Open Geospatial Consortium Inc.

Date: 2016-10-26

Reference number of this OpenGIS® project document: OGC 04-094r1

Version: 1.1.3

Category: OpenGIS® Implementation Specification

Status: Adopted Specification

Editor: Panagiotis A. Vretanos

Web Feature Service Implementation Specification with Corrigendum

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

Contents

i. Preface

The Open Geospatial Consortium (OGC) has developed a member consensus that when software vendors implement their products in compliance with open geospatial web service interface and data encoding specifications, end-users benefit from a larger pool of interoperable web based tools for geodata access and related geoprocessing services.

The Web Map Server products that have been developed to implement the OGC Web Map Service Implementation Specification [1] are prime examples of such tools. The GetCapabilities and GetMap interfaces defined in that specification give users on the web an interoperable way to combine and view map images from different sources. And the GetFeatureInfo interface gives those users a way to obtain attribute information about geographic features displayed in a map with a simple mouse click.

The OGC Geography Markup Language (GML) Implementation Specification [2] describes an encoding specification for geodata in XML that enables the storage, transport, processing, and transformation of geographic information.

This document, the OGC Web Feature Service (WFS) Implementation Specification, takes the next logical step of by defining interfaces for data access and manipulation operations on geographic features using HTTP as the distributed computing platform. Via these interfaces, a web user or service can combine, use and manage geodata

-- the feature information behind a map image -- from different sources

by invoking the following WFS operations on geographic features and elements:

- Create a new feature instance
- Delete a feature instance
- Update a feature instance
- Lock a feature instance
- Get or query features based on spatial and non-spatial constraints

ii. Submitting organizations

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

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

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

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

iii. Submission contact points

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

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

Additional contributors

Rob Atkinson (Social Change Online) rob@socialchange.net.au
Craig Bruce (CubeWerx) csbruce@cubwerx.com
Simon Cox (CSIRO) Simon.Cox@csiro.au
Paul Daisey (U.S. Census) mailto: Paul.W.Daisey@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
Edric Keighan (CubeWerx) ekeighan@cubewerx.com
Martin Kyle (Galdos Systems Inc.) mkyle@galdosinc.com
Ron Lake (Galdos Systems Inc.) rlake@galdosinc.com
Jeff Lansing (Polexis) jeff@polexis.com
Seb Lessware (Laser-Scan Ltd.) sebl@lsl.co.uk
Marwa Mabrouk (ESRI) mmabrouk@esri.com
Serge Margoulies (Ionic) Serge.Margoulies@ionicsoft.com
Richard Martell (Galdos Systems Inc.) rmartell@galdosinc.com
Aleksander Milanovic
Dimitri Monie (Ionic) dimitri.monie@ionicsoft.com
Paul Pilkington (Laser-Scan Ltd.) paulpi@lsiva.com
Keith Pomakis (CubeWerx) pomakis@cubewerx.com
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
Bernard Snyers (Ionic) Bernard.Snyers@ionicsoft.com
James T. Stephens (Lockheed Martin) james.t.stephens@lmco.com
Glenn Stowe (CubeWerx) gstowe@cubwerx.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
Tim Wilson (Galdos Systems Inc.) twilson@galdosinc.com
Nami Yamashita (Dawn Corp.) yamashita@dawn-corp.co.jp

iv. Revision history

1.1.0	Revise document based on change proposals 02-063,03-012,03-033r4,03-052,03-082
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 subclause 8.3.3 of Topic 12.

vi. Future work

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

1. Determine whether WFS should optionally offer operations beyond generic get/lock/update operations on elements of GML version 3 types that are not feature types, to provide specialized capabilities for such types, for example, topological types.

2. Support an update/merge operation that allow features to be updated without having to fetch the entire feature. For example adding vertices to a geometry without having to specify the whole geometry in the update request.

3. Integrate change proposal 03-097 with this specification.

Foreword

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

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

Normative annexes

Annex A is 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 and update 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.

