Open GIS Consortium Inc.

Date: 17-MAY-2002

Reference number of this OpenGIS® project document: OGC 02-058

Version: 1.0.0

Category: OpenGIS® Implementation Specification

Status: Adopted Specification

Editor: Panagiotis A. Vretanos

Web Feature Service Implementation Specification

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

Contents

i. Preface

One important achievement of the Open GIS Consortium (OGC) Web Mapping Test bed (WMT1) initiative was the development of a large consensus around open web based interface specifications. Such specifications allow software vendors to implement their products using interoperable interfaces and provide end-users a larger pool of interoperable web based tools for geodata access and related geoprocessing services.

During the OGC WMT1 project, two web based draft specification documents were developed:

1. OpenGIS© Web Map Service Implementation Specification [1]

2. OpenGIS© Geography Markup Language (GML) 2.0 Implementation Specification

The first document specifies web interfaces based on a model supporting general request and response rules using Hypertext Transfer Protocol (HTTP) and the eXtensible Markup Language (XML). Web Map Server products have been developed as the result of the adoption by OGC of the OpenGIS© Web Map Service Implementation Specification [1]. The interfaces defined in that specification include *GetCapabilities*, *GetMap* and *GetFeatureInfo*. The second document describes an encoding specification for geodata in XML. The encoding described in that specification is intended to enable the transport and storage of geographic information in XML including both properties and the geometry of geographic features.

This document (OpenGIS© Web Feature Service Implementation Specification) takes the next logical step and proposes interfaces for describing data manipulation operations on geographic features using HTTP as the distributed computing platform. Data manipulation operations include the ability to:

1. Create a new feature instance

2. Delete a feature instance

3. Update a feature instance

4. Get or Query features based on spatial and non-spatial constraints

A Web Feature Service (WFS) request consists of a description of query or data transformation operations that are to be applied to one or more features. The request is generated on the client and is posted to a web feature server using HTTP. The web feature server then reads and (in a sense) executes the request.

ii. Submitting organizations

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

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

Intergraph Corp.
Jonathan Clark
1881 Campus Commons Drive
Reston, VA 20191
U.S.A
jrclark@intergraph.com

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

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

iii. Submission contact points

All questions regarding this submission should be directed to the Editor or to the WWW Mapping SIG chair:

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

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

Additional contributors

Rob Atkinson (Social Change Online) rob@socialchange.net.au
Craig Bruce (CubeWerx) csbruce@cubwerx.com
Jonathan Clark (Intergraph) jrclark@intergraph.com
Adrian Cuthbert (SpotOn MOBILE) adrian@spotonmobile.com
Paul Daisey (U.S. Census) pdaisey@geo.census.gov
John Davidson georef@erols.com
John D. Evans (NASA) john.evans@gsfc.nasa.gov
Ron Fresne (ObjectFX) RonF@ObjectFX.com
Ignacio Guerrero (Intergraph) IGuerrer@ingr.com
John Herring (Oracle Corp.) John.Herring@oracle.com
Sandra Johnson (Mapinfo) Sandra_Johnson@mapinfo.com
Edric Keighan (CubeWerx) ekeighan@cubewerx.com
Ron Lake (Galdos Systems Inc.) rlake@galdosinc.com
Jeff Lansing (Polexis) jeff@polexis.com
Seb Lessware (Laser-Scan Ltd.) sebl@lsl.co.uk
Marwa Mabrouk (ESRI) mmabrouk@esri.com
Serge Margoulies (Ionic) Serge.Margoulies@ionicsoft.com
Brian May (CubeWerx) bmay@cubewerx.com
Richard Martell (Galdos Systems Inc.) rmartell@galdosinc.com
Aleksander Milanovic (Glados Systems Inc.) amilanovic@gladosinc.com
Dimitri Monie (Ionic) dimitri.monie@ionicsoft.com
Paul Pilkington (Laser-Scan Ltd.) paulpi@lsiva.com
Keith Pomakis (CubeWerx) pomakis@cubewerx.com
Christopher C. Pried (Polexis) ccp@polexis.com
Lou Reich (NASA) louis.i.reich@gsfc.nasa.gov
Carl Reed (Open GIS Consortium) creediii@mindspring.com
Martin Schaefer (Cadcorp Ltd.) martins@cadcorpdev.co.uk
Lacey T. Sharpe (Intergraph Corp.) ltsharpe@ingr.com
Raj R. Singh (Syncline Inc.) rs@syncline.com
Bernard Snyers (Ionic) Bernard.Snyers@ionicsoft.com
Daniel Specht (TEC) specht@tec.army.mil
James T. Stephens (Lockheed Martin) james.t.stephens@lmco.com
Glenn Stowe (CubeWerx) gstowe@cubwerx.com
Tom Strickland (Byers) tom.strickland@byers.com
Shuichi Takino (Dawn Corp.) takino@dawn-corp.co.jp
Milan Trninic (Galdos Systems Inc.) mtrninic@galdosinc.com
John T. Vincent (Intergraph Corp.) jtvincen@intergraph.com
Peter Woodsford (Laser-Scan Ltd.) peterw@lsl.co.uk
Arliss Whitesize (BAE Systems) Arliss.Whiteside@baesystems.com
Nami Yamashita (Dawn Corp.) yamashita@dawn-corp.co.jp

iv. Revision history

0.0.0	Address RFC comments.
0.0.14	Reformat document in ISO format; Relate document to OGC abstract specification (specifically Topic 12 / 19119); Include rules for property naming; Use XPath expressions for referencing properties in complex attributes; More synchronization between WMS and WFS with respect to keyword-value pair encoding; Add annex for conformance testing.
0.0.13	Prepare document for RFC submission; include XML-Schema encoding of WFS interfaces; align URL-encoding with BSM
0.0.12	Add complete list of contributors; align with latest GML 2.0 draft specification; add lock controls and versioning.

0.0.11	Correct typographical errors.
0.0.10	Server FeatureId and Filter elements into their own specification documents.
0.0.9	Review U.S.Census revisions
0.0.8	Review Galdos revisions
0.0.7	Review LaserScan revisions
0.0.6	Remove "Small XML-Schema Description Language"
0.0.5	Define "Small XML-Schema Description Language"
0.0.4	Use GML2 with application defined schema using XML-Schema. Remove dependency on featureType attribute.
0.0.3	Define GET request semantics.
0.0.2	Update <FeatureId> element to include <Scope>. Make handle attribute #IMPLIED. Add functions on properties and literals to <Filter>.
0.0.1	First version derived from the OpenGIS Transaction Encoding Specification [3] and the Spatial Object Transfer Format (SOTF) [4] specification.

v. Changes to the OpenGIS Abstract Specification

No further revisions to the OGC Abstract Specification are required. The revisions previously approved for Topic 12, "Service Architecture," including definitions of the terms "operation", "interface" and "service" are relevant to and sufficient for this specification. The essential operation of a web feature service, as a feature access and management service, is described in section 8.3.3 of Topic 12.

vi. Future work

Further work is desirable in the next version on the following work items.

1. XLink traversal capabilities,
2. UML models,
3. Change the abstract specification to include complex features.
4. Develop specific conformance tests.
5. GML3 integration.

Foreword

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

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

Normative annexes

Annexes A and B are normative.

Introduction

The OGC Web Map Service allows a client to overlay map images for display served from multiple Web Map Services on the Internet. In a similar fashion, the OGC Web Feature Service allows a client to retrieve geospatial data encoded in Geography Markup Language (GML) from multiple Web Feature Services.

The requirements for a Web Feature Service are:

1. The interfaces must be defined in XML.
2. GML must be used to express features within the interface.
3. At a minimum a WFS must be able to present features using GML.
4. The predicate or filter language will be defined in XML and be derived from CQL as defined in the OpenGIS Catalogue Interface Implementation Specification.
5. The datastore used to store geographic features should be opaque to client applications and their only view of the data should be through the WFS interface.
6. The use of a subset of XPath expressions for referencing properties.

The purpose of this document is to describe the Web Feature Service interface, as illustrated in figure 1.

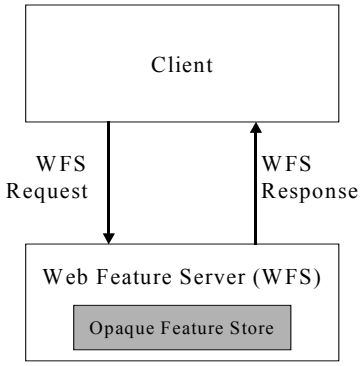

Figure 1 – Web feature service

This document is derived from a large consensus among its contributors and takes its roots from two independently proposed specifications titled OGC Transaction Encoding Specification [3] and Spatial Object Transfer Format (SOTF) [4]. In addition a number

of concepts, common among all OGC services, are taken from the Web Map Service Implementation Specification [1].

Web Feature Service Implementation Specification

1 Scope

This document describes the OGC Web Feature Service (WFS) operations. The WFS operations support INSERT, UPDATE, DELETE, QUERY and DISCOVERY operations on geographic features using HTTP as the distributed computing platform.

In the context of this document, a transaction is a logical unit of work that is composed of one or more data manipulation operations. Since the manner in which geographic features are persistently stored is not addressed in this document, no transaction semantics, such as atomic failure, are assumed to exist. It is the function of a web feature service, in its interaction with the data storage system used to persistently store features, to ensure that changes to data are consistent. However, the document also acknowledges the fact that many systems do support standard concurrent transaction semantics and so proposes optional operations that will allow a web feature service to take advantage of such systems (e.g. relational database systems based on SQL).

Geographic features

This document adopts the same concept of a geographic feature as described in the OGC Abstract Specification (http://www.opengis.org/techno/spec.htm) and interpreted in the OpenGIS© Geographic Markup Language(GML) Implementation Specification [2]. That is to say that the state of a geographic feature is described by a set of properties where each property can be thought of as a {name, type, value} tuple. The name and type of each feature property is determined by its type definition. Geographic features are those that may have at least one property that is geometry-valued. This, of course, implies that features can be defined with no geometric properties at all. The geometries of geographic features are restricted to what OGC calls simple geometries. A simple geometry is one for which coordinates are defined in two dimensions and the delineation of a curve is subject to linear interpolation. The traditional 0, 1 and 2-dimensional geometries defined in a 2-dimensional spatial reference system are represented by points, line strings and polygons. In addition, the OGC geometry model allows for geometries that are collections of other geometries - either homogeneous multi-point, multi-line string, and multi-polygon collections or heterogeneous geometry collections. Finally, GML allows features that have complex or aggregate non-geometric properties.

Processing requests

This section of the document outlines, in general terms, the protocol to be followed in order to process web feature service requests. Processing requests would proceed as follows:

1. A client application would request a capabilities document from the WFS. Such a document contains a description of all the operations that the WFS supports and a list of all feature types that it can service.

2. A client application (optionally) makes a request to a web feature service for the definition of one or more of the feature types that the WFS can service.

3. Based on the definition of the feature type(s), the client application generates a request as specified in this document.

4. The request is posted to a web server.

5. The WFS is invoked to read and service the request.

6. When the WFS has completed processing the request, it will generate a status report and hand it back to the client. In the event that an error has occurred, the status report will indicate that fact.

Operations

To support transaction and query processing, the following operations are defined:

GetCapabilities

A web feature service must be able to describe its capabilities. Specifically, it must indicate which feature types it can service and what operations are supported on each feature type.

DescribeFeatureType

A web feature service must be able, upon request, to describe the structure of any feature type it can service.

GetFeature

A web feature service must be able to service a request to retrieve feature instances. In addition, the client should be able to specify which feature properties to fetch and should be able to constrain the query spatially and non-spatially.

Transaction

A web feature service may be able to service transaction requests. A transaction request is composed of operations that modify features; that is create, update, and delete operations on geographic features.

LockFeature

A web feature service may be able to process a lock request on one or more instances of a feature type for the duration of a transaction. This ensures that serializable transactions are supported.

Based on the operation descriptions above, two classes of web feature services can be defined:

Basic WFS

A basic WFS would implement the GetCapabilities, DescribeFeatureType and GetFeature operations. This would be considered a READ-ONLY web feature service.

Transaction WFS

A transaction web feature service would support all the operations of a basic web feature service and in addition it would implement the Transaction operation. Optionally, a transaction WFS could implement the LockFeature operation.

Figure 2 is a simplified protocol diagram illustrating the messages that might be passed back and forth between a client application and a web feature service in order to process a typical transaction request. The elements referenced in the diagram are defined in this document.

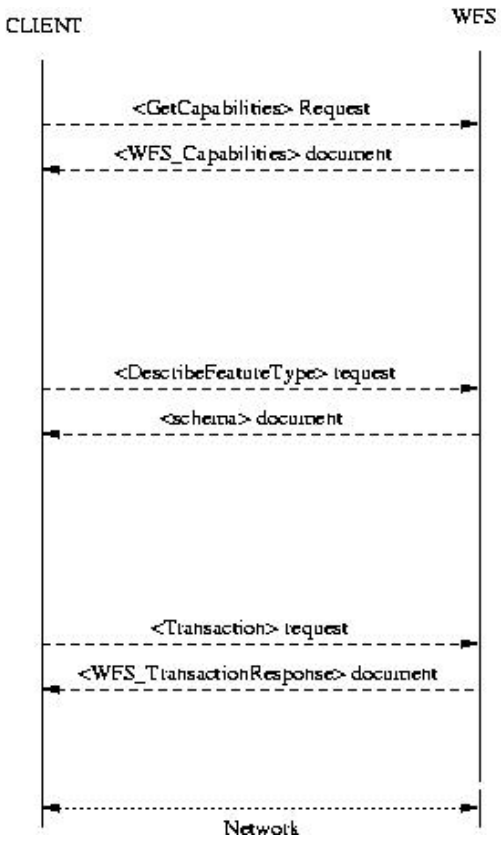

Figure 2 – Protocol diagram

2 Conformance

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

3 Normative references

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

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

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

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

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

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

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

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

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

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

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

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

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

4 Terms and definitions

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

4.1
operation
specification of a transformation or query that an object may be called to execute [4]

4.2
interface
a named set of operations that characterize the behavior of an entity [4]

4.3
service
a distinct part of the functionality that is provided by an entity through interfaces [4]

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

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

4.6
request
an invocation by a client of an operation.

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

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

4.9
spatial reference system
as defined in ISO19111

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

5 Conventions

5.1 Normative verbs

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

5.2 Abbreviated terms

CGI	Common Gateway Interface
DCP	Distributed Computing Platform
DTD	Document Type Definition
EPSG	European Petroleum Survey Group
GIS	Geographic Information System
GML	Geography Markup Language
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
MIME	Multipurpose Internet Mail Extensions
OGC	Open GIS Consortium
OWS	OGC Web Service
URL	Uniform Resource Locator
WFS	Web Feature Service
XML	Extensible Markup Language

5.3 Use of examples

This specification makes extensive use of XML examples. They are meant to illustrate the various aspects of a web feature service discussed in this specification. While every effort has been made to ensure that the examples are well formed and valid, this goal was sacrificed for the sake of clarity in many cases. For example, many examples are formatted in a specific way to highlight a particular aspect that would render the example invalid from the perspective of an XML validation tool. Further, most examples reference fictitious servers and data.

Thus, this specification does not assert that any XML or keyword-value pair encoded example, copied from this document, will necessarily execute correctly or validate using a particular XML validation tool. Only sections marked as *normative* should be expected to be well formed and valid XML or XML Schema documents.

6 Basic service elements

6.1 Introduction

This section describes aspects of OGC Web Feature Service behavior 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

The published specification version number contains three positive integers, separated by decimal points, in the form "x.y.z". The numbers "y" and "z" will never exceed 99. Each OWS specification is numbered independently.

6.2.2 Version changes

A particular specification's version number **shall** be changed with each revision. The number **shall** increase monotonically and **shall** comprise no more than three integers separated by decimal points, with the first integer being the most significant. There may be gaps in the numerical sequence. Some numbers may denote experimental or interim versions. Service instances and their clients need not support all defined versions, but **must** obey the negotiation rules below.

6.2.3 Appearance in requests and in service metadata

The version number appears in at least two places: in the Capabilities XML describing a service, and in the parameter list of client requests to that service. The version number used in a client's request of a particular service instance **must** be equal to a version number which that instance has declared it supports (except during negotiation as described below). A service instance may support several versions whose values clients may discover according to the negotiation rules.

6.2.4 Version number negotiation

An OWS Client may negotiate with a Service Instance to determine a mutually agreeable specification version. Negotiation is performed using the GetCapabilities operation [sec. 12] according to the following rules.

All Capabilities XML must include a protocol version number. In response to a GetCapabilities request containing a version number, an OGC Web Service **must** either respond with output that conforms to that version of the specification, **or** negotiate a mutually agreeable version if the requested version is not implemented on the server. If no version number is specified in the request, the server **must** respond with the highest version it understands and label the response accordingly.

Version number negotiation occurs as follows:

1. If the server implements the requested version number, the server **must** send that version.

2. If the client request is for an unknown version greater than the lowest version that the server understands, the server **must** send the highest version less than the requested version.

3. If the client request is for a version lower than any of those known to the server, then the server **must** send the lowest version it knows.

4. If the client does not understand the new version number sent by the server, it **may** either cease communicating with the server **or** send a new request with a new version number that the client does understand, but which is less than that sent by the server (if the server had responded with a lower version).

5. If the server had responded with a higher version (because the request was for a version lower than any known to the server), and the client does not understand the proposed higher version, then the client **may** send a new request with a version number higher than that sent by the server.

The process is repeated until a mutually understood version is reached, or until the client determines that it will not or cannot communicate with that particular server.

Example 1: Server understands versions 1, 2, 4, 5 and 8. Client understands versions 1, 3, 4, 6, and 7. Client requests version 7. Server responds with version 5. Client requests version 4. Server responds with version 4, which the client understands, and the negotiation ends successfully.

Example 2: Server understands versions 4, 5 and 8. Client understands version 3. Client requests version 3. Server responds with version 4. Client does not understand that version or any higher version, so negotiation fails and client ceases communication with that server.

6.3 General HTTP request rules

6.3.1 Introduction

At present, the only distributed computing platform (DCP) explicitly supported by OGC Web Services is the World Wide Web itself, or more specifically, Internet hosts implementing the Hypertext Transfer Protocol (HTTP)[9]. Thus the Online Resource of each operation supported by a service instance is located by an HTTP Uniform Resource Locator (URL). The URL may be different for each operation, or the same, at the discretion of the service provider. Each URL **must** conform to the description in [9], but is otherwise implementation-dependent; only the parameters comprising the service request itself are mandated by the OGC Web Services specifications.

HTTP supports two request methods: GET and POST. One or both of these methods may be defined for a particular OGC Web Service type and offered by a service instance. The use of the Online Resource URL differs in each case.

6.3.2 HTTP GET

An Online Resource URL intended for HTTP GET requests, is, in fact, only a URL prefix to which additional parameters must be appended in order to construct a valid

Operation request. A URL prefix is defined as an opaque string including the protocol, hostname, optional port number, path, a question mark '?', and, **optionally**, one or more server-specific parameters ending in an ampersand '&'. The prefix uniquely identifies the particular service instance. A client appends the necessary request parameters as name/value pairs in the form "name=value&". The resulting URL **must** be valid according to the HTTP Common Gateway Interface (CGI) standard [7], which mandates the presence of '?' before the sequence of query parameters and the '&' between each parameter. As with all CGI applications, the query URL is encoded [10] to protect special characters.

The URL prefix **must** end in either a '?' (in the absence of additional server-specific parameters) or a '&'. In practice, however, Clients **should** be prepared to add a necessary trailing '?' or '&' before appending the Operation parameters defined in this specification in order to construct a valid request URL.

Table 1 summarizes the components of an operation request URL.

Table 1 – A general OGC Web Service Request

URL Component	Description
http://host[:port]/path?{name[=value]&}	URL prefix of service operation. [ ] denotes 0 or 1 occurrence of an optional part; {} denotes 0 or more occurrences. The prefix is entirely at the discretion of the service provider.
name=value&	One or more standard request parameter name/value pairs defined by an OGC Web Service. The actual list of required and optional parameters is mandated for each operation by the appropriate OWS specification.

6.3.3 HTTP POST

An Online Resource URL intended for HTTP POST requests is a complete and valid URL to which Clients transmit encoded requests in the body of the POST document. A WFS **must not** require additional parameters to be appended to the URL in order to construct a valid target for the Operation request.

6.4 General HTTP response rules

Upon receiving a valid request, the service **must** send a response corresponding exactly to the request as detailed in the appropriate specification. Only in the case of Version Negotiation (described above) may the server offer a differing result.

Upon receiving an invalid request, the service **must** issue a Service Exception as described in Section 7.7.

NOTE: As a practical matter, in the WWW environment a client should be prepared to receive either a valid result, or nothing, or any other result. This is because the client may itself have formed a non-conforming request that inadvertently triggered a reply by something other than an OGC Web Service, because the Service itself may be non-conforming.

Response objects **must** be accompanied by the appropriate Multipurpose Internet Mail Extensions (MIME) type [9] for that object.

Response objects **should** be accompanied by other HTTP entity headers as appropriate and to the extent possible. In particular, the Expires and Last-Modified headers provide important information for caching; Content-Length may be used by clients to know when data transmission is complete and to efficiently allocate space for results, and Content-Encoding or Content-Transfer-Encoding may be necessary for proper interpretation of the results.

6.5 Request encoding

This document defines two methods of encoding WFS requests. The first uses XML as the encoding language. The second method uses keyword-value pairs to encode the various parameters of a request. An example of a keyword value pair is:

```
"REQUEST=GetCapabilities"
```

where "REQUEST" is the keyword and "GetCapabilities" is the value. In both cases, the response to a request or exception reporting must be identical.

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

Table 2 – Operation Request Encoding

Operation	Request Encoding
GetCapabilities	XML & KVP
DescribeFeatureType	XML & KVP
GetFeature / GetFeatureWithLock	XML & KVP
LockFeature	XML & KVP
Transaction	XML & limited KVP

KVP = keyword-value pair

This document mandates the use of GML for the XML encoding of the state of geographic features. A complete description of this encoding can be found in document [2].

6.6 Namespaces

Namespaces (17) are used to discriminate XML vocabularies from one another. For the WFS there are three normative namespace definitions, namely:

- (http://www.opengis.net/wfs) - for the WFS interface vocabulary
- (http://www.opengis.net/gml) - for the GML vocabulary

- (http://www.opengis.net/ogc) - for the OGC Filter vocabulary

A given WFS implementation will make use of one or more GML Application Schemas and these schemas will use, in turn, one or more application namespaces (e.g. http://www.someserver.com/myns). While many of the examples in this document use a single namespace, multiple namespaces can be used, as shown section 11.2.6.

7 Common elements

7.1 Feature identifier

This document assumes that every feature instance that a particular WFS implementation can operate upon is uniquely identifiable. That is to say, when a WFS implementation reports a feature identifier for a feature instance, that feature identifier is unique to the server and can be used to repeatedly reference the same feature instance (assuming it has not been deleted). It is further assumed that a feature identifier is encoded as described in the OpenGIS© Filter Encoding Implementation Specification [3]. A feature identifier can be used wherever a feature reference is required. For reference purposes, the XML Schema fragment that defines the feature identifier element is copied from the filter encoding specification:

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

The purpose of the feature identifier is to make database operations possible.

7.1.1 Globally unique identifiers (Informative)

For the purposes of a web feature service, a locally unique identifier is sufficient. However, there is a need within OGC web services to have unique identifiers for objects of all kinds. The approach thus far has been to reference objects using independent *scope* and *feature-id* components, where the scope it the URL of the server serving a feature and the feature-id is the local identifier for the feature. This approach, however, may be awkward to transport and use in other contexts, such as in a registry if one wanted to create metadata for a single repository data instance (such as a satellite image).

The purpose of this section of the specification is to point out that a single globally-unique string would be more convenient to use in multiple contexts, and that such a string may be generated by a web feature service using some combination of the URL of the service and the local identifier.

