136:2007, B.2.1 is satisfied.

c) Reference: 7.6.4

d) Test type: Capability

A.1.10 Response paging

a) Test purpose: Verify that the server implements response paging.

b) Test method: See A.2.20. For test A.2.23, verify that the value of the constraint ImplementsResultPaging is set to TRUE. Verify that conformance test ISO 19136:2007, B.3 is satisfied.

c) Reference: 7.7.4.4

d) Test type: Capability

A.1.11 Standard joins

a) Test purpose: Verify that the server implements standard joins.

b) Test method: See A.2.22.2.1. For test A.2.23, verify that the value of the constraint ImplementsStandardJoins is set to TRUE. Verify that conformance tests ISO 19143:2010, A.8 and A.9 are satisfied.

c) Reference: 7.9.2.5.3.1

d) Test type: Capability

A.1.12 Spatial joins

a) Test purpose: Verify that the server implements spatial joins.

b) Test method: See A.2.22.2.2. For test A.2.23, verify that the value of the constraint ImplementsSpatialJoins is set to TRUE. Verify that conformance tests ISO 19143:2010, A.11 and A.12 are satisfied.

c) Reference: 7.9.2.5.3.1

d) Test type: Capability

A.1.13 Temporal joins

a) Test purpose: Verify that the server implements temporal joins

b) Test method: See A.2.22.2.3. For test A.2.23, verify that the value of the constraint ImplementsTemporalJoins is set to TRUE. Verify that conformance tests ISO 19143:2010, A.9 and A.10 are satisfied.

c) Reference: 7.9.2.5.3.1

d) Test type: Capability

A.1.14 Feature versions

a) Test purpose: Verify that the server implements the Feature versions conformance class.

b) Test method: See A.2.11. For test A.2.23, verify that the value of the constraint ImplementsFeatureVersioning is set to TRUE. Verify that conformance test ISO 19143:2010, A.11 is satisfied.

c) Reference: ISO 19143:2010, 7.11

d) Test type: Capability

A.1.15 Manage stored queries

a) Test purpose: Verify that the server implements the Manage stored queries conformance class.

b) Test method: Verify that the server implements the Basic WFS conformance class. Verify in the capabilities document that the server includes the CreateStoredQuery and the DropStoredQuery operations. Submit requests to the server to verify the operation of the CreateStoredQuery and the DropStoredQuery operations. For test A.2.23, verify that the value of the constraint ManageStoredQueries is set to TRUE. Verify that conformance test ISO 19143:2010, A.1 is satisfied.

c) References: 14.4, 14.5

d) Test type: Capability

A.2 Basic tests

A.2.1 Version negotiation

a) Test purpose: To verify that the server correctly handles version negotiation and that the server supports the specific version "2.0.0".

b) Test method: Make a GetCapabilities request with the version parameter set to "2.0.0" and verify that the response is a valid capabilities document as described in this International Sandard (see 8.3).

c) References: 6.2.1, conformance test OGC 06-121r3:2009, A.4.2.3

d) Test type: Basic

A.2.2 Lists version number 2.0.0 as a supported request version number

a) Test purpose: To verify that the server lists the version number "2.0.0" as a supported request version in its capabilities document.

b) Test method: Execute a GetCapabilities request and verify that the version number "2.0.0" is listed as one of the items in the content of the ows:ServiceTypeVersion element.

c) Reference: 6.2.2

d) Test type: Basic

A.2.3 Invalid version number

a) Test purpose: To verify that a request, other than a GetCapabilities request, with the version number set to one that the server does not claim to support in its capabilities document fails.

b) Test method: Review the response to the GetCapabilites request and determine which request version(s) the server claims to support. Execute one or more WFS requests with a version that is not in the list of supported versions and verify that the server generates an InvalidParameterValue exception.

c) Reference: 6.2.2

d) Test type: Basic

A.2.4 Version negotiation for the GetCapabilities request

a) Test purpose: To verify that the server correctly handles version negotiation for the GetCapabilities operation.

b) Test method: Verify that the server conforms to the test described in OGC 06-121r3.

c) References: 6.2.3, OGC 06-121r3:2009, A.4.2.3

d) Test type: Basic

A.2.5 Response to XML- and KVP-encoded requests

a) Test purpose: To verify that the response to a request is identical, regardless of its encoding.

b) Test method: Verify that the server supports both XML- and KVP-encoded requests. Pick a selection of requests and encode then using both XML and KVP encodings and verify that the response for each corresponding set of XML- and KVP-encoded requests is the same.

c) Reference: 6.2.4

d) Test type: Basic

A.2.6 Parameter ordering and case

A.2.6.1 KVP-encoded requests

a) Test purpose: To verify that the order and case of parameters in KVP-encoded requests does not affect the response.

b) Test method: Pick a selection of KVP-encoded requests and invoke them with the parameters specified in a different order each time and with the mixed-case parameter names. Verify that the server's response is unaffected in each case.

c) Reference: 6.2.5.2

d) Test type: Basic

A.2.6.2 XML-encoded requests

Parameter ordering and case does not apply to XML-encoded requests which are encoded according to a fixed schema.

A.2.7 Unrecognized parameters

A.2.7.1 KVP-encoded requests

a) Test purpose: To verify that the server ignores any unrecognized parameters in a KVP-encoded request.

b) Test method: Generate a selection of valid KVP-encoded requests and add one or more parameters to the request that are not defined in this International Standard. Ensure that these additional parameters are not VSPs declared in the server's capabilities document. Verify that the server responds to the request, thus ignoring the additional unrecognized parameters.

c) Reference: 6.2.5.2

d) Test type: Basic

A.2.7.2 XML-encoded requests

Unrecognized parameters do not apply to XML-encoded requests which are encoded according to a fixed schema.

A.2.8 Server operates on GML features

A.2.8.1 Server generates GML features

a) Test purpose: To verify that the server can generate valid GML features.

b) Test method: Select a request whose response contains features (GetFeature, GetFeatureWithLock or GetPropertyValue). Set the outputFormat parameter to "application/gml+xml; version=3.2". Verify that the response is valid relative to the application schema that the server claims to support, thus verifying that the generated features are valid GML features.

c) Reference: 7.1

d) Test type: Basic

A.2.8.2 Server ingests GML features

a) Test purpose: To verify that the server can ingest features encoded using GML.

b) Test method: Generate a transaction that creates a new feature with the inputFormat parameter set to "application/gml+xml; version=3.2". Verify that the operation executes successfully.

c) Reference: 7.1

d) Test type: Basic

A.2.9 Feature identifiers

a) Test purpose: To verify that the server assigns a unique persistent identifier.

b) Test method: Create a new feature instance and verify that an identifier is assigned to that feature in the transaction response. Using that identifier, query to server to retrieve the feature. Verify that only a single feature is returned in the response that corresponds to the queried identifier. Verify that the identifier is encoded in the response using the gml:id attribute.

c) References: 7.2.1, 7.2.2

d) Test type: Basic

A.2.10 Invariant identifier

a) Test purpose: To verify that feature identifiers are invariant under WFS operations.

b) Test method: Update a number of features and verify that the identifier has not changed as a result of the operation. Delete a number of features and then create one or more new features. Verify that the identifiers for the deleted features are not reused for the newly created features.

c) Reference: 7.2.1

d) Test type: Basic

A.2.11 Versioning

A.2.11.1 Version creation

a) Test purpose: To verify that the server creates new feature versions when creating and modifying features.

b) Test method: Create a new feature. Modify the feature several times to force the creation of several versions of the feature. Verify that the feature versions have been created using the A.2.11.2 test.

c) Reference: 7.2.3

d) Test type: Basic

A.2.11.2 Version navigation

a) Test purpose: To verify that the server maintains version information for versioned features and can use that version information to navigate the feature versions when executing a query.

b) Test method: Select one or more versions of a feature using predicates that use the version navigation controls (see ISO 19143:2010, 7.11).

c) Reference: 7.2.3

d) Test type: Basic

A.2.12 XPath subset

a) Test purpose: Verify that the server supports the required XPath subset.

b) Test method: See test A.14 in ISO 19143:2010.

c) References: 7.3.1, 7.3.2

d) Test type: Basic

A.2.13 Predicate encoding

A.2.13.1 XML-encoded requests

a) Test purpose: Verify that the server can execute XML-encoded operations with predicates that are encoded using fes:Filter.

b) Test method: Pick an operation that uses a predicate. Encode the predicate for that operation using the fes:Filter element and verify that the operation executes successfully.

c) Reference: 7.4

d) Test type: Basic

A.2.13.2 KVP-encoded requests

a) Test purpose: Verify that the server can execute KVP-encoded operations with predicates that are encoded using the KVP parameters for encoding predicates.

b) Test method: Pick an operation that uses a predicate. Encode and execute several operations using the KVP parameters for encoding predicates.

c) Reference: 7.4

d) Test type: Basic

A.2.14 Exception reporting

A.2.14.1 Exception report validity

a) Test purpose: To verify that the exception reports that the server generates validate, according to the schema defined in Clause 8 of OGC 06-1212r3:2009.

b) Test method: Devise and execute a request that generates an error. Verify that the exception report that the server generates is valid.

c) Reference: 7.5

d) Test type: Basic

A.2.14.2 Exception report appropriateness

a) Test purpose: Verify that the server generates an appropriate exception report by setting the value of the code and locator parameters to an appropriate value.

b) Test method: Devise a series of requests that generate an error for each error code in Table 25 of OGC 06-121r3:2009 and each error code in Table 3. Verify that the server generates an appropriate exception report for each case by verifying that the code and locator parameters have been set to the correct value.

c) Reference: 7.5

d) Test type: Basic

A.2.14.3 Exception report version

a) Test purpose: Verify that the version parameter of the exception reports that a server generates is set to "2.0.0".

b) Test method: Devise and execute a request that generates an exception and verify that the version parameter of the resulting exception report is set to "2.0.0".

c) Reference: 7.5

d) Test type: Basic

A.2.15 Common request parameters

A.2.15.1 Service and version parameters

a) Test purpose: Verify that the server correctly handles the service and version parameters.

b) Test method: Devise a set of requests that lack the service parameter and the version parameter. Verify that the server responds with a MissingParameterValue exception and the locator parameter names the missing parameter.

c) References: 7.6.2.4, 7.6.2.5

d) Test type: Basic

A.2.15.2 Handle parameter

a) Test purpose: Verify that the server correctly reports the value of the handle parameter in an exception report.

b) Test method: Devise a request that includes multiple actions (e.g. GetFeature or Transaction) and include the handle parameter on each action. Arrange for one of the actions to fail and verify that the exception report uses the value of the handle attribute as the value of the location parameter in the exception response to indicate which action failed.

c) Reference: 7.6.2.6

d) Test type: Basic

A.2.16 Standard presentation parameters

A.2.16.1 startIndex parameter

a) Test purpose: Verify that the server correctly handles the startIndex parameter.

b) Test method: Devise and execute a GetFeature request without the startIndex parameter and note the response. Execute the same GetFeature request a second time including the startIndex parameter. Verify that the response document begins at the feature whose index is the value of the startIndex.

c) Reference: 7.6.3.4

d) Test type: Basic

A.2.16.2 count parameter

A.2.16.2.1 Processing

a) Test purpose: Verify that the server correctly handles the count parameter.

b) Test method: Devise and execute a query and note the number of features in the response. Execute the same query, including a count parameter whose value is less than the number of features in the response. Verify that the response now includes only count records.

c) Reference: 7.6.3.5

d) Test type: Basic

A.2.16.2.2 Configured default

a) Test purpose: Verify that the server correctly handles a configured default value for the count parameter.

b) Test method: Generate a capabilities document and determine the default value for the count parameter. Execute a query that generates more features than this value and verify that the response contains only as many features as the configured default count value.

c) Reference: 7.6.3.5

d) Test type: Basic

A.2.16.3 resultType parameter

a) Test purpose: Verify that the server correctly handles the resultType parameter.

b) Test method: Devise and execute a query with the resultType parameter set to "results". Verify that the server generates a feature collection that includes features as its content. Execute the same query with the resultType parameter set to "hits". Verify that the server generates an empty feature collection with the numberReturned parameter in the response set to zero and the numberMatched parameter in the response set to the count of the number of features the query would return if the resultType parameter was set to "results".

c) Reference: 7.6.3.6

d) Test type: Basic

A.2.16.4 outputFormat parameter

a) Test purpose: Verify that the server implements the mandatory outputFormat value.

b) Test method: Execute requests that accept the outputFormat parameter with the value set to "application/gml+xml; version=3.2" and verify that the output is either a valid GML 3.2 application schema or validates against a GML 3.2 application schema.

c) Reference: 7.6.3.7

d) Test type: Basic

A.2.17 Standard resolve parameters

A.2.17.1 Declaring support for remote resource resolution

a) Test purpose: Verify that the server advertises its remote resolve capability.

b) Test method: Generate a capabilities document and verify that the there are ows:Parameter elements advertising which values from the domain of the resolve parameters (i.e. none, local, remote, all) the server supports.

c) Reference: 7.6.4.4

d) Test type: Basic

A.2.17.2 Resolve parameter processing

A.2.17.2.1 No resource resolution

a) Test purpose: Verify that, if the value of the resolve parameter is set to "none", no resource resolution is performed in the response.

b) Test method: Devise a data set that contains references to local resources. Execute a query with the value of the resolve parameter set to "none" and verify that the resource references are not resolved in the response.

c) Reference: 7.6.4.4

d) Test type: Basic

A.2.17.2.2 Local resource resolution

a) Test purpose: Verify that, if the value of the resolve parameter is set to "local", the local resource resolution is performed in the response.

b) Test method: Devise a dataset that includes features with local resource references. Execute a query with the value of the resolve parameter set to "local" and verify that the local resource references have been resolved in the response.

c) Reference: 7.6.4.4

d) Test type: Basic

A.2.17.2.3 Remote resource resolution

A.2.17.2.3.1 Declare ability to resolve remote references

a) Test purpose: Verify that the server advertises the ability to resolve remote references.

b) Test method: Generate a capabilities document and verify that the ImplementsRemoteResolve service constraint is set to true.

c) Reference: 7.6.4.4

d) Test type: Basic

A.2.17.2.3.2 Remote resource resolution

a) Test purpose: Verify that, if the value of resolve parameter is set to "remote", the remote resource resolution is performed in the response.

b) Test method: Devise a dataset that includes features with remote resource references. Execute a query with the value of the resolve parameter set to "remote" and verify that the remote resource references have been resolved in the response.

c) Reference: 7.6.4.4

d) Test type: Basic

A.2.17.2.3.3 Resolve all resource references

a) Test purpose: Verify that, if the value of the resolve parameter is set to "all", the resource resolution is performed on all references in the response.

b) Test method: Devise a dataset with features that include both local and remote resource references. Execute a query with the value of the resolve parameter set to "all" and verify that all resource references have been resolved in the response.

c) Reference: 7.6.4.4

d) Test type: Basic

A.2.17.3 Resolve depth processing

A.2.17.3.1 Local resources

a) Test purpose: Verify that, when resource resolution is performed, it is performed to the number of levels specified by the resolveDepth parameter for local resource references.

b) Test method: Devise a dataset that includes features that reference local resources that, in turn, reference other local resources to various levels of recursion and also includes a set of features that have a circular reference chain. Execute a set of queries, with the value of the resolveDepth parameter set to various values, and verify that the resource resolution is performed in the response to the specified depth. Ensure that the resolveDepth values of "0", "1" and "*" are tested to verify that no resolutions, only immediately referenced local resources and all referenced local resources, are resolved in the response. Also verify that the circular reference chain has been correctly resolved.

c) Reference: 7.6.4.5

d) Test type: Basic

A.2.17.3.2 Remote resources

a) Test purpose: Verify that, when resource resolution is performed, it is performed to the number of levels specified by the resolveDepth parameter for local remote resource references.

b) Test method: Devise a database that includes features that reference remote resources that, in turn, reference other remote resources to various levels of recursion and also includes a set of features that have a circular reference chain. Execute a set of queries, with the value of the resolveDepth set to various values, and verify that the resource resolution has been performed in the resolve to the specified depth. Ensure that the resolveDepth values of "0", "1" and "*" are tested to verify that no resolutions, only immediately referenced remote resources and all referenced remote resources, are resolved in the response. Also verify that the circular reference chain has been correctly resolved.

c) Reference: 7.6.4.5

d) Test type: Basic

A.2.17.3.3 All resources

a) Test purpose: Verify that, when resource resolution is performed, it is performed to the number of levels specified by the resolveDepth parameter for all resource references.

b) Test method: Devise a database that includes features that reference a mix of local and remote resources that, in turn, reference other local or remote resources to various levels of recursion and also includes a set of features that have a circular reference chain. Execute a set of queries, with the value of the resolveDepth set to various values and verify that resource resolution has been performed in the response to the specified depth. Ensure that the resolveDepth values of "0", "1" and "*" are tested to verify that no resolution, only immediately referenced resources and all referenced resources are resolved in the response. Also verify that the circular reference chain has been correctly resolved.

c) Reference: 7.6.4.5

d) Test type: Basic

A.2.17.3.4 Configured defaults

a) Test purpose: Verify that the server behaves correctly if there is a configured default value for the resolveDepth parameter for local and remote resource resolution.

b) Test method: Generate a capabilities document and obtain the values of the ResolveLocalScope and ResolveRemoteScope operation constraints. If a value is configured for the ResolveLocalScope constraint, rerun test A.2.17.3.1 without the resolveDepth parameter set so that the default value is used. If a value is configured for the ResolveRemoteScope constraint, rerun test A.2.17.3.2 without the resolveDepth parameter set so that the default is used. If both values are configures then rerun tests A.2.17.3.1, A.2.17.3.2, A.2.17.3.3 without the resolveDepth parameter so that the default is set.

c) Reference: 7.6.4.5

d) Test type: Basic

A.2.17.4 Resolve timeout processing

A.2.17.4.1 Processing

a) Test purpose: Verify that the server stops trying to resolve a resource after the resolveTimeout period has elapsed.

b) Test method: Devise a dataset with a single feature that references a non-existent resource. Execute a query with the resolve parameter and the resolveTimeout parameter set and verify that the server responds after the resolveTimeout has elapsed.

c) Reference: 7.6.4.6

d) Test type: Basic

A.2.17.4.2 Configured default

a) Test purpose: Verify the server's behaviour when there is a configured resolveTimeout parameter.

b) Test method: Generate a capabilities document to determine if a default timeout value is configured by looking for the ResolveTimeoutDefault constraint. If a value is configured then rerun test A.2.17.4.1. without setting the resolveTimeout parameter so that the configured value is used.

c) Reference: 7.6.4.6

d) Test type: Basic

A.2.17.5 Unable to resolve resource reference

a) Test purpose: Verify that the server reports the original unresolved URI in the response if it cannot resolve the resource.

b) Test method: Devise a dataset with a single feature that references a non-existent resource. Execute a query with it, and verify that, in the response, the reference is the original unresolved URI.

c) Reference: 7.6.4.7

d) Test type: Basic

A.2.18 Standard input parameters

A.2.18.1 inputFormat parameter

a) Test purpose: Verify that the server correctly handles the inputFormat parameter when its value is "application/gml+xml; version=3.2".

b) Test method: Create a transaction with an insert action having the inputFormat set to "application/gml+xml; version=3.2" and the content of the action being an invalid feature. The server should detect the invalid feature encoding and raise an exception.

c) Reference: 7.6.5.4

d) Test type: Basic

A.2.18.2 srsName parameter

a) Test purpose: Verify that the server correctly handles the srsName parameter on input.

b) Test method: Insert a feature with geometries where the value of the srsName is one of the SRS values the server claims to support in its capabilities document. Verify that the operation succeeds. Insert another feature with geometries where the value of the srsName parameter is not one of the SRS values the server claims to support in its capabilities document. Verify that the server responds with an exception. Insert a final feature with geometries where the value of the srsName is one of the SRS values the server claims to support in its capabilities, but the geometries in the insert action are encoded with an SRS that does not match the one asserted as the value of the srsName parameter. Verify that the server responds with an exception.

c) Reference: 7.6.5.5

d) Test type: Basic

A.2.19 Standard response parameters

A.2.19.1 timeStamp parameter

a) Test purpose: Verify that the server populates the timeStamp parameter.

b) Test method: Have the server generate a feature collection response and verify that the timeStamp parameter is present and populated with a valid xsd:dateTime value.

c) Reference: 7.7.4.1

d) Test type: Basic

A.2.19.2 numberMatched parameter

A.2.19.2.1 Standard processing

a) Test purpose: Verify that the server populates the numberMatched parameter.

b) Test method: Generate a feature collection response and verify that the numberMatched parameter is populated with a positive integer or the value "unavailable".

c) Reference: 7.7.4.2

d) Test type: Basic

A.2.19.2.2 Processing with the resultType parameter

A.2.19.2.2.1 Non-paging response

a) Test purpose: Verify that the numberMatched parameter is correctly populated when the resultType parameter is set to "hits" for a server that does not support response paging.

b) Test method: Generate a feature collection response with a request having the resultType parameter set to "hits". Verify that the response is an empty feature collection with the numberMatched parameter containing a count of the number of features expected in the response.

c) Reference: 7.7.4.2

d) Test type: Basic

A.2.19.2.2.2 Paging response

a) Test purpose: Verify that the numberMatched parameter is correctly populated when the resultType parameter is set to "hits" for a server that does support response paging.

b) Test method: Generate a feature collection response with a request having the resultType parameter set to "hits". Verify that the response is an empty feature collection with the numberMatched parameter containing a count of the number of features expected in the response. Verify that the next parameter is present in the response and its value is a URI that fetches the first set of results.

c) Reference: 7.7.4.2

d) Test type: Basic

A.2.20 Response paging

A.2.20.1 Declaring support to response paging

a) Test purpose: Verify that the server advertises its ability to support response paging.

b) Test method: Generate a capabilities document and verify that the ImplementsResponsePaging service constraint is set.

c) Reference: 7.7.4.4.1

d) Test type: Basic

A.2.20.2 Processing

a) Test purpose: Verify that the server behaves correctly when paging through a response.

b) Test method: Execute a query that selects a set of features and set the count parameter to some small value to ensure that the entire response will not be presented in less than three response documents. Verify that the first response document contains a next parameter populated with a URI. Resolve the URI to retrieve the next page of the response. Verify that the second page has both the next and previous parameters populated. Resolve the URI that is the value of the next parameter. Verify that the last page in the response contains only the previous parameter. Resolve the URI that is the value of the previous parameter and verify that the response is the second page.

c) Reference: 7.7.4.4.1

d) Test type: Basic

A.2.20.3 Transactional consistency

A.2.20.3.1 Declaring transactional consistency

a) Test purpose: Verify that the server advertises if its response paging support is transactionally consistent or not.

b) Test method: Generate a capabilities document and verify that the PagingIsTransactionSafe operation constraint is present.

c) Reference: 7.7.4.4.2.1

d) Test type: Basic

A.2.20.3.2 Response paging is transaction safe

a) Test purpose: Verify that response paging is transaction safe.

b) Test method: Execute the test described in A.2.20.2, except that, between fetching the next or previous page, execute, in another context, a transaction that adds or removes records from the result set. Verify that no records are added or removed from the response pages.

c) Reference: 7.7.4.4.2.2

d) Test type: Basic

A.2.20.3.3 Response paging is not transaction safe

a) Test purpose: Verify that response paging is not transaction safe.

b) Test method: Execute the test described in A.2.20.2, except that, between fetching the next or previous page, execute, in another context, a transaction that adds or removes records from the result set. Verify that records have been added or removed from the response pages.

c) Reference: 7.7.4.4.2.3

d) Test type: Basic

A.2.21 schemaLocation parameter

a) Test purpose: Verify that the schemaLocation attribute is used in a feature collection response to aid its response validation.

b) Test method: Generate a feature collection and verify that the schemaLocation attribute is present and its value includes a reference to schemas that can be used to validate the features in the collection.

c) Reference: 7.8

d) Test type: Basic

A.2.22 Query expressions

A.2.22.1 Adhoc query expressions

A.2.22.1.1 typeNames parameter

a) Test purpose: Verify the value domain of the typeNames parameter.

b) Test method: From the server's capabilities document, obtain the list of feature types that the server offers. Encode a request that references a feature type that the server does not offer and verify that the response is an InvalidParameterValue exception report.

c) Reference: 7.9.2.4.1

d) Test type: Basic

A.2.22.1.2 schema-element() function

a) Test purpose: Verify that the server that supports the schema-element() function advertises it in its capabilities document.

b) Test method: Generate a capabilities document and verify that the ImplementsInheritance service constraint is present and its value is "TRUE".

c) Reference: 7.9.2.4.2

d) Test type: Basic

A.2.22.1.3 aliases parameter

a) Test purpose: Verify the correct handling of the aliases parameter.

b) Test method: Encode a query that includes the aliases parameter and where the number of aliases does not match the number of feature types listed in the typeNames parameter. Verify that the server raises an InvalidParameterValue exception.

c) Reference: 7.9.2.4.3

d) Test type: Basic

A.2.22.1.4 srsName parameter

a) Test purpose: Verify the correct handling of the srsName parameter.