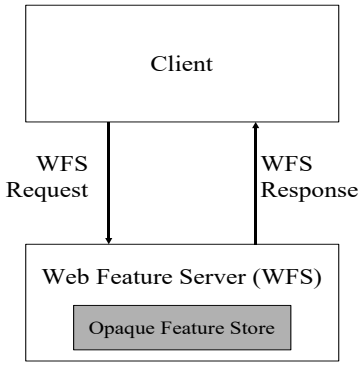

Figure 1 – Web feature service

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, LOCK, 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.opengeospatial.org/specs/?page=abstract) 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, also implies that features can be defined with no geometric properties at all.

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 or element 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.

Note that "client application" may include Registries and other middleware, as well as conventionally understood "end-users".

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.

GetGmlObject

A web feature service may be able to service a request to retrieve element instances by traversing XLinks that refer to their XML IDs. In addition, the client should be able to specify whether nested XLinks embedded in returned element data should also be retrieved.

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, three 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.

XLink WFS

An XLink WFS would support all the operations of a basic web feature service and in addition it would implement the GetGmlObject operation for local and/or remote XLinks, and offer the option for the GetGmlObject operation to be performed during GetFeature operations.

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 GetGmlObject and/or LockFeature operations.

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.

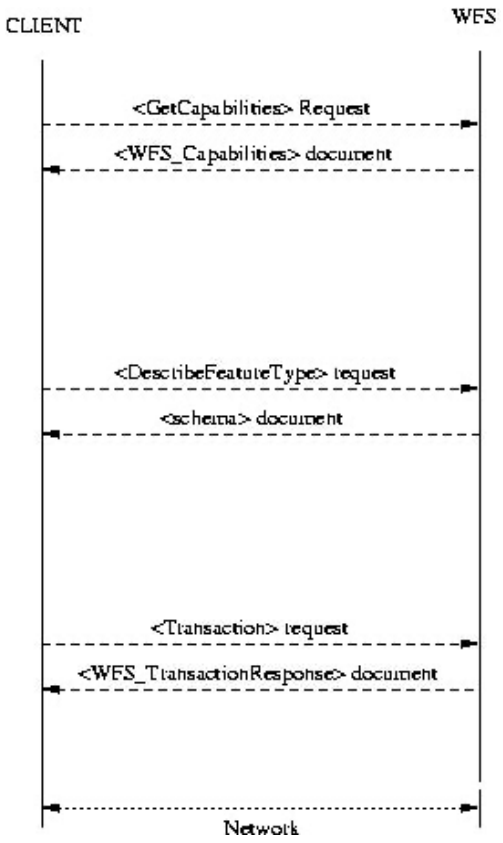

Figure 2 – Protocol diagram

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

[3] Vretanos, Panagiotis (ed.), "OpenGIS Implementation Specification #04-095: Filter Encoding Implementation Specification", Dec 2004

[4] Percivall, George, ed., OpenGIS Document 02-112, "The OpenGIS Abstract Specification, Topic 12: OpenGIS Service Architecture, Version 4.3", 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.

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

[12] 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.

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

[14] Steve DeRose, Eve Maler, David Orchard "XML Linking Language (XLink) Version 1.0" W3C Recommendation 27 June 2001, http://www.w3.org/TR/xlink/

[15] Whiteside, Arliss (ed.), "OWS Common Implementation Specification, V0.3.0", Open Geospatial Consortium Inc. document 04-016r3

[16] Rescorla et. al., "The Secure Hypertext Transfer Protocol", IETF RFC 2660, August 1999, http://www.ietf.org/rfc/rfc2660.txt.

[17] DeRose et al., "XPointer xmlns() Scheme", W3C Recommendation March 2003, http://www.w3.org/TR/xptr-xmlns/

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

4.11
XLink link
an explicit relationship between resources or portions of resources [14]

4.12
XLink linking element
an XLink-conforming XML element that asserts the existence of an XLink link [14]

4.13
locator attribute (href)
the attribute that supplies the data (a URI reference) that allows an XLink application to find a remote resource (or resource fragment) [14]

4.14
traversal
using or following an XLink link for any purpose [14]

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
HTTPS	Secure Hypertext Transfer Protocol
SSL	Secure Socket Layer
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. 13] according to the following rules.