This string could be used as if it were fully opaque in many contexts, but it would be more useful if it were actually a URL or URN which could be used to directly access the object it identifies in the native format of the object. The encoding of the URL or URN would be entirely implementation-specific. One note on the use of URNs: not many implementations will actually be able to resolve and fetch data objects; it may be mostly only usable as a unique identifier string.

Using a URL or URN is helpful for applications that need only simple access to the raw objects since no interface details need to be known. This mode of access/identification is also helpful for integration with high-level XML technologies such as RDF or XSLT, and even for debugging purposes.

7.2 Feature state

The definition of features served by a WFS is provided by a GML application schema. Section 8 describes how a client can request an XML document containing the GML application schema definition of one or more feature types served by a WFS. Such application schema definitions **shall** conform to the OpenGIS Geography Markup Language(GML) Implementation Specification, version 2.1.1 [2].

A client application uses the GML application schema definition of a feature type to refer to feature instances of that feature type, and to refer to the names and types of all the properties of those feature instances. The values of all properties of a feature instance constitute the state of that feature instance. A client application references feature instances by the name of their feature type and the names and values of feature properties. A client application asks a transactional WFS to alter the state of a feature through insert, update, and delete operation requests.

7.3 Property names

A web feature service refers to feature property names defined in a GML application schema. However, since the state of a feature must be expressed in GML and thus XML, the property names used by a web feature service must also be valid element and attribute names as described in the Extensible Markup Language (XML) 1.0 [5] specification. In addition, property names may be namespace qualified as described in Namespaces in XML [7] . The following definitions are taken from sections 2 & 3 of that document:

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

The definitions of the components Letter, Digit, CombiningChar and Extender are defined in annex B of [5].

Example

Examples of valid property names are:

Age, Temperature, _KHz, myns:INWATERA_1M.WKB_GEOM

Examples of invalid property names are:

+Domain, 123_SomeName

7.4 Property references

7.4.1 Introduction

As mentioned in the introduction, GML allows geographic features to have complex or aggregate non-geometric properties. A problem thus arises about how such properties should be referenced in the various places where property references are required (e.g. query and filter expressions). A WFS **must** use XPath [8] expressions, as defined in this document, for referencing the properties and sub-properties of a feature encoded as XML elements or attributes.

7.4.2 XPath expressions

The XML Path Language [8] specification is a language for addressing parts of a XML document, or in the case of this specification, for referencing feature properties referred to by means of XML elements or attributes.

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

1. A WFS implementation **must** support *abbreviated relative location* paths.

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

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

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

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

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

Example

To practically illustrate the use of XPath expressions for referencing the properties of a complex feature (encoded as elements or attributes), consider the fictitious complex feature *Person* defined by the following XML Schema document:

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

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

   <element name="Person" type="myns:PersonType"
            substitutionGroup="gml:_Feature"/>
   <complexType name="PersonType">
      <complexContent>
         <extension base="gml:AbstractFeatureType">
            <sequence>
               <element name="LastName" nillable="true">
                  <simpleType>
                     <restriction base="string">
                        <maxLength value="30"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="FirstName" nillable="true">
                  <simpleType>
                     <restriction base="string">
                        <maxLength value="10"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="Age" type="integer" nillable="true"/>
               <element name="Sex" type="string"/>
               <element name="Spouse">
                  <complexType>
                     <attribute name="sin" type="xsd:anyURI" use="required" />
                  </complexType>
               </element>
               <element name="Location"
                        type="gml:PointPropertyType"
                        nillable="true"/>
               <element name="Address" type="myns:AddressType" nillable="true"/>
               <element name="Phone" type="xsd:string"
                        minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
            <attribute name="sin" type="xsd:anyURI" use="required"/>
         </extension>
      </complexContent>
   </complexType>

   <complexType name="AddressType">
      <sequence>
         <element name="StreetName" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
               </restriction>
            </simpleType>
         </element>
         <element name="StreetNumber" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="10"/>
               </restriction>
            </simpleType>
         </element>
         <element name="City" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
               </restriction>
            </simpleType>
         </element>
         <element name="Province" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
```

```
            </restriction>
         </simpleType>
      </element>
      <element name="PostalCode" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="15"/>
            </restriction>
         </simpleType>
      </element>
      <element name="Country" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="30"/>
            </restriction>
         </simpleType>
      </element>
   </sequence>
</complexType>
</schema>
```

Note that the property *Address* is a complex property of type *AddressType*. An example instance of the feature *Person* might be:

```
<?xml version="1.0" ?>
<myns:Person
   sin="111222333"
   xmlns:myns="http://www.opengis.net/myns"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/myns Person.xsd">

   <myns:LastName>Smith</myns:LastName>
   <myns:FirstName>Fred</myns:FirstName>
   <myns:Age>35</myns:Age>
   <myns:Sex>Male</myns:Sex>
   <myns:Spouse sin="444555666" />
   <myns:Location>
      <gml:Point><gml:coordinates>15,15</gml:coordinates></gml:Point>
   </myns:Location>
   <myns:Address>
      <myns:StreetName>Main St.</myns:StreetName>
      <myns:StreetNumber>5</myns:StreetNumber>
      <myns:City>SomeCity</myns:City>
      <myns:Province>SomeProvince</myns:Province>
      <myns:PostalCode>X1X 1X1</myns:PostalCode>
      <myns:Country>Canada</myns:Country>
   </myns:Address>
   <myns:Phone>416-123-4567</myns:Phone>
   <myns:Phone>416-890-1234</myns:Phone>
</myns:Person>
```

Using XPath [8] expressions, each property of a *Person* feature can be referenced as follows (omitting the namespace qualifiers for clarities sake):

```
LastName
FirstName
Age
Sex
Source
Location
Address
Address/StreetNumber
Address/StreetName
Address/City
Address/Province
Address/Postal_Code
Address/Country
Phone[1]
Phone[2]
```

Notice that in this instance, each relative location path begins with the root element of the feature property being referenced. This simply corresponds to the name of the feature property. Optionally, each feature property may be referenced with the relative location

path beginning with root element of the feature (i.e. the name of the feature type). Thus the *LastName* property could be reference as *Person/LastName*, the *City* property could be referenced as *Person/Address/City* and so on.

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

The element *Phone* appears multiple times and the predicates *[1]* and *[2]* are used to indicate the specific elements. The predicate *[1]* is used to indicate the first occurrence of the *Phone* element. The predicate *[2]* is used to indicate the second occurrence of the *Phone* element.

In addition, the **sin**[1] attribute on the **<Person>** and **<Spouse>** elements can be referenced using the following XPath [8] expressions:

```
Person/@sin
Person/Spouse/@sin
```

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

7.5 <Native> element

It is clear that an open interface can only support a certain common set of capabilities. The <Native> element is intended to allow access to vendor specific capabilities of any particular web feature server or datastore.

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

```
<xsd:element name="Native" type="wfs:NativeType"/>
<xsd:complexType name="NativeType">
   <xsd:any />
   <xsd:attribute name="vendorId" type="xsd:string" use="required"/>
   <xsd:attribute name="safeToIgnore" type="xsd:boolean" use="required"/>
</xsd:complexType>
```

The <Native> element simply contains the vendor specific command or operation.

The **vendorId** attribute is used to identify the vendor that recognizes the command or operation enclosed by the <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.

The **safeToIgnore** attribute is used to guide the actions of a web feature service when the <Native> command or operation 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 <Native> element cannot be ignored and the operation that the element is associated with must fail if the web feature service cannot deal with it.

---

[1] SIN = Social Insurance Number

safeToIgnore=True

A value of *True* indicates that the <Native> element can be safely ignored.

Example

This example illustrates the use of the <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 can be safely ignored.

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

7.6 Filter

A filter is used to define a set of feature instances that are to be operated upon. The operating set can be comprised of one or more enumerated features or a set of features defined by specifying spatial and non-spatial constraints on the geometric and scalar properties of a feature type. Filter specifications **shall** be encoded as described in the OGC Filter Encoding Implementation Specification [3].

7.7 Exception reporting

In the event that a web feature service encounters an error while processing a request or receives an unrecognized request, it **shall** generate an XML document indicating that an error has occurred. The format of the XML error response is specified by, and **must** validate against, the exception response schema defined in section A.2.

A **<ServiceExceptionReport>** element can contain one or more WFS processing exceptions. The mandatory **version** attribute is used to indicate the version of the service exception report schema. For this version of the specification, this value is fixed at 1.2.0.

Individual exception messages are contained within the **<ServiceException>** element. The optional **code** attribute may 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. A number of elements defined in this document include a **handle** attribute that can be used to associate a mnemonic name with the element. If such a **handle** exists, its value may be reported using the **locator** attribute of the **<ServiceException>** element. If the **handle** attribute is not specified, then a web feature server implementation may attempt to locate the error using other means such as line numbers, etc...

Example

The following is an example of an exception report. This exception indicates that the first insert statement failed because of a missing closing XML tag in the request.

```
<?xml version="1.0" ?>
<ServiceExceptionReport
   version="1.2.0"
   xmlns="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/ogc ../wfs/1.0.0/OGC-exception.xsd">
   <ServiceException code="999" locator="INSERT STMT 01">
      parse error: missing closing tag for element WKB_GEOM
```

```
    </ServiceException>
  </ServiceExceptionReport>
```

It should be noted that this sample output validates against the exception report schema presented above.

7.8 Common XML attributes

7.8.1 Version attribute

All XML encoded WFS requests include an attribute called **version**. The mandatory **version** attribute is used to indicate to which version of the WFS specification the request encoding conforms and is used in version negotiation as described in section 6.2.4. The default value of the **version** attributed is set to 1.0.0, which corresponds to the version of this document.

7.8.2 Service attribute

All XML encoded WFS requests include an attribute called **service**. The mandatory **service** attribute is used to indicate which of the available service types, at a particular service instance, is being invoked. When invoking a web feature service, the value of the **service** attribute shall be WFS.

7.8.3 Handle attribute

The purpose of the **handle** attribute is to allow a client application to associate a mnemonic name with a request for error handling purposes. If a **handle** is specified, and an exception is encountered, a Web Feature Service may use the **handle** to identify the offending element.

8 DescribeFeatureType operation

8.1 Introduction

The function of the **DescribeFeatureType** operation is to generate a schema description of feature types serviced by a WFS implementation. The schema descriptions define how a WFS implementation expects feature instances to be encoded on input and how feature instances will be generated on output.

8.2 Request

A **DescribeFeatureType** element contains zero or more **TypeName** elements that encode the names of feature types that are to be described. If the content of the **DescribeFeatureType** element is empty, then that shall be interpreted as requesting a description of all feature types that a WFS can service. The XML encoding of a **DescribeFeatureType** request is defined by the following XML Schema fragment:

```
<xsd:element name="DescribeFeatureType" type="wfs:DescribeFeatureTypeType"/>
<xsd:complexType name="DescribeFeatureTypeType">
   <xsd:sequence>
      <xsd:element name="TypeName" type="xsd:QName"
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
```

```
    <xsd:attribute name="version"
                   type="xsd:string" use="required" fixed="1.0.0"/>
    <xsd:attribute name="service"
                   type="xsd:string" use="required" fixed="WFS"/>
    <xsd:attribute name="outputFormat"
                   type="xsd:string" use="optional" default="XMLSCHEMA"/>
</xsd:complexType>
```

The **outputFormat** attribute, is used to indicate the schema description language that should be used to describe feature type schemas. The only mandatory output format in response to a **DescribeFeatureType** operation is XML Schema, denoted by the value **XMLSCHEMA** for the **outputFormat** attribute. Other vendor specific formats are also possible, but they **must** be advertised on the capabilities document [sec. 12].

As specified by GML [2], the feature schema definition is entirely at the discretion of the particular WFS implementation that is describing its feature types. The only caveats are:

1. Feature geometry must be expressed using the GML geometry description. (geometry.xsd).
2. Spatial Reference Systems must be expressed as defined in the OpenGIS® Geography Markup Language (GML) Implementation Specification, version 2.1.1 [2].
3. The feature schema must be consistent with the OGC feature model. This means that the feature schema defines properties of the feature. The GML interpretation of this statement is that the elements nested below the root element of a feature type define properties of that feature.

8.3 Response

In response to a **DescribeFeatureType** request, where the value of the **outputFormat** attribute has been set to **XMLSCHEMA**, a WFS implementation must be able to present an XML Schema [6] document that is a valid GML [2] application schema and defines the schema of the feature types listed in the request. The document(s) presented 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.

Schema descriptions using other schema description languages, such as DTD, are also possible as long as such capabilities are declared in the capabilities document [sec. 12].

8.3.1 Supporting multiple namespaces

An XML Schema[6] document can only describe elements that belong to a single namespace. This means that a Web Feature Service cannot describe features from multiple namespaces in a single XML Schema document. To overcome this limitation, a WFS may generate an XML Schema document that is a "wrapper" schema that imports the schemas of the features from the various namespaces in the request. For example, consider the following request:

```
<?xml version="1.0" ?>
<DescribeFeatureType
   version="1.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   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 ../wfs/1.0.0/WFS-basic.xsd">
   <TypeName>ns01:TREESA_1M</TypeName>
   <TypeName>ns02:ROADL_1M</TypeName>
</DescribeFeatureType>
```

A WFS may generate the following response to this request:

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

   <import namespace="http://www.server01.com/ns01"
           schemaLocation="http://www.myserver.com/wfs.cgi?
                           request=DescribeFeatureType&amp;typeName=ns01:TREESA_1M"/>

   <import namespace="http://www.server02.com/ns02"
           schemaLocation="http://www.yourserver.com/wfs.cgi?
                           request=DescribeFeatureType&amp;typeName=ns02:ROADL_1M"/>

</schema>
```

In this example, the WFS is using a **DescribeFeatureType** request to obtain the schemas of the features in the various namespaces. This is simply an example, other methods of obtaining the schemas may be implemented (for example: referencing static schema documents).

8.4 Exceptions

In the event that a web feature service encounters an error servicing a **DescribeFeatureType** request, it shall raise an exception as described Section 7.7.

8.5 Examples

Example 1

Consider geographic features of types *TREESA_1*M and *ROADL_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
------------------------  -------- -------------
WKB_GEOM                  NOT NULL LONG RAW
ID                                 NUMBER(10)
TREE_TYPE                          VARCHAR2(80)

SQL> describe ROADL_1M
Name                       Null?    Type
-------------------------- -------- ------------
WKB_GEOM                   NOT NULL LONG RAW
DESIGNATION                         VARCHAR2(30)
SURFACE_TYPE                        VARCHAR2(30)
NLANES                              NUMBER(2)
```

In response to the **DescribeFeatureType** request:

```
<?xml version="1.0" ?>
<DescribeFeatureType
   version="1.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   xmlns:myns="http://www.myserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <TypeName>myns:TREESA_1M</TypeName>
   <TypeName>myns:ROADL_1M</TypeName>
</DescribeFeatureType>
```

a web feature service may 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"
   elementFormDefault="qualified" version="0.1">

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

   <!-- ============================================
        define global elements
        ============================================ -->
   <element name="TREESA_1M"
            type="myns:TREESA_1M_Type"
            substitutionGroup="gml:_Feature"/>

   <element name="ROADL_1M"
            type="myns:ROADL_1M_Type"
            substitutionGroup="gml:_Feature"/>

   <!-- ============================================
        define complex types (classes)
        ============================================ -->
   <complexType name="TREESA_1M_Type">
      <complexContent>
         <extension base="gml:AbstractFeatureType">
            <sequence>
               <element name="WKB_GEOM"
                        type="gml:PolygonPropertyType" nillable="false"/>
               <element name="ID" nillable="true" minOccurs="0">
                  <simpleType>
                     <restriction base="integer">
                        <totalDigits value="10"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="TREE_TYPE" 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="WKB_GEOM"
                        type="gml:LineStringPropertyType"
                        nillable="false"/>
               <element name="DESIGNATION" nillable="true" minOccurs="0">
                  <simpleType>
                     <restriction base="string">

                        <maxLength value="30"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="SURFACE_TYPE" 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>
</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
   xmlns="http://www.someserver.com/myns"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd
                       http://www.someserver.com/myns ex07.xsd">
   <gml:boundedBy>
      <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
         <gml:coordinates>-180.0,-90.0 180.0,90.0</gml:coordinates>
      </gml:Box>
   </gml:boundedBy>
   <gml:featureMember>
      <TREESA_1M>
         <WKB_GEOM>
            <gml:Polygon>
               <gml:outerBoundaryIs>
                  <gml:LinearRing>
                     <gml:coordinates decimal="." cs="," ts=" ">-120.000000,65.588264
-120.003571,65.590782 -120.011292,65.590965 -120.022491,65.595215 -
120.031212,65.592880 -120.019363,65.586121 -120.030350,65.585365 -
120.045082,65.581848 -120.059540,65.584938 -120.067284,65.590500 -
120.067284,65.595436 -120.067337,65.613441 -120.067337,65.613777 -
120.060997,65.606346 -120.045517,65.605545 -120.022675,65.599777 -
120.003975,65.601036 -120.000000,65.602081 -120.000000,65.602081 -
120.000000,65.588264</gml:coordinates>
                  </gml:LinearRing>
               </gml:outerBoundaryIs>
            </gml:Polygon>
         </WKB_GEOM>
         <ID>0000000002</ID>
         <TREE_TYPE>Maple</TREE_TYPE>
      </TREESA_1M>
   </gml:featureMember>
   <gml:featureMember>
      <ROADL_1M>
         <WKB_GEOM>
            <gml:LineString srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:coordinates decimal="." cs="," ts=" ">-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:coordinates>
            </gml:LineString>
         </WKB_GEOM>
         <DESIGNATION>HYW 401</DESIGNATION>
         <SURFACE_TYPE>ASPHALT</SURFACE_TYPE>
         <NLANES>12</NLANES>
      </ROADL_1M>
   </gml:featureMember>
</wfs:FeatureCollection>
```

Example 2

This example describes a collection type, *People*, composed of feature instances of the feature type *Person*, that includes a complex property *Address*.

In response to the **DescribeFeatureType** request:

```
<?xml version="1.0" ?>
<DescribeFeatureType
   version="1.0.0"
   service="WFS"
   outputFormat="XMLSCHEMA"
   xmlns="http://www.opengis.net/wfs"
   xmlns:myns="http://www.myserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <TypeName>myns:People</TypeName>
</DescribeFeatureType>
```

a web feature service might generate an XML Schema document that looks like:

```
<?xml version="1.0" ?>
<xsd:schema
   targetNamespace="http://www.someserver.com/myns"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified" version="0.1">

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

   <xsd:element name="Person"
                type="myns:PersonType"
                substitutionGroup="gml:_Feature"/>

   <xsd:complexType name="PersonType">
      <xsd:complexContent>
         <xsd:extension base="gml:AbstractFeatureType">
            <xsd:sequence>
               <xsd:element name="LastName" nillable="true">
                  <xsd:simpleType>
                     <xsd:restriction base="string">
                        <xsd:maxLength value="30"/>
                     </xsd:restriction>
                  </xsd:simpleType>
               </xsd:element>
               <xsd:element name="FirstName" nillable="true">
                  <xsd:simpleType>
                     <xsd:restriction base="string">
                        <xsd:maxLength value="10"/>
                     </xsd:restriction>
                  </xsd:simpleType>
               </xsd:element>
               <xsd:element name="Age"
                            type="integer"
                            nillable="true"/>
               <xsd:element name="Sex"
                            type="string"/>
               <xsd:element name="Location"
                            type="gml:PointPropertyType"
                            nillable="true"/>
               <xsd:element name="Address"
                            type="myns:AddressType"
                            nillable="true"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="AddressType">
      <xsd:sequence>
         <xsd:element name="StreetName" nillable="true">
            <xsd:simpleType>
               <xsd:restriction base="string">
                  <xsd:maxLength value="30"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="StreetNumber" nillable="true">
```

```
            <xsd:simpleType>
               <xsd:restriction base="string">
                  <xsd:maxLength value="10"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="City" nillable="true">
            <xsd:simpleType>
               <xsd:restriction base="string">
                  <xsd:maxLength value="30"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="Province" nillable="true">
            <xsd:simpleType>
               <xsd:restriction base="string">
                  <xsd:maxLength value="30"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="PostalCode" nillable="true">
            <xsd:simpleType>
               <xsd:restriction base="string">
                  <xsd:maxLength value="15"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="Country" nillable="true">
            <xsd:simpleType>
               <xsd:restriction base="string">
                  <xsd:maxLength value="30"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
      </xsd:sequence>
   </xsd:complexType>
</xsd:schema>
```

A sample instance document that validates against this schema might be:

```
<?xml version="1.0" ?>
<wfs:FeatureCollection
xmlns="http://www.someserver.com/myns"
xmlns:myns="http://www.someserver.com/myns"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd
                    http://www.someserver.com/myns ex10.xsd">
   <gml:boundedBy>
      <gml:Box>
         <gml:coord>
            <gml:X>10</gml:X>
            <gml:Y>10</gml:Y>
         </gml:coord>
         <gml:coord>
            <gml:X>20</gml:X>
            <gml:Y>20</gml:Y>
         </gml:coord>
      </gml:Box>
   </gml:boundedBy>
   <gml:featureMember>
      <Person>
         <myns:LastName>Smith</myns:LastName>
         <myns:FirstName>Fred</myns:FirstName>
         <myns:Age>35</myns:Age>
         <myns:Sex>Male</myns:Sex>
         <myns:Location>
            <gml:Point><gml:coordinates>15,15</gml:coordinates></gml:Point>
         </myns:Location>
         <myns:Address>
            <myns:StreetName>Main St.</myns:StreetName>
            <myns:StreetNumber>5</myns:StreetNumber>
            <myns:City>SomeCity</myns:City>
            <myns:Province>SomeProvince</myns:Province>
            <myns:PostalCode>X1X 1X1</myns:PostalCode>
            <myns:Country>Canada</myns:Country>
         </myns:Address>
```

```
        </Person>
      </gml:featureMember>
    </wfs:FeatureCollection>
```

9 GetFeature operation

9.1 Introduction

The **GetFeature** operation allows retrieval of features from a web feature service. A **GetFeature** request is processed by a WFS and an XML document, containing the result set, is returned to the client.