b) Test method: Generate a capabilities document and take notes of the default SRS and any other SRSs that the server supports. Execute a query without the srsName parameter and verify that the geometries in the response document are encoded using the default SRS. Select one or more SRSs from the list of supported SRSs to execute the query and verify that the geometries have been correctly encoded in the target SRS. Select an SRS that is not the default SRS, or any other SRS that the server claims to support and execute the query again. Verify that the server generates an InvalidParameterValue exception.

c) Reference: 7.9.2.4.4

d) Test type: Basic

A.2.22.1.5 Projection clause

A.2.22.1.5.1 Selecting optional properties

a) Test purpose: Verify that a feature collection response contains only the optional feature properties specified by the projections clause.

b) Test method: Execute a query that includes a projections clause and verify that the response includes only the optional properties selected by the projection clause.

c) Reference: 7.9.2.4.5.1

d) Test type: Basic

A.2.22.1.5.2 Invalid property name

a) Test purpose: Verify that the server generates an exception if an invalid property name is specified in the projections clause.

b) Test method: Execute a query with a non-existent property name in the projection clause of a query and verify that the server generates an InvalidParameterValue exception.

c) Reference: 7.9.2.4.6.1

d) Test type: Basic

A.2.22.1.5.3 resolvePath parameter

a) Test purpose: Verify the behaviour of the resolvePath parameter.

b) Test method: Execute a query that resolves resource references and specify the resolvePath parameter with an appropriate value. Verify that one resource along the resolve path has been verified in the response. Verify that the behaviour for local resources and remote resources in the server claims to support remote resource resolution.

c) Reference: 7.9.2.4.7

d) Test type: Basic

A.2.22.2 Selection clause

A.2.22.2.1 Standard join

A.2.22.2.1.1 Declaring support for standard join

a) Test purpose: Verify that the server correctly advertises and provides support for standard joins.

b) Test method: Generate a capabilities document and verify that the value of the ImplementsStandardJoin service constraint is set to "TRUE".

c) Reference: 7.9.2.5.3.1

d) Test type: Basic

A.2.22.2.1.2 Processing

a) Test purpose: Verify that the server correctly processes a standard join.

b) Test method: Execute a query that fetches data from at least two feature types and includes a join predicate that does not involve spatial or temporal operators. Verify that the response generates a valid response using the wfs:Tuple element to contain each feature tuple.

c) Reference: 7.9.2.5.3.1

d) Test type: Basic

A.2.22.2.2 Spatial join

A.2.22.2.2.1 Declaring support for spatial join

a) Test purpose: Verify that the server correctly advertises and provides support for spatial joins.

b) Test method: Generate a capabilities document and verify that the value of the ImplementsSpatialJoin service constraint is set to "TRUE".

c) Reference: 7.9.2.5.3.1

d) Test type: Basic

A.2.22.2.2.2 Processing

a) Test purpose: Verify that the server correctly processes a spatial join.

b) Test method: Execute a query that fetches data from at least two feature types and includes a join predicate that uses a spatial operator. Verify that the response generates a valid response using the wfs:Tuple element to contain each feature tuple.

c) Reference: 7.9.2.5.3.1

d) Test type: Basic

A.2.22.2.3 Temporal join

A.2.22.2.3.1 Declaring support for temporal join

a) Test purpose: Verify that the server correctly advertises and provides support for temporal join.

b) Test method: Generate a capabilities document and verify that the value of the ImplementsTemporalJoin service constraint is set to "TRUE".

c) Reference: 7.9.2.5.3.1

d) Test type: Basic

A.2.22.2.3.2 Processing

a) Test purpose: Verify that the server correctly processes a temporal join.

b) Test method: Execute a query that fetches data from at least two feature types and includes a join predicate that uses a temporal operator. Verify that the response generates a valid response using the wfs:Tuple element to contain each feature tuple.

c) Reference: 7.9.2.5.3.1

d) Test type: Basic

A.2.22.3 Sorting clause

A.2.22.3.1 Basic sorting

a) Test purpose: Verify the correct sorting behaviour.

b) Test method: Execute a query that includes a sorting clause and verify that the features in the response are sorted by the specified property or properties and in the specified sort order.

c) Reference: 7.9.2.5.4.4

d) Test type: Basic

A.2.22.3.2 Default order

a) Test purpose: Verify that the server returns features in the same default order.

b) Test method: Execute the same query on the same set of features repeatedly and verify that the features are presented in the same order each time. The specific order is not relevant; all that is relevant is that whatever order is presented is preserved across all invocations of the same query on the same set of features.

c) Reference: 7.9.2.5.4.4

d) Test type: Basic

A.2.22.3.3 Sorting level

a) Test purpose: Verify that the server honours any SortLevelLimit that it might advertise in its capabilities document.

b) Test method: Execute a query with a sorting clause that contains one more sort property that the server advertises that it supports in its capabilities document using the SortLevelLimit constraint. Verify that the server generates an exception.

c) Reference: Table 15

d) Test type: Basic

A.2.22.4 Stored queries

A.2.22.4.1 Stored query operations

a) Test purpose: Verify that the server implements the required stored query operations.

b) Test method: Generate a capabilities document and verify that the server implements the ListStoredQueries and DescribeStoredQueries operations. Using the ListStoredQueries operation, verify that the response lists a stored query with the identifier urn:ogc:def:query:OGC-WFS::GetFeatureById.

c) References: 7.9.3.6, 8.2, 14.3, 14.4

d) Test type: Basic

A.2.22.4.2 Returns feature types

a) Test purpose: Verify that all the return feature types for stored queries are also features listed in the feature type list.

b) Test method: Generate a capabilities document and get the list of feature types that the server offers. Generate a listing of all the stored queries that the server offers and verify that their return feature types are listed in the feature type list.

c) References: 7.9.3, 8.1, 14.3

d) Test type: Basic

A.2.23 Declaring conformance

a) Test purpose: Verify that the server declares the capabilities it implements in its capabilities document.

b) Test method: Generate a capabilities document and verify that all the service constraints listed in Table 13 are present in the capabilities document and have a value of "TRUE" or "FALSE", indicating that the server conforms to the corresponding class or not.

c) Reference: Table 13

d) Test type: Basic

Annex B
(informative)

Examples

B.1 Exception report example

The following is an example of an exception report. In this particular case, a CreateStoredQuery operation has failed because of a duplicate query identifier.

```
<?xml version="1.0" ?>
<ExceptionReport
   version="2.0.0"
   xmlns="http://www.opengis.net/ows/1.1"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/ows/1.1
                       http://schemas.opengis.net/ows/1.1.0/owsAll.xsd">
   <Exception exceptionCode="DuplicateStoredQueryIdValue" locator="FeatureInPolygon">
      <ExceptionText>The identifier urn:SomeServer:StoredQueries:FeaturesInPolygon has already been
        assigned to a stored query.</ExceptionText>
   </Exception>
</ExceptionReport>
```

B.2 DescribeFeatureType examples

B.2.1 Example 1

Consider geographic features of types *TreesA_1M, RoadL_1M* and *LakesA*_1M that are defined in a SQL database. The description of these feature types is reported by the database to be:

```
SQL> describe TREESA_1M
Name              Null?        Type
----------------------   --------   -------------
EXTENT                   NOT NULL     POLYGON
ID                                    NUMBER(10)
TREE_TYPE                             VARCHAR2(80)

SQL> describe ROADL_1M
Name                      Null?    Type
------------------------- -------- ------------
CENTERLINE                NOT NULL LINE
DESIGNATION                        VARCHAR2(30)
SURFACE_TYPE                       VARCHAR2(30)
NLANES                             NUMBER(2)

SQL> describe LAKESA_1M
Name                      Null?    Type
------------------------- -------- ------------
EXTENT                    NOT NULL POLYGON
NAME                           VARCHAR2(30)
AVG_DEPTH                          NUMBER
MAX_DEPTH                          NUMBER
```

In response to the DescribeFeatureType request:

```
<?xml version="1.0" ?>
<DescribeFeatureType
   service="WFS"
   version="2.0.0"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:myns="http://www.myserver.com/myns"
   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">
   <TypeName>myns:TreesA_1M</TypeName>
   <TypeName>myns:RoadL_1M</TypeName>
</DescribeFeatureType>
```

A web feature service might generate the following XML Schema document:

```
<?xml version="1.0" ?>
<schema
   targetNamespace="http://www.someserver.com/myns"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://www.w3.org/2001/XMLSchema"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   elementFormDefault="qualified" version="2.0.0">
   <import namespace="http://www.opengis.net/gml/3.2"
           schemaLocation="http://schemas.opengis.net/gml/3.2.1/gml.xsd"/>

   <!-- ============================================
        define global elements
        ============================================ -->
   <element name="TreesA_1M"
            type="myns:TreesA_1M_Type"
            substitutionGroup="gml:AbstractFeature"/>
   <element name="RoadL_1M"
            type="myns:RoadL_1M_Type"
            substitutionGroup="gml:AbstractFeature"/>
   <element name="LakesA_1M"
            type="myns:LakesA_1M_Type"
            substitutionGroup="gml:AbstractFeature"/>

   <!-- ============================================
        define complex types (classes)
        ============================================ -->
   <complexType name="TreesA_1M_Type">
      <complexContent>
         <extension base="gml:AbstractFeatureType">
            <sequence>
               <element name="extent"
                        type="gml:SurfacePropertyType" nillable="false"/>
               <element name="id" nillable="true" minOccurs="0">
                  <simpleType>
                     <restriction base="integer">
                        <totalDigits value="10"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="treeType" nillable="true" minOccurs="0">
                  <simpleType>
                     <restriction base="string">
                        <maxLength value="80"/>
                     </restriction>
                  </simpleType>
               </element>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="RoadL_1M_Type">
      <complexContent>
         <extension base="gml:AbstractFeatureType">
            <sequence>
               <element name="centerLine"
                        type="gml:CurvePropertyType"
                        nillable="false"/>
               <element name="designation" nillable="true" minOccurs="0">
                  <simpleType>
                     <restriction base="string">
                        <maxLength value="30"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="surfaceType" nillable="true" minOccurs="0">
                  <simpleType>
                     <restriction base="string">
```

```
                        <maxLength value="30"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="nLanes" nillable="true" minOccurs="0">
                  <simpleType>
                     <restriction base="integer">
                        <totalDigits value="2"/>
                     </restriction>
                  </simpleType>
               </element>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="LakesA_1M_Type">
      <complexContent>
         <extension base="gml:AbstractFeatureType">
            <sequence>
               <element name="extent" type="gml:SurfacePropertyType"/>
               <element name="name" type="xsd:string"/>
               <element name="waterType" type="xsd:string" minOccurs="0"/>
               <element name="avgDepth"
                        type="gml:MeasureType" minOccurs="0"/>
               <element name="maxDepth"
                        type="gml:MeasureType" minOccurs="0"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
</schema>
```

Notice that the response includes the requested feature TreesA_1M and RoadL_1M but also includes all the other features in the application schema (in this example, LakesA_1M). In other words, the server generates the complete application schema.

Using this schema description, a client could then express the state of a TreesA_1M feature instance and/or a *RoadL_1M* feature instance as shown in the following example:

```
<?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
                       ./DescribeFeatureType_Example01_Response.xsd">
   <wfs:member>
      <TreesA_1M gml:id="TRESSA_1M.1013">
         <extent>
            <gml:Polygon gml:id="GID_10" srsName="urn:ogc:def:crs:EPSG::4326">
               <gml:exterior>
                  <gml:LinearRing>
                     <gml:posList srsDimension="2">65.588264 -120.000000 65.590782 -120.003571
        65.590965 -120.011292 65.595215 -120.022491 65.592880 -120.031212 65.586121 -120.019363
        65.585365 -120.030350 65.581848 -120.045082 65.584938 -120.059540 65.590500 -120.067284
        65.595436 -120.067284 65.613441 -120.067337 65.613777 -120.067337 65.606346 -120.060997
        65.605545 -120.045517 65.599777 -120.022675 65.601036 -120.003975 65.602081 -120.000000
        65.602081 -120.000000 65.588264 -120.000000</gml:posList>
                  </gml:LinearRing>
               </gml:exterior>
            </gml:Polygon>
         </extent>
         <id>0000000002</id>
         <treeType>Maple</treeType>
      </TreesA_1M>
   </wfs:member>
   <wfs:member>
```

```
      <RoadL_1M gml:id="ROADL_1M.1914">
         <centerLine>
            <gml:LineString gml:id="GID_5" srsName="urn:ogc:def:crs:EPSG::4326">
               <gml:posList>-59.478340 -52.226578 -59.484871 -52.223564 -59.488991 -52.198524 -
        59.485958 -52.169559 -59.480400 -52.152615 -59.465576 -52.141491 -59.462002 -52.136417 -
        59.447968 -52.127190 -59.422928 -52.120701 -59.411915 -52.117844 -59.397972 -52.116440 -
        59.371311 -52.121300</gml:posList>
            </gml:LineString>
         </centerLine>
         <designation>HYW 401</designation>
         <surfaceType>ASPHALT</surfaceType>
         <nLanes>12</nLanes>
      </RoadL_1M>
   </wfs:member>
</wfs:FeatureCollection>
```

B.2.2 Example 2

In response to the DescribeFeatureType request:

```
<?xml version="1.0" ?>
<DescribeFeatureType
   service="WFS"
   version="2.0.0"
   outputFormat="application/gml+xml; version=3.2"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:myns="http://www.someserver.com/myns"
   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">
   <TypeName>myns:Person</TypeName>
</DescribeFeatureType>
```

A web feature service might generate an XML Schema document that looks like:

```
<?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"/>

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

```
                    nillable="true"/>
            <element name="mailAddress"
                    type="myns:AddressPropertyType" nillable="true"/>
            <element name="phone" type="xsd:string"
                    minOccurs="0" maxOccurs="unbounded"/>
            <element name="livesIn" type="myns:HousePropertyType"
                    minOccurs="0"/>
            <element name="isDriving" type="myns:CarPropertyType"
                    minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
<complexType name="PersonPropertyType">
   <sequence>
      <element ref="myns:Person"  minOccurs="0"/>
   </sequence>
   <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
<complexType name="AddressPropertyType">
   <sequence>
      <element name="Address"
               type="myns:AddressType" minOccurs="0" />
   </sequence>
   <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
<complexType name="AddressType">
   <sequence>
      <element name="streetName" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="30"/>
            </restriction>
         </simpleType>
      </element>
      <element name="streetNumber" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="10"/>
            </restriction>
         </simpleType>
      </element>
      <element name="city" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="30"/>
            </restriction>
         </simpleType>
      </element>
      <element name="province" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="30"/>
            </restriction>
         </simpleType>
      </element>
      <element name="postalCode" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="15"/>
            </restriction>
         </simpleType>
      </element>
      <element name="country" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="30"/>
            </restriction>
         </simpleType>
      </element>
   </sequence>
   <attribute ref="gml:id" use="required"/>
</complexType>

<element name="Car" type="myns:CarType"
         substitutionGroup="gml:AbstractFeature"/>
```

```
   <complexType name="CarType">
      <complexContent>
         <extension base="gml:AbstractFeatureType">
            <sequence>
               <element name="model" type="xsd:string"/>
               <element name="age" type="xsd:nonNegativeInteger"/>
               <element name="colour">
                  <simpleType>
                     <restriction base="string">
                        <enumeration value="red"/>
                        <enumeration value="green"/>
                        <enumeration value="blue"/>
                        <enumeration value="yellow"/>
                        <enumeration value="black"/>
                        <enumeration value="white"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="location" type="gml:PointPropertyType"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="CarPropertyType">
      <sequence>
         <element ref="myns:Car"  minOccurs="0"/>
      </sequence>
      <attributeGroup ref="gml:AssociationAttributeGroup"/>
   </complexType>

   <element name="House" type="myns:HouseType"
            substitutionGroup="gml:AbstractFeature"/>
   <complexType name="HouseType">
      <complexContent>
         <extension base="gml:AbstractFeatureType">
            <sequence>
               <element name="numFloors" type="xsd:nonNegativeInteger"/>
               <element name="area" type="gml:MeasureType"/>
               <element name="location" type="gml:PointPropertyType"/>
               <element name="frontsOn" type="gml:CurvePropertyType"/>
               <element name="address" type="myns:AddressPropertyType"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="HousePropertyType">
      <sequence>
         <element ref="myns:House"  minOccurs="0"/>
      </sequence>
      <attributeGroup ref="gml:AssociationAttributeGroup"/>
   </complexType>
</schema>
```

As in B.2.1, the server generates a complete application schema that includes the requested feature type (i.e. myns:Person) but also includes all the other feature types in the application schema.

A sample instance document that validates against this schema might be:

```
<?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:rds="http://www.someserver.com/rds"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   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
                       ./DescribeFeatureType_Example02_Response.xsd
                       http://www.someserver.com/rds
```

```
                    ./DescribeFeatureType_Example02_Road.xsd">

<wfs:member>
   <myns:Person  gml:id="p4456">
      <gml:identifier
      codeSpace="http://www.canadaSIN.com">424679374</gml:identifier>
      <myns:lastName>Smith</myns:lastName>
      <myns:firstName>Fred</myns:firstName>
      <myns:age>35</myns:age>
      <myns:sex>male</myns:sex>
      <myns:spouse xlink:href="#p4467"/>
      <myns:location xlink:href="#pt102" />
      <myns:mailAddress xlink:href="#a201"/>
      <myns:phone>416-123-4567</myns:phone>
      <myns:phone>416-890-1234</myns:phone>
      <myns:livesIn xlink:href="#h32"/>
   </myns:Person>
</wfs:member>

<wfs:member>
   <myns:Person  gml:id="p4467">
      <gml:identifier
      codeSpace="http://www.canadaSIN.com">424679360</gml:identifier>
      <myns:lastName>Smith</myns:lastName>
      <myns:firstName>Mary</myns:firstName>
      <myns:age>18</myns:age>
      <myns:sex>female</myns:sex>
      <myns:spouse xlink:href="#p4456"/>
      <myns:location xlink:href="#pt101" />
      <myns:mailAddress xlink:href="#a201"/>
      <myns:phone>416-123-4567</myns:phone>
      <myns:phone>416-890-4532</myns:phone>
      <myns:livesIn xlink:href="#h32"/>
      <myns:isDriving xlink:href="#r1432"/>
   </myns:Person>
</wfs:member>

<wfs:member>
   <myns:Car gml:id="r1432">
      <gml:identifier
      codeSpace="http://www.carserial.org">51465243</gml:identifier>
      <myns:model>Ford Pinto</myns:model>
      <myns:age>4</myns:age>
      <myns:colour>red</myns:colour>
      <myns:location>
         <gml:Point gml:id="pt102">
            <gml:pos>-59.603958 -52.106559</gml:pos>
         </gml:Point>
      </myns:location>
   </myns:Car>
</wfs:member>
<wfs:member>
   <myns:House gml:id="h32">
      <gml:identifier
      codeSpace="http://www.toronto.ca/reg.xml">654365143</gml:identifier>
      <myns:numFloors>2</myns:numFloors>
      <myns:area>200</myns:area>
      <myns:location>
         <gml:Point gml:id="pt101">
            <gml:pos>16 18</gml:pos>
         </gml:Point>
      </myns:location>
      <myns:frontsOn xlink:href="#rs11"/>
      <myns:address>
         <myns:Address gml:id="a201">
            <myns:streetName>Main St.</myns:streetName>
            <myns:streetNumber>5</myns:streetNumber>
            <myns:city>SomeCity</myns:city>
            <myns:province>Someprovince</myns:province>
            <myns:postalCode>X1X 1X1</myns:postalCode>
            <myns:country>Canada</myns:country>
         </myns:Address>
      </myns:address>
   </myns:House>
</wfs:member>
```

```
   <wfs:member>
      <rds:Road gml:id="rs11">
         <rds:numLanes>3</rds:numLanes>
         <rds:centerline>
            <gml:LineString gml:id="GID_5" srsName="urn:ogc::def:crs:EPSG::4326">
               <gml:posList>-59.478340 -52.226578 -59.484871 -52.223564 -59.488991 -52.198524 -
      59.485958 -52.169559 -59.480400 -52.152615 -59.465576 -52.141491 -59.462002 -52.136417 -
      59.447968 -52.127190 -59.422928 -52.120701 -59.411915 -52.117844 -59.397972 -52.116440 -
      59.371311 -52.121300</gml:posList>
            </gml:LineString>
         </rds:centerline>
      </rds:Road>
   </wfs:member>
</wfs:FeatureCollection>
```

B.3 GetFeature examples

B.3.1 Introduction

This clause contains numerous examples of the GetFeature request. Some examples include sample output.

B.3.2 Example 1

This example fetches a specific instance of the feature type *InWaterA_1M* identified by the feature identifier "InWaterA_1M.1234".

```
<?xml version="1.0" ?>
<wfs:GetFeature
   service="WFS"
   version="2.0.0"
   outputFormat="application/gml+xml; version=3.2"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   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">
   <wfs:Query typeName="myns:InWaterA_1M">
      <fes:Filter>
         <fes:ResourceId rid="InWaterA_1M.1234"/>
      </fes:Filter>
   </wfs:Query>
</wfs:GetFeature>
```

B.3.3 Example 2

This example fetches a subset of properties of the feature type InWaterA_1M. The specific instance that is retrieved by the request is identified by the feature identifier "InWaterA_1M.1013".

```
<?xml version="1.0" ?>
<wfs:GetFeature
   service="WFS"
   version="2.0.0"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:myns="http://www.someserver.com/myns"
   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">
   <wfs:Query typeNames="myns:InWaterA_1M">
      <wfs:PropertyName>myns:wkbGeom</wfs:PropertyName>
      <wfs:PropertyName>myns:tileId</wfs:PropertyName>
      <wfs:PropertyName>myns:facId</wfs:PropertyName>
      <fes:Filter>
         <fes:ResourceId rid="InWaterA_1M.1013"/>
      </fes:Filter>
   </wfs:Query>
</wfs:GetFeature>
```

B.3.4 Example 3

In this example, all the properties of feature type *InWaterA_1M* are fetched for an enumerated list of feature instances. The fes:ResourceId element is used to identify each feature to be fetched.

```
<?xml version="1.0" ?>
<GetFeature
   version="2.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:myns="http://www.someserver.com/myns"
   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">
   <Query typeNames="myns:InWaterA_1M">
     <fes:Filter>
        <fes:ResourceId rid="InWaterA_1M.1013"/>
        <fes:ResourceId rid="InWaterA_1M.1014"/>
        <fes:ResourceId rid="InWaterA_1M.1015"/>
      </fes:Filter>
   </Query>
</GetFeature>
```

B.3.5 Example 4

This example is similar to the previous example except that, in this case, only some of the properties of an enumerated set of features are fetched. The wfs:PropertyName element is used to list the properties to be retrieved.

```
<?xml version="1.0" ?>
<GetFeature
   version="2.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:myns="http://www.someserver.com/myns"
   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">
   <Query typeNames="myns:InWaterA_1M">
      <wfs:PropertyName>myns:wkbGeom</wfs:PropertyName>
      <wfs:PropertyName>myns:tileId</wfs:PropertyName>
      <fes:Filter>
         <fes:ResourceId rid="InWaterA_1M.1013"/>
         <fes:ResourceId rid="InWaterA_1M.1014"/>
         <fes:ResourceId rid="InWaterA_1M.1015"/>
      </fes:Filter>
   </Query>
</GetFeature>
```

The sample response fragment to such a request might be:

```
<?xml version="1.0"?>
<wfs:FeatureCollection
   timeStamp="2010-02-01T22:56:09"
   numberMatched="3"
   numberReturned="3"
   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.someserver.com/myns ./GetFeature_05.xsd
                       http://www.opengis.net/wfs/2.0
                       http://schemas.opengis.net/wfs/2.0.0/wfs.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:boundedBy>
      <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
         <gml:lowerCorner>-31.27141761779785 115.001335144043</gml:lowerCorner>
         <gml:upperCorner>-30.10202789306641 117.9325866699219</gml:upperCorner>
      </gml:Envelope>
   </wfs:boundedBy>
```

```
   <wfs:member>
      <InWaterA_1M gml:id="InWaterA_1M.1013">
         <wkbGeom>
            <gml:Polygon srsName="urn:ogc:def:crs:EPSG::4326" gml:id="P1">
               <gml:exterior>
                  <gml:LinearRing>
                     <gml:posList>-30.93597221374512 117.6290588378906 -30.94830513000489
       117.6447219848633 -30.95219421386719 117.6465530395508 -30.95219421386719 117.6431121826172 -
       30.94802856445312 117.6386108398438 -30.94799995422363 117.6314163208008 -30.946138381958
       117.62850189209 -30.94430541992188 117.6295852661133 -30.93280601501464 117.6240539550781 -
       30.92869377136231 117.624641418457 -30.92386054992676 117.6201400756836 -30.92111206054688
       117.6206970214844 -30.92458343505859 117.6275863647461 -30.93597221374512
       117.6290588378906</gml:posList>
                  </gml:LinearRing>
               </gml:exterior>
            </gml:Polygon>
         </wkbGeom>
         <id>28022</id>
         <tileId>177</tileId>
      </InWaterA_1M>
   </wfs:member>
   <wfs:member>
      <InWaterA_1M gml:id="InWaterA_1M.1014">
         <wkbGeom>
            <gml:Polygon srsName="urn:ogc:def:crs:EPSG::4326" gml:id="P2">
               <gml:exterior>
                  <gml:LinearRing>
                     <gml:posList>-30.92013931274414 117.6552810668945 -30.92383384704589
       117.661361694336 -30.93005561828613 117.6666412353516 -30.93280601501464 117.6663589477539 -
       30.93186187744141 117.6594467163086 -30.93780517578125 117.6511137695312 -30.94397163391114
       117.6519470214844 -30.94255638122559 117.6455535888672 -30.94402862548828 117.6336364746094 -
       30.92874908447266 117.6355285644531 -30.92138862609864 117.6326370239258 -30.92236137390137
       117.6395568847656 -30.91708374023438 117.6433029174805 -30.91711044311523 117.6454467773437 -
       30.92061042785645 117.6484985351563 -30.92061042785645 117.6504135131836 -30.91638946533203
       117.6504440307617 -30.92013931274414 117.6552810668945</gml:posList>
                  </gml:LinearRing>
               </gml:exterior>
            </gml:Polygon>
         </wkbGeom>
         <id>28021</id>
         <tileId>177</tileId>
      </InWaterA_1M>
   </wfs:member>
   <wfs:member>
      <InWaterA_1M gml:id="InWaterA_1M.1015">
         <wkbGeom>
            <gml:Polygon srsName="urn:ogc:def:crs:EPSG::4326" gml:id="P3">
               <gml:exterior>
                  <gml:LinearRing>
                     <gml:posList>-31.27141761779785 117.9237747192383 -31.2668342590332
       117.9219131469727 -31.26474952697754 117.9165802001953 -31.26177787780761 117.9173889160156 -
       31.25813865661621 117.9240798950195 -31.25927734375 117.928337097168 -31.26297187805175
       117.9320526123047 -31.26572227478028 117.9325866699219 -31.27097129821777 117.9283065795898 -
       31.27141761779785 117.9237747192383</gml:posList>
                  </gml:LinearRing>
               </gml:exterior>
            </gml:Polygon>
         </wkbGeom>
         <id>28074</id>
         <tileId>177</tileId>
      </InWaterA_1M>
   </wfs:member>
</wfs:FeatureCollection>
```

Notice that the feature instances include the requested properties wkbGeom and tileId and also include the property id which has been included by the web feature server in order to generate a valid response document (see 7.9.2.4.5.1).

B.3.6 Example 5

Select all instances of the feature type *InWaterA_1M* to a maximum of 10 000 features.