All Capabilities XML must include a protocol version number. In response to a GetCapabilities request containing a version number, a web feature 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 an HTTP Uniform Resource Locator (URL) locates the Online Resource of each operation supported by a service instance. 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 web feature implementation service specification.

HTTP supports two request methods: GET and POST. One or both of these methods may be defined for a particular web feature service 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 [11], 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 a web feature 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.

6.3.4 HTTPS

In addition to or instead of offering web feature services using the HTTP protocol, a service provider may offer web feature services using HTTPS [16]. HTTPS is HTTP over a secure communication channel which allows encrypted information to be transferred between machines over the World Wide Web.

The use of HTTPS does not affect the description of the requests and responses described in this specification but may require additional actions to be taken on both the client and the service in order to initiate the secure communication. The description of those actions is beyond the scope of the this specification.

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 subclause 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 a web feature 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, and is intended to be used with HTTP POST method. The second encoding uses keyword-value pairs (KVP) to encode the various parameters of a request and is intended to be used with HTTP GET. 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
GetGmlObject	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].

6.5.1 Request encoding and HTTP method

The following matrix correlated WFS request encoding with each of the supported HTTP methods (GET and POST). The value in each cell defines the expected MIME type for the encoding/request method combination. The value ***Not Applicable*** means that the encoding/request method combination is supported but a MIME type is not applicable. An empty cell indicates that the combination is not supported.

Table 2b – Request encoding and transport methods

	HTTP GET METHOD	HTTP POST METHOD
XML encoded requests		text/xml
KVP encoded requests	Not applicable	application/x-www-form-urlencoded

When using the HTTP POST method, the content type for XML encoded WFS requests **must** be set to *text/xml*.

When using the HTTP POST method, the content type for KVP encoded WFS requests must be set to *application/x-www-form-urlencoded* and the content of the document must be equivalent to the query string of an HTTP GET request. That is, the content must be equivalent to the string that follows the '?' character in a URL encoded GET request. Of course, the content must be encoded [10] to protect special characters.

When using the HTTP GET method and KVP encoded WFS requests, a MIME type is not applicable as the entire request is encoded in the URL as keyword-value pairs that follow the '?' character.

The combination of XML encoded requests and the HTTP GET method is not supported.

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 in subclause 11.2.6.

6.7 Simple object access protocol (SOAP)

This subclause specifies the use of SOAP messages for communication between a web feature service client and a web feature service using the HTTP POST method.

The Simple Object Access Protocol (SOAP) is a communication protocol for communication between applications. It defines a format for sending messages between communicating applications via the Internet and specifically using HTTP. Soap is platform independent, language independent and SOAP messages are encoded using

XML. This means that SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

A SOAP message is an ordinary XML document containing the following elements:

a) A required Envelope element that identifies the XML document as a SOAP message

b) An optional Header element that contains header information

c) A required Body element that contains call and response information

d) An optional Fault element that provides information about errors that occurred while processing the message

All the elements above are declared in the default namespace for the SOAP envelope:

```
http://www.w3.org/2003/05/soap-envelope
```

and the default namespace for SOAP encoding and data types is:

```
http://www.w3.org/2003/05/soap-encoding
```

The SOAP specification defines a number of syntax rules. Among the most important are:

e) A SOAP message shall be encoded using XML

f) A SOAP message shall use the SOAP Envelope namespace

g) A SOAP message shall use the SOAP Encoding namespace

h) A SOAP message shall not contain a DTD reference

i) A SOAP message shall not contain XML Processing Instructions

The following XML fragment illustrates a skeleton SOAP message:

```
<?xml version="1.0"?>
<soap:Envelope
   xmlns:soap=http://www.w3.org/2003/05/soap-envelope
   soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
   <soap:Header>
      ...
      ...
   </soap:Header>
   <soap:Body>
      ...
      ...
      <soap:Fault>
         ...
         ...
      </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

A client may send WFS requests to a compatible server using the body of a SOAP envelope. The client simply encodes the WFS request as the content of the <**soap:Body**> element in the request message.

The WFS may then response by generating a SOAP message where the response to the client's request is the content of the **<soap:Body>** element.

In the event the an exception is encountered while processing a WFS request encoded in a SOAP envelope, the web feature service must generate a SOAP response message where the content of the **<soap:Body>** element is a **<soap:Fault>** element. The following skeleton XML fragment must be used when generating the **<soap:Body>** element in the event that a WFS encounters an exception:

```
<soap:Body>
   <soap:Fault>
      <soap:faultcode>soap:Server</soap:faultode>
      <soap:faultstring>A server exception was encountered.</soap:faultstring>
      <soap:faultactor>http://www.url_of_WFS_server.com/</soap:faultactor>
      <soap:detail>
         <ows:ExceptionReport>
            …
         </ows:ExpetionReport>
      </soap:detail>
   </soap:Fault>
</soap:Body>
```

The **<soap:faultcode>** element must have the content *soap:Server* indicating that this is a server exception. The **<soap:faultstring>** element must have the content "Server exception was encountered.". This fixed string is used since the details of the exception will be specified in the **<soap:detail>** element using an **<ows:ExceptionRecport>** element as defined in document [15].

The **<soap:detail>** element must contain an **<ows:ExceptionReport>** element detailing the specific exception that the server encountered.

The use of the **<soap:Header>** element is not discussed in this version of this specification.

7 Common elements

7.1 Feature and Element Identifiers

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.

This document further assumes that if a particular WFS implementation supports the GetGmlObject operation, each element that the WFS can operate on is uniquely identifiable. That is to say, when a WFS implementation reports an element identifier for an element, that identifier is unique to the server and can be user to reference the same element (assuming it has not been deleted). Element identifiers are encoded as described in the OpenGIS® Filter Encoding Implementation Specification[3]. An element identifier can be used wherever a element reference is required.

Note that the feature identifier element used in version 1.0.0 of OpenGIS® Filter Encoding Implementation Specification and version 1.0.0 of this specification has been deprecated in favor of an element identifier element that is used as an identifier for both features and elements. See clause **11 Element Identifiers** in [3] for details.

For reference purposes, the XML Schema fragment that defines identifier elements is copied from the filter encoding specification:

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

The GML Application schema that defines the features and elements served by a WFS must require the use of feature and element identifiers that make WFS 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 is 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

A GML application schema provides the definition of features served by a WFS. Clause 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 3.1.1 [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.

Likewise, a client application uses the GML application schema definitions of XML identifiers for feature and feature property element types to refer to feature and feature property element instances and to access them using the GetGmlObject operation.

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 clauses 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]. Finally, note that the standard (though not mandatory) GML lexical pattern is for property names to be in lowerCamelCase [2].

Example

Examples of valid property names are:

age, Temperature, _KHz, myns:InWaterA_1M.wkbGeom

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 components of the complex value of 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 components of the value of 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 XML elements and attributes within the description of a feature.

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 XML elements and attributes within the description of a feature consider the fictitious complex feature *Person* defined by the following XML Schema document:

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

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

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

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

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

```
<?xml version="1.0" ?>
<myns:Person
   SIN="s111222333"
```

```
    xmlns:myns="http://www.someserver.com/myns"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.someserver.com/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="#s444555666"/>
    <myns:location>
       <gml:Point><gml:pos>15 15</gml:pos></gml:Point>
    </myns:location>
    <myns:mailAddress>
       <myns:Address>
          <myns:streetName>Main St.</myns:streetName>
          <myns:streetNumber>5</myns:streetNumber>
          <myns:city>SomeCity</myns:city>
          <myns:province>Someprovince</myns:province>
          <myns:postalCode>X1X 1X1</myns:postalCode>
          <myns:country>Canada</myns:country>
       </myns:Address>
    </myns:mailAddress>
    <myns:phone>416-123-4567</myns:phone>
    <myns:phone>416-890-1234</myns:phone>
</myns:Person>
```

Using XPath [8] expressions, each XML element within the description of a *Person* feature can be referenced as follows (omitting the namespace qualifiers for the sake of clarity):

```
lastName
firstName
age
sex
spouse@SIN
location
mailAddress/Address/streetNumber
mailAddress/Address/streetName
mailAddress/Address/city
mailAddress/Address/province
mailAddress/Address/postalCode
mailAddress/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 XML element within the description may be referenced with the relative location path beginning with root element of the feature (i.e. the name of the feature type). Thus the *lastName* property could be reference as *Person/lastName*, the *city* element could be referenced as *Person/mailAddress/Address/city* and so on.