9.2 Request

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:sequence>
      <xsd:element ref="wfs:Query" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="version"
                  type="xsd:string" use="required" fixed="1.0.0"/>
   <xsd:attribute name="service"
                  type="xsd:string" use="required" fixed="WFS"/>
   <xsd:attribute name="handle"
                  type="xsd:string" use="optional"/>
   <xsd:attribute name="outputFormat"
                  type="xsd:string" use="optional" default="GML2"/>
   </xsd:attribute>
   <xsd:attribute name="maxFeatures" type="xsd:positiveInteger"
                  use="optional"/>
</xsd:complexType>
<xsd:element name="Query" type="wfs:QueryType"/>
<xsd:complexType name="QueryType">
  <xsd:sequence>
    <xsd:element ref="ogc:PropertyName" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="handle"
                 type="xsd:string" use="optional"/>
  <xsd:attribute name="typeName"
                 type="xsd:QName" use="required"/>
  <xsd:attribute name="featureVersion"
                 type="xsd:string" use="optional"/>
</xsd:complexType>
```

The **<GetFeature>** element contains one or more <**Query**> elements, each of which in turn contain the description of a query. The results of all queries contained in a **GetFeature** request are concatenated to produce the result set.

The **outputFormat** attribute defines the format to use to generate the result set. The default value is **GML2** indicating that GML [2] shall be used. Vendor specific formats (including non-XML and binary formats), declared in the capabilities document are also possible.

The optional **maxFeatures** attribute can be used to limit the number of features that a **GetFeature** request retrieves. Once the **maxFeatures** limit is reached, the result set is truncated at that point.

Each individual query packaged in a **GetFeature** request is defined using the **<Query>** element. The **<Query>** element defines which feature type to query, what properties to retrieve and what constraints (spatial and non-spatial) to apply to those properties.

The **typeName** attribute is used to indicate the name of the feature type or class to be queried.

The **featureVersion** attribute is included in order to accommodate systems that support feature versioning. A value of **ALL** indicates that all versions of a feature should be fetched. Otherwise, an integer, *n*, can be specified to return the n^{th} version of a feature. The version numbers start at *1*, which is the oldest version. If a version value larger than the largest version number is specified, then the latest version is returned. The default action shall be for the query to return the latest version. Systems that do not support versioning can ignore the parameter and return the only version that they have.

The **<PropertyName>** element is used to enumerate the feature properties that should be selected during a query and whose values should be included in the response to a **GetFeature** request. A client application can determine the properties of a feature by making a **DescribeFeatureType** request before composing a **GetFeature** request. The **DescribeFeatureType** operation [sec. 8] will generate a GML application schema defining the schema of the feature type. The client can then select the properties to be fetched. In addition, the client can determine which feature properties are mandatory and must be fetched in order for the WFS to be able to generate an instance of the feature type that will validate againt the generated GML application schema. In the event that a WFS encounters a query that does not select all mandatory properties of a feature, the WFS will internally augment the property name list to include all necessary property names. A WFS client must thus be prepared to deal with a situation where it receives more property values than it requests.

If no **<PropertyName>** elements are specified, then all feature properties should be fetched.

The **<Filter>** element can be used to define constraints on a query. Both spatial and/or non-spatial constraints can be specified as described in the Filter Encoding Specification [3]. If no **<Filter>** element is contained within the <**Query>** element, then the query is unconstrained and all feature instances should be retrieved.

The **<GetFeatureWithLock>** element is functionally similar to the **<GetFeature>** element, except that it indicates to a web feature service to attempt to lock the features that are selected; presumably to update the features.

9.3 Response

The format of the response to a **GetFeature** request is controlled by the **outputFormat** attribute. The default value for the **outputFormat** attribute shall be **GML2.** This will indicate that a WFS must generate a GML document of the result set that conforms to the OpenGIS® Geography Markup Language Implementation Specification, version 2.1.1 [2], and more specifically, the output must validate againt the GML application schema generated by the **DescribeFeatureType** operation [sec. 8].

Any GML document generated by a WFS implementation, in response to a query where the **outputFormat** is **GML2**, must reference an appropriate GML application schema document so that the output can be validated. This can be accomplished using the **schemaLocation** attribute, as defined in [6]. This attribute provides hints as to the

physical location of one or more schema documents which may be used for local validation and schema-validity assessment. The **schemaLocation** attribute value contains pairs of values. The first member of each pair is the namespace for which the second member is the hint describing where to find to an appropriate schema document. The physical location of the schema documents is specified using a URI [10].

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

```
<?xml version="1.0" ?>
<wfs:FeatureCollection
   xmlns="http://www.opengis.net/myns"
   xmlns:myns="http://www.opengis.net/myns"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
   xsi:schemaLocation="http://www.opengis.net/myns
                       http://www.someserver.com/wfs.cgi?
                       request=DescribeFeatureType&typename=TREESA_1M,ROADL_1M"> …
```

In this instance, the schema document corresponding to the *myns* namespace is dynamically generated by making a **DescribeFeatureType** request back to the server that generated the output, requesting the schema. This **DescribeFeatureType** operation [sec. 8] requests the schema of the feature types TREESA_1M and ROADL_1M, both in the *myns* namespace.

It is up to each WFS implementation to arrange that the GML output makes the appropriate **schemaLocation** reference(s) such that the output can be validated.

For the **<GetFeatureWithLock>** request, a WFS must generate a result that includes the lock identifier. The lock identifier is encoded using the **lockId** attribute that is defined on the <**wfs:FeatureCollection**> element. The following XML fragment illustrates how to include the **lockId** attribute in the response to the 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.

9.4 Exceptions

In the event that a web feature service encounters an error servicing a **GetFeature** request, it shall raise an exception as described in Section 7.7.

9.5 Examples

This section contains numerous examples of the **GetFeature** request. Some examples include sample output.

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="1.0.0"
   outputFormat="GML2"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <wfs:Query typeName="myns:INWATERA_1M">
      <ogc:Filter>
         <ogc:FeatureId fid="INWATERA_1M.1234"/>
      </ogc:Filter>
   </wfs:Query>
</wfs:GetFeature>
```

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="1.0.0"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <wfs:Query typeName="myns:INWATERA_1M">
      <ogc:PropertyName>myns:WKB_GEOM</ogc:PropertyName>
      <ogc:PropertyName>myns:TILE_ID</ogc:PropertyName>
      <ogc:PropertyName>myns:FAC_ID</ogc:PropertyName>
      <ogc:Filter>
         <ogc:FeatureId fid="INWATERA_1M.1013"/>
      </ogc:Filter>
   </wfs:Query>
</wfs:GetFeature>
```

Example 3

In this example, all the properties of feature type *INWATERA_1M* are fetched for an enumerated list of feature instances. The **<FeatureId>** element is used to identify each feature to be fetched.

```
<?xml version="1.0" ?>
<GetFeature
   version="1.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <Query typeName="myns:INWATERA_1M">
     <ogc:Filter>
        <ogc:FeatureId fid="INWATERA_1M.1013"/>
        <ogc:FeatureId fid="INWATERA_1M.1014"/>
        <ogc:FeatureId fid="INWATERA_1M.1015"/>
      </ogc:Filter>
   </Query>
</GetFeature>
```

Example 4

This example is similar to the previous example except in this case only some of the properties of an enumerated set of features are fetched. The **<PropertyName>** element is used to list the properties to be retrieved.

```
<?xml version="1.0" ?>
<GetFeature
   version="1.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <Query typeName="myns:INWATERA_1M">
      <ogc:PropertyName>myns:WKB_GEOM</ogc:PropertyName>
      <ogc:PropertyName>myns:TILE_ID</ogc:PropertyName>
      <ogc:Filter>
         <ogc:FeatureId fid="INWATERA_1M.1013"/>
         <ogc:FeatureId fid="INWATERA_1M.1014"/>
         <ogc:FeatureId fid="INWATERA_1M.1015"/>
      </ogc:Filter>
   </Query>
</GetFeature>
```

Example 5

Select all instances of the feature type *INWATERA_1M* to a maximum of 10000 features.

```
<?xml version="1.0" ?>
<GetFeature
   version="1.0.0"
   service="WFS"
   maxFeatures="10000"
   xmlns="http://www.opengis.net/wfs"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <Query typeName="myns:INWATERA_1M"/>
</GetFeature>
```

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="1.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:yourns="http://demo.cubewerx.com/yourns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <Query typeName="myns:INWATERA_1M"/>
   <Query typeName="myns:BUILTUPA_1M"/>
   <Query typeName="yourns:ROADL_1M"/>
</GetFeature>
```

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: [-57.9118,46.2023,-46.6873,51.8145].

```
<?xml version="1.0" ?>
<GetFeature
   version="1.0.0"
   service="WFS"
   handle="Query01"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   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 ../wfs/1.0.0/WFS-basic.xsd">
   <Query typeName="myns:HYDROGRAPHY">
```

```
      <ogc:PropertyName>myns:GEOTEMP</ogc:PropertyName>
      <ogc:PropertyName>myns:DEPTH</ogc:PropertyName>
      <ogc:Filter>
         <ogc:Not>
            <ogc:Disjoint>
               <ogc:PropertyName>myns:GEOTEMP</ogc:PropertyName>
               <gml:Box>
               <gml:coordinates>-57.9118,46.2023 -46.6873,51.8145</gml:coordinates>
               </gml:Box>
            </ogc:Disjoint>
         </ogc:Not>
      </ogc:Filter>
   </Query>
</GetFeature>
```

The output from such a request might be:

```
<?xml version="1.0" ?>
<wfs:FeatureCollection
   xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.someserver.com/myns HYDROGRAPHY.xsd
                       http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <gml:boundedBy>
      <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
         <gml:coordinates>10,10 20,20</gml:coordinates>
      </gml:Box>
   </gml:boundedBy>
   <gml:featureMember>
      <HYDROGRAPHY fid="HYDROGRAPHY.450">
         <GEOTEMP>
            <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:coordinates>10,10</gml:coordinates>
            </gml:Point>
         </GEOTEMP>
         <DEPTH>565</DEPTH>
      </HYDROGRAPHY>
   </gml:featureMember>
   <gml:featureMember>
      <HYDROGRAPHY fid="HYDROGRAPHY.450">
         <GEOTEMP>
            <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:coordinates>10,11</gml:coordinates>
            </gml:Point>
         </GEOTEMP>
         <DEPTH>566</DEPTH>
      </HYDROGRAPHY>
   </gml:featureMember>
   <!—
   .
   . … more HYDROGRAPHY instances …
   .
   -->
</wfs:FeatureCollection>
```

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="1.0.0"
   service="WFS"
   handle="Example Query"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   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 ../wfs/1.0.0/WFS-basic.xsd">
   <Query typeName="myns:ROADS">
      <ogc:PropertyName>myns:PATH</ogc:PropertyName>
      <ogc:PropertyName>myns:LANES</ogc:PropertyName>
```

```
      <ogc:PropertyName>myns:SURFACETYPE</ogc:PropertyName>
      <ogc:Filter>
         <ogc:Within>
            <ogc:PropertyName>myns:PATH</ogc:PropertyName>
            <gml:Box>
               <gml:coordinates>50,40 100,60</gml:coordinates>
            </gml:Box>
         </ogc:Within>
      </ogc:Filter>
   </Query>
   <Query typeName="myns:RAILS">
      <ogc:PropertyName>myns:TRACK</ogc:PropertyName>
      <ogc:PropertyName>myns:GAUGE</ogc:PropertyName>
      <ogc:Filter>
         <ogc:Within>
            <ogc:PropertyName>myns:TRACK</ogc:PropertyName>
            <gml:Box>
               <gml:coordinates>50,40 100,60</gml:coordinates>
            </gml:Box>
         </ogc:Within>
      </ogc:Filter>
   </Query>
</GetFeature>
```

The results of each query are concatenated to form the output feature collection.

```
<?xml version="1.0" ?>
<wfs:FeatureCollection
   xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd
                       http://www.someserver.com/myns ROADSRAILS.xsd">
   <gml:boundedBy>
      <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
         <gml:coordinates>0,0 180,360</gml:coordinates>
      </gml:Box>
   </gml:boundedBy>
   <gml:featureMember>
      <ROADS fid="ROADS.100">
         <PATH>
            <gml:LineString gid="1"
                 srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:coordinates>10,10 10,11 10,12 10,13</gml:coordinates>
            </gml:LineString>
         </PATH>
         <SURFACE_TYPE>ASPHALT</SURFACE_TYPE>
         <NLANES>4</NLANES>
      </ROADS>
   </gml:featureMember>
   <gml:featureMember>
      <ROADS fid="ROADS.105">
         <PATH>
            <gml:LineString gid="2"
                 srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:coordinates>10,10 10,11 10,12</gml:coordinates>
            </gml:LineString>
         </PATH>
         <SURFACE_TYPE>GRAVEL</SURFACE_TYPE>
         <NLANES>2</NLANES>
      </ROADS>
   </gml:featureMember>
   <!--
   ... more ROADS features ....
   -->
   <gml:featureMember>
      <RAILS fid="RAILS.119">
         <TRACK>
            <gml:LineString gid="n"
                 srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:coordinates>15,10 16,11 17,12</gml:coordinates>
            </gml:LineString>
         </TRACK>
         <GAUGE>24</GAUGE>
      </RAILS>
   </gml:featureMember>
   <!--
```

```
   ... more RAILS features ....
   -->
</wfs:FeatureCollection>
```

Example 9

This example illustrates how complex properties of features can be referenced using XPath expressions. Consider the feature type Person defined as:

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

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

   <element name="Person" type="myns:PersonType"
            substitutionGroup="gml:_Feature"/>
   <complexType name="PersonType">
      <complexContent>
         <extension base="gml:AbstractFeatureType">
            <sequence>
               <element name="LastName" nillable="true">
                  <simpleType>
                     <restriction base="string">
                        <maxLength value="30"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="FirstName" nillable="true">
                  <simpleType>
                     <restriction base="string">
                        <maxLength value="10"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="Age" type="integer" nillable="true"/>
               <element name="Sex" type="string"/>
               <element name="Spouse">
                  <complexType>
                     <attribute name="sin" type="xsd:anyURI" use="required" />
                  </complexType>
               </element>
               <element name="Location"
                        type="gml:PointPropertyType"
                        nillable="true"/>
               <element name="Address" type="myns:AddressType" nillable="true"/>
            </sequence>
            <attribute name="sin" type="xsd:anyURI" use="required"/>
         </extension>
      </complexContent>
   </complexType>

   <complexType name="AddressType">
      <sequence>
         <element name="StreetName" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
               </restriction>
            </simpleType>
         </element>
         <element name="StreetNumber" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="10"/>
               </restriction>
            </simpleType>
         </element>
         <element name="City" nillable="true">
            <simpleType>
```

```
                <restriction base="string">
                   <maxLength value="30"/>
                </restriction>
             </simpleType>
          </element>
          <element name="Province" nillable="true">
             <simpleType>
                <restriction base="string">
                   <maxLength value="30"/>
                </restriction>
             </simpleType>
          </element>
          <element name="PostalCode" nillable="true">
             <simpleType>
                <restriction base="string">
                   <maxLength value="15"/>
                </restriction>
             </simpleType>
          </element>
          <element name="Country" nillable="true">
             <simpleType>
                <restriction base="string">
                   <maxLength value="30"/>
                </restriction>
             </simpleType>
          </element>
       </sequence>
    </complexType>
</schema>
```

The Address property is a complex property.

The following example fetches the last name of all the people who live on the 10000 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="1.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   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 ../wfs/1.0.0/WFS-basic.xsd
                       http://www.someserver.com/myns Person.xsd">
   <Query typeName="Person">
      <ogc:PropertyName>myns:Person/myns:LastName</ogc:PropertyName>
      <ogc:Filter>
         <ogc:And>
            <ogc:And>
               <ogc:PropertyIsGreaterThanOrEqualTo>
<ogc:PropertyName>myns:Person/myns:Address/myns:StreetNumber</ogc:PropertyName>
                  <ogc:Literal>10000</ogc:Literal>
               </ogc:PropertyIsGreaterThanOrEqualTo>
               <ogc:PropertyIsLessThanOrEqualTo>
<ogc:PropertyName>myns:Person/myns:Address/myns:StreetNumber</ogc:PropertyName>
                  <ogc:Literal>10999</ogc:Literal>
               </ogc:PropertyIsLessThanOrEqualTo>
            </ogc:And>
            <ogc:And>
               <ogc:PropertyIsEqualTo>
<ogc:PropertyName>myns:Person/myns:Address/myns:StreetName</ogc:PropertyName>
                  <ogc:Literal>Main St.</ogc:Literal>
               </ogc:PropertyIsEqualTo>
               <ogc:PropertyIsEqualTo>
<ogc:PropertyName>myns:Person/myns:Address/myns:City</ogc:PropertyName>
                  <ogc:Literal>SomeTown</ogc:Literal>
               </ogc:PropertyIsEqualTo>
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>myns:Person/myns:Sex</ogc:PropertyName>
                  <ogc:Literal>Female</ogc:Literal>
               </ogc:PropertyIsEqualTo>
```

```
                <ogc:PropertyIsGreaterThan>
                   <ogc:PropertyName>myns:Person/myns:Salary</ogc:PropertyName>
                   <ogc:Literal>35000</ogc:Literal>
                </ogc:PropertyIsGreaterThan>
             </ogc:And>
          </ogc:And>
       </ogc:Filter>
    </Query>
 </GetFeature>
```

10 LockFeature operation

10.1 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 database 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 that 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 can modify the same data item. This can 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 long term because network latency would make feature locks last relatively longer than native commercial database locks.

The **LockFeature** operation is optional and does not need to be implemented for a WFS implementation to conform to this specification. If a WFS implements the **LockFeature** operation, this fact must be advertised in the capabilities document [sec. 12].

10.2 Request

10.2.1 Schema definition

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

```
<xsd:element name="LockFeature" type="wfs:LockFeatureType"/>
<xsd:complexType name="LockFeatureType">
   <xsd:sequence>
      <xsd:element name="Lock" type="wfs:LockType" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="version"
                  type="xsd:string" use="required" fixed="1.0.0"/>
   <xsd:attribute name="service"
                  type="xsd:string" use="required" fixed="WFS"/>
   <xsd:attribute name="expiry"
                  type="xsd:positiveInteger" use="optional"/>
   <xsd:attribute name="lockAction"
                  type="wfs:AllSomeType" use="optional"/>
</xsd:complexType>
```

```
<xsd:complexType name="LockType">
   <xsd:sequence>
      <xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
   <xsd:attribute name="handle"
                  type="xsd:string" use="optional"/>
   <xsd:attribute name="typeName"
                  type="xsd:QName" use="required"/>
</xsd:complexType>
```

The <**LockFeature**> element contains one or more <**Lock**> elements that define a locking operation on feature instances of one feature type.

The **expiry** attribute is used to set a limit on how long a web feature service should hold a lock on feature instances in the event that a transaction is never issued that will release the lock. The *expiry* limit is specified in minutes. Once the specified number of minutes have elapsed, a web feature service may release the lock if it exists. Any further transactions issued against that lock using a lock identifier generated by the service will fail. This specification does not constrain how long a lock should be held if the **expiry** attribute is not specified. However, it would be prudent for a web feature service implementation to include methods to detect and release locks that have been maintained for a long period of time without any transactions being executed to release them.

The **<Lock>** element contains a single **<Filter>** element that is used to define the set of feature instances of the specified feature type to be locked. Using the **<Filter>** element, one or more feature instances can be enumerated using their identifiers; or a set of features can be identified by specifying spatial and non-spatial constrains for the lock operation. The <**Filter**> element is defined in the Filter Encoding Implementation Specification [3].

The optional **lockAction** attribute is used to control how feature locks are acquired. A lock action of **ALL** indicates that a web feature service should try to acquire a lock on all requested feature instances. If all feature instances cannot be locked, then the operation should fail, and no feature instances should remain locked. If the lock action is set to **SOME**, then a web feature service shall attempt to lock as many of the requested feature instances as it can. The default lock action shall be **ALL**. Section 10.2.2 presents a state machine for the **LockFeature** operation.

10.2.2 State machine notation from UML

The approach to dynamic modeling 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 4.

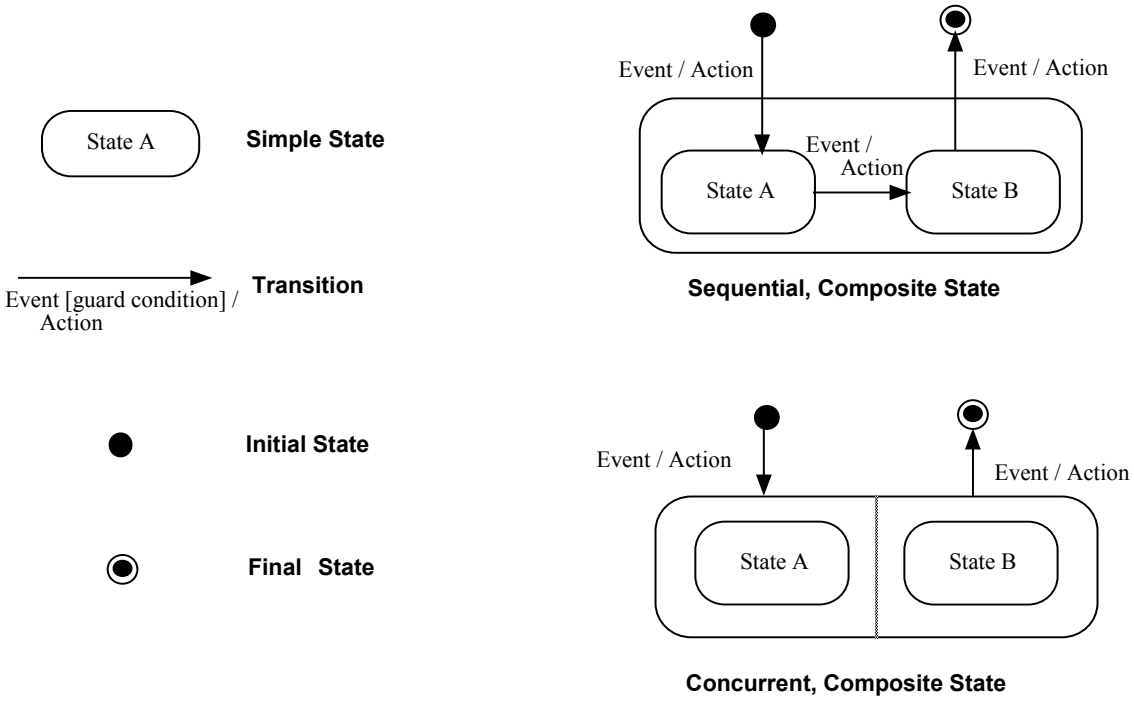

Figure 4 – Summary of UML state diagram notation

10.2.3 State machine for WFS locking

This section defines the state machine for the Lock State for a server that provides the Web Feature Service interface. The state diagram shows the allowed transitions between the states. All other state transitions are disallowed and are consider errors if exhibited by a server.

A physical server may support more than one lock. Each of the locks are independent when viewed from the service defined by the WFS specification.

In the state model below, a transition is typically triggered by a request. Following the messaging model, a WFS Request is paired with a WFS Response. Note that a request-response pair cannot be started while it is active. The request may be cancelled by a HTTP-level command.

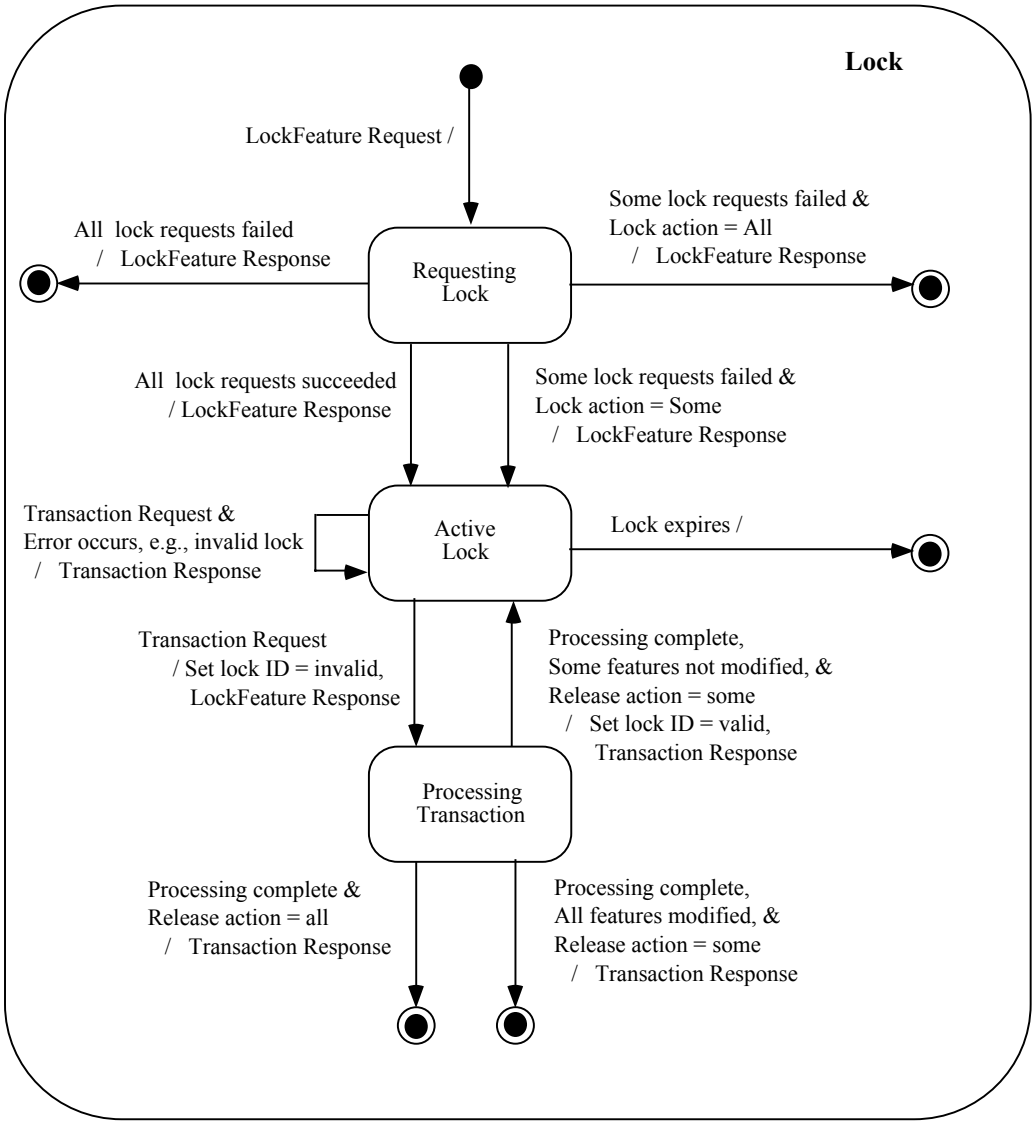

Figure 5 –State diagram for a WFS lock

10.3 Response

The XML encoding of the response to a **LockFeature** request is defined by the following XML Schema fragment:

```
<xsd:element name="WFS_LockFeatureResponse"
             type="wfs:WFS_LockFeatureResponseType"/>
<!-- RESPONSE TYPES -->
<xsd:complexType name="WFS_LockFeatureResponseType">
   <xsd:sequence>
      <xsd:element ref="wfs:LockId"/>
      <xsd:element name="FeaturesLocked"
                   type="wfs:FeaturesLockedType" minOccurs="0"/>
      <xsd:element name="FeaturesNotLocked"
                   type="wfs:FeaturesNotLockedType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="FeaturesLockedType">
  <xsd:sequence maxOccurs="unbounded">
    <xsd:element ref="ogc:FeatureId"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="FeaturesNotLockedType">
  <xsd:sequence maxOccurs="unbounded">
    <xsd:element ref="ogc:FeatureId"/>
```

```
  </xsd:sequence>
</xsd:complexType>
```

In response to a **LockFeature** request, a web feature service shall generate an XML document. This document will contain a lock identifier that a client application can use in subsequent WFS operations to operate upon the set of locked feature instances. The response may also contain the optional elements <**FeaturesLocked**> and <**FeaturesNotLocked**> depending on the value of the **lockAction** attribute.

If the lock action is specified as **SOME**, then the <**WFS_LockFeatureResponse**> element must contain the <**FeaturesLocked**> and <**FeatureNotLocked**> elements.. The **<FeaturesLocked>** element shall list the feature identifiers of all the feature instances that were locked by the **LockFeature** request. The **<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).

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.

10.4 Exceptions

If a WFS does not implement the **LockFeature** operation then it should generate an 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 servicing the request, it shall raise an exception as described in Section 7.7.

10.5 Examples

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="1.0.0"
   service="WFS"
   lockAction="SOME"
   xmlns="http://www.opengis.net/wfs"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <Lock typeName="myns:INWATERA_1M">
      <ogc:Filter>
         <ogc:FeatureId fid="INWATERA_1M.1013"/>
         <ogc:FeatureId fid="INWATERA_1M.1014"/>
         <ogc:FeatureId fid="INWATERA_1M.1015"/>
         <ogc:FeatureId fid="INWATERA_1M.1016"/>
         <ogc:FeatureId fid="INWATERA_1M.1017"/>
      </ogc:Filter>
   </Lock>
</LockFeature>
```

Sample response:

```
<?xml version="1.0" ?>
<WFS_LockFeatureResponse
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <LockId>1</LockId>
   <FeaturesLocked>
      <ogc:FeatureId fid="INWATERA_1M.1013"/>
      <ogc:FeatureId fid="INWATERA_1M.1014"/>
      <ogc:FeatureId fid="INWATERA_1M.1016"/>
      <ogc:FeatureId fid="INWATERA_1M.1017"/>
   </FeaturesLocked>
   <FeaturesNotLocked>
      <ogc:FeatureId fid="INWATERA_1M.1015"/>
   </FeaturesNotLocked>
</WFS_LockFeatureResponse>
```

Example 2

Lock all the feature instances of type INWATERA_1M.

Request:

```
<?xml version="1.0" ?>
<LockFeature
   version="1.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <Lock typeName="myns:INWATERA_1M"/>
</LockFeature>
```

Sample response:

```
<?xml version="1.0" ?>
<WFS_LockFeatureResponse
   xmlns="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <LockId>2</LockId>
</WFS_LockFeatureResponse>
```

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="1.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   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 ../wfs/1.0.0/WFS-transaction.xsd">
   <Lock handle="Lock1" typeName="myns:INWATERA_1M">
      <ogc:Filter>
         <ogc:Within>
            <ogc:PropertyName>myns:WKB_GEOM</ogc:PropertyName>
            <gml:Polygon gid="1"
                 srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:outerBoundaryIs>
                  <gml:LinearRing>
                  <gml:coordinates>-95.7,38.1 -97.8,38.2 ...</gml:coordinates>
                  </gml:LinearRing>
```

```
               </gml:outerBoundaryIs>
            </gml:Polygon>
         </ogc:Within>
      </ogc:Filter>
   </Lock>
</LockFeature>
```

Sample response:

```
<?xml version="1.0" ?>
<WFS_LockFeatureResponse
   xmlns="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <LockId>A1014375BD</LockId>
</WFS_LockFeatureResponse>
```

Example 4

This example locks features of type BUILTUPA_1M and INWATERA_1M. The lock labeled with the handle *LOCK1* locks all the features inside the defined window. The lock labeled with the handle *LOCK2* locks the features INWATERA_1M.1212, INWATERA_1M.1213 and INWATERA_1M.10.

Request:

```
<LockFeature
   version="1.0.0"
   service="WFS"
   expiry="4"
   lockAction="SOME"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   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 ../wfs/1.0.0/WFS-transaction.xsd">
   <Lock handle="LOCK1" typeName="myns:BUILTUPA_1M">
      <ogc:Filter>
         <ogc:Within>
            <ogc:PropertyName>BUILTUPA_1M/WKB_GEOM</ogc:PropertyName>
            <gml:Polygon gid="1"
               srsName="http://www.opengis.net/gml/epsg.xml#4326">
               <gml:outerBoundaryIs>
                  <gml:LinearRing>
                  <gml:coordinates>-95.7,38.1 -97.8,38.2 ...</gml:coordinates>
                  </gml:LinearRing>
               </gml:outerBoundaryIs>
            </gml:Polygon>
         </ogc:Within>
      </ogc:Filter>
   </Lock>
   <Lock handle="LOCK2" typeName="myns:INWATERA_1M">
      <ogc:Filter>
         <ogc:FeatureId fid="INWATERA_1M.1212"/>
         <ogc:FeatureId fid="INWATERA_1M.1213"/>
         <ogc:FeatureId fid="INWATERA_1M.10"/>
      </ogc:Filter>
   </Lock>
</LockFeature>
```

Sample response:

```
<?xml version="1.0" ?>
<WFS_LockFeatureResponse
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <LockId>LOCK1A</LockId>
   <FeaturesLocked>
      <ogc:FeatureId fid="BUILTUPA_1M.1" />
      <ogc:FeatureId fid="BUILTUPA_1M.10" />
```

```
        <ogc:FeatureId fid="BUILTUPA_1M.34" />
        <ogc:FeatureId fid="BUILTUPA_1M.786" />
        <ogc:FeatureId fid="BUILTUPA_1M.3" />
        <ogc:FeatureId fid="BUILTUPA_1M.13" />
        <ogc:FeatureId fid="BUILTUPA_1M.47563" />
        <ogc:FeatureId fid="INWATERA_1M.1212" />
        <ogc:FeatureId fid="INWATERA_1M.1213" />
        <ogc:FeatureId fid="INWATERA_1M.10" />
     </FeaturesLocked>
  </WFS_LockFeatureResponse>
```

11 Transaction operation

11.1 Introduction

The **Transaction** operation is used to describe data transformation operations that are to be applied to web accessible feature instances. A web feature service may process a **Transaction** operation directly or possibly translate it into the language of a target datastore to which it is connected and then have the datastore execute the transaction. When the transaction has been completed, a web feature service will generate an XML response document indicating the completion status of the transaction.

The **Transaction** operation is optional and a WFS implementation does not need to support it to conform to this specification. If the **Transaction** operation is supported then this fact must be advertised on the capabilities document as describes in section 12.

11.2 Request

11.2.1 Schema definition

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:sequence>
      <xsd:element ref="wfs:LockId" minOccurs="0"/>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
         <xsd:element ref="wfs:Insert"/>
         <xsd:element ref="wfs:Update"/>
         <xsd:element ref="wfs:Delete"/>
         <xsd:element ref="wfs:Native"/>
      </xsd:choice>
   </xsd:sequence>
   <xsd:attribute name="version"
                  type="xsd:string" use="required" fixed="1.0.0"/>
   <xsd:attribute name="service"
                  type="xsd:string" use="required" fixed="WFS"/>
   <xsd:attribute name="handle"
                  type="xsd:string" use="optional"/>
   <xsd:attribute name="releaseAction"
                  type="wfs:AllSomeType" use="optional"/>
</xsd:complexType>
<xsd:element name="LockId" type="xsd:string"/>
<xsd:element name="Insert" type="wfs:InsertElementType"/>
<xsd:complexType name="InsertElementType">
   <xsd:sequence>
      <xsd:element ref="gml:_Feature" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="handle" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:element name="Update" type="wfs:UpdateElementType"/>
<xsd:complexType name="UpdateElementType">
   <xsd:sequence>
      <xsd:element ref="wfs:Property" maxOccurs="unbounded"/>
      <xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1"/>
```

```
      </xsd:sequence>
      <xsd:attribute name="handle" type="xsd:string" use="optional"/>
      <xsd:attribute name="typeName" type="xsd:QName" use="required"/>
   </xsd:complexType>
   <xsd:element name="Delete" type="wfs:DeleteElementType"/>
   <xsd:complexType name="DeleteElementType">
      <xsd:sequence>
         <xsd:element ref="ogc:Filter" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="handle" type="xsd:string" use="optional"/>
      <xsd:attribute name="typeName" type="xsd:QName" use="required"/>
   </xsd:complexType>

   <xsd:element name="Property" type="wfs:PropertyType"/>
   <xsd:complexType name="PropertyType">
      <xsd:sequence>
         <xsd:element name="Name" type="xsd:QName"/>
         <xsd:element name="Value"/>
      </xsd:sequence>
   </xsd:complexType>
```

11.2.2 Attribute descriptions

As described in section 7.8, the **handle** attribute can be used to assign a mnemonic name to the element with which it is associated. Its intended use, is to make error reporting more meaningful for a client application. In the event that an error is encountered, the **handle** attribute can be used by a web feature service to locate the error when generating an exception report. In the event that no **handle** is specified, a web feature service may attempt to report the location of the exception relative to the current **Transaction** request using line numbers or some other convenient mechanism.

Assuming that a WFS implementation supports the optional **LockFeature** and/or **GetFeatureWithLock** operations, the **releaseAction** attribute is used to control 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>** should be released when the transaction completes, regardless of whether or not a particular feature instance in the locked set was actually operated upon. A value of **SOME** indicates that only the locks on feature instances modified by the transaction should be released. The other, unmodified, feature instances should remain locked using the same **<LockId>** so that subsequent transactions can operate on those feature instances. In the event that the **releaseAction** is set to **SOME**, and an expiry period was specified on the <**LockFeature**> or <**GetFeatureWithLock**> elements using the **expiry** attribute, the expiry counter must be reset to zero after each transaction unless all feature instances in the locked set have been operated upon. The default **releaseAction** value is **ALL.**

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 should remain locked when the transaction terminates. Subsequent transaction operations can then be submitted by the client application, using the same lock identifier to modify the remaining 10 feature instances.

11.2.3 <Transaction> element

A **<Transaction>** element may contain zero or more **<Insert>**, **<Update>**, or **<Delete>** elements that describe operations to create, modify or destroy feature instances. An empty **<Transaction>** request is valid but not very useful.

The optional **<LockId>** element is used to specify that the transaction will be applied to previously locked set of feature instances. Section 10 presents a full description of a feature locking mechanism. If the WFS does not support feature locking, then the **<LockId>** element can be ignored. If a WFS does support locking and an invalid lock identifier is specified in the transaction, then the transaction shall fail and the web feature service shall report the error as described in Section 7.7.

At the end of a transaction, the web feature service shall apply transaction semantics appropriate to the particular system used to persistently store features. For example, if the datastore is a SQL based RDBMS, then a *commit* will be executed at the end of the transaction (or a *rollback* should the transaction fail). Any locks maintained by the web feature service for the duration of the transaction shall be released according to the value of the **releaseAction** attribute described above.

The **<Native>** element is defined in Section 7.5.

11.2.4 <Insert> element

The **<Insert>** element is used to create new feature instances. The initial state of a feature to be created is expressed using GML and must validate relative to a GML application schema generated by the **DescribeFeatureType** operation [sec. 8]. Multiple **<Insert>** elements can be enclosed in a single **Transaction** request and multiple feature instances can be created using a single **<Insert>** element.

In response to an **<Insert>** operation, a web feature service shall generate a list of newly generated feature identifiers assigned to the new feature instances. The feature identifiers must be presented in the order in which the **<Insert>** operations were encountered in the **Transaction** request.

Example

The following transaction creates two instances of feature type INWATERA_1M.

```
<?xml version="1.0"?>
<wfs:Transaction
   version="1.0.0"
   service="WFS"
   xmlns="http://www.someserver.com/myns"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.someserver.com/myns
                       http://www.someserver.com/wfs/cwwfs.cgi?
                       request=describefeaturetype&amp;typename=INWATERA_1M.xsd
                       http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:Insert>
      <INWATERA_1M>
         <WKB_GEOM>
            <gml:Polygon gid="1"
               srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:outerBoundaryIs>
                  <gml:LinearRing>
                     <gml:coordinates>-98.54,24.26 ...</gml:coordinates>
                  </gml:LinearRing>
               </gml:outerBoundaryIs>
            </gml:Polygon>
         </WKB_GEOM>
         <ID>150</ID>
         <F_CODE>ABCDE</F_CODE>
         <HYC>152</HYC>
```

```
            <TILE_ID>250</TILE_ID>
            <FAC_ID>111</FAC_ID>
         </INWATERA_1M>
         <INWATERA_1M>
            <WKB_GEOM>
               <gml:Polygon gid="1"
                  srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                  <gml:outerBoundaryIs>
                     <gml:LinearRing>
                        <gml:coordinates>-99.99,22.22 ...</gml:coordinates>
                     </gml:LinearRing>
                  </gml:outerBoundaryIs>
               </gml:Polygon>
            </WKB_GEOM>
            <ID>111</ID>
            <F_CODE>FGHIJ</F_CODE>
            <HYC>222</HYC>
            <TILE_ID>333</TILE_ID>
            <FAC_ID>444</FAC_ID>
         </INWATERA_1M>
      </wfs:Insert>
   </wfs:Transaction>
```

The schema reference to INWATERA_1M.xsd is assumed to be created using the **DescribeFeatureType** operation [sec. 8]. In this example, the document is statically referenced, but could just as easily have been dynamically referenced.

11.2.5 <Update> element

The **<Update>** element describes one update operation that is to be applied to a feature or set of features of a single feature type. Multiple **<Update>** operations can be contained in a single **Transaction** request.

An **<Update>** element contains one or more <**Property>** elements that specify the name and replacement value for a property that belongs to the feature type specified using the mandatory **typeName** attribute. A **<Property>** element contains a **<Name>** element that, in turn, contains the name of the feature property to be modified and an optional **<Value>** element that contains the replacement value for the named feature property. The omission of the **<Value>** element means that the property should be assigned a NULL value. In the event that the property is not nillable, a WFS **must** raise an exception indicating that NULL value is not allowed.

The scope of the **<Update>** element is constrained by using the **<Filter>** element. The **<Filter>** element can be used to limit the scope of an update operation to an enumerated set of features or a set of features defined using spatial and non-spatial constraints.

The full definition of the **<Filter>** element is described in the Filter Encoding Implementation Specification [3].

Example

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="1.0.0"
   service="WFS"
   xmlns="http://www.someserver.com/myns"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
```

```
      <wfs:Update typeName="BUILTUPA_1M">
         <wfs:Property>
            <wfs:Name>POPULATION</wfs:Name>
            <wfs:Value>4070000</wfs:Value>
         </wfs:Property>
         <ogc:Filter>
            <ogc:FeatureId fid="BUILTUPA_1M.10131"/>
         </ogc:Filter>
      </wfs:Update>
   </wfs:Transaction>
```

Example

Update the *POPULATION_TYPE* property of an enumerated set of features. In this example, the features identified by feature identifiers:

```
BUILTUPA_1M.1013
BUILTUPA_1M.34
BUILTUPA_1M.24256
```

have their *POPULATION_TYPE* attribute set to the value "CITY".

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.0.0"
   service="WFS"
   xmlns="http://www.someserver.com/myns"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:Update typeName="BUILTUPA_1M">
      <wfs:Property>
         <wfs:Name>POPULATION_TYPE</wfs:Name>
         <wfs:Value>CITY</wfs:Value>
      </wfs:Property>
      <ogc:Filter>
         <ogc:FeatureId fid="BUILTUPA_1M.1013"/>
         <ogc:FeatureId fid="BUILTUPA_1M.34"/>
         <ogc:FeatureId fid="BUILTUPA_1M.24256"/>
      </ogc:Filter>
   </wfs:Update>
</wfs:Transaction>
```

Example

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 *TILE_ID* property to values greater than 1000.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.0.0"
   service="WFS"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:Update typeName="myns:BUILTUPA_1M">
      <wfs:Property>
         <wfs:Name>myns:NAME</wfs:Name>
         <wfs:Value>somestring</wfs:Value>
      </wfs:Property>
      <ogc:Filter>
         <ogc:FeatureId fid="BUILTUPA_1M.1013"/>
         <ogc:FeatureId fid="BUILTUPA_1M.34"/>
         <ogc:FeatureId fid="BUILTUPA_1M.24256"/>
      </ogc:Filter>
   </wfs:Update>
   <wfs:Update typeName="myns:BUILTUPA_1M">
      <wfs:Property>
         <wfs:Name>myns:FAC_ID</wfs:Name>
```

```
            <wfs:Value>100</wfs:Value>
         </wfs:Property>
         <ogc:Filter>
            <ogc:PropertyIsGreaterThan>
               <ogc:PropertyName>BUILTUPA_1M/TILE_ID</ogc:PropertyName>
               <ogc:Literal>1000</ogc:Literal>
            </ogc:PropertyIsGreaterThan>
         </ogc:Filter>
      </wfs:Update>
   </wfs:Transaction>
```

Example

This example updates two feature classes, OCEANSA_1M and TREESA_1M. All features of OCEANSA_1M with a DEPTH greater than 2400m are updated and feature TREESA_1M.1010 is also updated.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.0.0"
   service="WFS"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:Update typeName="myns:OCEANSA_1M">
      <wfs:Property>
         <wfs:Name>myns:DEPTH</wfs:Name>
         <wfs:Value>2400</wfs:Value>
      </wfs:Property>
      <ogc:Filter>
         <ogc:PropertyIsGreaterThan>
            <ogc:PropertyName>OCEANSA_1M.DEPTH</ogc:PropertyName>
            <ogc:Literal>2400</ogc:Literal>
         </ogc:PropertyIsGreaterThan>
      </ogc:Filter>
   </wfs:Update>
   <wfs:Update typeName="myns:TREESA_1M">
      <wfs:Property>
         <wfs:Name>myns:TREETYPE</wfs:Name>
         <wfs:Value>CONIFEROUS</wfs:Value>
      </wfs:Property>
      <ogc:Filter>
         <ogc:FeatureId fid="TREESA_1M.1010"/>
      </ogc:Filter>
   </wfs:Update>
</wfs:Transaction>
```

11.2.6 <Delete> element

The **<Delete>** element is used to indicate that one or more feature instances should be deleted. The scope of a delete operation is constrained by using the **<Filter>** element as described in the Filter Encoding Implementation Specification [3]**.**

Example

Delete a single feature.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.0.0"
   service="WFS"
   xmlns="http://www.someserver.com/myns"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:Delete typeName="INWATERA_1M">
```

```
      <ogc:Filter>
         <ogc:FeatureId fid="INWATERA_1M.1013"/>
      </ogc:Filter>
   </wfs:Delete>
</wfs:Transaction>
```

Example

This examples deletes an enumerated set of feature instances.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.0.0"
   service="WFS"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:Delete typeName="myns:INWATERA_1M">
      <ogc:Filter>
         <ogc:FeatureId fid="INWATERA_1M.1013"/>
         <ogc:FeatureId fid="INWATERA_1M.10"/>
         <ogc:FeatureId fid="INWATERA_1M.13"/>
         <ogc:FeatureId fid="INWATERA_1M.140"/>
         <ogc:FeatureId fid="INWATERA_1M.5001"/>
         <ogc:FeatureId fid="INWATERA_1M.2001"/>
      </ogc:Filter>
   </wfs:Delete>
</wfs:Transaction>
```

Example

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 **<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="1.0.0"
   service="WFS"
   xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:Delete typeName="INWATERA_1M">
      <ogc:Filter>
         <ogc:Within>
            <ogc:PropertyName>WKB_GEOM</ogc:PropertyName>
            <gml:Polygon gid="pp9"
               srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:outerBoundaryIs>
                  <gml:LinearRing>
                  <gml:coordinates>-95.7,38.1 -97.8,38.2 ...</gml:coordinates>
                  </gml:LinearRing>
               </gml:outerBoundaryIs>
            </gml:Polygon>
         </ogc:Within>
      </ogc:Filter>
   </wfs:Delete>
</wfs:Transaction>
```

11.3 Response

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 **<Insert>** operations, then the web feature service must report the feature identifiers of all newly created features. In the event that the

transaction fails to execute, the web feature service shall also indicate this in the response.

The XML encoding of the WFS transaction response is defined by the following XML Schema fragment:

```
<xsd:element name="WFS_TransactionResponse"
             type="wfs:WFS_TransactionResponseType"/>
<xsd:complexType name="WFS_TransactionResponseType">
   <xsd:sequence>
      <xsd:element name="InsertResult"
                   type="wfs:InsertResultType"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="TransactionResult"
                   type="wfs:TransactionResultType"/>
   </xsd:sequence>
   <xsd:attribute name="version"
                  type="xsd:string" use="required" fixed="1.0.0"/>
</xsd:complexType>
<xsd:complexType name="TransactionResultType">
   <xsd:sequence>
      <xsd:element name="Status" type="wfs:StatusType"/>
      <xsd:element name="Locator" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Message" type="xsd:string" minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="handle" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:complexType name="InsertResultType">
   <xsd:sequence>
      <xsd:element ref="ogc:FeatureId" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="handle" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:complexType name="StatusType">
   <xsd:choice>
      <xsd:element ref="wfs:SUCCESS"/>
      <xsd:element ref="wfs:FAILED"/>
      <xsd:element ref="wfs:PARTIAL"/>
   </xsd:choice>
</xsd:complexType>
<xsd:element name="SUCCESS" type="wfs:EmptyType"/>
<xsd:element name="FAILED" type="wfs:EmptyType"/>
<xsd:element name="PARTIAL" type="wfs:EmptyType"/>
```

The **<WFS_TransactionResponse>** element contains zero or more **<InsertResult>** elements and a **<TransactionResult>** element.

The **<InsertResult>** element contains one or more feature identifiers of newly created feature instances. One **<InsertResult>** element is reported per **<Insert>** element in the request. The insert results are reported in the order in which the **<Insert>** operations were contained in the **<Transaction>** element. Additionally, they can be correlated using the **handle** attribute if it was specified.

The overall result of a transaction request is specified using the **<TransactionResult>** element. The **<TransactionResult>** element must contain a **<Status>** element and may contain **<Locator>** and **<Message>** elements.

The **<Status>** element is used to indicate the completion status of the transaction. Transactions can terminate with a status of:

SUCCESS	The transaction was successfully completed.
FAILED	An exception was encountered while processing one or more

	elements contained in a **Transaction** request.
PARTIAL	The transaction partially succeeded and the data may be in an inconsistent state. For systems that do not support atomic transactions, this outcome is a distinct possibility.

In the event that a transaction request fails, the **<Locator>** element can be used to indicate which part of the transaction failed. If the element at which the failure occured is labeled using a **handle** attribute, then a web feature service may report its value to locate the failure. Otherwise, a web feature service may try to identify the failure relative to the beginning of the transaction request, possibly using line numbers or some other convenient mechanism.

The **<Message>** element is used to report any error messages.

Example

Consider a transaction request (labeled "TX01") that creates a number of new feature instances. The feature instances are created using three **<Insert>** elements labeled "STMT1", "STMT2" and "STMT3". A typical response to such a request might be:

```
<?xml version="1.0" ?>
<wfs:TransactionResponse
   version="1.0.0"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:InsertResult handle="STMT1">
      <ogc:FeatureId fid="SOMEFEATURE.4567"/>
      <ogc:FeatureId fid="SOMEFEATURE.4568"/>
      <ogc:FeatureId fid="SOMEFEATURE.4569"/>
   </wfs:InsertResult>
   <wfs:InsertResult handle="STMT2">
      <ogc:FeatureId fid="FEATURE1.4569"/>
   </wfs:InsertResult>
   <wfs:InsertResult handle="STMT3">
      <ogc:FeatureId fid="FEATURE2.389345"/>
   </wfs:InsertResult>
   <wfs:TransactionResult handle="TX01">
      <wfs:Status>
         <wfs:SUCCESS/>
      </wfs:Status>
   </wfs:TransactionResult>
</wfs:TransactionResponse>
```

11.4 Exceptions

In the event that a web feature service encounters an error servicing a **Transaction** request, it shall raise an exception as described in Section 7.7.

In the event that a web feature service encounters an error while processing a particular element contained in a **Transaction** request, the web feature service shall queue the result and report the failure in a **<WFS_TransactionResponse>** element [sec. 11.3].

Example

In this example, the second statement of a **Transaction** request has failed. The WFS might generate a response such as:

```
<?xml version="1.0" ?>
<WFS_TransactionResponse
```

```
   version="1.0.0"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">

   <TransactionResult handle="TX01">
      <Status><FAILED/></Status>
      <Locator>STMT2</Locator>
      <Message>ORA-00942: table or view does not exist</Message>
   </TransactionResult>
</WFS_TransactionResponse>
```

11.5 Examples

This example defines a complex transaction, labeled "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="1.0.0"
   service="WFS"
   handle="Transaction 01"
   xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.someserver.com/myns
                       http://www.someserver.com/wfs/cwwfs.cgi?
                       request=DESCRIBEFEATURETYPE&amp;
                       typename=ELEVP_1M,ROADL_1M,BUILTUPA_1M
                       http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">

   <!-- Create a new instance of feature type ELEVP_1M -->
   <wfs:Insert handle="Statement 1">
      <ELEVP_1M>
         <ID>167928</ID>
         <F_CODE>CA</F_CODE>
         <ACC>2</ACC>
         <ELA>1</ELA>
         <ZV2>152</ZV2>
         <TILE_ID>250</TILE_ID>
         <END_ID>111</END_ID>
         <LOCATION>
            <gml:Point gid="e33"
               srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:coordinates>-98.5485,24.2633</gml:coordinates>
            </gml:Point>
         </LOCATION>
      </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>
         <NAME>Highway 401</NAME>
         <SEGMENT>
            <DESIGNATION>SEG_A41</DESIGNATION>
            <GEOMETRY>
               <gml:LineString gid="e3"
                  srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                  <gml:coordinates>...</gml:coordinates>
               </gml:LineString>
            </GEOMETRY>
         </SEGMENT>
         <ROADTYPE>
            <SURFACE_TYPE>Asphalt</SURFACE_TYPE>
            <NLANES>12</NLANES>
```

```
            <GRADE>15</GRADE>
         </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:Name>ROADL_1M/SEGMENT/DESIGNATION</wfs:Name>
         <wfs:Value>SEG_A60</wfs:Value>
      </wfs:Property>
      <ogc:Filter>
         <ogc:PropertyIsBetween>
            <ogc:PropertyName>ROADL_1M/SEGMENT/DESIGNATION</ogc:PropertyName>
            <ogc:LowerBoundary>
               <ogc:Literal>SEG_A60</ogc:Literal>
            </ogc:LowerBoundary>
            <ogc:UpperBoundary>
               <ogc:Literal>SEG_A69</ogc:Literal>
            </ogc:UpperBoundary>
         </ogc:PropertyIsBetween>
      </ogc:Filter>
   </wfs:Update>

   <!-- Create 2 feature instances of feature type BUILTUPA_1M -->
   <wfs:Insert handle="Statement 2">
      <BUILTUPA_1M>
         <PLACEID>4070</PLACEID>
         <NAME>Toronto</NAME>
         <POPULATION>4000000</POPULATION>
         <BNDRY>
            <gml:Polygon gid="g3"
               srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:outerBoundaryIs>
                  <gml:LinearRing>
                  <gml:coordinates>-95.7,38.1 -97.8,-38.2 ...</gml:coordinates>
                  </gml:LinearRing>
               </gml:outerBoundaryIs>
            </gml:Polygon>
         </BNDRY>
      </BUILTUPA_1M>
      <BUILTUPA_1M>
         <PLACEID>1725</PLACEID>
         <NAME>Montreal</NAME>
         <POPULATION>2000000</POPULATION>
         <BNDRY>
            <gml:Polygon gid="e4"
               srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:outerBoundaryIs>
                  <gml:LinearRing>
                  <gml:coordinates>-89.8,44.3 -89.9,44.4 ...</gml:coordinates>
                  </gml:LinearRing>
               </gml:outerBoundaryIs>
            </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:Name>NAME</wfs:Name>
         <wfs:Value>SMALLVILLE</wfs:Value>
      </wfs:Property>
      <ogc:Filter>
         <ogc:FeatureId fid="BUILTUPA_1M.1210"/>
      </ogc:Filter>
   </wfs:Update>

   <!-- Update the geometry of the feature BUILTUPA_1M.1725. -->
   <wfs:Update typeName="BUILTUPA_1M">
      <wfs:Property>
         <wfs:Name>BNDRY</wfs:Name>
         <wfs:Value>
            <gml:Polygon gid="g5"
               srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
```

```
                <gml:outerBoundaryIs>
                   <gml:LinearRing>
                   <gml:coordinates>-89.8,44.3 -89.9,44.4 ...</gml:coordinates>
                   </gml:LinearRing>
                </gml:outerBoundaryIs>
             </gml:Polygon>
          </wfs:Value>
       </wfs:Property>
       <ogc:Filter>
          <ogc:FeatureId fid="BUILTUPA_1M.1725"/>
       </ogc:Filter>
    </wfs:Update>

    <!-- Delete the feature instance BUILTUPA_1M.1013. -->
    <wfs:Delete typeName="BUILTUPA_1M">
       <ogc:Filter>
          <ogc:FeatureId fid="BUILTUPA_1M.1013"/>
       </ogc: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">
       <ogc:Filter>
          <ogc:And>
             <ogc:Within>
                <ogc:PropertyName>BUILTUPA_1M/BNDRY</ogc:PropertyName>
                <gml:Polygon gid="WINDOW"
                   srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                   <gml:outerBoundaryIs>
                      <gml:LinearRing>
                         <gml:coordinates>0,0 0,5 5,5 5,0 ...</gml:coordinates>
                      </gml:LinearRing>
                   </gml:outerBoundaryIs>
                </gml:Polygon>
             </ogc:Within>
             <ogc:And>
                <ogc:PropertyIsGreaterThanOrEqualTo>
                   <ogc:PropertyName>BUILTUPA_1M/POPULATION</ogc:PropertyName>
                   <ogc:Literal>100</ogc:Literal>
                </ogc:PropertyIsGreaterThanOrEqualTo>
                <ogc:PropertyIsLessThanOrEqualTo>
                   <ogc:PropertyName>BUILTUPA_1M/POPULATION</ogc:PropertyName>
                   <ogc:Literal>2000</ogc:Literal>
                </ogc:PropertyIsLessThanOrEqualTo>
             </ogc:And>
          </ogc:And>
       </ogc:Filter>
    </wfs:Delete>
</wfs:Transaction>
```

In response to the successful completion of this request, a web feature service would generate the following response document:

```
<?xml version="1.0" ?>
<wfs:TransactionResponse
   version="1.0.0"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:InsertResult handle="Statement 1">
      <ogc:FeatureId fid="ELEVP_1M.1001"/>
   </wfs:InsertResult>
   <wfs:InsertResult handle="ComplexInsert">
      <ogc:FeatureId fid="ROADL_1M.1553"/>
   </wfs:InsertResult>
   <wfs:InsertResult handle="Statement 2">
      <ogc:FeatureId fid="BUILTUPA_1M.509876"/>
      <ogc:FeatureId fid="BUILTUPA_1M.509877"/>
   </wfs:InsertResult>
   <wfs:TransactionResult handle="Transaction 01">
      <wfs:Status>
         <wfs:SUCCESS/>
      </wfs:Status>
```

```
    </wfs:TransactionResult>
</wfs:TransactionResponse>
```

12 GetCapabilities operation

12.1 Introduction

A web feature service must have the ability to describe its capabilities. This section defines an XML document that a web feature service must generate to define its capabilities.

The capabilities document defined in this specification is closely modeled after the capabilities document defined for map servers as defined in the Web Map Service Implementation Specification [1].

12.2 Request

The **<GetCapabilities>** element is used to request a capabilities document from a web feature service.

It is defined by the following XML Schema fragment:

```
<xsd:element name="GetCapabilities" type="wfs:GetCapabilitiesType"/>
<xsd:complexType name="GetCapabilitiesType">
   <xsd:attribute name="version"
                  type="xsd:string" use="optional"/>
   <xsd:attribute name="service"
                  type="xsd:string" use="required" fixed="WFS"/>
</xsd:complexType>
```

The **service** attribute is described in section 7.8. The **version** attribute, unlike the normal case, is not mandatory since a client application may not have apriori knowledge about what versions a server may support. The client and the server will need to apply the rules of version negotiation as described in Section 6.2.4.

12.3 Response

12.3.1 Response schema

The schema of the response to a **GetCapabilities** request is normatively defined using XML Schema in Annex A.4.

12.3.2 Capabilities document

The capabilities document is composed of four main sections:

1. **Service section**
   The service section provides information about the service itself.

2. **Capabilities section**
   The capabilities section specifies the list of requests that the WFS can handle. Two classes of web feature services, based on the capabilities they support, are defined in the Overview [see Section i].

3. **FeatureType list**
   This section defines the list of feature types (and operations on each feature type) that are available from a web feature service. Additional information, such as SRS, about each feature type is also provided.

4. **Filter capabilities section**
   The schema of the Filter Capabilities Section is defined in the Filter Encoding Implementation Specification [3]. This is an optional section. If it exists, then the WFS should support the operations advertised therein. If the Filter Capabilities Section is not defined, then the client should assume that the server only supports the minimum default set of filter operators as defined in the Filter Encoding Implementation Specification [3].

The **version** attribute specifies the specification revision to which this schema applies. Its format is one, two or three integers separated by periods: "x", or "x.y", or "x.y.z", with the most significant number appearing first. Future revisions are guaranteed to be numbered in a monotonically increasing fashion, though gaps may appear in the sequence.

The **updateSequence** attribute is a sequence number for managing propagation of the contents of the capabilities document. For example, if a feature server adds some feature types, it can increment the update sequence to inform catalog servers that their previously cached versions are now stale. The format is a positive integer.

12.3.3 Service section

The **<Service>** element contains metadata for the feature service as a whole. The following content can be specified for the **<Service>** element:

Table 3 – Service section elements

ELEMENT NAME	DESCRIPTION
Name	A name the service provider assigns to the web feature service instance.
Title	The **<Title>** is a human-readable title to briefly identify this server in menus.
Abstract	The **<Abstract>** is a descriptive narrative for more information about the server.
Keyword	The **<Keyword>** element contains short words to aid catalog searching.
OnlineResource	The **<OnlineResource>** element defines the top-level HTTP URL of this service. Typically the URL of a "home page" for the service.
Fees	The **<Fees>** element contains a text block indicating any fees imposed by the service provider for usage of the service or for data retrieved from the WFS. The keyword NONE is reserved to mean no fees.
AccessConstraints	The **<AccessConstraints>** element constain a text block describing any access constraints imposed by the service provider on the WFS or data retrieved from that service. The keyword NONE is reserved to indicate no access constraints are imposed.

12.3.4 Capabilities Section

The capabilities section is used to specifically define the list of WFS operations that a particular WFS implements. A basic WFS would include the **GetCapabilities**, **DescribeFeatureType** and the **GetFeature** operations. A transactional WFS would also include the **Transaction** operation, possibly the **LockFeature** operation and/or the **GetFeatureWithLock** operation.

The specific capabilities implemented by a WFS are denoted by the following elements:

Table 4 – Web feature service operations

ELEMENT NAME	DESCRIPTION
GetCapabilities	The **<GetCapabilities>** element is included to define the available distributed computing platforms for this service.
DescribeFeatureType	The **<DescribeFeatureType>** element is used to define the available distributed computing platforms for this service and indicate what schema description languages can be used to describe the schema of a feature type when a client requests such a description. **XMLSCHEMA** is the only mandatory language that must be available. The **SCHEMALANGUAGES** entity can be redefined to include vendor specific languages.
Transaction	The **<Transaction>** element is included to define the available distributed computing platforms for this service
GetFeature	The **<GetFeature>** element is used to define the available distributed computing platforms for this service and enumerate the formats available for expressing the results of a query. The **RESULTFORMATS** entity defines the mandatory output format of **GML** but can be redefined to include additional vendor specific formats.
LockFeature	The **<LockFeature>** element is included to define the available distributed computing platforms.

The only available distributed computing platform is HTTP, for which two request methods are defined; GET and POST. The **onlineResource** attribute indicates the URL prefix for HTTP GET requests (everything before the question mark and query string: http://hostname[:port]/path/scriptname); for HTTP POST requests, **onlineResource** is the complete URL.

The **<VendorSpecificCapabilities>** element can be defined to include vendor specific extensions.

12.3.5 FeatureTypeList section

The purpose of the **<FeatureTypeList>** element is to contain a list of feature types that a WFS can service and define the transaction and query elements that are supported on each feature type. The possible transaction and query elements are:

Table 5 – Transaction and Query Elements on Features

ELEMENT NAME	DESCRIPTION
Insert	The **<Insert>** element is used to indicate that the WFS is capable of creating new instances of a feature type.
Update	The **<Update>** element indicates that the WFS can change the existing state of a feature.

Delete	The **<Delete>** element indicates that the WFS can delete or remove instances of a feature type from the datastore.
Query	The **<Query>** element indicates that the WFS is capable of executing a query on a feature type.
Lock	The **<Lock>** element indicates that the WFS is capable of locking instances of a feature type.

Transaction and query elements can be specified globally for all feature types or locally for each specific feature type contained in the **<FeatureTypeList>** element. Globally specified transaction and query elements are inherited by every feature type contained in the **<FeatureTypeList>** element and can be augmented by specifying local transaction and query elements. For example, the **<Query>** element may be specified globally for all feature types contained in the **<FeatureTypeList>,** but the **<Update>** element may only be specified locally for a small number of feature types. If no transaction or query elements are defined anywhere, then the default element **<Query>** will be implied for all feature types contained in the **<FeatureTypeList>** element.

The following elements can be used to describe each feature type contained in a **<FeatureTypeList>** element:

Table 6 – Elements to describe feature types

ELEMENT	DESCRIPTION
Name	The namespace qualified name of the feature type. This element is mandatory.
Title	The **<Title>** is a human-readable title to briefly identify this feature type in menus.
Abstract	The **<Abstract>** is a descriptive narrative for more information about the feature type.
Keyword	The **<Keyword>** element contains short words to aid catalog searching.
SRS	The **<SRS>** element is used to indicate which spatial reference system should be used to express the state of a feature. The SRS may be indicated using either the Petrotechnical Open Software Corporation form 'EPSG:<POSC Code>' or the URL format defined in section 4.3.2 of reference [2].
Operations	The **<Operations>** element defines which are operations are supported on a feature type. Any locally defined operations take precedence over any globally defined operations.
LatLongBoundingBox	The **LatLongBoundingBox** element is used to indicate the edges of an enclosing rectangle in the SRS of the associated feature type. Its purpose is to facilitate geographic searches by indicating where instances of the particular feature type exist. Since multiple LatLongBoundingBoxes can be specified, a WFS can indicate where various clusters of data may exist. This knowledge aids client applications by letting them know where they should query in order to have a high probability of finding data.
MetadataURL	A WFS may use zero or more **<MetadataURL>** elements to offer detailed, standardized metadata about the data in a particular feature type. The type attribute indicates the standard to which the metadata complies; the format attribute indicates how the metadata is structured. Two types are defined at present: 'TC211' = ISO TC211 19115; 'FGDC' = FGDC CSDGM.

12.4 Exceptions

In the event that a web feature service encounters an error servicing a **GetCapabilities** request, it shall raise an exception as described in Section 7.7.

12.5 Examples

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"
   xmlns="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd"/>
```

In response to such a request, a web feature service might generate a document that looks like:

```
<?xml version="1.0" ?>
<WFS_Capabilities
   version="1.0.0"
   xmlns="http://www.opengis.net/wfs"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:ogc="http://www.opengis.net/ogc"
   ">

   <!--  The SERVICE section says something about who is providing the  -->
   <!--  service and where one can go to obtain more information about  -->
   <!--  the service.                                                   -->
   <Service>
      <Name>CubeWerx WFS</Name>
      <Title>CubeWerx Web Feature Service</Title>
      <Abstract>Web Feature Server maintained by CubeWerx Inc.</Abstract>
      <OnlineResource>http://www.someserver.com/wfs/cwwfs.cgi?</OnlineResource>
   </Service>

   <!-- The CAPABILITY section defines which WFS operations this        -->
   <!-- service instance supports, what distributed computing platform  -->
   <!-- is supported for each service and what the entry point is for   -->
   <!-- each operation.                                                 -->
   <Capability>
      <Request>
         <GetCapabilities>
            <DCPType>
               <HTTP>
                  <Get onlineResource="http://www.someserver.com/wfs/cwwfs.cgi?"/>
               </HTTP>
            </DCPType>
            <DCPType>
               <HTTP>
                  <Post onlineResource="http://www.someserver.com/wfs/cwwfs.cgi"/>
               </HTTP>
            </DCPType>
         </GetCapabilities>
         <DescribeFeatureType>
            <SchemaDescriptionLanguage>
               <XMLSCHEMA/>
            </SchemaDescriptionLanguage>
            <DCPType>
               <HTTP>
                  <Get onlineResource="http://www.someserver.com/wfs/cwwfs.cgi?"/>
               </HTTP>
            </DCPType>
            <DCPType>
```

```
            <HTTP>
               <Post onlineResource="http://www.someserver.com/wfs/cwwfs.cgi"/>
            </HTTP>
         </DCPType>
      </DescribeFeatureType>
      <GetFeature>
         <ResultFormat>
            <GML2/>
         </ResultFormat>
         <DCPType>
            <HTTP>
               <Get onlineResource="http://www.someserver.com/wfs/cwwfs.cgi?"/>
            </HTTP>
         </DCPType>
         <DCPType>
            <HTTP>
               <Post onlineResource="http://www.someserver.com/wfs/cwwfs.cgi"/>
            </HTTP>
         </DCPType>
      </GetFeature>
      <GetFeatureWithLock>
         <ResultFormat>
            <GML2/>
         </ResultFormat>
         <DCPType>
            <HTTP>
               <Get onlineResource="http://www.someserver.com/wfs/cwwfs.cgi?"/>
            </HTTP>
         </DCPType>
         <DCPType>
            <HTTP>
               <Post onlineResource="http://www.someserver.com/wfs/cwwfs.cgi"/>
            </HTTP>
         </DCPType>
      </GetFeatureWithLock>
      <Transaction>
         <DCPType>
            <HTTP>
               <Get onlineResource="http://www.someserver.com/wfs/cwwfs.cgi?"/>
            </HTTP>
         </DCPType>
         <DCPType>
            <HTTP>
               <Post onlineResource="http://www.someserver.com/wfs/cwwfs.cgi"/>
            </HTTP>
         </DCPType>
      </Transaction>
      <LockFeature>
         <DCPType>
            <HTTP>
               <Get onlineResource="http://www.someserver.com/wfs/cwwfs.cgi?"/>
            </HTTP>
         </DCPType>
         <DCPType>
            <HTTP>
               <Post onlineResource="http://www.someserver.com/wfs/cwwfs.cgi"/>
            </HTTP>
         </DCPType>
      </LockFeature>
   </Request>
</Capability>

<!-- The FEATURETYPELIST section defines the list of feature types   -->
<!-- that this service instance can operate upon as well as which    -->
<!-- operations are supported on each feature type.                  -->
<FeatureTypeList>
   <Operations>
      <Query/>
   </Operations>
   <FeatureType>
      <Name>myns:BUILTUPA_1M</Name>
      <SRS>EPSG:4326</SRS>
      <Operations>
         <Insert/>
         <Update/>
         <Delete/>
      </Operations>
      <LatLongBoundingBox minx="-179.1296081543" miny="-53.167423248291"
                          maxx="178.44325256348" maxy="70.992721557617"/>
```

```
    </FeatureType>
    <FeatureType>
       <Name>myns:COASTL_1M</Name>
       <SRS>EPSG:4326</SRS>
       <LatLongBoundingBox minx="-179.99942016602" miny="-85.582763671875"
                           maxx="179.9999" maxy="83.627418518066"/>
    </FeatureType>
    <FeatureType>
       <Name>myns:ELEVP_1M</Name>
       <SRS>EPSG:4326</SRS>
       <Operations>
          <Insert/>
          <Update/>
          <Delete/>
       </Operations>
       <LatLongBoundingBox minx="-179.9984893715" miny="-89.83837892767"
                           maxx="179.99234007206" maxy="83.520408603363"/>
    </FeatureType>
    <FeatureType>
       <Name>myns:OCEANSEA_1M</Name>
       <SRS>EPSG:4326</SRS>
       <Operations>
          <Insert/>
          <Update/>
          <Delete/>
       </Operations>
       <LatLongBoundingBox minx="-179.9999" miny="-85.582763671875"
                           maxx="179.99996948242" maxy="89.9999"/>
    </FeatureType>
    <FeatureType>
       <Name>myns:RAILRDL_1M</Name>
       <SRS>EPSG:4326</SRS>
       <Operations>
          <Insert/>
          <Update/>
          <Delete/>
       </Operations>
       <LatLongBoundingBox minx="-165.24467468262" miny="-53.138427734375"
                           maxx="179.60989379883" maxy="78.16796875"/>
    </FeatureType>
    <FeatureType>
       <Name>myns:TREESA_1M</Name>
       <SRS>EPSG:4326</SRS>
       <Operations>
          <Insert/>
          <Update/>
          <Delete/>
       </Operations>
       <LatLongBoundingBox minx="-139.99757385254" miny="25.281270980835"
                           maxx="-52.661720275879" maxy="66.718765258789"/>
    </FeatureType>
  </FeatureTypeList>

  <!-- The FILTER_CAPABILITIES section defines the capabilities of the -->
  <!-- filter supported by this feature instance.  For example, in     -->
  <!-- this case all spatial operator are supported.  Another, simpler -->
  <!-- WFS implementation, may only support the BBOX operator.         -->
  <ogc:Filter_Capabilities>
     <ogc:Spatial_Capabilities>
        <ogc:Spatial_Operators>
           <ogc:BBOX/>
           <ogc:Equals/>
           <ogc:Disjoint/>
           <ogc:Intersect/>
           <ogc:Touches/>
           <ogc:Crosses/>
           <ogc:Contains/>
           <ogc:Overlaps/>
        </ogc:Spatial_Operators>
     </ogc:Spatial_Capabilities>
     <ogc:Scalar_Capabilities>
        <ogc:Logical_Operators/>
        <ogc:Comparison_Operators>
           <ogc:Simple_Comparisons/>
           <ogc:Like/>
           <ogc:Between/>
           <ogc:NullCheck/>
        </ogc:Comparison_Operators>
        <ogc:Arithmetic_Operators>
```

```
            <ogc:Simple_Arithmetic/>
            <ogc:Functions>
               <ogc:Function_Names>
                  <ogc:Function_Name nArgs="1">MIN</ogc:Function_Name>
                  <ogc:Function_Name nArgs="1">MAX</ogc:Function_Name>
                  <ogc:Function_Name nArgs="1">COUNT</ogc:Function_Name>
                  <ogc:Function_Name nArgs="1">DISTINCT</ogc:Function_Name>
               </ogc:Function_Names>
            </ogc:Functions>
         </ogc:Arithmetic_Operators>
      </ogc:Scalar_Capabilities>
   </ogc:Filter_Capabilities>
</WFS_Capabilities>
```

13 Keyword-value pair encoding

13.1 Introduction

This section describes how to encode WFS operations using the standard CGI style of keyword-value pairs. This means that parameters consist of name-value pairs in the form of "name=value" and the pairs are separated by the "&" character. This form of encoding is also known as URL-Encoding.

13.1.1 A note about the examples

In general, URL-encoding requires that certain characters, such as '&', be escaped [10] when they are not used in their intended manner. In this section, however, such characters may not be escaped for the sake of clarity.

In addition, many of the examples in this section include a **FILTER** parameter whose value is an XML encoded filter as specified in the Filter Encoding Implementation Specification [3]. To be rigorously correct, these examples should include namespace and schema location information in the root element <**Filter**>, such that the XML may be validated. Thus the parameter:

```
FILTER=<Filter><Within><PropertyName>INWATERA_1M/WKB_GEOM<PropertyName>
       <gml:Box><gml><coordinates>10,10 20,20</gml:coordinates></gml:Box>
       </Within></Filter>
```

should more correctly be specified as:

```
FILTER=<Filter xmlns="http://www.opengis.net/ogc"
               xmlns:gml="http://www.opengis.net/gml"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://www.opengis.net/ogc
                                   ../filter/1.0.0/filter.xsd
                                   http://www.opengis.net/gml
                                   ../gml/2.1/geometry.xsd">
       <Within><PropertyName>INWATERA_1M/WKB_GEOM<PropertyName>
       <gml:Box><gml><coordinates>10,10 20,20</gml:coordinates></gml:Box>
       </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 section. 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 clarities sake. A **FILTER** parameter's value, in practice, would be composed of a single long string.