```
<?xml version="1.0" ?>
<GetFeature
   version="2.0.0"
   service="WFS"
   count="10000"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:myns="http://www.someserver.com/myns"
   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">
   <Query typeNames="myns:InWaterA_1M"/>
</GetFeature>
```

B.3.7 Example 6

The following unconstrained request fetches all the instances of an enumerated set of feature types. Notice that the feature types are not all in the same namespace.

```
<?xml version="1.0" ?>
<GetFeature
   version="2.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:yourns="http://demo.someserver.com/yourns"
   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">
   <Query typeNames="myns:InWaterA_1M"/>
   <Query typeNames="myns:BuiltUpA_1M"/>
   <Query typeNames="yourns:RoadL_1M"/>
</GetFeature>
```

B.3.8 Example 7

The following example selects the geometry and depth from the Hydrography feature for the area of the Grand Banks. The Grand Banks are bounded by the following box: [46.2023,-57.9118, 51.8145,-46.6873].

```
<?xml version="1.0" ?>
<GetFeature
   version="2.0.0"
   service="WFS"
   handle="Query01"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:myns="http://www.someserver.com/myns"
   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.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd
                       http://www.someserver.com/myns ./GetFeature_07.xsd">

   <Query typeNames="myns:Hydrography">
      <PropertyName>myns:geoTemp</PropertyName>
      <PropertyName>myns:depth</PropertyName>
      <PropertyName>myns:temperature</PropertyName>
      <fes:Filter>
         <fes:Not>
            <fes:Disjoint>
               <fes:ValueReference>myns:geoTemp</fes:ValueReference>
               <gml:Envelope srsName="urn:ogc;def:crs:EPSG::4326">
                  <gml:lowerCorner>46.2023 -57.9118 </gml:lowerCorner>
                  <gml:upperCorner>51.8145 -46.6873</gml:upperCorner>
               </gml:Envelope>
            </fes:Disjoint>
         </fes:Not>
      </fes:Filter>
```

```
      <fes:SortBy>
         <fes:SortProperty>
            <fes:ValueReference>myns:depth</fes:ValueReference>
         </fes:SortProperty>
         <fes:SortProperty>
            <fes:ValueReference>myns:temperature</fes:ValueReference>
            <fes:SortOrder>DESC</fes:SortOrder>
         </fes:SortProperty>
      </fes:SortBy>
   </Query>
</GetFeature>
```

The output from such a request might be:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection
  timeStamp="2009-11-04T11:05:00"
  numberMatched="3"
  numberReturned="3"
  xmlns="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.someserver.com/myns  ./GetFeature_07.xsd
                      http://www.opengis.net/wfs/2.0
                      http://schemas.opengis.net/wfs/2.0.0/wfs.xsd
                      http://www.opengis.net/gml/3.2
                      http://schemas.opengis.net/gml/3.2.1/gml.xsd">
  <wfs:member>
    <Hydrography gml:id="HydrographyHydrography.450">
      <geoTemp>
        <gml:Point gml:id="GID_3" srsName="urn:ogc:def:crs:EPSG::4326">
          <gml:pos>47.2002 -56.314</gml:pos>
        </gml:Point>
      </geoTemp>
      <depth uom="mm">565</depth>
      <temperature uom="C">10</temperature>
    </Hydrography>
  </wfs:member>
  <wfs:member>
    <Hydrography gml:id="HydrographyHydrography.451">
      <geoTemp>
        <gml:Point gml:id="GID_4" srsName="urn:ogc:def:crs:EPSG::4326">
          <gml:pos>47.2003 -56.313</gml:pos>
        </gml:Point>
      </geoTemp>
      <depth uom="mm">1015</depth>
      <temperature uom="C">7</temperature>
    </Hydrography>
  </wfs:member>
  <wfs:member>
    <Hydrography gml:id="HydrographyHydrography.452">
      <geoTemp>
        <gml:Point gml:id="GID_5" srsName="urn:ogc:def:crs:EPSG::4326">
          <gml:pos>47.2003 -56.313</gml:pos>
        </gml:Point>
      </geoTemp>
      <depth uom="mm">2456</depth>
      <temperature uom="C">4</temperature>
    </Hydrography>
  </wfs:member>
</wfs:FeatureCollection>
```

B.3.9 Example 8

This example describes two queries that fetch instances of *ROADS* and *RAILS* that lie within a single region of interest.

```
<?xml version="1.0"?>
<GetFeature
   version="2.0.0"
   service="WFS"
   handle="Example Query"
   xmlns="http://www.opengis.net/wfs/2.0"
```

```
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:myns="http://www.someserver.com/myns"
   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.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <Query typeNames="myns:Roads" handle="Q01">
      <PropertyName>myns:path</PropertyName>
      <PropertyName>myns:nLanes</PropertyName>
      <PropertyName>myns:surfaceType</PropertyName>
      <fes:Filter>
         <fes:Within>
            <fes:ValueReference>myns:path</fes:ValueReference>
            <fes:Literal>
               <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
                  <gml:lowerCorner>50 40</gml:lowerCorner>
                  <gml:upperCorner>100 60</gml:upperCorner>
               </gml:Envelope>
            </fes:Literal>
         </fes:Within>
      </fes:Filter>
   </Query>
   <Query typeNames="myns:Rails" handle="Q02">
      <PropertyName>myns:track</PropertyName>
      <PropertyName>myns:gauge</PropertyName>
      <fes:Filter>
         <fes:Within>
            <fes:ValueReference>myns:track</fes:ValueReference>
            <fes:Literal>
               <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
                  <gml:lowerCorner>-33 103</gml:lowerCorner>
                  <gml:upperCorner>1 135</gml:upperCorner>
               </gml:Envelope>
            </fes:Literal>
         </fes:Within>
      </fes:Filter>
   </Query>
</GetFeature>
```

The results of each query are combined according to 11.3.3.5 to form the output feature collection.

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection
   timeStamp="2008-08-01T22:47:02"
   numberMatched="unknown"
   numberReturned="200"
   xmlns="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.someserver.com/myns ./RoadRail.xsd
                       http://www.opengis.net/wfs/2.0
                       http://schemas.opengis.net/wfs/2.0.0/wfs.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">

   <wfs:boundedBy>
      <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
         <gml:lowerCorner>-32.7638053894043 104.1429748535156</gml:lowerCorner>
         <gml:upperCorner>0 133.3553924560547</gml:upperCorner>
      </gml:Envelope>
   </wfs:boundedBy>

   <!-- Road collection -->
   <wfs:member>
      <wfs:FeatureCollection
         timeStamp="2008-08-01T22:47:02"
         numberMatched="unknown"
         numberReturned="167">
         <wfs:member>
            <Road gml:id="Road.100">
               <centerLine>
                  <gml:LineString gml:id="LS1" srsName="urn:ogc:def:crs:EPSG::4326">
```

```
                  <gml:posList>-24.90258407592773 113.6248092651367 -24.90200042724609
        113.6278305053711 -24.89602851867676 113.6336364746094 -24.89147186279297 113.6345825195312 -
        24.8884449005127 113.6405029296875 -24.87958335876465 113.6336669921875 -24.8785285949707
        113.6328582763672 -24.87274932861328 113.6313323974609 -24.87163925170898 113.6195526123047 -
        24.87466621398926 113.614387512207 -24.87716674804688 113.6137771606445 -24.88877868652344
        113.6236114501953 -24.89338874816895 113.6241683959961 -24.89838981628418 113.622444152832 -
        24.90258407592773 113.6248092651367</gml:posList>
                </gml:LineString>
             </centerLine>
             <surfaceType>ASPHALT</surfaceType>
             <nLanes>4</nLanes>
          </Road>
       </wfs:member>
       <wfs:member>
          <Road gml:id="Road.105">
             <centerLine>
                <gml:LineString gml:id="LS2" srsName="urn:ogc:def:crs:EPSG::4326">
                   <gml:posList>-0.3605277836322785 104.2599411010742 -0.3567777872085571
        104.2643585205078 -0.3565555512905121 104.2669982910156 -0.3578888773918152 104.2689743041992
        -0.3623055517673492 104.2698364257812 -0.3702777922153473 104.2678298950195 -
        0.3751111030578613 104.2592468261719 -0.3728888928890228 104.2574996948242 -0.3664722144603729
        104.2566375732422 -0.3605277836322785 104.2599411010742</gml:posList>
                </gml:LineString>
             </centerLine>
             <surfaceType>GRAVEL</surfaceType>
             <nLanes>2</nLanes>
          </Road>
       </wfs:member>
       <!-- ... more Road instances ... -->
    </wfs:FeatureCollection>
 </wfs:member>
 <!-- Rail collection -->
 <wfs:member>
    <wfs:FeatureCollection
       timeStamp="2008-08-01T22:47:02"
       numberMatched="unknown"
       numberReturned="33">
       <wfs:member>
          <Rail gml:id="Rail.119">
             <track>
                <gml:LineString gml:id="LS3" srsName="urn:ogc:def:crs:EPSG::4326">
                   <gml:posList>-15.34877777099609 124.3914184570313 -15.34572219848633
        124.3990859985352 -15.34027767181396 124.39111328125 -15.34066677093506 124.3884963989258 -
        15.34805583953857 124.3888320922852 -15.34877777099609 124.3914184570313</gml:posList>
                </gml:LineString>
             </track>
             <gauge>24</gauge>
          </Rail>
       </wfs:member>
       <!-- ... more Rail instances ... -->
    </wfs:FeatureCollection>
 </wfs:member>
</wfs:FeatureCollection>
```

B.3.10 Example 9

This example illustrates how complex properties of features may be referenced using XPath expressions. Consider the feature type *Person* defined as:

```
<?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"/>
   <element name="Person" type="myns:PersonType"
            substitutionGroup="gml:AbstractFeature"/>
   <complexType name="PersonType">
      <complexContent>
```

```
      <extension base="gml:AbstractFeatureType">
         <sequence>
            <element name="lastName" nillable="true">
               <simpleType>
                  <restriction base="string">
                     <maxLength value="30"/>
                  </restriction>
               </simpleType>
            </element>
            <element name="firstName" nillable="true">
               <simpleType>
                  <restriction base="string">
                     <maxLength value="10"/>
                  </restriction>
               </simpleType>
            </element>
            <element name="age" type="integer" nillable="true"/>
            <element name="sex" type="string"/>
            <element name="spouse">
               <complexType>
                  <attribute name="SIN" type="xsd:anyURI" use="required" />
               </complexType>
            </element>
            <element name="location"
                     type="gml:PointPropertyType"
                     nillable="true"/>
            <element name="mailAddress"
                     type="myns:AddressPropertyType" nillable="true"/>
            <element name="salary" type="positiveInteger" nillable="true"/>
         </sequence>
         <attribute name="SIN" type="xsd:anyURI" use="required"/>
      </extension>
   </complexContent>
</complexType>
<complexType name="AddressPropertyType">
   <sequence>
      <element name="Address"
               type="myns:AddressType" minOccurs="0" />
   </sequence>
</complexType>
<complexType name="AddressType">
   <sequence>
      <element name="streetName" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="30"/>
            </restriction>
         </simpleType>
      </element>
      <element name="streetNumber" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="10"/>
            </restriction>
         </simpleType>
      </element>
      <element name="city" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="30"/>
            </restriction>
         </simpleType>
      </element>
      <element name="province" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="30"/>
            </restriction>
         </simpleType>
      </element>
      <element name="postalCode" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="15"/>
            </restriction>
         </simpleType>
```

```
         </element>
         <element name="country" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
               </restriction>
            </simpleType>
         </element>
      </sequence>
   </complexType>
</schema>
```

The *mailAddress* property is a complex property.

The following example fetches the last name of all the people who live on the 10 000 block of "Main St." in the town of "SomeTown" who are female and make over $35 000 in salary. Note the use of XPath expressions in the predicate to reference complex properties.

```
<?xml version="1.0" ?>
<GetFeature
   version="2.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:myns="http://www.someserver.com/myns"
   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 ./GetFeature_09.xsd">
   <Query typeNames="Person">
      <PropertyName>myns:lastName</PropertyName>
      <fes:Filter>
         <fes:And>
            <fes:And>
               <fes:PropertyIsGreaterThanOrEqualTo>
<fes:ValueReference>myns:Person/myns:mailAddress/myns:Address/myns:streetNumber</fes:ValueReference>
                  <fes:Literal>10000</fes:Literal>
               </fes:PropertyIsGreaterThanOrEqualTo>
               <fes:PropertyIsLessThanOrEqualTo>
<fes:ValueReference>myns:Person/myns:mailAddress/myns:Address/myns:streetNumber</fes:ValueReference>
                  <fes:Literal>10999</fes:Literal>
               </fes:PropertyIsLessThanOrEqualTo>
            </fes:And>
            <fes:And>
               <fes:PropertyIsEqualTo>
<fes:ValueReference>myns:Person/myns:mailAddress/myns:Address/myns:streetName</fes:ValueReference>
                  <fes:Literal>Main St.</fes:Literal>
               </fes:PropertyIsEqualTo>
               <fes:PropertyIsEqualTo>
<fes:ValueReference>myns:Person/myns:mailAddress/myns:Address/myns:city</fes:ValueReference>
                  <fes:Literal>SomeTown</fes:Literal>
               </fes:PropertyIsEqualTo>
               <fes:PropertyIsEqualTo>
                  <fes:ValueReference>myns:Person/myns:sex</fes:ValueReference>
                  <fes:Literal>Female</fes:Literal>
               </fes:PropertyIsEqualTo>
               <fes:PropertyIsGreaterThan>
                  <fes:ValueReference>myns:Person/myns:salary</fes:ValueReference>
                  <fes:Literal>35000</fes:Literal>
               </fes:PropertyIsGreaterThan>
            </fes:And>
         </fes:And>
      </fes:Filter>
   </Query>
</GetFeature>
```

B.3.11 Example 10

This example illustrates a GetFeature request using wfs:PropertyName elements that serves as a base for comparison with Examples 11 and 12. The request:

```
<?xml version="1.0"?>
<wfs:GetFeature
```

```
   service="WFS"
   version="2.0.0"
   outputFormat="application/gml+xml; version=3.2"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   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/wfs/2.0
                       http://schemas.opengis.net/wfs/2.0.0/wfs.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:Query typeNames="Town">
      <wfs:PropertyName>gml:name</wfs:PropertyName>
      <wfs:PropertyName resolve="none">gml:directedNode</wfs:PropertyName>
      <fes:Filter>
         <fes:ResourceId rid="t1"/>
      </fes:Filter>
   </wfs:Query>
</wfs:GetFeature>
```

The response contains an xlink:href, but it is returned as-is because the resolve attribute is set to none:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection
   timeStamp="2009-02-14T01:27:44"
   numberMatched="1"
   numberReturned="1"
   xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   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.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:boundedBy>
      <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
         <gml:lowerCorner>10 10</gml:lowerCorner>
         <gml:upperCorner>20 20</gml:upperCorner>
      </gml:Envelope>
   </wfs:boundedBy>
   <wfs:member>
      <Town gml:id="t1">
         <gml:name>Bedford</gml:name>
         <gml:directedNode orientation="+" xlink:href="#n1"/>
      </Town>
   </wfs:member>
</wfs:FeatureCollection>
```

B.3.12 Example 11

This example illustrates the use of the resolveDepth and resolveTimeout attributes on the GetFeature request from Example 10. The request:

```
<?xml version="1.0"?>
<wfs:GetFeature
   service="WFS"
   version="2.0.0"
   resolveDepth="1"
   resolveTimeout="1"
   outputFormat="application/gml+xml; version=3.2"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   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/wfs/2.0
                       http://schemas.opengis.net/wfs/2.0.0/wfs.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:Query typeNames="Town">
      <wfs:PropertyName>gml:name</wfs:PropertyName>
      <wfs:PropertyName resolve="all">gml:directedNode</wfs:PropertyName>
      <fes:Filter>
```

```
         <fes:ResourceId rid="t1"/>
      </fes:Filter>
   </wfs:Query>
</wfs:GetFeature>
```

In the response, the first level xlink:href has been traversed, and its contents returned as:

```
<?xml version="1.0"?>
<wfs:FeatureCollection
   timeStamp="2009-02-14T01:27:44"
   numberMatched="1"
   numberReturned="1"
   xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   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.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:boundedBy>
      <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
         <gml:lowerCorner>10 10</gml:lowerCorner>
         <gml:upperCorner>20 20</gml:upperCorner>
      </gml:Envelope>
   </wfs:boundedBy>
   <wfs:member>
      <Town gml:id="t1">
         <gml:name>Bedford</gml:name>
         <gml:directedNode orientation="+" xlink:href="#n1"/>
         <!-- xlink:href="#n1"/> -->
         <gml:directedNode orientation="+">
            <gml:Node gml:id="n1">
               <gml:pointProperty
       xlink:href="http://www.bedford.town.uk/civilworks/gps.gml#townHall"/>
            </gml:Node>
         </gml:directedNode>
      </Town>
   </wfs:member>
</wfs:FeatureCollection>
```

B.3.13 Example 12

This example illustrates the use of the wfs:XlinkPropertyName element with resolveDepth and resolveTimeout attributes that override those attributes on the GetFeature request from Example 11. The request:

```
<?xml version="1.0"?>
<wfs:GetFeature
   service="WFS"
   version="2.0.0"
   resolveDepth="1"
   resolveTimeout="1"
   outputFormat="application/gml+xml; version=3.2"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   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/wfs/2.0
                       http://schemas.opengis.net/wfs/2.0.0/wfs.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:Query typeNames="Town">
      <wfs:PropertyName>gml:name</wfs:PropertyName>
      <wfs:PropertyName resolve="all"
                        resolveDepth="2">gml:directedNode</wfs:PropertyName>
      <fes:Filter>
         <fes:ResourceId rid="t1"/>
      </fes:Filter>
   </wfs:Query>
</wfs:GetFeature>
```

In the response, the first and second level xlink:href in the gml:directedNode properties have been traversed, and their contents returned as described in 7.6.4:

```
<?xml version="1.0"?>
<wfs:FeatureCollection
   timeStamp="2009-02-14T01:27:44"
   numberMatched="1"
   numberReturned="1"
   xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   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.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:boundedBy>
      <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
         <gml:lowerCorner>10 10</gml:lowerCorner>
         <gml:upperCorner>20 20</gml:upperCorner>
      </gml:Envelope>
   </wfs:boundedBy>
   <wfs:member>
      <Town gml:id="t1">
         <gml:name>Bedford</gml:name>
         <gml:directedNode orientation="+" xlink:href="#n1"/>
         <!-- xlink:href="#n1"/> -->
         <gml:directedNode orientation="+">
            <gml:Node gml:id="n1">
               <gml:pointProperty>
                  <gml:Point gml:id="townHall" srsName="urn:ogc:def:crs:EPSG::4326">
                     <gml:pos>-47 34</gml:pos>
                  </gml:Point>
               </gml:pointProperty>
            </gml:Node>
         </gml:directedNode>
      </Town>
   </wfs:member>
</wfs:FeatureCollection>
```

B.3.14 Example 13

The following example performs a join operation to find a lake contained within Algonquin Park.

```
<?xml version="1.0"?>
<GetFeature
   service="WFS"
   version="2.0.0"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   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">
   <Query typeNames="myns:Parks myns:Lakes">
      <fes:Filter>
         <fes:And>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>/myns:Parks</fes:ValueReference>
               <fes:Literal>Algonquin Park</fes:Literal>
            </fes:PropertyIsEqualTo>
            <fes:Contains>
               <fes:ValueReference>/myns:Parks/geometry</fes:ValueReference>
               <fes:ValueReference>/myns:Lakes/geometry</fes:ValueReference>
            </fes:Contains>
         </fes:And>
      </fes:Filter>
   </Query>
</GetFeature>
```

In response to this request, a WFS might generate:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection timeStamp="2008-08-15T11:36:00" numberMatched="12"
   numberReturned="12" xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.someserver.com/myns ./SampleSchema.xsd
                       http://www.opengis.net/wfs/2.0
                       http://schemas.opengis.net/wfs/2.0.0/wfs.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:member>
      <wfs:Tuple>
         <wfs:member>
            <Parks gml:id="Parks.287796">
               <Name>Algonquin Park</Name>
               <Boundary>
                  <gml:Polygon gml:id="GID_13"
                     srsName="urn:ogc:def:crs:EPSG::4326">
                     <gml:exterior>
                        <gml:LinearRing>
                           <gml:posList>-78.62456512451172 44.95297622680663 -78.62757110595703
        44.93315887451171 -78.62944030761717 44.93072891235352 -78.62944030761717 44.93072891235352 -
        78.64232635498047 44.93759918212891 -78.66877746582031 44.92557144165039 -78.66857147216798
        44.92765045166015 -78.65638732910156 44.93304443359375 -78.63591003417969 44.94966506958008 -
        78.62456512451172 44.95297622680663</gml:posList>
                        </gml:LinearRing>
                     </gml:exterior>
                  </gml:Polygon>
               </Boundary>
            </Parks>
         </wfs:member>
         <wfs:member>
            <Lakes gml:id="Lakes.287797">
               <Name>Canisbay Lake</Name>
               <Boundary>
                  <gml:Polygon gml:id="GID_14"
                     srsName="urn:ogc:def:crs:EPSG::4326">
                     <gml:exterior>
                        <gml:LinearRing>
                           <gml:posList>-78.70862579345703 44.96030044555664 -78.71179962158205
        44.96092224121094 -78.71607208251953 44.96665191650391 -78.71385955810547 44.97250747680664 -
        78.71767425537109 44.97959136962891 -78.71420288085938 44.98193359375 -78.71136474609375
        44.98753356933594 -78.70468902587891 44.98627471923828 -78.69895172119141 44.99493026733398 -
        78.69511413574219 44.99450683593375 -78.69327545166016 44.99028015136719 -78.70228576660156
        44.97490310668945 -78.69764709472655 44.96639633178711 -78.6983642578125 44.96253204345703 -
        78.70416259765625 44.96282958984375 -78.70862579345703 44.96030044555664</gml:posList>
                        </gml:LinearRing>
                     </gml:exterior>
                  </gml:Polygon>
               </Boundary>
            </Lakes>
         </wfs:member>
      </wfs:Tuple>
   </wfs:member>
   <wfs:member>
      <wfs:Tuple>
         <wfs:member xlink:href="#Parks.287796"/>
         <wfs:member>
            <Lakes gml:id="Lakes.287798">
               <Name>Kearney Lake</Name>
               <Boundary>
                  <gml:Polygon gml:id="GID_15"
                     srsName="urn:ogc:def:crs:EPSG::4326">
                     <gml:exterior>
                        <gml:LinearRing>
                           <gml:posList>-78.60539245605469 45 -78.59363555908202 45 -
        78.59363555908202 45 -78.59047698974609 44.99884414672852 -78.58409118652344 45 -
        78.58409118652344 45 -78.58390808105469 45 -78.58390808105469 45 -78.57343292236328 45 -
        78.57343292236328 45 -78.58406829833984 44.99701690673828 -78.59286499023438 44.98714447021484
        -78.5972900390625 44.97566604614257 -78.61244964599609 44.96060943603516 -78.62032318115234
        44.9564323425293 -78.62032318115234 44.9564323425293 -78.61922454833984 44.96395111083984 -
```