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

The 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.

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.

1 SIN = Social Insurance Number

```
<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 clause 8 of the OWS Common Implementation Specification [15].

An **<ExceptionReport>** element can contain one or more WFS processing exceptions specified using the <**Exception**> element. 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 1.1.0. The optional **language** attribute may be used to indicate the language used. The code list for the language parameter is defined in IETF RFC 1766.

Individual exception messages are contained within the **<ExceptionText>** element. The mandatory **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 **<ExceptionText>** 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" ?>
<ExceptionReport
   version="1.1.1"
   xmlns="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="owsExceptionReport.xsd">
   <Exception code="999" locator="INSERT STMT 01">
      <ExceptionText>parse error: missing closing tag for element
wkbGeom</ExceptionText>
   </Exception>
```

```
</ExceptionReport>
```

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 subclause 6.2.4. The default value of the **version** attributed is set to 1.1.0, 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 (via Insert and Update requests) and how feature instances will be generated on output (in response to GetFeature and GetGmlObject requests). The only mandatory output in response to a **DescribeFeatureType** request is a GML3 application schema defined using XML Schema. However, for the purposes of experimentation, vendor extension, or even extensions that serve a specific community of interest, other acceptable output format values may be advertised by a WFS service in the capabilities document [clause 13]. The meaning of such values in not defined in the WFS specification. The only proviso in such cases is that WFS clients may safely ignore outputFormat values that they do not recognize.