13.2 Request parameter rules

13.2.1 Parameter ordering and case

Parameter names **shall not** be case sensitive, but parameter values **shall** be case sensitive. In this document, parameter names are typically shown in uppercase for typographical clarity, not as a requirement.

Parameters in a request **may** be specified in any order.

An OGC Web Service **must** be prepared to encounter parameters that are not part of this specification. In terms of producing results per this specification, an OGC Web Service **shall** ignore such parameters.

13.2.2 Parameter lists

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

Example 1

An example of a list of items.

```
parameter=item1,item2,item2
```

Example 2

An example of multiple lists of items assigned to a single parameter.

```
parameter=(item11,item12,item13)(item21,item22,item23)
```

Parentheses may also be used to delimit multiple local filters when more than one feature type is specified for the **TYPENAME** parameter. The following URL fragment shows how this may be done:

```
typename=FEAT1,FEAT2&filter=(<Filter>… FEAT1 filter…</Filter>)(<Filter>… FEAT2
filter…</Filter>)
```

13.3 Common request parameters

13.3.1 Version parameter

The **VERSION** parameter specifies the protocol version number. The format of the version number and version negotiation are described in Section 6.2.4.

13.3.2 Request parameter

The **REQUEST** parameter indicates which service operation is being invoked. The value operation_name **must** be one of those offered by the OGC Web Service Instance.

13.3.3 Bounding box

The bounding box parameter, **BBOX**, is included in this specification for convenience as a shorthand representation of the very common a bounding box filter which would be

expressed in much longer form using XML and the filter encoding described in [3]. A **BBOX** applies to all feature types listed in the request.

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

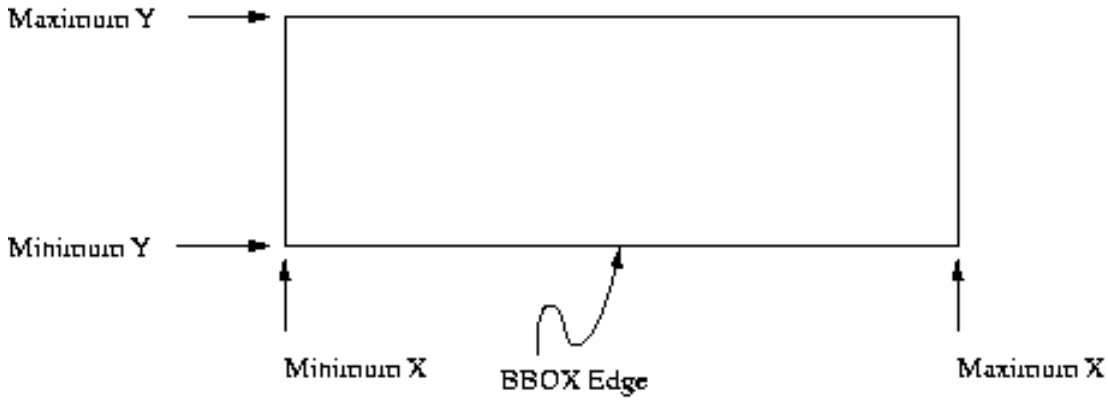

Figure 5 — Bounding Box representation

A Bounding Box **should not** have zero area.

If a request contains an invalid Bounding Box (e.g., one whose minimum X is greater than or equal to the maximum X, or whose minimum Y is greater than or equal to the maximum Y) the server **must** throw an exception.

If a request contains a Bounding Box whose area does not overlap at all with the **LatLongBoundingBox(s)** advertised in the Capabilities XML for the requested geodata object, the server **should** return empty content (e.g. null feature set) for that element. Any elements that are partly or entirely contained in the Bounding Box **should** be returned in the appropriate format.

The SRS of the bounding box **must** be the same as the SRS of the feature type(s) in a request. The SRS of a feature type is advertised by a WFS in the capabilities document. If more than one feature type is specified in a request, the feature types **must** all be in the same SRS and the BBOX must be specified in the common SRS as well.

If the Bounding Box values are not defined for the given SRS (e.g., latitudes greater than 90 degrees in EPSG:4326), the server **should** return empty content for areas outside the valid range of the SRS.

In the particular case of longitude, the following behavior applies regarding the anti-meridian at 180 degrees of longitude. There is a legitimate desire for maps that span the anti-meridian (for example, a map centered on the Pacific Ocean). If Xmin is the west-

most longitude in degrees and Xmax is the east-most, then the following constraint applies:

```
-180 <= Xmin < Xmax < 540
```

Example

Xmin,Xmax values and the corresponding scope of the bounding box:

-180,180 = Earth centered at Greenwich
0,360 = Earth with Greenwich at left edge
120,250 = Pacific Ocean

13.3.4 Vendor-specific parameters

Requests may allow for optional vendor-specific parameters (VSPs) that will enhance the results of a request. Typically, these are used for private testing of non-standard functionality prior to possible standardization. A generic client is **not** required or expected to make use of these VSPs.

An OGC Web Service **must** produce a valid result even if VSPs are missing or malformed (i.e., the Service **shall** supply a default value), or if VSPs are supplied that are not known to the Service (i.e., the Service **shall** ignore unknown request parameters).

An OGC Web Service **may** choose not to advertise some or all of its VSPs. If VSPs are included in the Capabilities XML, the **VendorSpecificCapabilities** element must be redefined accordingly. Additional schema documents may be imported containing the redefinition of the element.

Clients **may** read the vendor specific definition from the capabilities schemas and formulate requests using any VSPs advertised therein.

Vendors **should** choose vendor-specific parameter names with care to avoid clashes with standard parameters.

13.4 Common parameters

The following table describes parameters common to all WFS requests. Subsequent tables may redefine some of the facets of one or more of the parameters in this table.

Table 7 – Common parameters for WFS requests

URL Component	O/M[2]	DEFAULT	Description
http:/server_address/path/script	M		URL prefix of web feature service
VERSION	M[3]	1.0.0	Request version.
SERVICE	M	WFS	Service type.

[2] O = Optional, M=Mandatory

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

REQUEST	M		Name of WFS request.
Additional parameters	O		As described in this section.
Vendor-specific parameters	O		Optional vendor specific parameters.

The mandatory **VERSION** parameter specifies the protocol version number and allows for negotiation as described in Section 6.2.4.

The mandatory **SERVICE** parameter specifies which of the available service types at a particular service instance is being invoked. The value **WFS** is used to indicate that the Web Feature Service should be invoked.

The parameter **REQUEST** must also be included and it indicates which of the web feature service operations to invoke. The possible values of the **REQUEST** parameters are: DescribeFeatureType, LockFeature, Transaction, GetFeature, GetFeatureWithLock or GetCapabilities.

Additional **GET** parameters, as described in this section, shall be expressed as name-value pairs. Parameter names **shall not** be case sensitive. Parameter values **shall** be case sensitive. Parameters in a request may be specified in any order.

A WFS must be prepared to encounter parameters that are not part of the specification. These are known as vendor-specific parameters. Vendor-specific parameters allow vendors to specify additional parameters which will enhance the results of requests. A WFS must produce valid results even if the vendor-specific parameters are missing or malformed. A WFS may declare vendor-specific parameters within its capabilities XML. A WFS may choose to advertise some or all of its vendor specific parameters. Clients may read the capabilities schema and formulate requests using any vendor-specific parameters advertised therein.

13.5 Response

The response to any request encoded using keyword-value pair encoding shall be identical to the responses generated for requests encoded in XML and described in earlier sections of this document.

13.6 Exceptions

Exception reporting for requests encoded using keyword-value pairs shall be identical to that generated by requests encoded using XML. Refer to Sections 7.7 and 11.4 for a detailed discussion of exception reporting.

13.7 Operations

13.7.1 Introduction

This section describes how to formulate WFS requests using standard CGI style keyword-value pair encoding. Heavy use is made of examples that illustrate the various forms possible. In addition, for clarity, each parameter specification in the examples is placed on a separate line.

13.7.2 DescribeFeatureType operation

13.7.2.1 Request

Table 8 – DescribeFeatureType encoding

URL Component	O/M	DEFAULT	Description
REQUEST=DescribeFeatureType	M		Name of request.
TYPENAME	O		A comma separated list of feature types to describe. If no value is specified that is to be interpreted as all feature types.
OUTPUTFORMAT	O	XMLSCHEMA	The output format to use to describe the feature. XMLSCHEMA must be supported. Other output formats, such as DTD are possible.

13.7.2.2 Examples

Example 1

The following example requests the schema description of the feature type TREESA_1M.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=DescribeFeatureType&
       TYPENAME=TREESA_1M
```

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=1.0.0&
       REQUEST=DescribeFeatureType&
       TYPENAME=TREESA_1M,BUILTUPA_1M
```

13.7.3 GetFeature & GetFeatureWithLock operation

13.7.3.1 Request

Table 9 – GetFeature & GetFeatureWithLock encoding

URL Component	O/M	DEFAULT	Description
REQUEST=[GetFeature | GetFeatureWithLock]	M		The name of the WFS request.
PROPERTYNAME	O		A list of properties may be specified for each feature type that is being queried. Refer to Section [13.2.2] on how to form lists of parameters. A "*" character can be used to indicate that all properties should be retrieved. There is a 1:1 mapping between each element in a FEATUREID or TYPENAME list and the

			PROPERTYNAME list. The absense of a value also indicates that all properties should be fetched.
FEATUREVERSION=[ALL | N]	O		If versioning is supported, the FEATUREVERSION parameter directs the WFS on which feature version to fetch.. A value of ALL indicates to fetch all versions of a feature. An integer value fetches the Nth version of a feature. No value indicates that the latest version of the feature should be fetched.
MAXFEATURES=N	O		A positive integer indicating the maximum number of features that the WFS should return in response to a query. If no value is specified then all result instances should be presented.
TYPENAME (**Optional** if FEATUREID is specified.)	M		A list of feature type names to query.
FEATUREID (**Mutually exclusive** with FILTER and BBOX)	O		An enumerated list of feature instances to fetch identified by their feature identifiers.
FILTER (**Prerequisite**: TYPENAME) (**Mutually exclusive** with FEATUREID and BBOX)	O		A filter specification describes a set of features to operate upon. The filter is defined as specified in the Filter Encoding Specification [3]. If the FILTER parameter is used, one filter must be specified for each feature type listed in the TYPENAME parameter. Individual filters encoded in the FILTER parameter are enclosed in parentheses "(" and ")".
BBOX (**Prerequisite**: TYPENAME) (**Mutually exclusive** with FEATUREID and FILTER.)	O		In lieu of a FEATUREID or FILTER, a client may specify a bounding box as described in Section 13.3.3.

13.7.3.2 Examples

Many of the examples in this section include a FILTER parameter whose value is an XML encoded filter as specified in the Filter Encoding Implementation Specification [3]. To be rigorously correct, these examples should include namespace and schema location information in the root element <**Filter**>, such that the XML may be validated. Thus the parameter:

```
FILTER=<Filter><Within><PropertyName>INWATERA_1M/WKB_GEOM<PropertyName>
       <gml:Box><gml><coordinates>10,10 20,20</gml:coordinates></gml:Box>
       </Within></Filter>
```

should more correctly be:

```
FILTER=<Filter xmlns="http://www.opengis.net/ogc"
               xmlns:gml="http://www.opengis.net/gml"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://www.opengis.net/ogc
                                   ../filter/1.0.0/filter.xsd
                                   http://www.opengis.net/gml
```

```
                                ../gml/2.1/geometry.xsd">
<Within><PropertyName>INWATERA_1M/WKB_GEOM<PropertyName>
<gml:Box><gml><coordinates>10,10 20,20</gml:coordinates></gml:Box>
</Within></Filter>
```

For the sake of clarify, however, the namespace and schema location attribute tags have been omitted from the examples in this section. In addition, the schema locations shown are only example locations and the correct schema locations would need to be specified.

Finally, the value of the **FILTER** parameter is broken up over several lines again for clarities sake. An actual **FILTER** parameter would have a single long string as its argument.

Example 1

Query all properties of all instances of type INWATERA_1M.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       TYPENAME=INWATERA_1M
```

Example 2

Query some properties of all instances of type INWATERA_1M.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       PROPERTYNAME=INWATERA_1M/WKB_GEOM,INWATERA_1M/TILE_ID&
       TYPENAME=INWATERA_1M
```

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=1.0.0&
       REQUEST=GetFeature&
       FEATUREID=INWATERA_1M.1013
```

Example 4

Query some properties of the feature instance identified by the feature identifier "INWATERA_1M.1013".

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       PROPERTYNAME=INWATERA_1M/WKB_GEOM,INWATERA_1M/TILE_ID&
       FEATUREID=INWATERA_1M.1013
```

Example 5

Query all properties of an enumerated set of feature instances of type INWATERA_1M.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       FEATUREID=INWATERA_1M.1013,INWATERA_1M.1014,INWATERA_1M.1015
```

Example 6

Query some properties of an enumerated set of feature instances of type INWATERA_1M. In this example, the WKB_GEOM and TILE_ID properties are selected for each feature instance.

```
http://www.someserver.com/wfs.cgi&
     SERVICE=WFS&
     VERSION=1.0.0&
     REQUEST=GetFeature&
     PROPERTYNAME=(INWATERA_1M/WKB_GEOM,INWATERA_1M/TILE_ID)
                  (INWATERA_1M/WKB_GEOM,INWATERA_1M/TILE_ID)
                  (INWATERA_1M/WKB_GEOM,INWATERA_1M/TILE_ID)&
     FEATUREID=INWATERA_1M.1013,INWATERA_1M.1014,INWATERA_1M.1015
```

Example 7

Query all properties of a constrained set of feature instances of type INWATERA_1M.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       TYPENAME=INWATERA_1M&
       FILTER=<Filter><Within><PropertyName>INWATERA_1M/WKB_GEOM<PropertyName>
              <gml:Box><gml><coordinates>10,10 20,20</gml:coordinates></gml:Box>
              </Within></Filter>
```

Example 8

Query some properties of a constrained set of features instances of type INWATERA_1M.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       PROPERTYNAME=INWATERA_1M/WKB_GEOM,INWATERA_1M/TILE_ID&
       TYPENAME=INWATERA_1M&
       FILTER=<Filter><Within><PropertyName>INWATERA_1M/WKB_GEOM<PropertyName><gml:Box>
              <gml:coordinates>10,1020,20</gml:coordinates></gml:Box></Within></Filter>
```

Example 9

Query all properties of a list of feature types.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       TYPENAME=INWATERA_1M,BUILTUPA_1M
```

Example 10

Query some properties of a list of feature types. In this case, the attributes WKB_GEOM and TILE_ID 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=1.0.0&
       REQUEST=GetFeature&
       PROPERTY=(INWATERA_1M/WKB_GEOM,INWATERA_1M/TILE_ID)(BUILTUPA_1M/*)&
       TYPENAME=INWATERA_1M,BUILTUPA_1M
```

Example 11

Query all properties of an enumerated set of feature instances.

```
http://www.someserver.com/wfs.cgi&
```

```
        SERVICE=WFS&
        VERSION=1.0.0&
        REQUEST=GetFeature&
        FEATUREID=INWATERA_1M.1013,BUILTUP_1M.3456
```

Example 12

Query some properties of an enumerated set of feature instances. In this case, the WKB_GEOM and TILE_ID attributes are fetched for feature instance "INWATERA_1M.1013". The attribute WKB_GEOM is fetched for feature instance "BUILTUPA_1M.3456".

```
http://www.someserver.com/wfs.cgi&
        SERVICE=WFS&
        VERSION=1.0.0&
        REQUEST=GetFeature&
        PROPERTYNAME=INWATERA_1M/WKB_GEOM,INWATERA_1M/TILE_ID,BUILTUPA_1M/WKB_GEOM)&
        FEATUREID=INWATERA_1M.1013,BUILTUPA_1M.3456
```

Example 13

Query all properties of all feature instances of the feature type INWATERA_1M and BUILTUPA_1M that lie within the specified box.

```
http://www.someserver.com/wfs.cgi&
        SERVICE=WFS&
        VERSION=1.0.0&
        REQUEST=GetFeature&
        TYPENAME=INWATERA_1M,BUILTUPA_1M&
        FILTER=(<Filter><Within><PropertyName>INWATERA_1M/WKB_GEOM
                <PropertyName><gml:Box><gml:coordinates>10,10 20,20</gml:Box>
                </gml:coordinates></Within></Filter>)(<Filter><Within><PropertyName>
                BUILTUPA_1M/WKB_GEOM<PropertyName><gml:Box><gml:coordinates>10,10
                20,20</gml:Box></gml:coordinates></Within></Filter>)
```

Example 14

Query some properties of a constrained set of feature instances of types INWATERA_1M and BUILTUPA_1M.

```
http://www.SiriusCyberneticsCorp.com/wfs.cgi&
        SERVICE=WFS&
        VERSION=1.0.0&
        REQUEST=GetFeature&
        PROPERTYNAME=(INWATERA_1M/WKB_GEOM,INWATERA_1MTILE_ID)(BUILTUPA_1M/WKB_GEOM)&
        TYPENAME=INWATERA_1M,BUILTUPA_1M&
        FILTER=(<Filter><Within><PropertyName>INWATERA_1M/WKB_GEOM|INWATERA_1M/WKB_GEOM
                <PropertyName><gml:Box><gml:coordinates>10,10 20,20</gml:coordinates>
                </gml:Box></Within></Filter>)(<Filter><Within><PropertyName>
                INWATERA_1M/WKB_GEOM<PropertyName><gml:Box><gml:coordinates>10,10
                20,20</gml:coordinates></gml:Box></Within></Filter>)
```

13.7.4 LockFeature operation

13.7.4.1 Request

Table 10 – LockFeature encoding

URL Component	O/M	DEFAULT	Description
REQUEST=LockFeature	M		The name of the request.
TYPENAME (**Optional** if FEATUREID is specified.)	M		Names or one or more feature types whose feature instances are to be locked.
EXPIRY	O		The number of minutes the lock should persist before being cleared. If no value is specified then the lock should persist indefinitely.

LOCKACTION=[ALL | SOME]	O		Specify how the lock should be acquired. ALL indicates to try to get all feature locks otherwise fail. SOME indicates to try to get as many feature locks as possible.
FEATUREID (**Mutually exclusive** with FILTER and BBOX.)	O		An enumerated list of feature instance identifiers indicating which feature instances to lock.
FILTER (**Prerequisite**: TYPENAME) (**Mutually exclusive** with FEATUREID and BBOX.)	O		An XML string encoded as described in [3] indicating which features to operate upon. If the FILTER parameter is used, one filter must be specified for each feature type listed in the TYPENAME parameter. Individual filters encoded in the FILTER parameter are enclosed in parentheses "(" and ")".
BBOX (**Prerequisite**: TYPENAME) (**Mutually exclusive** with FEATUREID and FILTER.)	O		In lieu of a FEATUREID or FILTER, a client may specify a bounding box as described in Section 13.3.3.

13.7.4.2 Examples

Example 1

The following example locks all instances of feature type INWATERA_1M.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=LockFeature&
       TYPENAME=INWATERA_1M
```

Example 2

The following example locks the feature identified by "ROADL_1M.1013".

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=LockFeature&
       FEATUREID=ROADL_1M.1013
```

Example 3

The following example locks all feature instances of feature type INWATERA_1M and BUILTUPA_1M.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=LockFeature&
       TYPENAME=INWATER_1M,BUILTUPA_1M
```

Example 4

The following example locks all features of feature type INWATERA_1M and BUILTUPA_1M that lie INSIDE a user specified region of interest.

```
http://www.someserver.com/wfs.cgi?
     SERVICE=WFS&
     VERSION=1.0.0&
     REQUEST=LockFeature&
     LOCKACTION=ALL&
```

```
TYPENAME=INWATER_1M,BUILTUPA_1M&
FILTER=(<Filter><Within><PropertyName>WKB_GEOM</PropertyName><gml:Box>
        <gml:coordinates>10,10 20,20</gml:coordinates></gml:Box></Within>
        </Filter>)(<Filter><Within><PropertyName>WKB_GEOM</PropertyName><gml:Box>
        <gml:coordinates>10,10 20,20</gml:coordinates></gml:Box></Within>
        </Filter>)
```

13.7.5 Transaction operation

The only supported operation of the transaction interface is the DELETE operation. Expressing INSERT or UPDATE requests, which can be quite lengthy, is not convenient using keyword-value pair encoding.