```
      78.60328674316406 44.9897575378418 -78.60232543945312 44.99590301513671 -78.60539245605469
      45</gml:posList>
                     </gml:LinearRing>
                  </gml:exterior>
               </gml:Polygon>
            </Boundary>
         </Lakes>
      </wfs:member>
   </wfs:Tuple>
</wfs:member>
<wfs:member>
   <wfs:Tuple>
      <wfs:member xlink:href="#Parks.287796"/>
      <wfs:member>
         <Lakes gml:id="Lakes.287799">
            <Name>Lake Of Two Rivers</Name>
            <Boundary>
               <gml:Polygon gml:id="GML_16"
                  srsName="urn:ogc:def:crs:EPSG::4326">
                  <gml:exterior>
                     <gml:LinearRing>
                        <gml:posList>-78.50907135009766 44.964599609375 -78.51042175292969
      44.96719360351563 -78.50795745849609 44.97211456298828 -78.49131011962891 44.97031402587891 -
      78.47936248779297 44.97610855102539 -78.46945190429688 44.97489166259765 -78.46963500976562
      44.9700813293457 -78.46161651611328 44.96322250366211 -78.45389556884766 44.96281051635742 -
      78.45217895507812 44.96065139770508 -78.46006774902344 44.95647811889649 -78.46690368652344
      44.94995498657227 -78.4732666015625 44.95718383789062 -78.47340393066406 44.9620132446289 -
      78.47900390625 44.96414947509766 -78.48902893066406 44.96123886108398 -78.49602508544922
      44.96551132202148 -78.50370788574219 44.96638107299804 -78.50907135009766
      44.964599609375</gml:posList>
                     </gml:LinearRing>
                  </gml:exterior>
               </gml:Polygon>
            </Boundary>
         </Lakes>
      </wfs:member>
   </wfs:Tuple>
</wfs:member>
<wfs:member>
   <wfs:Tuple>
      <wfs:member xlink:href="#Parks.287796"/>
      <wfs:member>
         <Lakes gml:id="Lakes.287806">
            <Name>Mew Lake</Name>
            <Boundary>
               <gml:Polygon gml:id="GID_17"
                  srsName="urn:ogc:def:crs:EPSG::4326">
                  <gml:exterior>
                     <gml:LinearRing>
                        <gml:posList>-78 44.96173095703125 -78.01906585693359 44.95022583007812 -
      78.02398681640625 44.95257568359375 -78.03146362304686 44.94656753540039 -78.03414916992188
      44.95430755615234 -78.03836059570312 44.95431900024414 -78.04071044921875 44.95148086547851 -
      78.04399108886719 44.94751739501953 -78.05403900146484 44.94165802001952 -78.06121063232422
      44.93264770507813 -78.06439971923828 44.93305587768555 -78.06134033203125 44.94873428344727 -
      78.05697631835938 44.95009231567382 -78.05594635009766 44.9534797668457 -78.05167388916016
      44.95691299438477 -78.06182861328125 44.96736907958984 -78.06884765625 44.98270034790039 -
      78.0685577392578 44.98819732666016 -78.06369781494141 44.98815155029297 -78.05403900146484
      44.97611618041992 -78.04621124267578 44.97658920288086 -78.04701232910156 44.96951293945312 -
      78.03591156005859 44.97025680541992 -78.02657318115234 44.9812126159668 -78.02198791503906
      44.98163986206055 -78.02023315429688 44.97993469238281 -78.02134704589844 44.97585678100586 -
      78.03009033203125 44.97015380859375 -78.04158782958984 44.9630012512207 -78.03964996337891
      44.96013259887695 -78.03326416015626 44.95930862426758 -78.02040863037109 44.96408843994141 -
      78.01665496826172 44.96295166015625 -78.00716400146486 44.96378707885742 -78.00435638427734
      44.96569442749023 -78.00571441650391 44.97082901000977 -78 44.97296905517578 -78
      44.97296905517578 -78 44.96173095703125</gml:posList>
                     </gml:LinearRing>
                  </gml:exterior>
               </gml:Polygon>
            </Boundary>
         </Lakes>
      </wfs:member>
   </wfs:Tuple>
</wfs:member>
<wfs:member>
   <wfs:Tuple>
      <wfs:member xlink:href="#Parks.287796"/>
```

```
         <wfs:member>
            <Lakes gml:id="Lakes.287817">
               <Name>Pog Lake</Name>
               <Boundary>
                  <gml:Polygon gml:id="GID_18"
                     srsName="urn:ogc:def:crs:EPSG::4326">
                     <gml:exterior>
                        <gml:LinearRing>
                           <gml:posList>-77.97476959228516 44.97800827026367 -77.97828674316406
      44.97269821166992 -77.99055480957031 44.9674301147461 -78 44.96173095703125 -78
      44.96173095703125 -78 44.97296905517578 -78 44.97296905517578 -77.99826812744141
      44.97361755371094 -77.98856353759766 44.97351837158203 -77.98063659667969 44.97765731811523 -
      77.97476959228516 44.97800827026367</gml:posList>
                        </gml:LinearRing>
                     </gml:exterior>
                  </gml:Polygon>
               </Boundary>
            </Lakes>
         </wfs:member>
      </wfs:Tuple>
   </wfs:member>
   <wfs:member>
      <wfs:Tuple>
         <wfs:member xlink:href="#Parks.287796"/>
         <wfs:member>
            <Lakes gml:id="Lakes.291062">
               <Name>Rock Lake</Name>
               <Boundary>
                  <gml:Polygon gml:id="GID_19"
                     srsName="urn:ogc:def:crs:EPSG::4326">
                     <gml:exterior>
                        <gml:LinearRing>
                           <gml:posList>-79.13716125488281 45.33119964599609 -79.14396667480469
      45.32807922363281 -79.15678405761719 45.32684707641602 -79.16410827636719 45.31824493408203 -
      79.16771697998047 45.31795501708984 -79.17977142333984 45.3109474182129 -79.18424224853516
      45.31184387207031 -79.18424224853516 45.31184387207031 -79.18721008300781 45.31496810913086 -
      79.19295501708984 45.31615447998048 -79.19607543945312 45.32111740112305 -79.19607543945312
      45.32111740112305 -79.19659423828125 45.32251739501953 -79.18892669677734 45.33133697509766 -
      79.1656494140625 45.33644104003906 -79.15970611572266 45.33386993408203 -79.14437866210938
      45.33406448364258 -79.13716125488281 45.33119964599609</gml:posList>
                        </gml:LinearRing>
                     </gml:exterior>
                  </gml:Polygon>
               </Boundary>
            </Lakes>
         </wfs:member>
      </wfs:Tuple>
   </wfs:member>
</wfs:FeatureCollection>
```

B.3.15 Example 14

In this example, the myns:RoadSegments feature is joined to the myns:Bridges feature using a standard join to find all bridges along a named road. The example assumes that both feature types include a RoadCode property. In the myns:RoadSegments feature, the RoadCode is used to indicate segments that are all part of the same named road. In the myns:Bridges feature, the RoadCode attribute indicates the named road along which the bridge exists.

```
<?xml version="1.0"?>
<GetFeature
   service="WFS"
   version="2.0.0"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   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">
   <Query typeNames="myns:RoadSegments myns:Bridges">
      <fes:Filter>
         <fes:And>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>/myns:RoadSegments/RoadName</fes:ValueReference>
               <fes:Literal>Main St.</fes:Literal>
```

```
            </fes:PropertyIsEqualTo>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>/myns:RoadSegments/RoadCode</fes:ValueReference>
               <fes:ValueReference>/myns:Bridges/RoadCode</fes:ValueReference>
            </fes:PropertyIsEqualTo>
         </fes:And>
      </fes:Filter>
   </Query>
</GetFeature>
```

B.3.16 Example 15

The following example shows a GetFeature request that uses a self-join with aliases to find all road segments that cross within some specified area of interest.

```
<?xml version="1.0"?>
<GetFeature
   service="WFS"
   version="2.0.0"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:myns="http://www.someserver.com/myns"
   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/wfs/2.0
                       http://schemas.opengis.net/wfs/2.0.0/wfs.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <Query typeNames="myns:RoadSegments myns:RoadSegments"
          aliases="RS1 RS2">
      <fes:Filter>
         <fes:And>
            <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:BBOX>
               <fes:ValueReference>/RS2/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:Crosses>
               <fes:ValueReference>/RS1/geometry</fes:ValueReference>
               <fes:ValueReference>/RS2/geometry</fes:ValueReference>
            </fes:Crosses>
         </fes:And>
      </fes:Filter>
   </Query>
</GetFeature>
```

B.3.17 Example 16

The following GetFeature example finds all the roads that intersect a specified polygon.

```
<?xml version="1.0" ?>
<GetFeature
   version="2.0.0"
   service="WFS"
   handle="Example Query"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:myns="http://www.someserver.com/myns"
   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.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
```

```
   <Query typeNames="myns:Roads" handle="Q01">
      <fes:Filter>
         <fes:Intersects>
            <fes:ValueReference>myns:path</fes:ValueReference>
            <gml:Polygon srsName="urn:ogc:def:crs:EPSG::4326" gml:id="P1">
               <gml:exterior>
                  <gml:LinearRing>
                     <gml:posList>-19.06099128723145 -169.9416961669922 -19.0565
3190612793 -169.9346008300781 -19.0523681640625 -169.9278564453125 -19.047290802
00195 -169.9230346679688 -19.03918266296387 -169.9215698242188 -19.0405883789062
5 -169.9138641357422 -19.04656600952148 -169.9136047363281 -19.05992698669434 -1
69.9196014404297 -19.06432342529297 -169.9275665283203 -19.06826400756836 -169.9
364929199219 -19.06099128723145 -169.9416961669922</gml:posList>
                  </gml:LinearRing>
               </gml:exterior>
            </gml:Polygon>
         </fes:Intersects>
      </fes:Filter>
   </Query>
</GetFeature>
```

B.3.18 Example 17

This International Standard defines two elements, wfs:Query and wfs:StoredQuery (see 7.9), that can be used to XML-encode query expressions.

This example shows how an additional element might be defined to encode query expressions, as might be the case for a vendor extension or a profile of this International Standard. In this example, an element named sql:Query is defined that allows a SQL query to be presented to the server.

This example assumes the following.

a) The server is implemented to understand the kinds of queries contained in the sql:Query element.

b) The newly defined query expression element is advertised in the server's capabilities document using the QueryExpression constraint (see Table 14).

c) The tables used in the query are materialized as feature types through the WFS interface.

The following XML fragment defines the schema of our new query element, sql:Query:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.someserver.com/sql/1.0"
   xmlns:sql="http://www.someserver.com/sql/1.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:xml="http://www.w3.org/XML/1998/namespace"
   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="Query" type="sql:QueryType"
                substitutionGroup="fes:AbstractQueryExpression"/>
   <xsd:complexType name="QueryType">
      <xsd:complexContent>
         <xsd:extension base="fes:AbstractQueryExpressionType">
            <xsd:sequence>
               <xsd:element name="Text" type="xsd:string"/>
            </xsd:sequence>
            <xsd:attribute name="targetDb" type="xsd:string" use="optional"/>
            <xsd:attribute name="targetDbVer" type="xsd:string" use="optional"/>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:schema>
```

The following XML-encoded GetFeature request uses the sql:Query element to identify all buildings that lie within a specified flood zone:

```
<?xml version="1.0" ?>
<wfs:GetFeature
   service="WFS"
   version="2.0.0"
   outputFormat="application/gml+xml; version=3.2"
   xmlns:sql="http://www.someserver.com/sql/1.0"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   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/sql/1.0 ./SqlQuery.xsd">
   <sql:Query targetDb="SQLMM">
      <sql:Text>
         SELECT * FROM buildings AS b, rivers AS r
         WHERE b.ground_plot.ST_Within(r.flood_zones) = 1
      </sql:Text>
   </sql:Query>
</wfs:GetFeature>
```

B.3.19 Example 18

The following example obtains a count of the number of InWaterA_1M feature instances by setting the resultType parameter to "hits".

```
<?xml version="1.0" ?>
<GetFeature
   version="2.0.0"
   service="WFS"
   resultType="hits"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:myns="http://www.someserver.com/myns"
   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">
   <Query typeNames="myns:InWaterA_1M"/>
</GetFeature>
```

The response shows that there are 339 963 features. The numberReturned parameter is set to 0, indicating that no features are actually returned in the response.

```
<?xml version="1.0"?>
<wfs:FeatureCollection
   timeStamp="2010-02-01T22:56:09"
   numberMatched="339963"
   numberReturned="0"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   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"/>
```

B.3.20 Example 19

Although GML 3.2 (see ISO 19136:2007) is the canonical version of GML supported by this International Standard, other versions of the GML may be used as well. The following GetFeature example includes a spatial predicate where the geometry has been encoded using GML 2.1.2.

```
<?xml version="1.0" ?>
<GetFeature
   version="2.0.0"
   service="WFS"
   handle="Example Query"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:myns="http://www.someserver.com/myns"
   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.opengis.net/gml
                       http://schemas.opengis.net/gml/2.1.2/geometry.xsd">
   <Query typeNames="myns:Roads" handle="Q01">
      <fes:Filter>
         <fes:Intersects>
            <fes:ValueReference>myns:path</fes:ValueReference>
            <gml:Polygon srsName="urn:ogc:def:crs:EPSG::4326">
               <gml:outerBoundaryIs>
                  <gml:LinearRing>
                     <gml:coordinates>-19.06099128723145,-169.9416961669922 -19.05653190612793,-
      169.9346008300781 -19.0523681640625,-169.9278564453125 -19.04729080200195,-169.9230346679688 -
      19.03918266296387,-169.9215698242188 -19.04058837890625,-169.9138641357422 -
      19.04656600952148,-169.9136047363281 -19.05992698669434,-169.9196014404297 -
      19.06432342529297,-169.9275665283203 -19.06826400756836,-169.9364929199219 -
      19.06099128723145,-169.9416961669922</gml:coordinates>
                  </gml:LinearRing>
               </gml:outerBoundaryIs>
            </gml:Polygon>
         </fes:Intersects>
      </fes:Filter>
   </Query>
</GetFeature>
```

B.4 GetPropertyValue examples

B.4.1 Introduction

The examples in this clause assume that a web feature service is offering the following fictitious set of feature instances:

```
      <myns:Person gml:id="p4456">
         <gml:identifier codeSpace="http://www.canadaSIN.com">424679374</gml:identifier>
         <myns:lastName>Smith</myns:lastName>
         <myns:firstName>Fred</myns:firstName>
         <myns:age>35</myns:age>
         <myns:sex>Male</myns:sex>
         <myns:spouse xlink:href="#p4467"/>
         <myns:location xlink:href="#p102"/>
         <myns:mailAddress>
            <myns:Address gml:id="a201">
               <myns:streetName>Main St.</myns:streetName>
               <myns:streetNumber>5</myns:streetNumber>
               <myns:city>SomeCity</myns:city>
               <myns:province>Someprovince</myns:province>
               <myns:postalCode>X1X 1X1</myns:postalCode>
               <myns:country>Canada</myns:country>
            </myns:Address>
         </myns:mailAddress>
         <myns:phone>416-123-4567</myns:phone>
         <myns:phone>416-890-1234</myns:phone>
      </myns:Person>

      <myns:Car gml:id="r1432">
         <gml:identifier codeSpace="http://www.carserial.org">51465243</gml:identifier>
         <myns:model>Ford Pinto</myns:model>
         <myns:age>4</myns:age>
         <myns:colour>red</myns:colour>
         <myns:location>
            <gml:Point gml:id="p102">
               <gml:pos>15 15</gml:pos>
            </gml:Point>
         </myns:location>
      </myns:Car>

      <myns:House gml:id="h32">
         <gml:identifier codeSpace="http://www.google.org/houses.xml">654365143</gml:identifier>
         <myns:numFloors>2</myns:numFloors>
         <myns:area uom="sqm">200</myns:area>
         <myns:location>
            <gml:Point gml:id="p101">
```

```
            <gml:pos>16 18</gml:pos>
         </gml:Point>
      </myns:location>
      <myns:frontsOn xlink:href="#rs11"/>
      <myns:address xlink:href="#a201"/>
   </myns:House>

   <abc:Road gml:id="rs11">
      <abc:numLanes>3</abc:numLanes>
      <abc:centerline>
         <gml:LineString gml:id="l123">?</gml:LineString>
      </abc:centerline>
   </abc:Road>

   <abc:RoadNetwork gml:id="rn202">
      <abc:operator>RTA</abc:operator>
      <abc:members xlink:href="#rs11"/>
      <abc:topology>
         <gml:TopoComplex>
            <gml:Edge gml:id="e1">
               <gml:pointProperty xlink:href="#l123"/>
            </gml:Edge>
         </gml:TopoComplex>
      </abc:topology>
   </abc:RoadNetwork>

   <myns:Person gml:id="p4467">
      <gml:identifier codeSpace="http://www.canadaSIN.com">424679360</gml:identifier>
      <myns:lastName>Smith</myns:lastName>
      <myns:firstName>Mary</myns:firstName>
      <myns:age>31</myns:age>
      <myns:sex>Female</myns:sex>
      <myns:spouse xlink:href="#p4456"/>
      <myns:location xlink:href="#p101"/>
      <myns:mailAddress xlink:href="#a201"/>
      <myns:phone>416-123-4567</myns:phone>
      <myns:phone>416-890-1234</myns:phone>
      <myns:livesIn xlink:href="#h32"/>
      <myns:isDriving xlink:href="r1432"/>
   </myns:Person>
```

B.4.2 Example 1

The following example finds the location of Fred Smith using a GetFeature request. The use of the resolve and resolveDepth parameters instruct the web feature service to resolve any local references.

```
<?xml version="1.0" ?>
<GetFeature
   service="WFS"
   version="2.0.0"
   resolveDepth="*"
   outputFormat="application/xml; subtype=gml/3.2"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:myns="http://www.myserver.com/myns"
   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/Common/SampleSchema.xsd">
   <Query typeNames="myns:Person">
      <PropertyName resolve="local">myns:location</PropertyName>
      <fes:Filter>
         <fes:And>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>myns:firstName</fes:ValueReference>
               <fes:Literal>Fred</fes:Literal>
            </fes:PropertyIsEqualTo>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>myns:lastName</fes:ValueReference>
               <fes:Literal>Smith</fes:Literal>
            </fes:PropertyIsEqualTo>
         </fes:And>
      </fes:Filter>
```

```
   </Query>
</GetFeature>
```

The response document is:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:ValueCollection timeStamp="2008-09-07T19:00:00" numberReturned="2"
   numberMatched="unknown" xmlns="http://www.someserver.com/myns"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:abc="http://www.someserver.com/abc"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   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 ./myns.xsd
                       http://www.someserver.com/abc ./abc.xsd">
   <wfs:member>
      <myns:Person gml:id="p4456">
         <gml:identifier codeSpace="http://www.canadaSIN.com">424679374</gml:identifier>
         <myns:lastName>Smith</myns:lastName>
         <myns:firstName>Fred</myns:firstName>
         <myns:age>35</myns:age>
         <myns:sex>male</myns:sex>
         <myns:spouse xlink:href="#p4467"/>
         <myns:location xlink:href="#pt102"/>
         <myns:mailAddress>
            <myns:Address gml:id="a201">
               <myns:streetName>Main St.</myns:streetName>
               <myns:streetNumber>5</myns:streetNumber>
               <myns:city>SomeCity</myns:city>
               <myns:province>Someprovince</myns:province>
               <myns:postalCode>X1X 1X1</myns:postalCode>
               <myns:country>Canada</myns:country>
            </myns:Address>
         </myns:mailAddress>
         <myns:phone>416-123-4567</myns:phone>
         <myns:phone>416-890-1234</myns:phone>
      </myns:Person>
   </wfs:member>
   <wfs:additionalValues>
      <wfs:ValueCollection
         timeStamp="2008-09-07T19:00:00"
         numberReturned="2"
         numberMatched="2">
         <wfs:member>
            <myns:Person gml:id="p4467">
               <gml:identifier codeSpace="http://www.canadaSIN.com">424679360</gml:identifier>
               <myns:lastName>Smith</myns:lastName>
               <myns:firstName>Mary</myns:firstName>
               <myns:age>18</myns:age>
               <myns:sex>Female</myns:sex>
               <myns:spouse xlink:href="#p4456"/>
               <myns:location xlink:href="#p101"/>
               <myns:mailAddress xlink:href="#a201"/>
               <myns:phone>416-123-4567</myns:phone>
               <myns:phone>416-890-1234</myns:phone>
               <myns:livesIn xlink:href="#h32"/>
               <myns:isDriving xlink:href="r1432"/>
            </myns:Person>
         </wfs:member>
         <wfs:member>
            <myns:Car gml:id="r1432">
               <gml:identifier codeSpace="http://www.carserial.org">51465243</gml:identifier>
               <myns:model>Ford Pinto</myns:model>
               <myns:age>4</myns:age>
               <myns:colour>red</myns:colour>
               <myns:location>
                  <gml:Point gml:id="pt102">
                     <gml:pos>-59.603958 -52.106559</gml:pos>
                  </gml:Point>
               </myns:location>
            </myns:Car>
         </wfs:member>
      </wfs:ValueCollection>
```

```
   </wfs:additionalValues>
</wfs:ValueCollection>
```

The response documents contain Fred Smith's myns:Person feature, and the wfs:additionalObjects section contains the resolved location reference and spouse reference. The resolved location reference identifies Fred Smith's location.

B.4.3 Example 2

The following operation performs the same query as Example 1 (see B.4.2), but instead uses the GetPropertyValue operation.

```
<?xml version="1.0"?>
<GetPropertyValue
   service="WFS"
   version="2.0.0"
   valueReference="myns:location"
   resolve="local"
   resolveDepth="*"
   outputFormat="application/xml; subtype=gml/3.2"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:myns="http://www.myserver.com/myns"
   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 ./myns.xsd">
   <Query typeNames="myns:Person">
      <fes:Filter>
         <fes:And>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>myns:firstName</fes:ValueReference>
               <fes:Literal>Fred</fes:Literal>
            </fes:PropertyIsEqualTo>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>myns:lastName</fes:ValueReference>
               <fes:Literal>Smith</fes:Literal>
            </fes:PropertyIsEqualTo>
         </fes:And>
      </fes:Filter>
   </Query>
</GetPropertyValue>
```

The response document contains Fred Smith's location:

```
<?xml version="1.0" ?>
<wfs:ValueCollection
   timeStamp="2008-09-07T19:00:00"
   numberReturned="1"
   numberMatched="unknown"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   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">
   <wfs:member>
      <gml:Point gml:id="pt102">
         <gml:pos>-59.603958 -52.106559</gml:pos>
      </gml:Point>
   </wfs:member>
</wfs:ValueCollection>
```

B.4.4 Example 3

The following GetFeature request uses a join of three feature types to find the number of lanes of the road that fronts the house that Mary Smith lives in.