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

```
<xsd:element name="DescribeFeatureType" type="wfs:DescribeFeatureTypeType"/>
<xsd:complexType name="DescribeFeatureTypeType">
   <xsd:complexContent>
      <xsd:extension base="wfs:BaseRequestType">
         <xsd:sequence>
            <xsd:element name="TypeName" type="xsd:QName"
                         minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="outputFormat"
                        type="xsd:string" use="optional"
                        default="text/xml; subtype=gml/3.1.1"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

The **outputFormat** attribute is used to indicate the schema description language that should be used to generate the description of feature types. The default value of *application/gml+xml; version=3.1* is used to indicate that a GML3 application schema, using XML Schema, should be generated. Other schema languages may also be used to describe feature types as long as the MIME type value for the **outputFormat** attribute is advertised in the capabilities document. Table 2a defines other values for the outputFormat attribute that may be specified for backward compatibility with version 1.0 of this specification:

Table 2a – Values for the outputFormat attribute

outputFormat Value	Description
XMLSCHEMA	This value is kept for backward compatability and is used to indicate that a GML2 application schema should be generated in response to a **DescribeFeatureType** request.
application/gml+xml; version=2.1	Same as XMLSCHEMA
application/gml+xml; version=3.1	This values indicates that a GML3 application schema, using XML Schema, should be generated in response to a **DescribeFeatureType** request. This is the default values for the **outputFormat** attribute in the event that it is not specified.

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. (gml.xsd).
2. Spatial Reference Systems must be expressed as defined in the OpenGIS® Geography Markup Language (GML) Implementation Specification, version 3.1 [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 immediately 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 **application/gml+xml; version=3.1**, 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 [clause 13].

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="1.1.3"
   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/1.1.3/wfs.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
   targetNamespace="http://www.someserver.com/myns"
   xmlns:myns=http://www.someserver.com/myns
   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 subclause 7.7.

8.5 Examples

Example 1

Consider geographic features of types *TreesA_1M* 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="1.1.3"
   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/1.1.3/wfs.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/3.1.1/base/gml.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="wkbGeom"
                     type="gml:PolygonPropertyType" nillable="false"/>
            <element name="id" nillable="true" minOccurs="0">
               <simpleType>
                  <restriction base="integer">
                     <totalDigits value="10"/>
                  </restriction>
               </simpleType>
            </element>
            <element name="treeType" nillable="true" minOccurs="0">
               <simpleType>
                  <restriction base="string">
                     <maxLength value="80"/>
                  </restriction>
               </simpleType>
            </element>
         </sequence>
      </extension>
   </complexContent>
</complexType>

<complexType name="RoadL_1M_Type">
   <complexContent>
      <extension base="gml:AbstractFeatureType">
         <sequence>
            <element name="wkbGeom"
                     type="gml:LineStringPropertyType"
                     nillable="false"/>
            <element name="designation" nillable="true" minOccurs="0">
               <simpleType>
                  <restriction base="string">

                     <maxLength value="30"/>
                  </restriction>
               </simpleType>
            </element>
            <element name="surfaceType" nillable="true" minOccurs="0">
               <simpleType>
                  <restriction base="string">
                     <maxLength value="30"/>
                  </restriction>
               </simpleType>
            </element>
            <element name="nLANES" nillable="true" minOccurs="0">
               <simpleType>
                  <restriction base="integer">
                     <totalDigits value="2"/>
                  </restriction>
```

```
                  </simpleType>
               </element>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
</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/1.1.3/wfs.xsd
                       http://www.someserver.com/myns ex07.xsd">
   <gml:boundedBy>
      <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
         <gml:lowerCorner>-180.0 -90.0</gml:lowerCorner>
         <gml:upperCorner>180.0 90.0</gml:upperCorner>
      </gml:Envelope>
   </gml:boundedBy>
   <gml:featureMember>
      <TreesA_1MTreesA_1M>
         <wkbGeom>
            <gml:Polygon>
               <gml:exterior>
                  <gml:LinearRing srsName="EPSG:63266405">
                     <gml:posList srsDimensions="2">-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:posList>
                  </gml:LinearRing>
               </gml:exterior>
            </gml:Polygon>
         </wkbGeom>
         <ID>0000000002</ID>
         <treeType>Maple</treeType>
      </TreesA_1MTreesA_1M>
   </gml:featureMember>
   <gml:featureMember>
      <RoadL_1M>
         <wkbGeom>
            <gml:LineString
srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
               <gml:posList>-59.478340 -52.226578 -59.484871 -52.223564 -59.488991 -
52.198524 -59.485958 -52.169559 -59.480400 -52.152615 -59.465576 -52.141491 -
59.462002 -52.136417 -59.447968 -52.127190 -59.422928 -52.120701 -59.411915 -
52.117844 -59.397972 -52.116440 -59.371311 -52.121300</gml:posList>
            </gml:LineString>
         </wkbGeom>
         <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 *mailAddress*.

In response to the **DescribeFeatureType** request:

```
<?xml version="1.0" ?>
<DescribeFeatureType
   version="1.1.3"
   service="WFS"
   outputFormat="text/xml; subtype=gml/3.1.1"
   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/1.1.3/wfs.xsd">
   <TypeName>myns:Person</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/3.1.1/base/gml.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="mailAddress"
                            type="myns:AddressPropertyType"
                            nillable="true"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
```