13.7.5.1 Request

Table 11 – Transaction encoding

URL Component	O/M	DEFAULT	Description
REQUEST=Transaction	M		The name of the WFS request.
OPERATION=Delete	M		Transaction operation to execute. Currently only *Delete* is defined.
TYPENAME (**Optional** if FEATUREID is specified.)	M		A list of feature types upon which to apply the operation.
RELEASEACTION=[ALL|SOME]	O		A value of ALL indicates that all feature locks should be released when a transaction terminates. A value of SOME indicates that only those records that are modified should be released. The remaining locks are maintained
FEATUREID (**Mutually exclusive** with FILTER and BBOX)	O		A list of feature identifiers upon which the specified operation shall be applied. Optional. No default.
FILTER (**Prerequisite**: TYPENAME) (**Mutually exclusive** with FEATUREID and BBOX)	O		A filter specification describes a set of features to operate upon. The format of the filter is defined in the Filter Encoding Specification [3]. If the FILTER parameter is used, one filter must be specified for each feature type listed in the TYPENAME parameter. Individual filters encoded in the FILTER parameter are enclosed in parentheses "(" and ")".
BBOX (**Prerequisite**: TYPENAME) (**Mutually exclusive** with FILTER and FEATUREID)	O		In lieu of a FEATUREID or FILTER, a client may specify a bounding box as described in Section 13.3.3.

13.7.5.2 Examples

Example 1

Delete the feature instance identified by "ROADL_1M.1013".

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=Transaction&
       OPERATION=Delete&
       FEATUREID=ROADL_1M.1013
```

Example 2

The following example deletes all features of type INWATERA_1M and BUILTUPA_1M that lie INSIDE the specified box.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=Transaction&
       OPERATION=Delete&
       TYPENAME=INWATER_1M,BUILTUPA_1M&
       FILTER=(<Filter><Within><PropertyName>INWATERA_1M/WKB_GEOM<PropertyName><gml:Box>
              <gml:coordinates>10,10 20,20</gml:coordinates></gml:Box></Within>
              </Filter>)(< Filter><Within><PropertyName>BUILTUPA_1M/WKB_GEOM
              <PropertyName><gml:Box><gml:coordinates>10,10 20,20</gml:coordinates>
              </gml:Box></Within></Filter>)
```

Example 3

The following example is the same as Example 2, except that the BBOX parameter is used to specify the spatial constraint on the Delete command.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=Transaction&
       OPERATION=Delete&
       TYPENAME=INWATER_1M,BUILTUPA_1M&
       BBOX=10,10,20,20
```

13.7.6 GetCapabilities Operation

13.7.6.1 Request

Table 12 – GetCapabilities encoding

URL Component	O/M	DEFAULT	Description
REQUEST=GetCapabilities	M		Name of request.

13.7.6.2 Examples

Example 1

Request the capabilities document from a WFS.

```
http://www.someserver.com/wfs.cgi?
       VERSION=1.0.0&
       SERVICE=WFS&
       REQUEST=GetCapabilities
```

ANNEX A – XML Schema definitions (Normative)

A.1 Introduction

This annex contains the normative XML Schema definitions of the WFS operations, capabilities document, and exceptions.

Annex A.2 contains the XML Schema definition for exception messages generated by a web feature service. All exception messages generated by a WFS must validate against this schema.

Annex A.3 contains the XML Schema definition of the basic operations that a web feature service must provide. The basic set of operations includes GetCapabilities, DescribeFeatureType and GetFeature. A web feature service must implement this set of operations.

Annex A.4 contains the XML Schema definitions of optional operations related to transaction processing. The set of operations includes GetFeatureWithLock, LockFeature and Transaction. A web feature service may implement some or all of these operations.

Annex A.5 contains the XML Schema definition of the capabilities document that a web feature service must generate in response to a GetCapabilities request. The GetCapabilities response of a web feature service must validate against this schema. Further, if a web feature service implements vendor specific extensions, this schema in Annex A.5 must be extended in the places indicated so that the vendor extensions can be discovered by a client application wishing to do so.

A.2 OGC-exception.xsd

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

   <xsd:element name="ServiceExceptionReport">
      <xsd:annotation>
         <xsd:documentation>
            The ServiceExceptionReport element contains one
            or more ServiceException elements that describe
            a service exception.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="ServiceException"
                         type="ogc:ServiceExceptionType"
                         minOccurs="0" maxOccurs="unbounded">
               <xsd:annotation>
                  <xsd:documentation>
                     The Service exception element is used to describe
                     a service exception.
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:element>
         </xsd:sequence>
```

```
         <xsd:attribute name="version" type="xsd:string" fixed="1.2.0"/>
      </xsd:complexType>
   </xsd:element>

   <xsd:complexType name="ServiceExceptionType">
      <xsd:annotation>
         <xsd:documentation>
            The ServiceExceptionType type defines the ServiceException
            element.  The content of the element is an exception message
            that the service wished to convey to the client application.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:attribute name="code" type="xsd:string">
               <xsd:annotation>
                  <xsd:documentation>
                     A service may associate a code with an exception
                     by using the code attribute.
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="locator" type="xsd:string">
               <xsd:annotation>
                  <xsd:documentation>
                     The locator attribute may be used by a service to
                     indicate to a client where in the client's request
                     an exception was encountered.  If the request included
                     a 'handle' attribute, this may be used to identify the
                     offending component of the request.  Otherwise the
                     service may try to use other means to locate the
                     exception such as line numbers or byte offset from the
                     begining of the request, etc ...
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:schema>
```

A.3 WFS-basic.xsd

```
<?xml version="1.0"?>
<xsd:schema
   targetNamespace="http://www.opengis.net/wfs"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified">

   <!-- ==========================================================
        Includes and Imports
        ========================================================== -->
   <xsd:import namespace="http://www.opengis.net/gml"
               schemaLocation="../../gml/2.1/feature.xsd"/>
   <xsd:import namespace="http://www.opengis.net/ogc"
               schemaLocation="../../filter/1.0.0/filter.xsd"/>

   <!-- ==========================================================
        REQUEST MESSAGES
        ========================================================== -->
   <xsd:element name="GetCapabilities" type="wfs:GetCapabilitiesType">
      <xsd:annotation>
         <xsd:documentation>
            The GetCapapbilities element is used to request that a Web Feature
            Service generate an XML document describing the organization
            providing the service, the WFS operations that the service
            supports, a list of feature types that the service can operate
            on and list of filtering capabilities that the service support.
            Such an XML document is called a capabilities document.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:element>
   <xsd:element name="DescribeFeatureType" type="wfs:DescribeFeatureTypeType">
```

```
   <xsd:annotation>
      <xsd:documentation>
         The DescribeFeatureType element is used to request that a Web
         Feature Service generate a document describing one or more
         feature types.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:element name="GetFeature" type="wfs:GetFeatureType">
   <xsd:annotation>
      <xsd:documentation>
         The GetFeature element is used to request that a Web Feature
         Service return feature instances of one or more feature types.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>

<!-- ==============================================================
     RESPONSE MESSAGES
     ============================================================== -->
<xsd:element name="FeatureCollection"
             type="wfs:FeatureCollectionType"
             substitutionGroup="gml:_FeatureCollection">
   <xsd:annotation>
      <xsd:documentation>
         This element is a container for the response to a GetFeature
         or GetFeatureWithLock (WFS-transaction.xsd) request.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>

<!-- ==============================================================
     COMMON ATTRIBUTES DOCUMENTATION
     ==============================================================
     VERSION:
        The version attribute is used to indicate to which version
        of the Web Feature Service Implementation Specification a
        request conforms.

     SERVICE:
        The service attribute is used to indicate which service
        should process an operation.  This attribute is particularly
        useful in the case where a single server implements multiple
        services (e.g. WMS, WFS, WCS, etc ...).

     HANDLE:
        The purpose of the handle attribute is to allow a client app
        to associate a mnemonic name with a request for error handling
        purposes.  If a "handle" is specified, and an exception occurs,
        a Web Feature Service may use the handle to identify the
        offending element.

     TYPENAME:
        The typeName attribute is used to specify the name of the
        feature type to be queried.  The term "feature type" is a
        term used by convention to refer to the container storing
        feature instances.  It does not mean type in the programmatic
        sense.  The typeName attribute should, instead, be thought
        of as the feature name. -->

<!-- ==============================================================
     TYPES
     ============================================================== -->
<!-- GETCAPABILITIES -->
<xsd:complexType name="GetCapabilitiesType">
   <xsd:annotation>
      <xsd:documentation>
         This type defines the GetCapabilities operation.  In response
         to a GetCapabilities request, a Web Feature Service must
         generate a capabilities XML document that validates against
         the schemas defined in WFS-capabilities.xsd.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:attribute name="version"
                  type="xsd:string" use="optional" fixed="1.0.0"/>
   <xsd:attribute name="service"
                  type="xsd:string" use="required" fixed="WFS"/>
</xsd:complexType>
<!-- DESCRIBEFEATURETYPE -->
```

```
<xsd:complexType name="DescribeFeatureTypeType">
   <xsd:annotation>
      <xsd:documentation>
         The DescribeFeatureType operation allows a client application
         to request that a Web Feature Service describe one or more
         feature types.   A Web Feature Service must be able to generate
         feature descriptions as valid GML2 application schemas.

         The schemas generated by the DescribeFeatureType operation can
         be used by a client application to validate the output.

         Feature instances within the WFS interface must be specified
         using GML2.  The schema of feature instances specified within
         the WFS interface must validate against the feature schemas
         generated by the DescribeFeatureType request.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:sequence>
      <xsd:element name="TypeName" type="xsd:QName"
                   minOccurs="0" maxOccurs="unbounded">
         <xsd:annotation>
            <xsd:documentation>
               The TypeName element is used to enumerate the feature types
               to be described.  If no TypeName elements are specified
               then all features should be described.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
   </xsd:sequence>
   <xsd:attribute name="version"
                  type="xsd:string" use="required" fixed="1.0.0"/>
   <xsd:attribute name="service"
                  type="xsd:string" use="required" fixed="WFS"/>
   <xsd:attribute name="outputFormat"
                  type="xsd:string" use="optional" default="XMLSCHEMA">
      <xsd:annotation>
         <xsd:documentation>
            The outputFormat attribute is used to specify what schema
            description language should be used to describe features.
            The default value of XMLSCHEMA means that the Web Feature
            Service must generate a GML2 application schema that can
            be used to validate the GML2 output of a GetFeature request
            or feature instances specified in Transaction operations.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:attribute>
</xsd:complexType>
<!-- GETFEATURE -->
<xsd:complexType name="GetFeatureType">
   <xsd:annotation>
      <xsd:documentation>
         A GetFeature element contains one or more Query elements
         that describe a query operation on one feature type.  In
         response to a GetFeature request, a Web Feature Service
         must be able to generate a GML2 response that validates
         using a schema generated by the DescribeFeatureType request.
         A Web Feature Service may support other possibly non-XML
         (and even binary) output formats as long as those formats
         are advertised in the capabilities document.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:sequence>
      <xsd:element ref="wfs:Query" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="version"
                  type="xsd:string" use="required" fixed="1.0.0"/>
   <xsd:attribute name="service"
                  type="xsd:string" use="required" fixed="WFS"/>
   <xsd:attribute name="handle"
                  type="xsd:string" use="optional"/>
   <xsd:attribute name="outputFormat"
                  type="xsd:string" use="optional" default="GML2">
      <xsd:annotation>
         <xsd:documentation>
            The outputFormat attribute is used to specify the output
            format that the Web Feature Service should generate in
            response to a GetFeature or GetFeatureWithLock element.
            The default value of GML2 indicates that the output is an
            XML document that conforms to the Geography Markup Language
```

```
            (GML) Implementation Specification V2.0.

            Other values may be used to specify other formats as long
            as those values are advertised in the capabilities document.
            For example, the value WKB may be used to indicate that a
            Well Known Binary format be used to encode the output.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:attribute>
   <xsd:attribute name="maxFeatures" type="xsd:positiveInteger"
                  use="optional">
      <xsd:annotation>
         <xsd:documentation>
            The maxFeatures attribute is used to specify the maximum
            number of features that a GetFeature operation should
            generate (regardless of the actual number of query hits).
         </xsd:documentation>
      </xsd:annotation>
   </xsd:attribute>
</xsd:complexType>
<xsd:element name="Query" type="wfs:QueryType">
   <xsd:annotation>
      <xsd:documentation>
         The Query element is used to describe a single query.
         One or more Query elements can be specified inside a
         GetFeature element so that multiple queries can be
         executed in one request.  The output from the various
         queries are combined in a wfs:FeatureCollection element
         to form the response to the request.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:complexType name="QueryType">
   <xsd:annotation>
      <xsd:documentation>
         The Query element is of type QueryType.
      </xsd:documentation>
   </xsd:annotation>
  <xsd:sequence>
    <xsd:element ref="ogc:PropertyName" minOccurs="0" maxOccurs="unbounded">
       <xsd:annotation>
          <xsd:documentation>
             The PropertyName element is used to specify one or more
             properties of a feature whose values are to be retrieved
             by a Web Feature Service.

             While a Web Feature Service should endeavour to satisfy
             the exact request specified, in some instance this may
             not be possible.  Specifically, a Web Feature Service
             must generate a valid GML2 response to a Query operation.
             The schema used to generate the output may include
             properties that are mandatory.  In order that the output
             validates, these mandatory properties must be specified
             in the request.  If they are not, a Web Feature Service
             may add them automatically to the Query before processing
             it.  Thus a client application should, in general, be
             prepared to receive more properties than it requested.

             Of course, using the DescribeFeatureType request, a client
             application can determine which properties are mandatory
             and request them in the first place.
          </xsd:documentation>
       </xsd:annotation>
    </xsd:element>
    <xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1">
       <xsd:annotation>
          <xsd:documentation>
             The Filter element is used to define spatial and/or non-spatial
             constraints on query.  Spatial constrains use GML2 to specify
             the constraining geometry.  A full description of the Filter
             element can be found in the Filter Encoding Implementation
             Specification.
          </xsd:documentation>
       </xsd:annotation>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="handle"
                 type="xsd:string" use="optional"/>
  <xsd:attribute name="typeName"
```

```
                  type="xsd:QName" use="required"/>
    <xsd:attribute name="featureVersion"
                  type="xsd:string" use="optional">
       <xsd:annotation>
          <xsd:documentation>
             For systems that implement versioning, the featureVersion
             attribute is used to specify which version of a particular
             feature instance is to be retrieved.  A value of ALL means
             that all versions should be retrieved.  An integer value
             'i', means that the ith version should be retrieve if it
             exists or the most recent version otherwise.
          </xsd:documentation>
       </xsd:annotation>
    </xsd:attribute>
  </xsd:complexType>
  <!-- RESPONSE TYPE -->
  <xsd:complexType name="FeatureCollectionType">
     <xsd:annotation>
        <xsd:documentation>
           This type defines a container for the response to a
           GetFeature or GetFeatureWithLock request.  If the
           request is GetFeatureWithLock, the lockId attribute
           must be populated.  The lockId attribute can otherwise
           be safely ignored.
        </xsd:documentation>
     </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="gml:AbstractFeatureCollectionType">
        <xsd:attribute name="lockId" type="xsd:string" use="optional">
           <xsd:annotation>
              <xsd:documentation>
                 The value of the lockId attribute is an identifier
                 that a Web Feature Service generates and which a
                 client application can use in subsequent operations
                 (such as a Transaction request) to reference the set
                 of locked features.
              </xsd:documentation>
           </xsd:annotation>
        </xsd:attribute>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

A.4 WFS-transaction.xsd

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

   <!-- ================================================================
        Includes and Imports
        ================================================================ -->
   <xsd:include schemaLocation="WFS-basic.xsd"/>
   <xsd:import namespace="http://www.opengis.net/gml"
               schemaLocation="../../gml/2.1/feature.xsd"/>
   <xsd:import namespace="http://www.opengis.net/ogc"
               schemaLocation="../../filter/1.0.0/filter.xsd"/>

   <!-- ================================================================
         REQUEST MESSAGES
         ================================================================ -->
   <xsd:element name="GetFeatureWithLock" type="wfs:GetFeatureWithLockType">
      <xsd:annotation>
         <xsd:documentation>
            This is the root element for the GetFeatureWithLock request.
            The GetFeatureWithLock operation performs identically to a
            GetFeature request except that the GetFeatureWithLock request
            locks all the feature instances in the result set and returns
            a lock identifier to a client application in the response.
         </xsd:documentation>
      </xsd:annotation>
```

```
</xsd:element>
<xsd:element name="LockFeature" type="wfs:LockFeatureType">
   <xsd:annotation>
      <xsd:documentation>
         This is the root element for a LockFeature request.
         The LockFeature request can be used to lock one or
         more feature instances.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:element name="Transaction" type="wfs:TransactionType">
   <xsd:annotation>
      <xsd:documentation>
         This is the root element for a Transaction request.
         A transaction request allows insert, update and
         delete operations to be performed to create, change
         or remove feature instances.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>

<!-- ============================================================
      RESPONSE MESSAGES
      ============================================================ -->
<xsd:element name="WFS_LockFeatureResponse"
             type="wfs:WFS_LockFeatureResponseType">
   <xsd:annotation>
      <xsd:documentation>
         The WFS_LockFeatureResponse element contains a report
         about the completion status of a LockFeature request.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:element name="WFS_TransactionResponse"
             type="wfs:WFS_TransactionResponseType">
   <xsd:annotation>
      <xsd:documentation>
         The WFS_TransactionResponse element contains a report
         about the completion status of a Transaction operation.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>

<!-- ============================================================
      COMMON ATTRIBUTE DOCUMENTATION
      ============================================================
      EXPIRY:
         The expiry attribute value is specified in minutes. It
         indicates how long a Web Feature Service should wait to
         receive a request from the client application that locked
         the feature instances.  If the specified time elapses and
         no request has been received by a Web Feature Service that
         references the lockId given to the client application, then
         the locks maintained by the Web Feature Service shall be
         released and the lockId that references the locked features
         shall now be invalid.  If the expiry attribute is not specified,
         then the feature instances shall be locked indefinitely and the
         intervention of an administrator may be required to release
         the locks. -->

<!-- ============================================================
      TYPES
      ============================================================ -->
<!-- GETFEATUREWITHLOCK -->
<xsd:complexType name="GetFeatureWithLockType">
   <xsd:annotation>
      <xsd:documentation>
         A GetFeatureWithLock request operates identically to a
         GetFeature request expect that it attempts to lock the
         feature instances in the result set and includes a lock
         identifier in its response to a client.  A lock identifier
         is an identifier generated by a Web Feature Service that
         a client application can use, in subsequent operations,
         to reference the locked set of feature instances.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:sequence>
      <xsd:element ref="wfs:Query" maxOccurs="unbounded"/>
   </xsd:sequence>
```

```
   <xsd:attribute name="version"
                  type="xsd:string" use="required" fixed="1.0.0"/>
   <xsd:attribute name="service"
                  type="xsd:string" use="required" fixed="WFS"/>
   <xsd:attribute name="handle"
                  type="xsd:string" use="optional"/>
   <xsd:attribute name="expiry"
                  type="xsd:positiveInteger" use="optional"/>
   <xsd:attribute name="outputFormat"
                  type="xsd:string" use="optional" default="GML2"/>
   <xsd:attribute name="maxFeatures"
                  type="xsd:positiveInteger" use="optional"/>
</xsd:complexType>
<!-- LOCKFEATURE -->
<xsd:complexType name="LockFeatureType">
   <xsd:annotation>
      <xsd:documentation>
         This type defines the LockFeature operation.  The LockFeature
         element contains one or more Lock elements that define
         which features of a particular type should be locked.  A lock
         identifier (lockId) is returned to the client application which
         can be used by subsequent operations to reference the locked
         features.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:sequence>
      <xsd:element name="Lock" type="wfs:LockType" maxOccurs="unbounded">
         <xsd:annotation>
            <xsd:documentation>
               The lock element is used to indicate which feature
               instances of particular type are to be locked.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
   </xsd:sequence>
   <xsd:attribute name="version"
                  type="xsd:string" use="required" fixed="1.0.0"/>
   <xsd:attribute name="service"
                  type="xsd:string" use="required" fixed="WFS"/>
   <xsd:attribute name="expiry"
                  type="xsd:positiveInteger" use="optional"/>
   <xsd:attribute name="lockAction"
                  type="wfs:AllSomeType" use="optional">
      <xsd:annotation>
         <xsd:documentation>
            The lockAction attribute is used to indicate what
            a Web Feature Service should do when it encounters
            a feature instance that has already been locked by
            another client application.

            Valid values are ALL or SOME.

            ALL means that the Web Feature Service must acquire
            locks on all the requested feature instances.  If it
            cannot acquire those locks then the request should
            fail.  In this instance, all locks acquired by the
            operation should be released.

            SOME means that the Web Feature Service should lock
            as many of the requested features as it can.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:attribute>
</xsd:complexType>
<xsd:complexType name="LockType">
   <xsd:annotation>
      <xsd:documentation>
         This type defines the Lock element.  The Lock element
         defines a locking operation on feature instances of
         a single type. An OGC Filter is used to constrain the
         scope of the operation.  Features to be locked can be
         identified individually by using their feature identifier
         or they can be locked by satisfying the spatial and
         non-spatial constraints defined in the filter.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:sequence>
      <xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
```

```
   <xsd:attribute name="handle"
                  type="xsd:string" use="optional"/>
   <xsd:attribute name="typeName"
                  type="xsd:QName" use="required"/>
</xsd:complexType>
<!-- TRANSACTION -->
<xsd:complexType name="TransactionType">
   <xsd:annotation>
      <xsd:documentation>
         The TranactionType defines the Transaction operation.  A
         Transaction element contains one or more Insert, Update
         Delete and Native elements that allow a client application
         to create, modify or remove feature instances from the
         feature repository that a Web Feature Service controls.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:sequence>
      <xsd:element ref="wfs:LockId" minOccurs="0">
         <xsd:annotation>
            <xsd:documentation>
               In order for a client application to operate upon locked
               feature instances, the Transaction request must include
               the LockId element.  The content of this element must be
               the lock identifier the client application obtained from
               a previous GetFeatureWithLock or LockFeature operation.

               If the correct lock identifier is specified the Web
               Feature Service knows that the client application may
               operate upon the locked feature instances.

               No LockId element needs to be specified to operate upon
               unlocked features.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
         <xsd:element ref="wfs:Insert"/>
         <xsd:element ref="wfs:Update"/>
         <xsd:element ref="wfs:Delete"/>
         <xsd:element ref="wfs:Native"/>
      </xsd:choice>
   </xsd:sequence>
   <xsd:attribute name="version"
                  type="xsd:string" use="required" fixed="1.0.0"/>
   <xsd:attribute name="service"
                  type="xsd:string" use="required" fixed="WFS"/>
   <xsd:attribute name="handle"
                  type="xsd:string" use="optional"/>
   <xsd:attribute name="releaseAction"
                  type="wfs:AllSomeType" use="optional">
      <xsd:annotation>
         <xsd:documentation>
            The releaseAction attribute is used to control how a Web
            Feature service releases locks on feature instances after
            a Transaction request has been processed.

            Valid values are ALL or SOME.

            A value of ALL means that the Web Feature Service should
            release the locks of all feature instances locked with the
            specified lockId, regardless or whether or not the features
            were actually modified.

            A value of SOME means that the Web Feature Service will
            only release the locks held on feature instances that
            were actually operated upon by the transaction.  The lockId
            that the client application obtained shall remain valid and
            the other, unmodified, feature instances shall remain locked.
            If the expiry attribute was specified in the original operation
            that locked the feature instances, then the expiry counter
            will be reset to give the client application that same amount
            of time to post subsequent transactions against the locked
            features.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:attribute>
</xsd:complexType>
<xsd:element name="LockId" type="xsd:string">
   <xsd:annotation>
```

```
      <xsd:documentation>
         The LockId element contains the value of the lock identifier
         obtained by a client application from a previous GetFeatureWithLock
         or LockFeature request.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:element name="Insert" type="wfs:InsertElementType">
   <xsd:annotation>
      <xsd:documentation>
         The Insert element is used to indicate that the Web Feature
         Service should create a new instance of a feature type.  The
         feature instance is specified using GML2 and one or more
         feature instances to be created can be contained inside the
         Insert element.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:complexType name="InsertElementType">
   <xsd:sequence>
      <xsd:element ref="gml:_Feature" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="handle" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:element name="Update" type="wfs:UpdateElementType">
   <xsd:annotation>
      <xsd:documentation>
         One or more existing feature instances can be changed by
         using the Update element.  Changing a feature instance
         means that the current value of one or more properties of
         the feature are replaced with new values.  The Update
         element contains  one or more Property elements.  A
         Property element contains the name or a feature property
         who's value is to be changed and the replacement value
         for that property.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:complexType name="UpdateElementType">
   <xsd:sequence>
      <xsd:element ref="wfs:Property" maxOccurs="unbounded" />
      <xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1">
         <xsd:annotation>
            <xsd:documentation>
               The Filter element is used to constrain the scope
               of the update operation to those features identified
               by the filter.  Feature instances can be specified
               explicitly and individually using the identifier of
               each feature instance OR a set of features to be
               operated on can be identified by specifying spatial
               and non-spatial constraints in the filter.
               If no filter is specified, then the update operation
               applies to all feature instances.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
   </xsd:sequence>
   <xsd:attribute name="handle" type="xsd:string" use="optional"/>
   <xsd:attribute name="typeName" type="xsd:QName" use="required"/>
</xsd:complexType>
<xsd:element name="Delete" type="wfs:DeleteElementType">
   <xsd:annotation>
      <xsd:documentation>
         The Delete element is used to indicate that one or more
         feature instances should be removed from the feature
         repository.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:complexType name="DeleteElementType">
   <xsd:sequence>
      <xsd:element ref="ogc:Filter" minOccurs="1" maxOccurs="1">
         <xsd:annotation>
            <xsd:documentation>
               The Filter element is used to constrain the scope
               of the delete operation to those features identified
               by the filter.  Feature instances can be specified
               explicitly and individually using the identifier of
               each feature instance OR a set of features to be
```

```
                operated on can be identified by specifying spatial
                and non-spatial constraints in the filter.
                If no filter is specified then an exception should
                be raised since it is unlikely that a client application
                intends to delete all feature instances.
             </xsd:documentation>
          </xsd:annotation>
       </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="handle" type="xsd:string" use="optional"/>
    <xsd:attribute name="typeName" type="xsd:QName" use="required"/>
 </xsd:complexType>
 <xsd:element name="Native" type="wfs:NativeType">
    <xsd:annotation>
       <xsd:documentation>
          Many times, a Web Feature Service interacts with a repository
          that may have special vendor specific capabilities.  The native
          element allows vendor specific command to be passed to the
          repository via the Web Feature Service.
       </xsd:documentation>
    </xsd:annotation>
 </xsd:element>
 <xsd:complexType name="NativeType">
    <xsd:attribute name="vendorId" type="xsd:string" use="required">
       <xsd:annotation>
          <xsd:documentation>
             The vendorId attribute is used to specify the name of
             vendor who's vendor specific command the client
             application wishes to execute.
          </xsd:documentation>
       </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="safeToIgnore" type="xsd:boolean" use="required">
       <xsd:annotation>
          <xsd:documentation>
             In the event that a Web Feature Service does not recognize
             the vendorId or does not recognize the vendor specific command,
             the safeToIgnore attribute is used to indicate whether the
             exception can be safely ignored.  A value of TRUE means that
             the Web Feature Service may ignore the command.  A value of
             FALSE means that a Web Feature Service cannot ignore the
             command and an exception should be raised if a problem is
             encountered.
          </xsd:documentation>
       </xsd:annotation>
    </xsd:attribute>
 </xsd:complexType>

 <!-- define structure to specify a property value -->
 <xsd:element name="Property" type="wfs:PropertyType">
    <xsd:annotation>
       <xsd:documentation>
          The Property element is used to specify the new
          value of a feature property inside an Update element.
       </xsd:documentation>
    </xsd:annotation>
 </xsd:element>
 <xsd:complexType name="PropertyType">
    <xsd:sequence>
       <xsd:element name="Name" type="xsd:string">
          <xsd:annotation>
             <xsd:documentation>
                The Name element contains the name of a feature property
                to be updated.
             </xsd:documentation>
          </xsd:annotation>
       </xsd:element>
       <xsd:element name="Value" minOccurs="0">
          <xsd:annotation>
             <xsd:documentation>
                The Value element contains the replacement value for the
                named property.
             </xsd:documentation>
          </xsd:annotation>
       </xsd:element>
    </xsd:sequence>
 </xsd:complexType>
 <!-- RESPONSE TYPES -->
 <xsd:complexType name="WFS_LockFeatureResponseType">
```

```
    <xsd:annotation>
       <xsd:documentation>
          The WFS_LockFeatureResponseType is used to define an
          element to contains the response to a LockFeature
          operation.
       </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
       <xsd:element ref="wfs:LockId">
          <xsd:annotation>
             <xsd:documentation>
                The WFS_LockFeatureResponse includes a LockId element
                that contains a lock identifier.  The lock identifier
                can be used by a client, in subsequent operations, to
                operate upon the locked feature instances.
             </xsd:documentation>
          </xsd:annotation>
       </xsd:element>
       <xsd:element name="FeaturesLocked"
                    type="wfs:FeaturesLockedType" minOccurs="0">
          <xsd:annotation>
             <xsd:documentation>
                The LockFeature or GetFeatureWithLock operations
                identify and attempt to lock a set of feature
                instances that satisfy the constraints specified
                in the request.  In the event that the lockAction
                attribute (on the LockFeature or GetFeatureWithLock
                elements) is set to SOME, a Web Feature Service will
                attempt to lock as many of the feature instances from
                the result set as possible.

                The FeaturesLocked element contains list of ogc:FeatureId
                elements enumerating the feature instances that a WFS
                actually managed to lock.
             </xsd:documentation>
          </xsd:annotation>
       </xsd:element>
       <xsd:element name="FeaturesNotLocked"
                    type="wfs:FeaturesNotLockedType" minOccurs="0">
          <xsd:annotation>
             <xsd:documentation>
                In contrast to the FeaturesLocked element, the
                FeaturesNotLocked element contains a list of
                ogc:Filter elements identifying feature instances
                that a WFS did not manage to lock because they were
                already locked by another process.
             </xsd:documentation>
          </xsd:annotation>
       </xsd:element>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="FeaturesLockedType">
  <xsd:sequence maxOccurs="unbounded">
    <xsd:element ref="ogc:FeatureId"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="FeaturesNotLockedType">
  <xsd:sequence maxOccurs="unbounded">
    <xsd:element ref="ogc:FeatureId"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="WFS_TransactionResponseType">
    <xsd:annotation>
       <xsd:documentation>
          The WFS_TransactionResponseType defines the format of
          the XML document that a Web Feature Service generates
          in response to a Transaction request.  The response
          includes the completion status of the transaction
          and the feature identifiers of any newly created
          feature instances.
       </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
       <xsd:element name="InsertResult"
                    type="wfs:InsertResultType"
                    minOccurs="0" maxOccurs="unbounded">
          <xsd:annotation>
             <xsd:documentation>
                The InsertResult element contains a list of ogc:FeatureId
```

```
                elements that identify any newly created feature instances.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
      <xsd:element name="TransactionResult"
                   type="wfs:TransactionResultType">
         <xsd:annotation>
            <xsd:documentation>
               The TransactionResult element contains a Status element
               indicating the completion status of a transaction.  In
               the event that the transaction fails, additional element
               may be included to help locate which part of the transaction
               failed and why.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
   </xsd:sequence>
   <xsd:attribute name="version"
                  type="xsd:string" use="required" fixed="1.0.0"/>
</xsd:complexType>
<xsd:complexType name="TransactionResultType">
   <xsd:sequence>
      <xsd:element name="Status" type="wfs:StatusType">
         <xsd:annotation>
            <xsd:documentation>
               The Status element contains an element indicating the
               completion status of a transaction.  The SUCCESS element
               is used to indicate successful completion.  The FAILED
               element is used to indicate that an exception was
               encountered.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
      <xsd:element name="Locator" type="xsd:string" minOccurs="0">
         <xsd:annotation>
            <xsd:documentation>
               In the event that an exception was encountered while
               processing a transaction, a Web Feature Service may
               use the Locator element to try and identify the part
               of the transaction that failed.  If the element(s)
               contained in a Transaction element included a handle
               attribute, then a Web Feature Service may report the
               handle to identify the offending element.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
      <xsd:element name="Message" type="xsd:string" minOccurs="0">
         <xsd:annotation>
            <xsd:documentation>
               The Message element may contain an exception report
               generated by a Web Feature Service when an exception
               is encountered.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
   </xsd:sequence>
   <xsd:attribute name="handle" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:complexType name="InsertResultType">
   <xsd:sequence>
      <xsd:element ref="ogc:FeatureId" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="handle" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:complexType name="StatusType">
   <xsd:choice>
      <xsd:element ref="wfs:SUCCESS"/>
      <xsd:element ref="wfs:FAILED"/>
      <xsd:element ref="wfs:PARTIAL"/>
   </xsd:choice>
</xsd:complexType>
<xsd:element name="SUCCESS" type="wfs:EmptyType"/>
<xsd:element name="FAILED" type="wfs:EmptyType"/>
<xsd:element name="PARTIAL" type="wfs:EmptyType"/>
<!-- MISC TYPES -->
<xsd:complexType name="EmptyType"/>
<xsd:simpleType name="AllSomeType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="ALL"/>
```

```
         <xsd:enumeration value="SOME"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:schema>
```