```
<?xml version="1.0" ?>
<GetFeature
   service="WFS"
```

```
   version="2.0.0"
   outputFormat="application/xml; subtype=gml/3.2"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:abc="http://www.myserver.com/abc"
   xmlns:myns="http://www.myserver.com/myns"
   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 ./myns.xsd
                       http://www.someserver.com/abc ./abc.xsd">
   <Query typeNames="myns:Person myns:House abc:Road">
     <PropertyName resolve="local">abc:numberOfLanes</PropertyName>
     <fes:Filter>
        <fes:And>
           <fes:PropertyIsEqualTo>
              <fes:ValueReference>myns:firstName</fes:ValueReference>
              <fes:Literal>Mary</fes:Literal>
           </fes:PropertyIsEqualTo>
           <fes:PropertyIsEqualTo>
              <fes:ValueReference>myns:lastName</fes:ValueReference>
              <fes:Literal>Smith</fes:Literal>
           </fes:PropertyIsEqualTo>
           <fes:PropertyIsEqualTo>
              <fes:ValueReference>myns:Person/valueOf(myns:livesIn)/@gml:id</fes:ValueReference>
              <fes:ValueReference>myns:House/@gml:id</fes:ValueReference>
           </fes:PropertyIsEqualTo>
           <fes:PropertyIsEqualTo>
              <fes:ValueReference>myns:House/valueOf(abc:frontsOn)/@gml:id</fes:ValueReference>
              <fes:ValueReference>abc:Road/@gml:id</fes:ValueReference>
           </fes:PropertyIsEqualTo>
        </fes:And>
      </fes:Filter>
   </Query>
</GetFeature>
```

The response to this request is a tuple that satisfies the join predicate:

```
<?xml version="1.0" ?>
<wfs:FeatureCollection
   timeStamp="2008-09-07T19:00:00"
   numberReturned="1"
   numberMatched="unknown"
   xmlns="http://www.someserver.com/myns"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:abc="http://www.someserver.com/abc"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   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 ./myns.xsd
                       http://www.someserver.com/abc ./abc.xsd">
      <wfs:member>
   <wfs:Tuple>
      <wfs:member>
         <myns:Person  gml:id="p4467">
            <gml:identifier
            codeSpace="http://www.canadaSIN.com">424679360</gml:identifier>
            <myns:lastName>Smith</myns:lastName>
            <myns:firstName>Mary</myns:firstName>
            <myns:age>18</myns:age>
            <myns:sex>female</myns:sex>
            <myns:spouse xlink:href="#p4456"/>
            <myns:location xlink:href="#pt101" />
            <myns:mailAddress>
               <myns:Address gml:id="a201">
                  <myns:streetName>Main St.</myns:streetName>
                  <myns:streetNumber>5</myns:streetNumber>
                  <myns:city>SomeCity</myns:city>
                  <myns:province>Someprovince</myns:province>
                  <myns:postalCode>X1X 1X1</myns:postalCode>
                  <myns:country>Canada</myns:country>
               </myns:Address>
            </myns:mailAddress>
```

```
            <myns:phone>416-123-4567</myns:phone>
            <myns:phone>416-890-4532</myns:phone>
            <myns:livesIn xlink:href="#h32"/>
            <myns:isDriving xlink:href="#r1432"/>
         </myns:Person>
      </wfs:member>
      <wfs:member>
         <myns:House gml:id="h32">
            <gml:identifier
            codeSpace="http://www.toronto.ca/reg.xml">654365143</gml:identifier>
            <myns:numFloors>2</myns:numFloors>
            <myns:area uom="sqm">200</myns:area>
            <myns:location>
               <gml:Point gml:id="pt101">
                  <gml:pos>16 18</gml:pos>
               </gml:Point>
            </myns:location>
            <myns:frontsOn xlink:href="#rs11"/>
            <myns:address xlink:href="#a201"/>
         </myns:House>
      </wfs:member>
      <wfs:member>
         <abc:Road gml:id="rs11">
            <abc:numLanes>3</abc:numLanes>
            <abc:centerLineOf>
               <gml:LineString gml:id="GID_5" srsName="urn:ogc:def:crs:EPSG::4326">
                  <gml:posList>-59.478340 -52.226578 -59.484871 -52.223564 -59.488991 -52.198524 -
        59.485958 -52.169559 -59.480400 -52.152615 -59.465576 -52.141491 -59.462002 -52.136417 -
        59.447968 -52.127190 -59.422928 -52.120701 -59.411915 -52.117844 -59.397972 -52.116440 -
        59.371311 -52.121300</gml:posList>
               </gml:LineString>
            </abc:centerLineOf>
         </abc:Road>
      </wfs:member>
   </wfs:Tuple>
      </wfs:member>
</wfs:FeatureCollection>
```

The response document shows that the road that fronts Mary Smith's house has three lanes.

B.4.5 Example 4

The following GetPropertyValue operation answers the same question as the previous example (see B.4.4). The wfs:Query action identifies Mary Smith's house and the path expression in the valueReference attribute retrieves the desired abc:numLanes value. The use of the valueOf() operator in the path expression de-references the myns:livesIn and myns:frontsOn references.

```
<?xml version="1.0" ?>
<GetPropertyValue
   service="WFS"
   version="2.0.0"
   valueReference="valueOf(myns:livesIn)/valueOf(myns:frontsOn)/abc:numLanes"
   outputFormat="application/xml; subtype=gml/3.2"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:myns="http://www.myserver.com/myns"
   xmlns:abc="http://www.myserver.com/abc"
   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 ./myns.xsd
                       http://www.someserver.com/abc ./abc.xsd">
   <Query typeNames="myns:Person">
      <fes:Filter>
         <fes:And>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>myns:firstName</fes:ValueReference>
               <fes:Literal>Fred</fes:Literal>
            </fes:PropertyIsEqualTo>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>myns:lastName</fes:ValueReference>
               <fes:Literal>Smith</fes:Literal>
            </fes:PropertyIsEqualTo>
```

```
        </fes:And>
      </fes:Filter>
   </Query>
</GetPropertyValue>
```

The response document:

```
<?xml version="1.0" ?>
<wfs:ValueCollection
   timeStamp="2008-09-07T19:00:00"
   numberReturned="1"
   numberMatched="1"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   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">
   <wfs:member>3</wfs:member>
</wfs:ValueCollection>
```

shows that the road that fronts Mary Smith's house has three lanes.

B.4.6 Example 5

The following GetFeature request uses a join to find the road that fronts each house. The join predicate uses the valueOf() operator to resolve the "frontsOn" reference to get the gml:id of each corresponding road.

```
<?xml version="1.0" ?>
<GetFeature
   service="WFS"
   version="2.0.0"
   outputFormat="application/xml; subtype=gml/3.2"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:myns="http://www.myserver.com/myns"
   xmlns:abc="http://www.myserver.com/abc"
   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 ./myns.xsd
                       http://www.someserver.com/abc ./abc.xsd">
   <Query typeNames="myns:House abc:Road">
     <fes:Filter>
         <fes:PropertyIsEqualTo>
            <fes:ValueReference>myns:House/valueOf(myns:frontsOn)/@gml:id</fes:ValueReference>
            <fes:ValueReference>abd:Road/@gml:id</fes:ValueReference>
         </fes:PropertyIsEqualTo>
      </fes:Filter>
   </Query>
</GetFeature>
```

The response is a set of house-road tuples showing which road fronts each house:

```
<?xml version="1.0" ?>
<wfs:FeatureCollection
   timeStamp="2008-09-07T19:00:00"
   numberReturned="1"
   numberMatched="unknown"
   xmlns="http://www.someserver.com/myns"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:abc="http://www.someserver.com/abc"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   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 ./myns.xsd
                       http://www.someserver.com/abc ./abc.xsd">
   <wfs:member>
   <wfs:Tuple>
      <wfs:member>
```

```
         <myns:House gml:id="h32">
            <gml:identifier
            codeSpace="http://www.toronto.ca/reg.xml">654365143</gml:identifier>
            <myns:numFloors>2</myns:numFloors>
            <myns:area uom="sqm">200</myns:area>
            <myns:location>
               <gml:Point gml:id="pt101">
                  <gml:pos>16 18</gml:pos>
               </gml:Point>
            </myns:location>
            <myns:frontsOn xlink:href="#rs11"/>
            <myns:address xlink:href="#a201"/>
         </myns:House>
      </wfs:member>
      <wfs:member>
         <abc:Road gml:id="rs11">
            <abc:numLanes>3</abc:numLanes>
            <abc:centerLineOf>
               <gml:LineString gml:id="GID_5" srsName="urn:ogc:def:crs:EPSG::4326">
                  <gml:posList>-59.478340 -52.226578 -59.484871 -52.223564 -59.488991 -52.198524 -
        59.485958 -52.169559 -59.480400 -52.152615 -59.465576 -52.141491 -59.462002 -52.136417 -
        59.447968 -52.127190 -59.422928 -52.120701 -59.411915 -52.117844 -59.397972 -52.116440 -
        59.371311 -52.121300</gml:posList>
               </gml:LineString>
            </abc:centerLineOf>
         </abc:Road>
      </wfs:member>
   </wfs:Tuple>
   </wfs:member>
</wfs:FeatureCollection>
```

B.4.7 Example 6

The following GetPropertyValue request retrieves Mary Smith's postal code. The wfs:Query action retrieves Mary Smith's myns:Person feature and the XPath (see W3C XML Path Language) that is the value of the valueReference parameter retrieves the postal code. The valueOf() operator resolves any myns:livesIn and myns:mailAddress references while evaluating the XPath expression.

```
<?xml version="1.0"?>
<GetPropertyValue
   service="WFS"
   version="2.0.0"
   valueReference="valueOf(myns:livesIn)/valueof(myns:mailAddress)/myns:postalCode"
   outputFormat="application/xml; subtype=gml/3.2"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:myns="http://www.myserver.com/myns"
   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 ./myns.xsd">
   <Query typeNames="myns:Person">
      <fes:Filter>
         <fes:And>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>myns:firstName</fes:ValueReference>
               <fes:Literal>Mary</fes:Literal>
            </fes:PropertyIsEqualTo>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>myns:lastName</fes:ValueReference>
               <fes:Literal>Smith</fes:Literal>
            </fes:PropertyIsEqualTo>
         </fes:And>
      </fes:Filter>
   </Query>
</GetPropertyValue>
```

The following response document shows that Mary Smith's postal code is X1X 1X1:

```
<?xml version="1.0" ?>
<wfs:ValueCollection
   timeStamp="2008-09-07T19:00:00"
   numberReturned="1"
```

```
   numberMatched="unknown"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   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">
   <wfs:member>X1X 1X1</wfs:member>
</wfs:ValueCollection>
```

B.4.8 Example 7

The following request obtains Mary Smith's age:

```
<?xml version="1.0" ?>
<GetPropertyValue
   service="WFS"
   version="2.0.0"
   valueReference="myns:age"
   outputFormat="application/xml; subtype=gml/3.2"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:myns="http://www.myserver.com/myns"
   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 ./myns.xsd">
   <Query typeNames="myns:Person">
      <fes:Filter>
         <fes:And>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>myns:firstName</fes:ValueReference>
               <fes:Literal>Mary</fes:Literal>
            </fes:PropertyIsEqualTo>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>myns:lastName</fes:ValueReference>
               <fes:Literal>Smith</fes:Literal>
            </fes:PropertyIsEqualTo>
         </fes:And>
      </fes:Filter>
   </Query>
</GetPropertyValue>
```

The following response document shows that Mary Smith's age is 31:

```
<?xml version="1.0" ?>
<wfs:ValueCollection
   timeStamp="2008-09-07T19:00:00"
   numberReturned="1"
   numberMatched="unknown"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   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">
   <wfs:member>31</wfs:member>
</wfs:ValueCollection>
```

B.4.9 Example 8

The following GetPropertyValue request fetches all of Fred Smith's phone numbers. The query expression fetches Fred Smith's myns:Person feature and the valueReference XPath expression gets the phone numbers from the feature.

```
<?xml version="1.0" ?>
<GetPropertyValue
   service="WFS"
   version="2.0.0"
   valueReference="myns:phone"
   outputFormat="application/xml; subtype=gml/3.2"
   xmlns="http://www.opengis.net/wfs/2.0"
```

```
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:myns="http://www.myserver.com/myns"
   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 ./myns.xsd">
   <Query typeNames="myns:Person">
      <fes:Filter>
         <fes:And>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>myns:firstName</fes:ValueReference>
               <fes:Literal>Fred</fes:Literal>
            </fes:PropertyIsEqualTo>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>myns:lastName</fes:ValueReference>
               <fes:Literal>Smith</fes:Literal>
            </fes:PropertyIsEqualTo>
         </fes:And>
      </fes:Filter>
   </Query>
</GetPropertyValue>
```

The response document, containing all the phone numbers, is:

```
<?xml version="1.0" ?>
<wfs:ValueCollection
   timeStamp="2008-09-07T19:00:00"
   numberReturned="2"
   numberMatched="2"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   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">
   <wfs:member>416-123-4567</wfs:member>
   <wfs:member>416-890-1234</wfs:member>
</wfs:ValueCollection>
```

B.4.10 Example 9

The following request retrieves Fred Smith's second phone number. The wfs:Query action retrieves Fred Smith's myns:Person records and the XPath expression myns:phone[2] gets the second phone number from the feature.

```
<?xml version="1.0" ?>
<GetPropertyValue
   service="WFS"
   version="2.0.0"
   valueReference="myns:phone[2]"
   outputFormat="application/xml; subtype=gml/3.2"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:myns="http://www.myserver.com/myns"
   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 ./myns.xsd">
   <Query typeNames="myns:Person">
      <fes:Filter>
         <fes:And>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>myns:firstName</fes:ValueReference>
               <fes:Literal>Fred</fes:Literal>
            </fes:PropertyIsEqualTo>
            <fes:PropertyIsEqualTo>
               <fes:ValueReference>myns:lastName</fes:ValueReference>
               <fes:Literal>Smith</fes:Literal>
            </fes:PropertyIsEqualTo>
         </fes:And>
      </fes:Filter>
   </Query>
</GetPropertyValue>
```

The response document contains the second phone number:

```
<?xml version="1.0" ?>
<wfs:ValueCollection
   timeStamp="2008-09-07T19:00:00"
   numberReturned="1"
   numberMatched="1"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   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">
   <wfs:member>416-890-1234</wfs:member>
</wfs:ValueCollection>
```

B.5 LockFeature examples

B.5.1 Example 1

Lock a set of enumerated features. The WFS is, in this case, directed to try and lock as many features as it can.

Request:

```
<?xml version="1.0"?>
<LockFeature
   version="2.0.0"
   service="WFS"
   lockAction="SOME"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:myns="http://www.someserver.com/myns"
   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">
   <Query typeNames="myns:InWaterA_1M">
      <fes:Filter>
         <fes:ResourceId rid="InWaterA_1M.1013"/>
         <fes:ResourceId rid="InWaterA_1M.1014"/>
         <fes:ResourceId rid="InWaterA_1M.1015"/>
         <fes:ResourceId rid="InWaterA_1M.1016"/>
         <fes:ResourceId rid="InWaterA_1M.1017"/>
      </fes:Filter>
   </Query>
</LockFeature>
```

Sample response:

```
<?xml version="1.0" ?>
<LockFeatureResponse
   lockId="1"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   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">
   <FeaturesLocked>
      <fes:ResourceId rid="InWaterA_1M.1013"/>
      <fes:ResourceId rid="InWaterA_1M.1014"/>
      <fes:ResourceId rid="InWaterA_1M.1016"/>
      <fes:ResourceId rid="InWaterA_1M.1017"/>
   </FeaturesLocked>
   <FeaturesNotLocked>
      <fes:ResourceId rid="InWaterA_1M.1015"/>
   </FeaturesNotLocked>
</LockFeatureResponse>
```

B.5.2 Example 2

Lock all the feature instances of type InWaterA_1M.

Request:

```
<?xml version="1.0" ?>
<LockFeature
   version="2.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:myns="http://www.someserver.com/myns"
   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">
   <Query typeNames="myns:InWaterA_1M"/>
</LockFeature>
```

Sample response:

```
<?xml version="1.0" ?>
<LockFeatureResponse
   lockId="2"
   xmlns="http://www.opengis.net/wfs/2.0"
   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"/>
```

B.5.3 Example 3

In this example, a filter expression using a spatial constraint is used to identify the set of feature instances to be locked.

Request:

```
<?xml version="1.0" ?>
<LockFeature
   version="2.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:myns="http://www.someserver.com/myns"
   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.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <Query handle="Lock1" typeNames="myns:InWaterA_1M">
      <fes:Filter>
         <fes:Within>
            <fes:ValueReference>myns:wkbGeom</fes:ValueReference>
            <gml:Polygon gml:id="GID_45"
                  srsName="urn:ogc:def:crs:EPSG::4326">
               <gml:exterior>
                  <gml:LinearRing>
                     <gml:posList>-95.7 38.1 -97.8 38.2 -98.9 38.2 -95.7 38.1 </gml:posList>
                  </gml:LinearRing>
               </gml:exterior>
            </gml:Polygon>
         </fes:Within>
      </fes:Filter>
   </Query>
</LockFeature>
```

Sample response:

```
<?xml version="1.0"?>
<LockFeatureResponse
   lockId="A1014375BD"
   xmlns="http://www.opengis.net/wfs/2.0"
   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"/>
```

B.5.4 Example 4

This example locks features of type BuiltUpA_1M and InWaterA_1M. The lock labelled with the handle *LOCK1* locks all the features inside the defined window. The lock labelled with the handle *LOCK2* locks the features InWaterA_1M.1212, InWaterA_1M.1213 and InWaterA_1M.10.

Request:

```
<?xml version="1.0"?>
<LockFeature
   version="2.0.0"
   service="WFS"
   expiry="4"
   lockAction="SOME"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:myns="http://www.someserver.com/myns"
   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.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <Query handle="LOCK1" typeNames="myns:BuiltUpA_1M">
      <fes:Filter>
         <fes:Within>
            <fes:ValueReference>BuiltUpA_1M/wkbGeom</fes:ValueReference>
            <gml:Polygon gml:id="GID_71" srsName="urn:ogc:def:crs:EPSG::4326">
               <gml:exterior>
                  <gml:LinearRing>
                     <gml:posList>-72.4343981 18.4774721 -72.4344232 18.4775742 -72.4343699 18.477586
      -72.4343448 18.4774839 -72.4343981 18.4774721</gml:posList>
                  </gml:LinearRing>
               </gml:exterior>
            </gml:Polygon>
         </fes:Within>
      </fes:Filter>
   </Query>
   <Query handle="LOCK2" typeNames="myns:InWaterA_1M">
      <fes:Filter>
         <fes:ResourceId rid="InWaterA_1M.1212"/>
         <fes:ResourceId rid="InWaterA_1M.1213"/>
         <fes:ResourceId rid="InWaterA_1M.10"/>
      </fes:Filter>
   </Query>
</LockFeature>
```

Sample response:

```
<?xml version="1.0" ?>
<LockFeatureResponse
   lockId="LOCK1A"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   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">
   <FeaturesLocked>
      <fes:ResourceId rid="BuiltUpA_1M.1"/>
      <fes:ResourceId rid="BuiltUpA_1M.10"/>
      <fes:ResourceId rid="BuiltUpA_1M.34"/>
      <fes:ResourceId rid="BuiltUpA_1M.786"/>
      <fes:ResourceId rid="BuiltUpA_1M.3"/>
      <fes:ResourceId rid="BuiltUpA_1M.13"/>
      <fes:ResourceId rid="BuiltUpA_1M.47563"/>
      <fes:ResourceId rid="InWaterA_1M.1212"/>
      <fes:ResourceId rid="InWaterA_1M.1213"/>
      <fes:ResourceId rid="InWaterA_1M.10"/>
   </FeaturesLocked>
</LockFeatureResponse>
```

B.6 Transaction examples

B.6.1 Insert example

The following transaction creates two instances of feature type InWaterA_1M.

```
<?xml version="1.0"?>
<wfs:Transaction
   version="2.0.0"
   service="WFS"
   xmlns="http://www.someserver.com/myns"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.someserver.com/myns ./SampleSchema.xsd
                       http://www.opengis.net/wfs/2.0
                       http://schemas.opengis.net/wfs/2.0.0/wfs.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:Insert>
    <InWaterA_1M gml:id="F1">
      <wkbGeom>
        <gml:Polygon srsName="urn:ogc:def:crs:EPSG::4326" gml:id="P1">
          <gml:exterior>
            <gml:LinearRing>
              <gml:posList>-30.93597221374512 117.6290588378906 -30.94830513000489 117.6447219848633
        -30.95219421386719 117.6465530395508 -30.95219421386719 117.6431121826172 -30.94802856445312
        117.6386108398438 -30.94799995422363 117.6314163208008 -30.946138381958 117.62850189209 -
        30.94430541992188 117.6295852661133 -30.93280601501464 117.6240539550781 -30.92869377136231
        117.624641418457 -30.92386054992676 117.6201400756836 -30.92111206054688 117.6206970214844 -
        30.92458343505859 117.6275863647461 -30.93597221374512 117.6290588378906</gml:posList>
            </gml:LinearRing>
          </gml:exterior>
        </gml:Polygon>
      </wkbGeom>
      <id>28022</id>
      <fCode>BH000</fCode>
      <hyc>6</hyc>
      <tileId>177</tileId>
      <facId>132</facId>
    </InWaterA_1M>
    <InWaterA_1M gml:id="F2">
      <wkbGeom>
        <gml:Polygon srsName="urn:ogc:def:crs:EPSG::4326" gml:id="P2">
          <gml:exterior>
            <gml:LinearRing>
               <gml:posList>-30.92013931274414 117.6552810668945 -30.92383384704589 117.661361694336
        -30.9300556182861 3 117.6666412353516 -30.93280601501464 117.6663589477539 -30.93186187744141
        117.6594467163086 -30.93780517578125 117.6541137695312 -30.94397163391114 117.6519470214844 -
        30.94255638122559 117.6455535888672 -30.93402862548828 117.6336364746094 -30.92874908447266
        117.6355285644531 -30.92138862609864 117.6326370239258 -30.92236137390137 117.6395568847656 -
        30.91708374023438 117.6433029174805 -30.91711044311523 117.6454467773437 -30.92061042785645
        117.6484985351563 -30.92061042785645 117.6504135131836 -30.91638946533203 117.6504440307617 -
        30.92013931274414 117.6552810668945 </gml:posList>
            </gml:LinearRing>
          </gml:exterior>
        </gml:Polygon>
      </wkbGeom>
      <id>28021</id>
      <fCode>BH000</fCode>
      <hyc>6</hyc>
      <tileId>177</tileId>
      <facId>131</facId>
    </InWaterA_1M>
   </wfs:Insert>
</wfs:Transaction>
```

In this example, the schemaLocation attribute references a static XML Schema document (SampleSchema.xsd) that declares the InWatera_1M element. This declaration of the InWatera_1M element could just as easily have been dynamically referenced using the DescribeFeatureType operation (see Clause 9).

B.6.2 Update examples

B.6.2.1 Example 1

The following example updates the population property of the feature identified by the feature identifier BuiltUpA_1M.1013.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="2.0.0"
   service="WFS"
   xmlns="http://www.someserver.com/myns"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   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">
   <wfs:Update typeName="BuiltUpA_1M">
      <wfs:Property>
         <wfs:ValueReference>population</wfs:ValueReference>
         <wfs:Value>4070000</wfs:Value>
      </wfs:Property>
      <fes:Filter>
         <fes:ResourceId rid="BuiltUpA_1M.10131"/>
      </fes:Filter>
   </wfs:Update>
</wfs:Transaction>
```

B.6.2.2 Example 2

Update the *populationType* property of an enumerated set of features. In this example, the features are identified by feature identifiers:

```
BuiltUpA_1M.1013
BuiltUpA_1M.34
BuiltUpA_1M.24256
```

have their *populationType* attribute set to the value "CITY".

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="2.0.0"
   service="WFS"
   xmlns="http://www.someserver.com/myns"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   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">
   <wfs:Update typeName="BuiltUpA_1M">
      <wfs:Property>
         <wfs:ValueReference>populationType</wfs:ValueReference>
         <wfs:Value>CITY</wfs:Value>
      </wfs:Property>
      <fes:Filter>
         <fes:ResourceId rid="BuiltUpA_1M.1013"/>
         <fes:ResourceId rid="BuiltUpA_1M.34"/>
         <fes:ResourceId rid="BuiltUpA_1M.24256"/>
      </fes:Filter>
   </wfs:Update>
</wfs:Transaction>
```

B.6.2.3 Example 3

Update the *name* property of an enumerated set of features, and update the *FAC_ID* property of another set of features defined by constraining the value of the *tileId* property to values greater than 1 000.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="2.0.0"
```

```
   service="WFS"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   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">
   <wfs:Update typeName="myns:BuiltUpA_1M">
      <wfs:Property>
         <wfs:ValueReference>myns:name</wfs:ValueReference>
         <wfs:Value>somestring</wfs:Value>
      </wfs:Property>
      <fes:Filter>
         <fes:ResourceId rid="BuiltUpA_1M.1013"/>
         <fes:ResourceId rid="BuiltUpA_1M.34"/>
         <fes:ResourceId rid="BuiltUpA_1M.24256"/>
      </fes:Filter>
   </wfs:Update>
   <wfs:Update typeName="myns:BuiltUpA_1M">
      <wfs:Property>
         <wfs:ValueReference>myns:facId</wfs:ValueReference>
         <wfs:Value>100</wfs:Value>
      </wfs:Property>
      <fes:Filter>
         <fes:PropertyIsGreaterThan>
            <fes:ValueReference>BuiltUpA_1M/tileId</fes:ValueReference>
            <fes:Literal>1000</fes:Literal>
         </fes:PropertyIsGreaterThan>
      </fes:Filter>
   </wfs:Update>
</wfs:Transaction>
```

B.6.2.4 Example 4

This example updates two feature classes, OceansA_1M and TreesA_1MTreesA_1M. All features of OceansA_1M with a depth greater than 2 400 m are updated and feature TreesA_1MTreesA_1M.1010 is also updated.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="2.0.0"
   service="WFS"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:myns="http://www.someserver.com/myns"
   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">
   <wfs:Update typeName="myns:OceansA_1M">
      <wfs:Property>
         <wfs:ValueReference>myns:depth</wfs:ValueReference>
         <wfs:Value>2400</wfs:Value>
      </wfs:Property>
      <fes:Filter>
         <fes:PropertyIsGreaterThan>
            <fes:ValueReference>OceansA_1M/depth</fes:ValueReference>
            <fes:Literal>2400</fes:Literal>
         </fes:PropertyIsGreaterThan>
      </fes:Filter>
   </wfs:Update>
   <wfs:Update typeName="myns:TreesA_1M">
      <wfs:Property>
         <wfs:ValueReference>myns:treeType</wfs:ValueReference>
         <wfs:Value>CONIFEROUS</wfs:Value>
      </wfs:Property>
      <fes:Filter>
         <fes:ResourceId rid="TreesA_1M.1010"/>
      </fes:Filter>
   </wfs:Update>
</wfs:Transaction>
```

B.6.3 Delete examples

B.6.3.1 Example 1

Delete a single feature.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="2.0.0"
   service="WFS"
   xmlns="http://www.someserver.com/myns"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   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">
   <wfs:Delete typeName="InWaterA_1M">
      <fes:Filter>
         <fes:ResourceId rid="InWaterA_1M.1013"/>
      </fes:Filter>
   </wfs:Delete>
</wfs:Transaction>
```

B.6.3.2 Example 2

This example deletes an enumerated set of feature instances.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="2.0.0"
   service="WFS"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   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">
   <wfs:Delete typeName="myns:InWaterA_1M">
      <fes:Filter>
         <fes:ResourceId rid="InWaterA_1M.1013"/>
         <fes:ResourceId rid="InWaterA_1M.10"/>
         <fes:ResourceId rid="InWaterA_1M.13"/>
         <fes:ResourceId rid="InWaterA_1M.140"/>
         <fes:ResourceId rid="InWaterA_1M.5001"/>
         <fes:ResourceId rid="InWaterA_1M.2001"/>
      </fes:Filter>
   </wfs:Delete>
</wfs:Transaction>
```

B.6.3.3 Example 3

This examples deletes the set of feature instances of feature type InWaterA_1M that lie inside a region defined by a polygon specified in the predicate. The fes:Filter element is used to constrain the scope of the operation, and GML is used to express the geometry of the polygon.