```
   </xsd:complexType>
   <complexType name="AddressPropertyType">
      <sequence>
         <element name="Address"
                  type="myns:AddressType" minOccurs="0" />
      </sequence>
   </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/1.1.3/wfs.xsd
                    http://www.someserver.com/myns ex10.xsd">
   <gml:boundedBy>
      <gml:Envelope>
         <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:Envelope>
   </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:pos>15 15</gml:pos></gml:Point>
         </myns:location>
         <myns:mailAddress>
            <myns:Address>
               <myns:streetName>Main St.</myns:streetName>
               <myns:streetNumber>5</myns:streetNumber>
               <myns:city>SomeCity</myns:city>
               <myns:province>Someprovince</myns:province>
               <myns:postalCode>X1X 1X1</myns:postalCode>
               <myns:country>Canada</myns:country>
            <myns:Address>
         </myns:mailAddress>
      </Person>
   </gml:featureMember>
</wfs:FeatureCollection>
```

9 GetFeature operation

9.1 Introduction

The canonical representation of Features uses GML, and the form of the **GetFeature** request is modeled after this representation of the result set. For this reason, it is necessary to have a clear understanding of how the information model for features is mapped into the GML representation. The reference description of GML is given by the specification [3] but the salient aspects are summarized here.

In GML a **feature** is represented as an XML element. The name of the feature element indicates the Feature Type, conventionally given in UpperCamelCase, such as *xmml:BoreHole* or *myns:SecondaryCollege*.

The content of a feature element is a set of elements, which describes the feature in terms of a set of properties. Each child element of the feature element is a **property**. The name of the property element indicates the property type, conventionally given in lowerCamelCase, such as *gml:boundedBy* or *xmml:collarLocation*.

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

property *myns:frontsOn*, whose value is a *myns:Road*, which will itself have further properties, etc. However, note that the properties of the second feature (the *myns:Road*) are ***not*** properties of the first feature (the *myns:School*) and it is an error to refer to them as such.

The **GetFeature** operation allows retrieval of features from a web feature service. A **GetFeature** request is processed by a WFS and when the value of the **outputFormat** attribute is set to **text/gml; subtype=gml/3.1.1**, a GML instance document, containing the result set, is returned to the client.

If a web feature service implements Xlink traversal, a WFS client can use the **traverseXlinkDepth** and **traverseXlinkExpiry** attributes to request that nested property XLink linking element locator attribute (href) XLinks are traversed and resolved if possible.

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:complexContent>
      <xsd:extension base="wfs:BaseRequestType">
         <xsd:sequence>
            <xsd:element ref="wfs:Query" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="resultType"
                        type="wfs:ResultTypeType" use="optional"
                        default="results"/>
         <xsd:attribute name="outputFormat"
                        type="xsd:string" use="optional"
                        default="text/xml; subtype=3.1.1"/>
         <xsd:attribute name="maxFeatures"
                        type="xsd:positiveInteger" use="optional"/>
         <xsd:attribute name="traverseXlinkDepth"
                        type="xsd:string" use="optional"/>
         <xsd:attribute name="traverseXlinkExpiry"
                        type="xsd:positiveInteger"
                        use="optional"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="ResultTypeType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="results"/>
      <xsd:enumeration value="hits"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:element name="Query" type="wfs:QueryType"/>
<xsd:complexType name="QueryType">
  <xsd:sequence>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:element ref="wfs:PropertyName"/>
       <xsd:element ref="ogc:Function"/>
    </xsd:choice>
    <xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="ogc:SortBy" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="handle"
                 type="xsd:string" use="optional"/>
```

```
    <xsd:attribute name="typeName"
                   type="wfs:TypeNameListType" use="required"/>
    <xsd:attribute name="featureVersion"
                   type="xsd:string" use="optional"/>
    <xsd:attribute name="srsName" type="xsd:anyURI" use="optional"/>
  </xsd:complexType>
  <xsd:simpleType name="Base_TypeNameListType">
     <xsd:list itemType="QName"/>
  </xsd:simpleType>
  <xsd:simpleType name="TypeNameListType">
     <xsd:restriction base="wfs:Base_TypeNameListType">
        <xsd:pattern value="((\w:)?\w(=\w)?){1,}"/>
     </xsd:restriction>
  </xsd:simpleType>
```

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

The optional **outputFormat** attribute specifies the format of the response to a **GetFeature** request. The default value is **application/gml+xml; version=3.1** indicating that a valid GML3 [2] document, that validates against a valid GML3 application schema (generated using the **DescribeFeatureType** request), must be generated. For backward compatability, the values **GML2** or **text/xml; subtype=gml/2.1.2** may be specified indicating that a valid GML2 document, that validates against a valid GML2 application schema, must be generated. Table 2b summarizes the possible values for the **outputFormat** attribute:

Table 9a – Values for outputFormat attribute

outputFormat Value	Description
GML2	This value is kept for backward compatability and indicates that an XML instance document must be generated that validates against a GML2 application schema.
application/gml+xml; version=2.1	Same as GML2
application/gml+xml; version=3.1	This value indicates that an XML instance document must be generated that validates against a GML3 application schema. This is the default values of the outputFormat attribute if the attribute is not specified in the GetFeature request.