A.5 WFS-capabilities.xsd

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

   <!-- Comments in this document may impose additional constraints
        beyond those codified in the schema syntax.  A conformant
        Web Feature Server must provide Capabilities XML that
           (1) validates against this schema
           (2) does not violate the constraints stated in
               comments herein. -->

   <!-- ============================================================
        Imports ...
        ============================================================ -->
   <xsd:import namespace="http://www.opengis.net/ogc"
               schemaLocation="../../filter/1.0.0/filterCapabilities.xsd" />

   <!-- ============================================================
        Global elements and attributes
        ============================================================ -->
   <!-- A descriptive narrative for more
        information about this server. -->
   <xsd:element name="Abstract" type="xsd:string"/>
   <!-- Elements containing text blocks indicating what
        fees or access constraints are imposed by the
        service provider on the service or data retrieved
        from the server. The reserved keyword "NONE"
        indicates no constraint exists. -->
   <xsd:element name="AccessConstraints" type="xsd:string"/>
   <xsd:element name="Fees" type="xsd:string"/>
   <!-- Short words to help catalog searching.
        Currently, no controlled vocabulary has
        been defined. -->
   <xsd:element name="Keywords" type="xsd:string"/>
   <!-- The top-level HTTP URL of this service.
        Typically the URL of a "home page" for
        the service.  See also the onlineResource
        attributes of <DCPType> children, below.
        Currently, no non-HTTP platforms have been
        specified. -->
   <xsd:element name="OnlineResource"/>
   <xsd:element name="SRS" type="xsd:string"/>
   <!-- A human-readable title to briefly identify
        this server in menus. -->
   <xsd:element name="Title" type="xsd:string"/>

   <xsd:element name="Query"  type="wfs:EmptyType"/>
   <xsd:element name="Insert" type="wfs:EmptyType"/>
   <xsd:element name="Update" type="wfs:EmptyType"/>
   <xsd:element name="Delete" type="wfs:EmptyType"/>
   <xsd:element name="Lock"   type="wfs:EmptyType"/>

   <!-- REDEFINE THIS ELEMENT AS NEEDED IN YOUR XML  -->
   <xsd:element name="VendorSpecificCapabilities" type="xsd:string"/>

   <!-- ============================================================
        Root element
        ============================================================ -->
   <!-- The parent element of the Capabilities document includes as
        children a Service element with general information about the
        server, a Capability element with specific information about
        the kinds of functionality offered by the server, a FeatureTypeList
        element defining the list of all feature types available from
        this server and a FeatureCapabilities element describing the
        filter capabilities of the server. -->
```

```
<xsd:element name="WFS_Capabilities" type="wfs:WFS_CapabilitiesType"/>

<!-- ================================================================
     Types
     ================================================================ -->
<xsd:complexType name="WFS_CapabilitiesType">
   <xsd:sequence>
      <!-- The Service element provides metadata for
           the service as a whole. -->
      <xsd:element name="Service" type="wfs:ServiceType"/>

      <!-- A Capability lists available request
           types, how exceptions may be reported, and
           whether any vendor-specific capabilities
           are defined.  It also lists all the
           feature types available from this feature
           server. -->
      <xsd:element name="Capability" type="wfs:CapabilityType"/>
      <xsd:element name="FeatureTypeList" type="wfs:FeatureTypeListType"/>
      <xsd:element ref="ogc:Filter_Capabilities" />
   </xsd:sequence>

   <!-- The version attribute specifies the specification revision
        to which this schema applies.  Its format is one,t two or three
        integers separated by periods: "x", or "x.y", or "x.y.z",
        with the most significant number appearing first.  Future
        revisions are guaranteed to be numbered in monotonically
        increasing fashion, though gaps may appear in the sequence. -->
   <xsd:attribute name="version"
                  type="xsd:string" fixed="1.0.0"/>

   <!-- The updateSequence attribute is a sequence number for
        managing propagation of the contents of this document.
        For example, if a Feature Server adds some data feature
        types it can increment the update sequence to inform
        catalog servers that their previously cached versions
        are now stale. The format is a positive integer. -->
   <xsd:attribute name="updateSequence"
                  type="xsd:nonNegativeInteger" default="0"/>
</xsd:complexType>

<xsd:complexType name="ServiceType">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element ref="wfs:Title"/>
      <xsd:element ref="wfs:Abstract" minOccurs="0"/>
      <xsd:element ref="wfs:Keywords" minOccurs="0"/>
      <xsd:element ref="wfs:OnlineResource"/>
      <xsd:element ref="wfs:Fees" minOccurs="0"/>
      <xsd:element ref="wfs:AccessConstraints" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CapabilityType">
   <xsd:sequence>
      <xsd:element name="Request" type="wfs:RequestType"/>
      <!-- The optional VendorSpecificCapabilities element lists any
           capabilities unique to a particular server.  Because the
           information is not known a priori, it cannot be constrained
           by this particular schema document.  A vendor-specific schema
           fragment must be supplied at the start of the XML capabilities
           document, after the reference to the general WFS_Capabilities
           schema. -->
      <xsd:element ref="wfs:VendorSpecificCapabilities" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="FeatureTypeListType">
   <xsd:sequence>
      <xsd:element name="Operations"
                   type="wfs:OperationsType" minOccurs="0"/>
      <xsd:element name="FeatureType"
                   type="wfs:FeatureTypeType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<!-- Available WFS-defined request types are listed here. At
     least one of the values is required, but more than one
     may be given. -->
<xsd:complexType name="RequestType">
```

```
   <xsd:choice maxOccurs="unbounded">
      <xsd:element name="GetCapabilities"
                   type="wfs:GetCapabilitiesType"/>
      <xsd:element name="DescribeFeatureType"
                   type="wfs:DescribeFeatureTypeType"/>
      <xsd:element name="Transaction"
                   type="wfs:TransactionType"/>
      <xsd:element name="GetFeature"
                   type="wfs:GetFeatureTypeType"/>
      <xsd:element name="GetFeatureWithLock"
                   type="wfs:GetFeatureTypeType"/>
      <xsd:element name="LockFeature"
                   type="wfs:LockFeatureTypeType"/>
   </xsd:choice>
</xsd:complexType>
<xsd:complexType name="GetCapabilitiesType">
   <xsd:sequence>
      <xsd:element name="DCPType"
                   type="wfs:DCPTypeType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="DescribeFeatureTypeType">
   <xsd:sequence>
      <xsd:element name="SchemaDescriptionLanguage"
                   type="wfs:SchemaDescriptionLanguageType"/>
      <xsd:element name="DCPType"
                   type="wfs:DCPTypeType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TransactionType">
   <xsd:sequence>
      <xsd:element name="DCPType"
                   type="wfs:DCPTypeType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="GetFeatureTypeType">
   <xsd:sequence>
      <xsd:element name="ResultFormat"
                   type="wfs:ResultFormatType"/>
      <xsd:element name="DCPType"
                   type="wfs:DCPTypeType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="LockFeatureTypeType">
   <xsd:sequence>
      <xsd:element name="DCPType"
                   type="wfs:DCPTypeType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<!-- Available Distributed Computing Platforms (DCPs) are
     listed here.  At present, only HTTP is defined. -->
<xsd:complexType name="DCPTypeType">
   <xsd:sequence>
      <xsd:element name="HTTP" type="wfs:HTTPType"/>
   </xsd:sequence>
</xsd:complexType>

<!-- A list of feature types available from
     this server.  The following table
     specifies the number and source of the
     various elements that are available for
     describing a feature type.

     element             number    comments
     =================   =======   ======================================
     Name                    1     this is the Name of the feature type

     Title                 0/1     an optional Meaningful title for the
                                   feature type  (e.g. "Ontario Roads"
                                   for ROADL_1M")

     Abstract              0/1     optional; no Default

     Keywords              0/1     optional; no Default

     SRS                     1     the SRS that should be used
                                   when specifying the state of
                                   the feature
```

```
     Operations            0/1     a list of available operations for
                                   the feature type

     LatLongBoundingBox      1+    bounding box(s) of data

     MetadataURL           0/1+    optional; no default
-->
<xsd:complexType name="FeatureTypeType">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:QName"/>
      <xsd:element ref="wfs:Title" minOccurs="0"/>
      <xsd:element ref="wfs:Abstract" minOccurs="0"/>
      <xsd:element ref="wfs:Keywords" minOccurs="0"/>
      <xsd:element ref="wfs:SRS"/>
      <xsd:element name="Operations"
                   type="wfs:OperationsType" minOccurs="0"/>
      <xsd:element name="LatLongBoundingBox"
                   type="wfs:LatLongBoundingBoxType"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="MetadataURL"
                   type="wfs:MetadataURLType"
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="GetType">
   <xsd:attribute name="onlineResource" type="xsd:string" use="required"/>
</xsd:complexType>

<!-- Available HTTP request methods. -->
<xsd:complexType name="HTTPType">
   <xsd:choice maxOccurs="unbounded">
      <!-- HTTP request methods.  The onlineResourc attribute
           indicates the URL prefix for HTTP GET requests
           (everything before the question mark and query string:
           http://hostname[:port]/path/scriptname); for HTTP POST
           requests, onlineResource is the complete URL. -->
      <xsd:element name="Get" type="wfs:GetType"/>
      <xsd:element name="Post" type="wfs:PostType"/>
   </xsd:choice>
</xsd:complexType>

<!-- The LatLongBoundingBox element is used to indicate the edges of
     an enclosing rectangle in the SRS of the associated feature type.
     Its purpose is to facilitate geographic searches  by indicating
     where instances of the particular feature type exist.  Since multiple
     LatLongBoundingBoxes can be specified, a WFS can indicate where
     various clusters of data may exist.  This knowledge aids client
     applications by letting them know where they should query in order
     to have a high probability of finding data. -->
<xsd:complexType name="LatLongBoundingBoxType">
   <xsd:attribute name="minx" type="xsd:string" use="required"/>
   <xsd:attribute name="miny" type="xsd:string" use="required"/>
   <xsd:attribute name="maxx" type="xsd:string" use="required"/>
   <xsd:attribute name="maxy" type="xsd:string" use="required"/>
</xsd:complexType>

<!-- A Web Feature Server MAY use zero or more MetadataURL
     elements to offer detailed, standardized metadata about
     the data underneath a particular feature type.  The type
     attribute indicates the standard to which the metadata
     complies; the format attribute indicates how the metadata is
     structured.  Two types are defined at present:
     'TC211' = ISO TC211 19115;
     'FGDC'  = FGDC CSDGM. -->
<xsd:complexType name="MetadataURLType">
   <xsd:simpleContent>
      <xsd:extension base="xsd:string">
         <xsd:attribute name="type" use="required">
            <xsd:simpleType>
               <xsd:restriction base="xsd:NMTOKEN">
                  <xsd:enumeration value="TC211"/>
                  <xsd:enumeration value="FGDC"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:attribute>
         <xsd:attribute name="format" use="required">
            <xsd:simpleType>
               <xsd:restriction base="xsd:NMTOKEN">
```

```
                     <xsd:enumeration value="XML"/>
                     <xsd:enumeration value="SGML"/>
                     <xsd:enumeration value="TXT"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:attribute>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <xsd:complexType name="OperationsType">
      <xsd:choice maxOccurs="unbounded">
         <xsd:element ref="wfs:Insert"/>
         <xsd:element ref="wfs:Update"/>
         <xsd:element ref="wfs:Delete"/>
         <xsd:element ref="wfs:Query"/>
         <xsd:element ref="wfs:Lock"/>
      </xsd:choice>
   </xsd:complexType>
   <xsd:complexType name="PostType">
      <xsd:attribute name="onlineResource" type="xsd:string" use="required"/>
   </xsd:complexType>

   <!-- The ResultFormatType type defines the output formats that the
        web feature server can generate.  The mandatory format "GML2"
        must always be available.  Individual servers can define
        additional elements representing other output formats. -->
   <xsd:element name="GML2" type="wfs:EmptyType"/>
   <xsd:complexType name="ResultFormatType">
      <xsd:sequence maxOccurs="unbounded">
         <xsd:element ref="wfs:GML2"/>
      </xsd:sequence>
   </xsd:complexType>

   <!-- The SchemaDescriptionLanguageType type defines the schema languages
        that a feature server is capable of using to describe the schema
        of a feature.  Individual servers can define additional elements
        representing other schema languages but XMLSCHEMA must always
        be defined. -->
   <xsd:element name="XMLSCHEMA" type="wfs:EmptyType"/>
   <xsd:complexType name="SchemaDescriptionLanguageType">
      <xsd:sequence maxOccurs="unbounded">
         <xsd:element ref="wfs:XMLSCHEMA"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="EmptyType" />
</xsd:schema>
```

Annex B - Conformance tests (normative)

Specific conformance tests for Web Feature Services have not yet been determined and will be added in a future revision of this specification.

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

1. The GetCapabilities, DescribeFeatureType and GetFeature operations must be supported.

2. The Extensible Markup Language (XML) document returned in response to a GetCapabilities request must be valid against the XML Schema definition in Annex A.4. Such validation may be performed using commonly available XML validating tools.

3. In response to a GetFeature operation, the WFS must be able to generate GML [2] as one of its output formats.

4. The Extensible Markup Language (XML) document returned in response to a GetFeature request must validate against the schema generated in response to a DescribeFeatureType request. Such validation may be performed using commonly available XML validating tools.

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

XML File
An XML file can be an actual operating system file or any valid XML stream.

2-PHASE COMMIT
A 2-phase commit protocol ensures that execution of data transactions are synchronized, either all committed or all rolled back to each of the distributed databases.

DTD
Document Type Definition; a description of the schema of an XML document.

XML-Schema
A schema description language, similar to DTD, based on XML itself. Like a DTD it can be used to describe the schema of an XML document, but it is flexible enough to be able to describe other structures as well. Unlike a DTD, XML-Schema include type information. XML-Schema is defined at http://www.w3c.org/TR/xmlschema-1/.

XPath
An XPath expression is a path expression, similar to the path expressions used to identify operating system files, that is used to reference elements in an XML document. XPath expressions are defined at http://www.w3c.org/TR/Xpath/.

Bibliography

[11] de La Beaujardière, Jeff (ed.), "OpenGIS Implementation Specification #01-047r2: Web Map Service Implementation Specification", June 2001

[12] Vretanos, Panagiotis, "OpenGIS Discussion Paper: Transaction Encoding Specification Version 0.0.5", March 2000

[13] Arctur, D., Pilkington, P., Cuthbert, A., "Spatial Object Transfer Format (SOTF): Initial High-Level Design, Version 1.2", Laser-Scan Inc., November 1999

[14] Rumbach, James, et al., "Unified Modeling Language Reference Manual", 1999

[15] Murata, St. Laurent, Kohn, "XML Media Types, January 2001, http://www.ietf.org/rfc/rfc3023.txt