```
<?xml version="1.0"?>
<wfs:Transaction
   version="2.0.0"
   service="WFS"
   xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   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.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:Delete typeName="InWaterA_1M">
      <fes:Filter>
```

```
         <fes:Within>
            <fes:ValueReference>wkbGeom</fes:ValueReference>
            <gml:Polygon srsName="urn:ogc:def:crs:EPSG::4326" gml:id="pp9">
               <gml:exterior>
                  <gml:LinearRing>
                  <gml:posList>-30.15600013732911 115.0352249145508 -30.17819404602051
        115.0252532958984 -30.16072273254395 115.021614074707 -30.1563892364502 115.0222473144531 -
        30.15555572509765 115.0259704589844 -30.1512508392334 115.0282211303711 -30.14588928222656
        115.0344696044922 -30.15600013732911 115.0352249145508 </gml:posList>
                  </gml:LinearRing>
               </gml:exterior>
            </gml:Polygon>
         </fes:Within>
      </fes:Filter>
   </wfs:Delete>
</wfs:Transaction>
```

B.6.4 Mixed transaction example

This example defines a complex transaction, labelled "Transaction 01", that performs insert, update and delete operations. Some of the feature types include complex properties, and XPath expressions are used in the filter expressions to unambiguously reference the desired properties. Comments contained in the example explain the various operations.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="2.0.0"
   service="WFS"
   handle="Transaction 01"
   xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   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.someserver.com/myns ./SampleSchema.xsd
                       http://www.opengis.net/wfs/2.0
                       http://schemas.opengis.net/wfs/2.0.0/wfs.xsd
                       http://www.opengis.net/gml/3.2
                       http://www.opengis.net/gml/3.2.1/gml.xsd">

   <wfs:Native vendorId="BigDbCorp" safeToIgnore="true">
      ALTER SESSION ENABLE PARALLEL DML;
      <bdbc:SomeXmlCommand xmlns:bdbc="http://www.bigdbcorp.com">
         <bdbc:Command>allow</bdbc:Command>
      </bdbc:SomeXmlCommand>
   </wfs:Native>

   <!-- Create a new instance of feature type ELEVP_1M -->

   <wfs:Insert handle="Statement 1">
      <ElevP_1M gml:id="E1">
         <location>
            <gml:Point srsName="urn:ogc:def:crs:EPSG::4326" gml:id="e33">
               <gml:pos>24.2633 -68.5485</gml:pos>
            </gml:Point>
         </location>
         <id>167928</id>
         <fCode>CA030</fCode>
         <acc>2</acc>
         <ela>1</ela>
         <zv2>29999</zv2>
         <tileId>250</tileId>
         <endId>111</endId>
      </ElevP_1M>
   </wfs:Insert>
   <!-- Create a new instance of feature type RoadL_1M
        which has complex properties segment and roadType. -->
   <wfs:Insert handle="ComplexInsert">
      <RoadL_1M gml:id="R1">
         <name>Highway 401</name>
         <segment>
            <SegmentDesc>
               <designation>SEG_A41</designation>
               <geometry>
```

```
                <gml:LineString gml:id="e3"
                   srsName="urn:ogc:def:crs:EPSG::4326">
                   <gml:posList>-46.32769393920898 168.9116668701172 -46.31716537475586
      168.9369659423828 -46.31291580200195 168.9402465820312 -46.30877685546875 168.9530792236328 -
      46.30380630493164 168.9554138183594 -46.30014038085938 168.9626159667969 -46.29613876342773
      168.9665222167969 -46.28755569458008 168.9909210205078</gml:posList>
                </gml:LineString>
             </geometry>
          </SegmentDesc>
       </segment>
       <roadType>
          <RoadDesc>
             <surfaceType>Asphalt</surfaceType>
             <nLanes>12</nLanes>
             <grade>15</grade>
          </RoadDesc>
       </roadType>
    </RoadL_1M>
 </wfs:Insert>
 <!-- Update the designation of a particular range of segments
      which are now being collapsed into a single segment. The
      The filter uses an XPath expression to reference the
      designation property -->
 <wfs:Update typeName="RoadL_1M">
    <wfs:Property>
       <wfs:ValueReference>RoadL_1M/segment/designation</wfs:ValueReference>
       <wfs:Value>SEG_A60</wfs:Value>
    </wfs:Property>
    <fes:Filter>
       <fes:PropertyIsBetween>
          <fes:ValueReference>RoadL_1M/segment/designation</fes:ValueReference>
          <fes:LowerBoundary>
             <fes:Literal>SEG_A60</fes:Literal>
          </fes:LowerBoundary>
          <fes:UpperBoundary>
             <fes:Literal>SEG_A69</fes:Literal>
          </fes:UpperBoundary>
       </fes:PropertyIsBetween>
    </fes:Filter>
 </wfs:Update>
 <!-- Create 2 feature instances of feature type BuiltUpA_1M -->
 <wfs:Insert handle="Statement 2">
    <BuiltUpA_1M gml:id="B1">
       <placeId>4070</placeId>
       <name>Toronto</name>
       <population>4000000</population>
       <bndry>
          <gml:Polygon gml:id="g3"
             srsName="urn:ogc:def:crs:EPSG::4326">
             <gml:exterior>
                <gml:LinearRing>
                <gml:posList>-32.49074935913086 137.7553100585938 -32.49797058105469
      137.7500762939453 -32.50219345092773 137.7501068115234 -32.49538803100586 137.7489929199219 -
      32.49352645874023 137.748779296875 -32.49258422851562 137.7469940185547 -32.4914436340332
      137.7448272705078 -32.49761199951172 137.7403869628906 -32.49774932861328 137.7377014160156 -
      32.48825073242188 137.7359161376953 -32.47955703735352 137.7342987060547 -32.4787483215332
      137.7353668212891 -32.47758483886719 137.7368927001953 -32.47419357299805 137.7501678466797 -
      32.47983169555664 137.7571411132812 -32.48297119140625 137.759033203125 -32.48714065551758
      137.7578887939453 -32.49074935913086 137.7553100585938</gml:posList>
                </gml:LinearRing>
             </gml:exterior>
          </gml:Polygon>
       </bndry>
    </BuiltUpA_1M>
    <BuiltUpA_1M gml:id="B2">
       <placeId>1725</placeId>
       <name>Montreal</name>
       <population>2000000</population>
       <bndry>
          <gml:Polygon gml:id="e4"
             srsName="urn:ogc:def:crs:EPSG::4326">
             <gml:exterior>
                <gml:LinearRing>
                <gml:posList>-33.20774841308594 138.0020904541016 -33.20772171020508
      137.997802734375 -33.20266723632812 137.9964141845703 -33.20236206054688 137.9871368408203 -
      33.19541549682617 137.9859924316406 -33.19369506835938 137.9819183349609 -33.17902755737305
```

```
      137.9871978759766 -33.18127822875977 137.9911346435547 -33.17658233642578 137.9950256347656 -
      33.18086242675781 138.0019378662109 -33.18441772460938 138.0037536621094 -33.18713760375977
      138.0029144287109 -33.19083404541016 137.9978637695312 -33.19633483886719 137.9975280761719 -
      33.19985961914062 137.9996185302734 -33.19947052001953 138.0031127929688 -33.18966674804688
      138.01025390625 -33.19049835205078 138.0133361816406 -33.19108200073242 138.0155792236328 -
      33.19660949707031 138.0187530517578 -33.20325088500977 138.0167236328125 -33.20430374145508
      138.0137481689453 -33.21133422851562 138.009033203125 -33.20774841308594 138.0041961669922 -
      33.20774841308594 138.0020904541016</gml:posList>
                </gml:LinearRing>
             </gml:exterior>
          </gml:Polygon>
       </bndry>
    </BuiltUpA_1M>
 </wfs:Insert>
 <!-- Update the name property of the feature instance BuiltUpA_1M.1210 -->
 <wfs:Update typeName="BuiltUpA_1M">
    <wfs:Property>
       <wfs:ValueReference>BuiltUpA_1M/name</wfs:ValueReference>
       <wfs:Value>SMALLVILLE</wfs:Value>
    </wfs:Property>
    <fes:Filter>
       <fes:ResourceId rid="BuiltUpA_1M.1210"/>
    </fes:Filter>
 </wfs:Update>
 <!-- Update the geometry of the feature BuiltUpA_1M.1725. -->
 <wfs:Update typeName="BuiltUpA_1M">
    <wfs:Property>
       <wfs:ValueReference>BuiltUpA_1M/bndry</wfs:ValueReference>
       <wfs:Value>
          <gml:Polygon gml:id="g5"
             srsName="urn:ogc:def:crs:EPSG::4326">
             <gml:exterior>
                <gml:LinearRing>
                <gml:posList>-34.08438873291016 151.1380767822266 -34.08183288574219
      151.1435852050781 -34.08427810668945 151.1475219726562 -34.08483505249023 151.1522216796875 -
      34.08349990844727 151.1563110351562 -34.09330368041992 151.1546325683594 -34.08549880981445
      151.1357269287109 -34.08438873291016 151.1380767822266</gml:posList>
                </gml:LinearRing>
             </gml:exterior>
          </gml:Polygon>
       </wfs:Value>
    </wfs:Property>
    <fes:Filter>
       <fes:ResourceId rid="BuiltUpA_1M.1725"/>
    </fes:Filter>
 </wfs:Update>
 <!-- Delete the feature instance BuiltUpA_1M.1013. -->
 <wfs:Delete typeName="BuiltUpA_1M">
    <fes:Filter>
       <fes:ResourceId rid="BuiltUpA_1M.1013"/>
    </fes:Filter>
 </wfs:Delete>
 <!-- Delete all instances of the feature type
      BuiltUpA_1M where:
      1. the geometry is INSIDE a polygonal window
      2. where the POPULATION is between 100 and 2000 -->
 <wfs:Delete typeName="BuiltUpA_1M">
    <fes:Filter>
       <fes:And>
          <fes:Within>
             <fes:ValueReference>BuiltUpA_1M/bndry</fes:ValueReference>
             <gml:Polygon gml:id="WINDOW" srsName="urn:ogc:def:crs:EPSG::4326">
                <gml:exterior>
                   <gml:LinearRing>
                      <gml:posList>-33.9866943359375 150.8492279052734 -33.98172378540039
      150.8462829589844 -33.97666549682618 150.8461151123047 -33.97611236572266 150.8504791259766 -
      33.97283172607422 150.8525543212891 -33.97174835205078 150.8591156005859 -33.97491836547852
      150.8614501953125 -33.98274993896484 150.8600769042969 -33.98555374145508 150.857421875 -
      33.9866943359375 150.8492279052734</gml:posList>
                   </gml:LinearRing>
                </gml:exterior>
             </gml:Polygon>
          </fes:Within>
          <fes:And>
             <fes:PropertyIsGreaterThanOrEqualTo>
                <fes:ValueReference>BuiltUpA_1M/population</fes:ValueReference>
```

```
                  <fes:Literal>100</fes:Literal>
               </fes:PropertyIsGreaterThanOrEqualTo>
               <fes:PropertyIsLessThanOrEqualTo>
                  <fes:ValueReference>BuiltUpA_1M/population</fes:ValueReference>
                  <fes:Literal>2000</fes:Literal>
               </fes:PropertyIsLessThanOrEqualTo>
            </fes:And>
         </fes:And>
      </fes:Filter>
   </wfs:Delete>
   <wfs:Replace>
      <BuiltUpA_1M gml:id="B01a">
         <placeId>34159</placeId>
         <name>MyTown</name>
         <population>14357</population>
         <bndry>
         <gml:Polygon srsName="urn:ogc:def:crs:EPSG::4326" gml:id="P01a">
            <gml:exterior>
               <gml:LinearRing>
                  <gml:posList>-31.121000289917 120.4429473876953 -31.12280464172363 120.4445580078125
       -31.12605476379395 120.442253112793 -31.12794494628907 120.4380264282227 -31.12230491638183
       120.4315032958984 -31.11816596984863 120.432502746582 -31.11672210693359 120.4343619537353 52 -
       31.11802864074707 120.4410247802734 -31.121000289917 120.4429473876953</gml:posList>
               </gml:LinearRing>
            </gml:exterior>
         </gml:Polygon>
         </bndry>
      </BuiltUpA_1M>
      <fes:Filter>
         <fes:ResourceId rid="BuiltUpA_1M.45783"/>
      </fes:Filter>
   </wfs:Replace>
</wfs:Transaction>
```

Response document:

```
<?xml version="1.0" ?>
<wfs:TransactionResponse
   version="2.0.0"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   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">
   <wfs:TransactionSummary>
      <wfs:totalInserted>3</wfs:totalInserted>
      <wfs:totalUpdated>3</wfs:totalUpdated>
      <wfs:totalReplaced>1</wfs:totalReplaced>
      <wfs:totalDeleted>2</wfs:totalDeleted>
   </wfs:TransactionSummary>
   <wfs:InsertResults>
      <wfs:Feature handle="Statement 1">
         <fes:ResourceId rid="ElevP_1M.1001"/>
      </wfs:Feature>
      <wfs:Feature handle="ComplexInsert">
         <fes:ResourceId rid="RoadL_1M.1553"/>
      </wfs:Feature>
      <wfs:Feature handle="Statement 2">
         <fes:ResourceId rid="BuiltUpA_1M.509876"/>
         <fes:ResourceId rid="BuiltUpA_1M.509877"/>
      </wfs:Feature>
   </wfs:InsertResults>
</wfs:TransactionResponse>
```

B.6.5 Transaction response example

Consider a transaction request that creates a number of new feature instances. The feature instances are created using three <wfs:Insert> elements labelled "STMT1", "STMT2" and "STMT3". A typical response to such a request might be:

```
<?xml version="1.0" ?>
<wfs:TransactionResponse
   version="2.0.0"
```

```
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   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">
   <wfs:TransactionSummary>
      <wfs:totalInserted>5</wfs:totalInserted>
   </wfs:TransactionSummary>
   <wfs:InsertResults>
      <wfs:Feature handle="STMT1">
         <fes:ResourceId rid="SomeFeature.4567"/>
      </wfs:Feature>
      <wfs:Feature handle="STMT1">
         <fes:ResourceId rid="SomeFeature.4568"/>
      </wfs:Feature>
      <wfs:Feature handle="STMT1">
         <fes:ResourceId rid="SomeFeature.4569"/>
      </wfs:Feature>
      <wfs:Feature handle="STMT2">
         <fes:ResourceId rid="Feature1.4569"/>
      </wfs:Feature>
      <wfs:Feature handle="STMT3">
         <fes:ResourceId rid="Feature2.389345"/>
      </wfs:Feature>
   </wfs:InsertResults>
</wfs:TransactionResponse>
```

B.7 GetCapabilities example

This example shows what a capabilities document might look like for a basic web feature service. To request a capabilities document, a client would issue the following request:

```
<?xml version="1.0" ?>
<GetCapabilities
   service="WFS"
   version="2.0.0"
   xmlns="http://www.opengis.net/wfs/2.0"
   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"/>
```

A server that implements the Simple WFS conformance class (see Table 1) might generate the following capabilities document:

```
<?xml version="1.0"?>
<WFS_Capabilities
   version="2.0.0"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:ows="http://www.opengis.net/ows/1.1"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   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.opengis.net/ows/1.1
                       http://schemas.opengis.net/ows/1.1.0/owsAll.xsd">
   <ows:ServiceIdentification>
      <ows:Title>OGC Member WFS</ows:Title>
      <ows:Abstract>Web Feature Service maintained by NSDI data provider, serving FGDC framework
        layer XXX; contact Paul.Bunyon@BlueOx.org</ows:Abstract>
      <ows:Keywords>
         <ows:Keyword>FGDC</ows:Keyword>
         <ows:Keyword>NSDI</ows:Keyword>
         <ows:Keyword>Framework Data Layer</ows:Keyword>
         <ows:Keyword>BlueOx</ows:Keyword>
         <ows:Type>String</ows:Type>
      </ows:Keywords>
      <ows:ServiceType>WFS</ows:ServiceType>
      <ows:ServiceTypeVersion>2.0.0</ows:ServiceTypeVersion>
      <ows:ServiceTypeVersion>1.1.0</ows:ServiceTypeVersion>
```

```
      <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
      <ows:Fees>NONE</ows:Fees>
      <ows:AccessConstraints>NONE</ows:AccessConstraints>
   </ows:ServiceIdentification>
   <ows:ServiceProvider>
      <ows:ProviderName>BlueOx Inc.</ows:ProviderName>
      <ows:ProviderSite xlink:href="http://www.cubewerx.com"/>
      <ows:ServiceContact>
         <ows:IndividualName>Paul Bunyon</ows:IndividualName>
         <ows:PositionName>Mythology Manager</ows:PositionName>
         <ows:ContactInfo>
            <ows:Phone>
               <ows:Voice>1.800.BIG.WOOD</ows:Voice>
               <ows:Facsimile>1.800.FAX.WOOD</ows:Facsimile>
            </ows:Phone>
            <ows:Address>
               <ows:DeliveryPoint>North Country</ows:DeliveryPoint>
               <ows:City>Small Town</ows:City>
               <ows:AdministrativeArea>Rural County</ows:AdministrativeArea>
               <ows:PostalCode>12345</ows:PostalCode>
               <ows:Country>USA</ows:Country>
               <ows:ElectronicMailAddress>Paul.Bunyon@BlueOx.org</ows:ElectronicMailAddress>
            </ows:Address>
            <ows:OnlineResource xlink:href="http://www.BlueOx.org/contactUs"/>
            <ows:HoursOfService>24x7</ows:HoursOfService>
            <ows:ContactInstructions>
               eMail Paul with normal requsts; Phone Paul for emergency
               requests; if you get voice mail and your request can't wait,
               contact another mythological figure listed on the contactUs
               page of our web site.
            </ows:ContactInstructions>
         </ows:ContactInfo>
         <ows:Role>PointOfContact</ows:Role>
      </ows:ServiceContact>
   </ows:ServiceProvider>
   <ows:OperationsMetadata>
      <ows:Operation name="GetCapabilities">
         <ows:DCP>
            <ows:HTTP>
               <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
               <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
            </ows:HTTP>
         </ows:DCP>
         <ows:Parameter name="AcceptVersions">
            <ows:AllowedValues>
            <ows:Value>2.0.0</ows:Value>
            </ows:AllowedValues>
         </ows:Parameter>
      </ows:Operation>
      <ows:Operation name="DescribeFeatureType">
         <ows:DCP>
            <ows:HTTP>
               <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
               <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
            </ows:HTTP>
         </ows:DCP>
      </ows:Operation>
      <ows:Operation name="ListStoredQueries">
         <ows:DCP>
            <ows:HTTP>
               <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
               <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
            </ows:HTTP>
         </ows:DCP>
      </ows:Operation>
      <ows:Operation name="DescribeStoredQueries">
         <ows:DCP>
            <ows:HTTP>
               <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
               <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
            </ows:HTTP>
         </ows:DCP>
      </ows:Operation>
      <ows:Operation name="GetFeature">
         <ows:DCP>
            <ows:HTTP>
```

```
            <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
            <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
         </ows:HTTP>
      </ows:DCP>
   </ows:Operation>
   <ows:Parameter name="version">
      <ows:AllowedValues>
      <ows:Value>2.0.0</ows:Value>
      </ows:AllowedValues>
   </ows:Parameter>
   <!-- ************************************************** -->
   <!-- *   CONFORMANCE DECLARATION                      * -->
   <!-- ************************************************** -->
   <ows:Constraint name="ImplementsBasicWFS">
      <ows:NoValues/>
      <ows:DefaultValue>FALSE</ows:DefaultValue>
   </ows:Constraint>
   <ows:Constraint name="ImplementsTransactionalWFS">
      <ows:NoValues/>
      <ows:DefaultValue>FALSE</ows:DefaultValue>
   </ows:Constraint>
   <ows:Constraint name="ImplementsLockingWFS">
      <ows:NoValues/>
      <ows:DefaultValue>FALSE</ows:DefaultValue>
   </ows:Constraint>
   <ows:Constraint name="KVPEncoding">
      <ows:NoValues/>
      <ows:DefaultValue>TRUE</ows:DefaultValue>
   </ows:Constraint>
   <ows:Constraint name="XMLEncoding">
      <ows:NoValues/>
      <ows:DefaultValue>FALSE</ows:DefaultValue>
   </ows:Constraint>
   <ows:Constraint name="SOAPEncoding">
      <ows:NoValues/>
      <ows:DefaultValue>FALSE</ows:DefaultValue>
   </ows:Constraint>
   <ows:Constraint name="ImplementsInheritance">
      <ows:NoValues/>
      <ows:DefaultValue>FALSE</ows:DefaultValue>
   </ows:Constraint>
   <ows:Constraint name="ImplementsRemoteResolve">
      <ows:NoValues/>
      <ows:DefaultValue>FALSE</ows:DefaultValue>
   </ows:Constraint>
   <ows:Constraint name="ImplementsResultPaging">
      <ows:NoValues/>
      <ows:DefaultValue>FALSE</ows:DefaultValue>
   </ows:Constraint>
   <ows:Constraint name="ImplementsStandardJoins">
      <ows:NoValues/>
      <ows:DefaultValue>FALSE</ows:DefaultValue>
   </ows:Constraint>
   <ows:Constraint name="ImplementsSpatialJoins">
      <ows:NoValues/>
      <ows:DefaultValue>FALSE</ows:DefaultValue>
   </ows:Constraint>
   <ows:Constraint name="ImplementsTemporalJoins">
      <ows:NoValues/>
      <ows:DefaultValue>FALSE</ows:DefaultValue>
   </ows:Constraint>
   <ows:Constraint name="ImplementsFeatureVersioning">
      <ows:NoValues/>
      <ows:DefaultValue>FALSE</ows:DefaultValue>
   </ows:Constraint>
   <ows:Constraint name="ManageStoredQueries">
      <ows:NoValues/>
      <ows:DefaultValue>FALSE</ows:DefaultValue>
   </ows:Constraint>
   <!-- ************************************************** -->
   <!-- *   CAPACITY CONSTRAINTS                         * -->
   <!-- ************************************************** -->
   <ows:Constraint name="CountDefault">
      <ows:NoValues/>
      <ows:DefaultValue>1000</ows:DefaultValue>
   </ows:Constraint>
```

```
      <ows:Constraint name="QueryExpressions">
         <ows:AllowedValues>
            <ows:Value>wfs:StoredQuery</ows:Value>
         </ows:AllowedValues>
      </ows:Constraint>
      <!-- ================================================== -->
   </ows:OperationsMetadata>
   <FeatureTypeList>
      <FeatureType xmlns:bo="http://www.BlueOx.org/BlueOx">
         <Name>bo:Woods</Name>
         <Title>The Great Northern Forest</Title>
         <Abstract>
            Describes the arborial diversity of the Great
            Northern Forest.
         </Abstract>
         <ows:Keywords>
            <ows:Keyword>forest</ows:Keyword>
            <ows:Keyword>north</ows:Keyword>
            <ows:Keyword>woods</ows:Keyword>
            <ows:Keyword>arborial</ows:Keyword>
            <ows:Keyword>diversity</ows:Keyword>
         </ows:Keywords>
         <DefaultCRS>urn:ogc:def:crs:EPSG::6269</DefaultCRS>
         <ows:WGS84BoundingBox>
            <ows:LowerCorner>-180 -90</ows:LowerCorner>
            <ows:UpperCorner>180 90</ows:UpperCorner>
         </ows:WGS84BoundingBox>
         <MetadataURL
        xlink:href="http://www.ogccatservice.com/csw.cgi?service=CSW&amp;version=2.0.0&amp;request=Get
        Records&amp;constraintlanguage=CQL&amp;recordid=urn:uuid:4ee8d2d3-9409-4a1d-b26b-
        6782e4fa3d59"/>
         <ExtendedDescription>
            <Element name="TemporalExtent" type="xsd:date">
               <ows:Metadata xlink:href="http://www.someserver.com/AboutTemporalExtent.html"/>
               <ValueList>
                  <Value>2000-01-01</Value>
                  <Value>2006-01-31</Value>
               </ValueList>
            </Element>
            <Element name="Classifications" type="xsd:anyURI">
               <ows:Metadata xlink:href="http://www.someserver.com/AboutClassifications.html"/>
               <ValueList>
                  <Value>urn:x-ogc:specification:csw-ebrim:ClassificationScheme:ISO-
        19115:biota</Value>
               </ValueList>
            </Element>
            <Element name="ArborialDiversityIndex" type="xsd:integer">
               <ows:Metadata xlink:href="http://www.someserver.com/ArborialDiversity.html"/>
               <ValueList>
                  <Value>14</Value>
               </ValueList>
            </Element>
         </ExtendedDescription>
      </FeatureType>
   </FeatureTypeList>
   <fes:Filter_Capabilities>
      <fes:Conformance>
         <fes:Constraint name="ImplementsQuery">
            <ows:NoValues/>
            <ows:DefaultValue>TRUE</ows:DefaultValue>
         </fes:Constraint>
         <fes:Constraint name="ImplementsAdHocQuery">
            <ows:NoValues/>
            <ows:DefaultValue>FALSE</ows:DefaultValue>
         </fes:Constraint>
         <fes:Constraint name="ImplementsFunctions">
            <ows:NoValues/>
            <ows:DefaultValue>FALSE</ows:DefaultValue>
         </fes:Constraint>
         <fes:Constraint name="ImplementsMinStandardFilter">
            <ows:NoValues/>
            <ows:DefaultValue>FALSE</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>FALSE</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="ImplementsTemporalFilter">
            <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:Constraint name="ImplementsExtendedOperators">
            <ows:NoValues/>
            <ows:DefaultValue>FALSE</ows:DefaultValue>
         </fes:Constraint>
      </fes:Conformance>
   </fes:Filter_Capabilities>
</WFS_Capabilities>
```

A server that implements the Basic WFS conformance class (see Table 1) might generate the following capabilities document:

```
< <?xml version="1.0"?>
<WFS_Capabilities
   version="2.0.0"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:ows="http://www.opengis.net/ows/1.1"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   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.opengis.net/ows/1.1
                       http://schemas.opengis.net/ows/1.1.0/owsAll.xsd">
   <ows:ServiceIdentification>
      <ows:Title>OGC Member WFS</ows:Title>
      <ows:Abstract>Web Feature Service maintained by NSDI data provider, serving FGDC framework
        layer XXX; contact Paul.Bunyon@BlueOx.org</ows:Abstract>
      <ows:Keywords>
         <ows:Keyword>FGDC</ows:Keyword>
         <ows:Keyword>NSDI</ows:Keyword>
         <ows:Keyword>Framework Data Layer</ows:Keyword>
         <ows:Keyword>BlueOx</ows:Keyword>
         <ows:Type>String</ows:Type>
      </ows:Keywords>
      <ows:ServiceType>WFS</ows:ServiceType>
      <ows:ServiceTypeVersion>2.0.0</ows:ServiceTypeVersion>
      <ows:ServiceTypeVersion>1.1.0</ows:ServiceTypeVersion>
      <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
      <ows:Fees>NONE</ows:Fees>
      <ows:AccessConstraints>NONE</ows:AccessConstraints>
   </ows:ServiceIdentification>
   <ows:ServiceProvider>
      <ows:ProviderName>BlueOx Inc.</ows:ProviderName>
      <ows:ProviderSite xlink:href="http://www.cubewerx.com"/>
      <ows:ServiceContact>
         <ows:IndividualName>Paul Bunyon</ows:IndividualName>
         <ows:PositionName>Mythology Manager</ows:PositionName>
         <ows:ContactInfo>
```

```
            <ows:Phone>
               <ows:Voice>1.800.BIG.WOOD</ows:Voice>
               <ows:Facsimile>1.800.FAX.WOOD</ows:Facsimile>
            </ows:Phone>
            <ows:Address>
               <ows:DeliveryPoint>North Country</ows:DeliveryPoint>
               <ows:City>Small Town</ows:City>
               <ows:AdministrativeArea>Rural County</ows:AdministrativeArea>
               <ows:PostalCode>12345</ows:PostalCode>
               <ows:Country>USA</ows:Country>
               <ows:ElectronicMailAddress>Paul.Bunyon@BlueOx.org</ows:ElectronicMailAddress>
            </ows:Address>
            <ows:OnlineResource xlink:href="http://www.BlueOx.org/contactUs"/>
            <ows:HoursOfService>24x7</ows:HoursOfService>
            <ows:ContactInstructions>
               eMail Paul with normal requsts; Phone Paul for emergency
               requests; if you get voice mail and your request can't wait,
               contact another mythological figure listed on the contactUs
               page of our web site.
            </ows:ContactInstructions>
         </ows:ContactInfo>
         <ows:Role>PointOfContact</ows:Role>
      </ows:ServiceContact>
   </ows:ServiceProvider>
   <ows:OperationsMetadata>
      <ows:Operation name="GetCapabilities">
         <ows:DCP>
            <ows:HTTP>
               <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
            </ows:HTTP>
         </ows:DCP>
         <ows:Parameter name="AcceptVersions">
            <ows:AllowedValues>
            <ows:Value>2.0.0</ows:Value>
            </ows:AllowedValues>
         </ows:Parameter>
      </ows:Operation>
      <ows:Operation name="DescribeFeatureType">
         <ows:DCP>
            <ows:HTTP>
               <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
            </ows:HTTP>
         </ows:DCP>
      </ows:Operation>
      <ows:Operation name="ListStoredQueries">
         <ows:DCP>
            <ows:HTTP>
               <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
            </ows:HTTP>
         </ows:DCP>
      </ows:Operation>
      <ows:Operation name="DescribeStoredQueries">
         <ows:DCP>
            <ows:HTTP>
               <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
            </ows:HTTP>
         </ows:DCP>
      </ows:Operation>
      <ows:Operation name="GetFeature">
         <ows:DCP>
            <ows:HTTP>
               <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
            </ows:HTTP>
         </ows:DCP>
      </ows:Operation>
      <ows:Parameter name="version">
         <ows:AllowedValues>
         <ows:Value>2.0.0</ows:Value>
         </ows:AllowedValues>
      </ows:Parameter>
      <!-- ************************************************** -->
      <!-- *   CONFORMANCE DECLARATION                      * -->
      <!-- ************************************************** -->
      <ows:Constraint name="ImplementsBasicWFS">
         <ows:NoValues/>
         <ows:DefaultValue>TRUE</ows:DefaultValue>
```

```
      </ows:Constraint>
      <ows:Constraint name="ImplementsTransactionalWFS">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </ows:Constraint>
      <ows:Constraint name="ImplementsLockingWFS">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </ows:Constraint>
      <ows:Constraint name="KVPEncoding">
         <ows:NoValues/>
         <ows:DefaultValue>TRUE</ows:DefaultValue>
      </ows:Constraint>
      <ows:Constraint name="XMLEncoding">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </ows:Constraint>
      <ows:Constraint name="SOAPEncoding">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </ows:Constraint>
      <ows:Constraint name="ImplementsInheritance">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </ows:Constraint>
      <ows:Constraint name="ImplementsRemoteResolve">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </ows:Constraint>
      <ows:Constraint name="ImplementsResultPaging">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </ows:Constraint>
      <ows:Constraint name="ImplementsStandardJoins">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </ows:Constraint>
      <ows:Constraint name="ImplementsSpatialJoins">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </ows:Constraint>
      <ows:Constraint name="ImplementsTemporalJoins">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </ows:Constraint>
      <ows:Constraint name="ImplementsFeatureVersioning">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </ows:Constraint>
      <ows:Constraint name="ManageStoredQueries">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </ows:Constraint>
      <!-- ************************************************** -->
      <!-- *          CAPACITY CONSTRAINTS                  * -->
      <!-- ************************************************** -->
      <ows:Constraint name="CountDefault">
         <ows:NoValues/>
         <ows:DefaultValue>1000</ows:DefaultValue>
      </ows:Constraint>
      <ows:Constraint name="QueryExpressions">
         <ows:AllowedValues>
            <ows:Value>wfs:StoredQuery</ows:Value>
         </ows:AllowedValues>
      </ows:Constraint>
      <!-- ************************************************** -->
   </ows:OperationsMetadata>
   <FeatureTypeList>
      <FeatureType xmlns:bo="http://www.BlueOx.org/BlueOx">
         <Name>bo:Woods</Name>
         <Title>The Great Northern Forest</Title>
         <Abstract>
            Describes the arborial diversity of the Great
            Northern Forest.
         </Abstract>
         <ows:Keywords>
```

```
            <ows:Keyword>forest</ows:Keyword>
            <ows:Keyword>north</ows:Keyword>
            <ows:Keyword>woods</ows:Keyword>
            <ows:Keyword>arborial</ows:Keyword>
            <ows:Keyword>diversity</ows:Keyword>
         </ows:Keywords>
         <DefaultCRS>urn:ogc:def:crs:EPSG::6269</DefaultCRS>
         <OtherCRS>urn:ogc:def:crs:EPSG::32615</OtherCRS>
         <OtherCRS>urn:ogc:def:crs:EPSG::32616</OtherCRS>
         <OtherCRS>urn:ogc:def:crs:EPSG::32617</OtherCRS>
         <OtherCRS>urn:ogc:def:crs:EPSG::32618</OtherCRS>
         <ows:WGS84BoundingBox>
            <ows:LowerCorner>-180 -90</ows:LowerCorner>
            <ows:UpperCorner>180 90</ows:UpperCorner>
         </ows:WGS84BoundingBox>
         <MetadataURL
      xlink:href="http://www.ogccatservice.com/csw.cgi?service=CSW&amp;version=2.0.0&amp;request=Get
      Records&amp;constraintlanguage=CQL&amp;recordid=urn:uuid:4ee8b2d3-9409-4a1d-b26b-
      6782e4fa3d59"/>
         <ExtendedDescription>
            <Element name="TemporalExtent" type="xsd:date">
               <ows:Metadata xlink:href="http://www.someserver.com/AboutTemporalExtent.html"/>
               <ValueList>
                  <Value>2000-01-01</Value>
                  <Value>2006-01-31</Value>
               </ValueList>
            </Element>
            <Element name="Classifications" type="xsd:anyURI">
               <ows:Metadata xlink:href="http://www.someserver.com/AboutClassifications.html"/>
               <ValueList>
                  <Value>urn:x-ogc:specification:csw-ebrim:ClassificationScheme:ISO-
      19115:biota</Value>
               </ValueList>
            </Element>
            <Element name="ArborialDiversityIndex" type="xsd:integer">
               <ows:Metadata xlink:href="http://www.someserver.com/ArborialDiversity.html"/>
               <ValueList>
                  <Value>14</Value>
               </ValueList>
            </Element>
         </ExtendedDescription>
      </FeatureType>
   </FeatureTypeList>
   <fes:Filter_Capabilities>
      <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="ImplementsFunctions">
            <ows:NoValues/>
            <ows:DefaultValue>FALSE</ows:DefaultValue>
         </fes:Constraint>
         <fes:Constraint name="ImplementsMinStandardFilter">
            <ows:NoValues/>
            <ows:DefaultValue>FALSE</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="ImplementsTemporalFilter">
               <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:Constraint name="ImplementsExtendedOperators">
               <ows:NoValues/>
               <ows:DefaultValue>FALSE</ows:DefaultValue>
            </fes:Constraint>
         </fes:Conformance>
         <fes:Id_Capabilities>
            <fes:ResourceIdentifier name="fes:ResourceId"/>
         </fes:Id_Capabilities>
         <fes:Spatial_Capabilities>
           <fes:GeometryOperands>
             <fes:GeometryOperand name="gml:Envelope"/>
           </fes:GeometryOperands>
           <fes:SpatialOperators>
             <fes:SpatialOperator name="BBOX"/>
           </fes:SpatialOperators>
         </fes:Spatial_Capabilities>
      </fes:Filter_Capabilities>
</WFS_Capabilities>
```

A complex server that implements all conformance classes (see Table 1) might generate the following capabilities document:

```
<WFS_Capabilities
   version="2.0.0"
   xmlns="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:ows="http://www.opengis.net/ows/1.1"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   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.opengis.net/ows/1.1
                       http://schemas.opengis/net/ows/1.1.0/owsAll.xsd">
   <ows:ServiceIdentification>
      <ows:Title>OGC Member WFS</ows:Title>
      <ows:Abstract>Web Feature Service maintained by NSDI data provider, serving FGDC framework
        layer XXX; contact Paul.Bunyon@BlueOx.org</ows:Abstract>
      <ows:Keywords>
         <ows:Keyword>FGDC</ows:Keyword>
         <ows:Keyword>NSDI</ows:Keyword>
         <ows:Keyword>Framework Data Layer</ows:Keyword>
         <ows:Keyword>BlueOx</ows:Keyword>
         <ows:Type>String</ows:Type>
      </ows:Keywords>
      <ows:ServiceType>WFS</ows:ServiceType>
      <ows:ServiceTypeVersion>2.0.0</ows:ServiceTypeVersion>
      <ows:ServiceTypeVersion>1.1.0</ows:ServiceTypeVersion>
      <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
      <ows:Fees>NONE</ows:Fees>
      <ows:AccessConstraints>NONE</ows:AccessConstraints>
   </ows:ServiceIdentification>
   <ows:ServiceProvider>
      <ows:ProviderName>BlueOx Inc.</ows:ProviderName>
      <ows:ProviderSite xlink:href="http://www.cubewerx.com"/>
      <ows:ServiceContact>
         <ows:IndividualName>Paul Bunyon</ows:IndividualName>
         <ows:PositionName>Mythology Manager</ows:PositionName>
         <ows:ContactInfo>
            <ows:Phone>
               <ows:Voice>1.800.BIG.WOOD</ows:Voice>
```

```
            <ows:Facsimile>1.800.FAX.WOOD</ows:Facsimile>
         </ows:Phone>
         <ows:Address>
            <ows:DeliveryPoint>North Country</ows:DeliveryPoint>
            <ows:City>Small Town</ows:City>
            <ows:AdministrativeArea>Rural County</ows:AdministrativeArea>
            <ows:PostalCode>12345</ows:PostalCode>
            <ows:Country>USA</ows:Country>
            <ows:ElectronicMailAddress>Paul.Bunyon@BlueOx.org</ows:ElectronicMailAddress>
         </ows:Address>
         <ows:OnlineResource xlink:href="http://www.BlueOx.org/contactUs"/>
         <ows:HoursOfService>24x7</ows:HoursOfService>
         <ows:ContactInstructions>eMail Paul with normal requsts; Phone Paul for emergency
            requests; if you get voice mail and your request can't wait,
            contact another mythological figure listed on the contactUs
            page of our web site.</ows:ContactInstructions>
      </ows:ContactInfo>
      <ows:Role>PointOfContact</ows:Role>
   </ows:ServiceContact>
</ows:ServiceProvider>
<ows:OperationsMetadata>
   <ows:Operation name="GetCapabilities">
      <ows:DCP>
         <ows:HTTP>
            <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
            <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
         </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="AcceptVersions">
         <ows:AllowedValues>
            <ows:Value>2.0.0</ows:Value>
            <ows:Value>1.1.0</ows:Value>
            <ows:Value>1.0.0</ows:Value>
         </ows:AllowedValues>
      </ows:Parameter>
   </ows:Operation>
   <ows:Operation name="DescribeFeatureType">
      <ows:DCP>
         <ows:HTTP>
            <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
            <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
         </ows:HTTP>
      </ows:DCP>
   </ows:Operation>
   <ows:Operation name="ListStoredQueries">
      <ows:DCP>
         <ows:HTTP>
            <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
            <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
         </ows:HTTP>
      </ows:DCP>
   </ows:Operation>
   <ows:Operation name="DescribeStoredQueries">
      <ows:DCP>
         <ows:HTTP>
            <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
            <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
         </ows:HTTP>
      </ows:DCP>
   </ows:Operation>
   <ows:Operation name="GetPropertyValue">
      <ows:DCP>
         <ows:HTTP>
            <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
            <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
         </ows:HTTP>
      </ows:DCP>
   </ows:Operation>
   <ows:Operation name="GetFeature">
      <ows:DCP>
         <ows:HTTP>
            <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
            <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
         </ows:HTTP>
      </ows:DCP>
   </ows:Operation>
```

```
<ows:Operation name="GetFeatureWithLock">
   <ows:DCP>
      <ows:HTTP>
         <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
         <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
      </ows:HTTP>
   </ows:DCP>
</ows:Operation>
<ows:Operation name="Transaction">
   <ows:DCP>
      <ows:HTTP>
         <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
      </ows:HTTP>
   </ows:DCP>
   <ows:Constraint name="AutomaticDataLocking">
      <ows:NoValues/>
      <ows:DefaultValue>TRUE</ows:DefaultValue>
   </ows:Constraint>
   <ows:Constraint name="PreservesSiblingOrder">
      <ows:NoValues/>
      <ows:DefaultValue>TRUE</ows:DefaultValue>
   </ows:Constraint>
</ows:Operation>
<ows:Operation name="LockFeature">
   <ows:DCP>
      <ows:HTTP>
         <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
         <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
      </ows:HTTP>
   </ows:DCP>
</ows:Operation>
<ows:Operation name="CreateStoredQuery">
   <ows:DCP>
      <ows:HTTP>
         <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
         <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
      </ows:HTTP>
   </ows:DCP>
</ows:Operation>
<ows:Operation name="DropStoredQuery">
   <ows:DCP>
      <ows:HTTP>
         <ows:Get xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
         <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
      </ows:HTTP>
   </ows:DCP>
</ows:Operation>
<ows:Parameter name="version">
   <ows:AllowedValues>
      <ows:Value>2.0.0</ows:Value>
      <ows:Value>1.1.0</ows:Value>
      <ows:Value>1.0.0</ows:Value>
   </ows:AllowedValues>
</ows:Parameter>
<!-- ************************************************** -->
<!-- *     CONFORMANCE DECLARATION                      * -->
<!-- ************************************************** -->
<ows:Constraint name="ImplementsBasicWFS">
   <ows:NoValues/>
   <ows:DefaultValue>TRUE</ows:DefaultValue>
</ows:Constraint>
<ows:Constraint name="ImplementsTransactionalWFS">
   <ows:NoValues/>
   <ows:DefaultValue>TRUE</ows:DefaultValue>
</ows:Constraint>
<ows:Constraint name="ImplementsLockingWFS">
   <ows:NoValues/>
   <ows:DefaultValue>TRUE</ows:DefaultValue>
</ows:Constraint>
<ows:Constraint name="KVPEncoding">
   <ows:NoValues/>
   <ows:DefaultValue>TRUE</ows:DefaultValue>
</ows:Constraint>
<ows:Constraint name="XMLEncoding">
   <ows:NoValues/>
   <ows:DefaultValue>TRUE</ows:DefaultValue>
```

```
    </ows:Constraint>
    <ows:Constraint name="SOAPEncoding">
       <ows:NoValues/>
       <ows:DefaultValue>TRUE</ows:DefaultValue>
    </ows:Constraint>
    <ows:Constraint name="ImplementsInheritance">
       <ows:NoValues/>
       <ows:DefaultValue>TRUE</ows:DefaultValue>
    </ows:Constraint>
    <ows:Constraint name="ImplementsRemoteResolve">
       <ows:NoValues/>
       <ows:DefaultValue>TRUE</ows:DefaultValue>
    </ows:Constraint>
    <ows:Constraint name="ImplementsResultPaging">
       <ows:NoValues/>
       <ows:DefaultValue>TRUE</ows:DefaultValue>
    </ows:Constraint>
    <ows:Constraint name="ImplementsStandardJoins">
       <ows:NoValues/>
       <ows:DefaultValue>TRUE</ows:DefaultValue>
    </ows:Constraint>
    <ows:Constraint name="ImplementsSpatialJoins">
       <ows:NoValues/>
       <ows:DefaultValue>TRUE</ows:DefaultValue>
    </ows:Constraint>
    <ows:Constraint name="ImplementsTemporalJoins">
       <ows:NoValues/>
       <ows:DefaultValue>TRUE</ows:DefaultValue>
    </ows:Constraint>
    <ows:Constraint name="ImplementsFeatureVersioning">
       <ows:NoValues/>
       <ows:DefaultValue>TRUE</ows:DefaultValue>
    </ows:Constraint>
    <ows:Constraint name="ManageStoredQueries">
       <ows:NoValues/>
       <ows:DefaultValue>TRUE</ows:DefaultValue>
    </ows:Constraint>
    <!-- ************************************************** -->
    <!-- *   CAPACITY CONSTRAINTS                         * -->
    <!-- ************************************************** -->
    <ows:Constraint name="PagingIsTransactionSafe">
       <ows:NoValues/>
       <ows:DefaultValue>FALSE</ows:DefaultValue>
    </ows:Constraint>
    <ows:Constraint name="CountDefault">
       <ows:NoValues/>
       <ows:DefaultValue>1000</ows:DefaultValue>
    </ows:Constraint>
    <ows:Constraint name="ResolveTimeoutDefault">
       <ows:NoValues/>
       <ows:DefaultValue>300</ows:DefaultValue>
    </ows:Constraint>
    <ows:Constraint name="SortLevelLimit">
       <ows:NoValues/>
       <ows:DefaultValue>1</ows:DefaultValue>
    </ows:Constraint>
    <ows:Constraint name="ResolveLocalScope">
       <ows:NoValues/>
       <ows:DefaultValue>*</ows:DefaultValue>
    </ows:Constraint>
    <ows:Constraint name="ResolveRemoteScope">
       <ows:NoValues/>
       <ows:DefaultValue>5</ows:DefaultValue>
    </ows:Constraint>
    <ows:Constraint name="ResponseCacheTimeout">
       <ows:NoValues/>
       <ows:DefaultValue>300</ows:DefaultValue>
    </ows:Constraint>
    <ows:Constraint name="QueryExpressions">
       <ows:AllowedValues>
          <ows:Value>wfs:Query</ows:Value>
          <ows:Value>wfs:StoredQuery</ows:Value>
       </ows:AllowedValues>
    </ows:Constraint>
    <!-- ************************************************** -->
 </ows:OperationsMetadata>
```

```
<FeatureTypeList>
   <FeatureType xmlns:bo="http://www.BlueOx.org/BlueOx">
      <Name>bo:Woods</Name>
      <Title>The Great Northern Forest</Title>
      <Abstract>Describes the arborial diversity of the Great
            Northern Forest.</Abstract>
      <ows:Keywords>
         <ows:Keyword>forest</ows:Keyword>
         <ows:Keyword>north</ows:Keyword>
         <ows:Keyword>woods</ows:Keyword>
         <ows:Keyword>arborial</ows:Keyword>
         <ows:Keyword>diversity</ows:Keyword>
      </ows:Keywords>
      <DefaultCRS>urn:ogc:def:crs:EPSG::6269</DefaultCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::32615</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::32616</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::32617</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::32618</OtherCRS>
      <ows:WGS84BoundingBox>
         <ows:LowerCorner>-180 -90</ows:LowerCorner>
         <ows:UpperCorner>180 90</ows:UpperCorner>
      </ows:WGS84BoundingBox>
      <MetadataURL

     xlink:href="http://www.ogccatservice.com/csw.cgi?service=CSW&amp;version=2.0.0&amp;request=Get
     Records&amp;constraintlanguage=CQL&amp;recordid=urn:uuid:4ee8b2d3-9409-4a1d-b26b-
     6782e4fa3d59"/>
      <ExtendedDescription>
         <Element name="TemporalExtent" type="xsd:date">
            <ows:Metadata
               xlink:href="http://www.someserver.com/AboutTemporalExtent.html"/>
            <ValueList>
               <Value>2000-01-01</Value>
               <Value>2006-01-31</Value>
            </ValueList>
         </Element>
         <Element name="Classifications" type="xsd:anyURI">
            <ows:Metadata
               xlink:href="http://www.someserver.com/AboutClassifications.html"/>
            <ValueList>
               <Value>urn:x-ogc:specification:csw-ebrim:ClassificationScheme:ISO-
     19115:biota</Value>
            </ValueList>
         </Element>
         <Element name="ArborialDiversityIndex" type="xsd:integer">
            <ows:Metadata
               xlink:href="http://www.someserver.com/ArborialDiversity.html"/>
            <ValueList>
               <Value>14</Value>
            </ValueList>
         </Element>
      </ExtendedDescription>
   </FeatureType>
   <FeatureType xmlns:bo="http://www.BlueOx.org/BlueOx">
      <Name>bo:Lakes</Name>
      <Title>The Great Northern Lakes</Title>
      <Abstract>Lake boundaries for all lakes in the
            Great Northern Forest.</Abstract>
      <ows:Keywords>
         <ows:Keyword>lakes</ows:Keyword>
         <ows:Keyword>boundaries</ows:Keyword>
         <ows:Keyword>water</ows:Keyword>
         <ows:Keyword>hydro</ows:Keyword>
      </ows:Keywords>
      <DefaultCRS>urn:ogc:def:crs:EPSG::6269</DefaultCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::32615</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::32616</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::32617</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::32618</OtherCRS>
      <ows:WGS84BoundingBox>
         <ows:LowerCorner>-180 -90</ows:LowerCorner>
         <ows:UpperCorner>180 90</ows:UpperCorner>
      </ows:WGS84BoundingBox>
      <MetadataURL

     xlink:href="http://www.ogccatservice.com/csw.cgi?service=CSW&amp;version=2.0.0&amp;request=Get
```

```
      Records&amp;constraintlanguage=CQL&amp;recordid=urn:uuid:57973502-29cb-4114-8d64-
      9939627a0414"/>
        <ExtendedDescription>
           <Element name="TemporalExtent" type="xsd:date">
              <ows:Metadata
                 xlink:href="http://www.someserver.com/AboutTemporalExtent.html"/>
              <ValueList>
                 <Value>2000-01-01</Value>
                 <Value>2006-01-31</Value>
              </ValueList>
           </Element>
        </ExtendedDescription>
     </FeatureType>
  </FeatureTypeList>
  <fes:Filter_Capabilities>
     <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="ImplementsFunctions">
           <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>TRUE</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>TRUE</ows:DefaultValue>
        </fes:Constraint>
        <fes:Constraint name="ImplementsMinTemporalFilter">
           <ows:NoValues/>
           <ows:DefaultValue>TRUE</ows:DefaultValue>
        </fes:Constraint>
        <fes:Constraint name="ImplementsTemporalFilter">
           <ows:NoValues/>
           <ows:DefaultValue>TRUE</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:Constraint name="ImplementsExtendedOperators">
           <ows:NoValues/>
           <ows:DefaultValue>FALSE</ows:DefaultValue>
        </fes:Constraint>
     </fes:Conformance>
     <fes:Id_Capabilities>
        <fes:ResourceIdentifier name="fes:ResourceId"/>
     </fes:Id_Capabilities>
     <fes:Scalar_Capabilities>
        <fes:LogicalOperators/>
        <fes:ComparisonOperators>
           <fes:ComparisonOperator name="PropertyIsLessThan"/>
           <fes:ComparisonOperator name="PropertyIsGreaterThan"/>
           <fes:ComparisonOperator name="PropertyIsLessThanOrEqualTo"/>
           <fes:ComparisonOperator name="PropertyIsGreaterThanOrEqualTo"/>
           <fes:ComparisonOperator name="PropertyIsEqualTo"/>
```

```
      <fes:ComparisonOperator name="PropertyIsNotEqualTo"/>
      <fes:ComparisonOperator name="PropertyIsLike"/>
      <fes:ComparisonOperator name="PropertyIsBetween"/>
      <fes:ComparisonOperator name="PropertyIsNull"/>
      <fes:ComparisonOperator name="PropertyIsNil"/>
   </fes:ComparisonOperators>
</fes:Scalar_Capabilities>
<fes:Spatial_Capabilities>
   <fes:GeometryOperands>
      <fes:GeometryOperand name="gml:Point"/>
      <fes:GeometryOperand name="gml:MultiPoint"/>
      <fes:GeometryOperand name="gml:LineString"/>
      <fes:GeometryOperand name="gml:MultiLineString"/>
      <fes:GeometryOperand name="gml:Curve"/>
      <fes:GeometryOperand name="gml:MultiCurve"/>
      <fes:GeometryOperand name="gml:Polygon"/>
      <fes:GeometryOperand name="gml:MultiPolygon"/>
      <fes:GeometryOperand name="gml:Surface"/>
      <fes:GeometryOperand name="gml:MultiSurface"/>
      <fes:GeometryOperand name="gml:MultiGeometry"/>
      <fes:GeometryOperand name="gml:Box"/>
      <fes:GeometryOperand name="gml:Envelope"/>
   </fes:GeometryOperands>
   <fes:SpatialOperators>
      <fes:SpatialOperator name="BBOX"/>
      <fes:SpatialOperator name="Equals"/>
      <fes:SpatialOperator name="Disjoint"/>
      <fes:SpatialOperator name="Intersects"/>
      <fes:SpatialOperator name="Touches"/>
      <fes:SpatialOperator name="Crosses"/>
      <fes:SpatialOperator name="Within"/>
      <fes:SpatialOperator name="Contains"/>
      <fes:SpatialOperator name="Overlaps"/>
      <fes:SpatialOperator name="Beyond"/>
      <fes:SpatialOperator name="DWithin"/>
   </fes:SpatialOperators>
</fes:Spatial_Capabilities>
<fes:Temporal_Capabilities>
   <fes:TemporalOperands>
      <fes:TemporalOperand name="gml:validTime"/>
      <fes:TemporalOperand name="gml:TimeInstant"/>
      <fes:TemporalOperand name="gml:TimePeriod"/>
      <fes:TemporalOperand name="gml:timePosition"/>
      <fes:TemporalOperand name="gml:timeInterval"/>
      <fes:TemporalOperand name="gml:duration"/>
   </fes:TemporalOperands>
   <fes:TemporalOperators>
      <fes:TemporalOperator name="After"/>
      <fes:TemporalOperator name="Before"/>
      <fes:TemporalOperator name="Begins"/>
      <fes:TemporalOperator name="BegunBy"/>
      <fes:TemporalOperator name="TContains"/>
      <fes:TemporalOperator name="During"/>
      <fes:TemporalOperator name="TEquals"/>
      <fes:TemporalOperator name="TOverlaps"/>
      <fes:TemporalOperator name="Meets"/>
      <fes:TemporalOperator name="OverlappedBy"/>
      <fes:TemporalOperator name="MetBy"/>
      <fes:TemporalOperator name="EndedBy"/>
   </fes:TemporalOperators>
</fes:Temporal_Capabilities>
<fes:Functions>
   <fes:Function name="min">
      <fes:Returns>xsd:double</fes:Returns>
      <fes:Arguments>
         <fes:Argument name="value1">
            <fes:Type>xsd:double</fes:Type>
         </fes:Argument>
         <fes:Argument name="value2">
            <fes:Type>xsd:double</fes:Type>
         </fes:Argument>
      </fes:Arguments>
   </fes:Function>
   <fes:Function name="max">
      <fes:Returns>xsd:double</fes:Returns>
      <fes:Arguments>
```

```
            <fes:Argument name="value1">
               <fes:Type>xsd:double</fes:Type>
            </fes:Argument>
            <fes:Argument name="value2">
               <fes:Type>xsd:double</fes:Type>
            </fes:Argument>
         </fes:Arguments>
      </fes:Function>
   </fes:Functions>
</fes:Filter_Capabilities>
</WFS_Capabilities>
```

B.8 KVP examples

B.8.1 Conventions

In general, URL encoding requires that certain characters, such as '&', be escaped (see IETF RFC 2396) when they are not used in their intended manner. In this clause, however, such characters are not escaped for the sake of clarity.

In addition, many of the examples in this clause include a FILTER parameter whose value is an XML-encoded filter as specified in ISO 19143:2010, Clause 7. To be rigorously correct, these examples should include namespace and schema location information in the root element, fes:Filter, such that the XML may be validated. Thus the parameter:

```
FILTER=<Filter><Within><PropertyName>InWaterA_1M/wkbGeom<PropertyName>
       <gml:Envelope><gml:lowerCorner>10 10</gml:lowerCorner><gml:upperCorner>20
       20</gml:upperCorner></gml:Envelope></Within></Filter>
```

should more correctly be specified as:

```
FILTER=<Filter xmlns="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/filter.xsd
                                   http://www.opengis.net/gml/3.2
                                   http://schemas.opengis.net/gml/3.2.1/gml.xsd">
       <Within><PropertyName>InWaterA_1M/wkbGeom<PropertyName>
       <gml:Envelope><gml:lowerCorner>10 10</lowerCorner>
       <gml:upperCorner>20,20</gml:upperCorner></gml:Envelope>
       </Within></Filter>
```

In order to not obfuscate the essential information, however, the namespace and schema location attribute tags have been omitted from the examples in this clause. In addition, the schema locations shown are only example locations and the correct schema locations would need to be substituted.

Finally, the values of the FILTER and other parameters are broken up over several lines, again for the sake of clarity. A FILTER parameter's value, in practice, would be composed of a single, long string.

B.8.2 DescribeFeatureType examples

B.8.2.1 Example 1

The following example requests the schema description of the feature type TreesA_1M.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.0&
       NAMESPACES=xmlns(myns,http://www.myserver.com/myns)
       REQUEST=DescribeFeatureType&
       TYPENAMES=myns:TreesA_1M
```

B.8.2.2 Example 2

The following example requests the schema description of the feature types InWaterA_1M and BuiltUpA_1M.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.0&
       NAMESPACES=xmlns(ns1,http://www.someserver.com/ns1),xmlns(ns2,http://someserver.com/ns2)&
       REQUEST=DescribeFeatureType&
       TYPENAMES=ns1:TreesA_1M,ns2:BuiltUpA_1M
```

B.8.3 GetPropertyValue examples

B.8.3.1 Introduction

The following set of examples assumes that the fictitious dataset described in B.4.1 is being queried.

B.8.3.2 Example 1

Find the city in which Fred Smith lives. This query uses the GetFeatureById stored query (see 7.9.3.6) to retrieve Fred Smith's record.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.0&
       REQUEST=GetPropertyValue&
       VALUEREFERENCE=myns:mailAddress/myns:Address/myns:city&
       STOREDQUERY_ID=urn:ogc:def:query:OGC-WFS::GetFeatureById&
       ID=p4456
```

B.8.3.3 Example 2

Find the location of Fred Smith.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.0&
       REQUEST=GetPropertyValue&
       VALUEREFERENCE=myns:location&
       RESOLVE=local&
       RESOLVEDEPTH=*&
       TYPENAMES=myns:Person&
       FILTER=<fes:Filter><fes:And><fes:PropertyIsEqualTo>
              <fes:ValueReference>myns:firstName</fes:ValueReference>
              <fes:Literal>Fred</fes:Literal></fes:PropertyIsEqualTo>
              <fes:PropertyIsEqualTo><fes:ValueReference>myns:lastName</fes:ValueReference>
              <fes:Literal>Smith</fes:Literal></fes:PropertyIsEqualTo></fes:And></fes:Filter>
```

B.8.3.4 Example 3

Find the number of lanes of the road that the house that Mary Smith lives in, fronts on.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.0&
       REQUEST=GetPropertyValue&
       VALUEREFERENCE=valueOf(myns:livesIn)/valueOf(myns:frontsOn)/abc:numLanes&
       TYPENAMES=myns:Person&
       FILTER=<fes:Filter><fes:And><fes:PropertyIsEqualTo>
              <fes:ValueReference>myns:firstName</fes:ValueReference>
              <fes:Literal>Fred</fes:Literal></fes:PropertyIsEqualTo>
              <fes:PropertyIsEqualTo><fes:ValueReference>myns:lastName</fes:ValueReference>
              <fes:Literal>Smith</fes:Literal></fes:PropertyIsEqualTo></fes:And></fes:Filter>
```

B.8.3.5 Example 4

What is Mary Smith's postal code?

```
http://www.someserver.com/wfs.cgi?
        SERVICE=WFS&
        VERSION=2.0.0&
        REQUEST=GetPropertyValue&
        VALUEREFERENCE=valueOf(myns:livesIn)/valueof(myns:mailAddress)/myns:postalCode&
        TYPENAMES=myns:Person&
        FILTER=<fes:Filter><fes:And><fes:PropertyIsEqualTo>
                <fes:ValueReference>myns:firstName</fes:ValueReference>
                <fes:Literal>Mary</fes:Literal></fes:PropertyIsEqualTo>
                <fes:PropertyIsEqualTo><fes:ValueReference>myns:lastName</fes:ValueReference>
                <fes:Literal>Smith</fes:Literal></fes:PropertyIsEqualTo></fes:And></fes:Filter>
```

B.8.3.6 Example 5

How old is Fred Smith?

```
http://www.someserver.com/wfs.cgi?
        SERVICE=WFS&
        VERSION=2.0.0&
        REQUEST=GetPropertyValue&
        VALUEREFERENCE=myns:age&
        TYPENAMES=myns:Person&
        FILTER=<fes:Filter><fes:And><fes:PropertyIsEqualTo>

        <fes:ValueReference>myns:firstName</fes:ValueReference>
        <fes:Literal>Fred</fes:Literal></fes:PropertyIsEqualTo><fes:PropertyIsEqualTo>
        <fes:ValueReference>myns:lastName</fes:ValueReference>
        <fes:Literal>Smith</fes:Literal></fes:PropertyIsEqualTo></fes:And></fes:Filter>
```

B.8.3.7 Example 6

What are Fred Smith's phone numbers?

```
http://www.someserver.com/wfs.cgi?
        SERVICE=WFS&
        VERSION=2.0.0&
        REQUEST=GetPropertyValue&
        VALUEREFERENCE=myns:phone&
        TYPENAMES=myns:Person&
        FILTER=<fes:Filter><fes:And><fes:PropertyIsEqualTo>
                <fes:ValueReference>myns:firstName</fes:ValueReference>
                <fes:Literal>Fred</fes:Literal></fes:PropertyIsEqualTo>
                <fes:PropertyIsEqualTo><fes:ValueReference>myns:lastName</fes:ValueReference>
                <fes:Literal>Smith</fes:Literal></fes:PropertyIsEqualTo></fes:And></fes:Filter>
```

B.8.3.8 Example 7

What is Fred Smith's second phone number?

```
http://www.someserver.com/wfs.cgi?
        SERVICE=WFS&
        VERSION=2.0.0&
        REQUEST=GetPropertyValue&
        VALUEREFERENCE=myns:phone[2]&
        TYPENAMES=myns:Person&
        FILTER=<fes:Filter><fes:And><fes:PropertyIsEqualTo>
                <fes:ValueReference>myns:firstName</fes:ValueReference>
                <fes:Literal>Fred</fes:Literal></fes:PropertyIsEqualTo>
                <fes:PropertyIsEqualTo><fes:ValueReference>myns:lastName</fes:ValueReference>
                <fes:Literal>Smith</fes:Literal></fes:PropertyIsEqualTo></fes:And></fes:Filter>
```

B.8.3.9 Example 8

Get the Road that Mary Smith's house fronts on.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.0&
       REQUEST=GetPropertyValue&
       VALUEREFERENCE= valueOf(myns:livesIn)/myns:frontsOn&
       TYPENAMES=myns:Person&
       FILTER=<fes:Filter><fes:And><fes:PropertyIsEqualTo>
              <fes:ValueReference>myns:firstName</fes:ValueReference>
              <fes:Literal>Mary</fes:Literal></fes:PropertyIsEqualTo>
              <fes:PropertyIsEqualTo><fes:ValueReference>myns:lastName</fes:ValueReference>
              <fes:Literal>Smith</fes:Literal></fes:PropertyIsEqualTo></fes:And></fes:Filter>
```

B.8.4 GetFeature examples

B.8.4.1 Example 1

Query all properties of all instances of type InWaterA_1M.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.0&
       REQUEST=GetFeature&
       TYPENAMES=InWaterA_1M
```

B.8.4.2 Example 2

Query some properties of all instances of type InWaterA_1M.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.0&
       REQUEST=GetFeature&
       PROPERTYNAME=InWaterA_1M/wkbGeom,InWaterA_1M/tileId&
       TYPENAMES=InWaterA_1M
```

B.8.4.3 Example 3

Query all properties of the feature instance identified by the feature identifier "InWaterA_1M.1013".

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.0&
       REQUEST=GetFeature&
       RESOURCEID=InWaterA_1M.1013
```

B.8.4.4 Example 4

Query all properties of an enumerated set of feature instances of type InWaterA_1M.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.0&
       REQUEST=GetFeature&
       RESOURCEID=InWaterA_1M.1013,InWaterA_1M.1014,InWaterA_1M.1015
```

B.8.4.5 Example 5

Query all properties of a spatially constrained set of feature instances of type InWaterA_1M.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.0&
```

```
        REQUEST=GetFeature&
        TYPENAMES=InWaterA_1M&
        BBOX=18.54,-72.3544,18.62,-72.2564
```

B.8.4.6 Example 6

Query some properties of a constrained set of feature instances of type InWaterA_1M. The result set is identified using a filter expression (see ISO 19143:2010, Clause 7) that combines spatial and non-spatial predicates.

```
http://www.someserver.com/wfs.cgi?
        SERVICE=WFS&
        VERSION=2.0.0&
        REQUEST=GetFeature&
        PROPERTYNAME=InWaterA_1M/wkbGeom,InWaterA_1M/tileId&
        TYPENAMES=InWaterA_1M&
        FILTER=<fes:Filter><fes:And><fes:PropertyIsBetween>
               <fes:ValueReference>InwaterA_1M/tileId</fes:ValueReference>
               <fes:LowerBoundary><fes:Literal>100</fes:Literal>
               </fes:LowerBoundary><fes:UpperBoundary>
               <fes:Literal>200</fes:Literal></fes:UpperBoundary>
               </fes:PropertyIsBetween><fes:Intersects>
               <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:Intersects></fes:And></fes:Filter>
```

B.8.4.7 Example 7

Find the fifth highest mountain in the region around the Himalayas.

```
http://www.someserver.com/wfs.cgi?
        SERVICE=WFS&
        VERSION=2.0.0&
        REQUEST=GetFeature&
        TYPENAMES=Mountains&
        BBOX=27.6669,86.4800,28.2709,87.2120&
        SORTBY=Elevation DESC&
        STARTINDEX=5&
        COUNT=1
```

B.8.4.8 Example 8

Query all properties of a list of feature types in different namespaces.

```
http://www.someserver.com/wfs.cgi?
        SERVICE=WFS&
        VERSION=2.0.0&
        REQUEST=GetFeature&
        NAMESPACES=xmlns(myns,http://www.someserver.com),xmlns(yourns,http://www.someotherserver.com)
        TYPENAMES=(myns:InWaterA_1M)(yourns:BuiltUpA_1M)
```

B.8.4.9 Example 9

Query some properties of a list of feature types. In this case, the attributes wkbGeom and tileId are fetched for the InWaterA_1M feature type and all the attributes of feature type BuiltUpA_1M are fetched.

```
http://www.someserver.com/wfs.cgi?
        SERVICE=WFS&
        VERSION=2.0.0&
        REQUEST=GetFeature&
        PROPERTY=(InWaterA_1M/wkbGeom,InWaterA_1M/tileId)(BuiltUpA_1M/*)&
        TYPENAMES=(InWaterA_1M)(BuiltUpA_1M)
```

B.8.4.10 Example 10

Query all properties of all feature instances of the feature type InWaterA_1M and BuiltUpA_1M that lie within the specified box. This example uses an equivalent filter expression rather than the BBOX parameter.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.0&
       REQUEST=GetFeature&
       TYPENAMES=(InWaterA_1M)(BuiltUpA_1M)&
       FILTER=(<Filter><Within><PropertyName>InWaterA_1M/wkbGeom
              <PropertyName><gml:Envelope><gml:lowerCorner>43.5707 -79.5797</gml:lowerCorner>
              <gml:upperCorner>43.8219 -79.2693</gml:upperCorner></gml:Envelope>
              </Within></Filter>)(<Filter><Within><PropertyName>
              BuiltUpA_1M/wkbGeom<PropertyName><gml:Envelope><gml:lowerCorner>43.5705 -79.5797
              </gml:lowerCorner><gml:upperCorner>43.8219 -79.2693</gml:upperCorner>
              </gml:Envelope></Within></Filter>)
```

B.8.4.11 Example 11

This example illustrates a GetFeature request using PROPERTYNAME components that serves as a base for comparison with Examples 12 and 13.

The request:

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.0&
       REQUEST=GetFeature&
       PROPERTYNAME=uk:Town/gml:name,uk:Town/gml:directedNode&
       TYPENAMES=uk:Town&
       RESOURCEID=t1
```

The response contains an **xlink:href**, but it is returned as-is:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection
   timeStamp="2009-02-14T01:27:44"
   numberMatched="1"
   numberReturned="1"
   xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs/2.0
                       http://schema.opengis.net/wfs/2.0.0/wfs.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:boundedBy>
      <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
         <gml:lowerCorner>49.7330 -11.5210</gml:lowerCorner>
         <gml:upperCorner>59.4518 -1.1409</gml:upperCorner>
      </gml:Envelope>
   </wfs:boundedBy>
   <wfs:member>
      <Town gml:id="t1">
         <gml:name>Bedford</gml:name>
         <gml:directedNode orientation="+" xlink:href="#n1"/>
      </Town>
   </wfs:member>
</wfs:FeatureCollection>
```

B.8.4.12 Example 12

This example illustrates the use of the RESOLVEDEPTH and RESOLVETIMEOUT components on the GetFeature request from Example 11.

The request:

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.0&
       REQUEST=GetFeature&
       PROPERTYNAME=uk:Town/gml:name,uk:Town/gml:directedNode&
       RESOLVEDEPTH=1&
       RESOLVETIMEOUT=1&
       TYPENAMES=uk:Town&
       RESOURCEID=t1
```

In the response, the first level xlink:href has been traversed, and its contents returned as described in 7.6.4:

```
<?xml version="1.0"?>
<wfs:FeatureCollection
   timeStamp="2009-02-14T01:27:44"
   numberMatched="1"
   numberReturned="1"
   xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs/2.0
                       http://schema.opengis.net/wfs/2.0.0/wfs.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:boundedBy>
      <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
         <gml:lowerCorner>10 10</gml:lowerCorner>
         <gml:upperCorner>20 20</gml:upperCorner>
      </gml:Envelope>
   </wfs:boundedBy>
   <wfs:member>
      <Town gml:id="t1">
         <gml:name>Bedford</gml:name>
         <gml:directedNode orientation="+" xlink:href="#n1"/>
         <!-- xlink:href="#n1"/> -->
         <gml:directedNode orientation="+">
            <gml:Node gml:id="n1">
               <gml:pointProperty xlink:href="http://www.bedford.town.uk/civilwo
rks/gps.gml#townHall"/>
            </gml:Node>
         </gml:directedNode>
      </Town>
   </wfs:member>
</wfs:FeatureCollection>
```

B.8.4.13 Example 13

This example illustrates the use of the RESOLVEDEPTH and RESOLVETIMEOUT components on the GetFeature request from Example 11.

The request:

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.0&
       REQUEST=GetFeature&
       PROPERTYNAME=uk:Town/gml:name,uk:Town/gml:directedNode&
       RESOLVEDEPTH=2&
       RESOLVETIMEOUT=2&
       TYPENAMES=uk:Town&
       RESOURCEID=t1
```

In the response, the first and second level xlink:href in the gml:directedNode property have been traversed, and their contents returned as described in 7.6.4:

```
<?xml version="1.0"?>
<wfs:FeatureCollection
   timeStamp="2009-02-14T01:27:44"
```

```
   numberMatched="1"
   numberReturned="1"
   xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs/2.0
                       http://schema.opengis.net/wfs/2.0.0/wfs.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <wfs:boundedBy>
      <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
         <gml:lowerCorner>10 10</gml:lowerCorner>
         <gml:upperCorner>20 20</gml:upperCorner>
      </gml:Envelope>
   </wfs:boundedBy>
   <wfs:member>
      <Town gml:id="t1">
         <gml:name>Bedford</gml:name>
         <gml:directedNode orientation="+" xlink:href="#n1"/>
         <!-- xlink:href="#n1"/> -->
         <gml:directedNode orientation="+">
            <gml:Node gml:id="n1">
               <gml:pointProperty>
                  <gml:Point gml:id="townHall" srsName="urn:ogc:def:crs:EPSG::4326">
                     <gml:pos>-47 34</gml:pos>
                  </gml:Point>
               </gml:pointProperty>
            </gml:Node>
         </gml:directedNode>
      </Town>
   </wfs:member>
</wfs:FeatureCollection>
```

B.8.4.14 Example 14

The following example encodes a join query between two feature types, myns:Parks and myns:Lakes:

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.2
       NAMESPACES=xmlns(myns,http://www.someserver.com/myns)&
       REQUEST=GetFeature&
       TYPENAMES=myns:Parks,myns:Lakes&
        FILTER=<fes:Filter><fes:And><fes:PropertyIsEqualTo><fes:ValueReference>/myns:Parks</fes:ValueR
        eference><fes:Literal>Algonquin+Park</fes:Literal></fes:PropertyIsEqualTo><fes:Contains><fes:V
        alueReference>/myns:Parks/geometry</fes:ValueReference><fes:ValueReference>/myns:Lakes/geometr
        y</fes:ValueReference></fes:Contains></fes:And></fes:Filter>
```

B.8.4.15 Example 15

The following example encodes two separate ad-hoc queries on two feature types, myns:Parks and myns:Lakes:

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=2.0.2
       NAMESPACES=xmlns(myns,http://www.someserver.com/myns)&
       REQUEST=GetFeature&
       TYPENAMES=(myns:Parks)(myns:Lakes)&
        FILTER=(<fes:Filter><fes:BBOX><fes:ValueReference>/RS1/geometry</fes:ValueReference>
                <gml:Envelope+srsName="urn:ogc:def:crs:EPSG::4326">
                <gml:lowerCorner>49.1874 -123.2778</gml:lowerCorner>
                <gml:upperCorner>49.3504 -122.8892</gml:upperCorner>
                </gml:Envelope></fes:BBOX><fes:Filter>)(<fes:Filter>
                <fes:BBOX><fes:ValueReference>/RS1/geometry</fes:ValueReference>
                <gml:Envelope+srsName="urn:ogc:def:crs:EPSG::4326">
                <gml:lowerCorner>44.6113 -63.7058</gml:lowerCorner>
                <gml:upperCorner>44.7256 -63.4641</gml:upperCorner>
                </gml:Envelope></fes:BBOX><fes:Filter>)
```