Other output formats (including older versions of GML, non-XML, binary and vendor specific formats) are also possible as long at the appropriate values for the **outputFormat** attribute are advertised in the capabilities document [clause 13]. This specification recommends that a descriptive narative be included in the capabilities document for each output format listed there. This, however, is not a mandatory requirement.

A web feature service may respond to a **GetFeature** request in one of two ways. It can either generate a complete response document or it may simply return a count of the number of features that a **GetFeature** request would return. The optional **resultType**

attribute is used to control how a web feature service responds to a **GetFeature** request. The possible values for the attribute are summarized in the following table:

Table 9b – Values for resultType attribute

resultType Value	Description
Results	The default value **results** indicates that a web feature service should generate a complete response that contains all the features that satisfy the request. The exact structure of the response is defined in clause 9.3.
Hits	The value **hits** indicates that a web feature service should process the GetFeature request and rather than return the entire result set, it should simply indicate the number of feature instance of the requested feature type(s) that satisfy the request. That is that the count should only include instances of feature types specified in the typeName attribute (i.e. GetFeature/Query/@typeName). The exact way in which the feature count is conveyed to a client application is described in clause 9.3.

The optional **maxFeatures** attribute can be used to limit the number of explicitly requested features (i.e. features specified via the `GetFeature/Query/@typeName` attribute) that a **GetFeature** request presents in the response document. The maxFeatures value applies to whole result set and the constraint in applied to the features in the order in which they are presented. In addition, feature members contained in a requested feature collection do not count – the requested feature collection counts as one feature. Once the **maxFeatures** limit is reached, the result set is truncated at that point. There is no default value defined and the absense of the attribute means that all feature type instances in the result should should be returned to the client.

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 the feature properties in order to select the valid feature set.

The mandatory **typeName** attribute is used to indicate the name of one or more feature type instances or class instances to be queried. Its value is a list of namespace-qualified names (XML Schema type QName, e.g. *myns:School*) whose value must match one of the feature types advertised in the Capabilities document of the WFS. Specifying more than one typename indicates that a join operation is being performed. All the names in the typeName list must be valid types that belong to this query's feature content as defined by the GML Application Schema. Optionally, individual feature type names in the typeName list may be aliased using the format *QName=Alias*. The following is an example typeName value that indicates that a join operation is to be performed and includes aliases:

typeName="ns1:InwaterA_1m=A, ns1:InwaterA_1m=B, ns2:CoastL_1M=C"

This example encodes a join between three feature types aliased as A, B and C. The join between feature type A and B is a self-join.

The optional **featureVersion** attribute on the <**Query**> element 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 nth 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. It should be noted, that if the value of the **featureVersion** parameter is set to **ALL**, the resulting XML document will contain duplicate feature identifiers and thus cannot be validated.

The optional **srsName** attribute of the <**Query**> element is used to specify a specific WFS-supported SRS to be used for returned feature geometries. Its value may be the <**DefaultSRS**> or any of the <**OtherSRS**> values listed for the feature type in WFS capabilities document. If no **srsName** value is supplied, then the features shall be returned using the advertised <**DefaultSRS**> value. This attribute has no meaning for feature types with no spatial properties; if an **srsName** value is specified for a feature with no spatial properties, a web feature service may ignore the parameter and its value.

Any valid URI value can be assigned to the **srsName** attribute. However, in order to enhance interoperability, a web feature service must be able to process **srsName** attribute values with the following format models:

- `EPSG:<EPSG code>`
- `http://www.opengis.net/gml/srs/epsg.xml#<EPSG code>`
- `urn:EPSG:geographicCRC:<epsg code>`

In these format models, the values <*EPSG code*> are placeholders for actual EPSG code values. Here is an example of the **srsName** where the assigned value follows one of the required format models: `srsName="urn:EPSG:geographicCRS:63266405"`.

The optional **traverseXlinkDepth** attribute indicates the depth to which nested property XLink linking element locator attribute (href) XLinks in all properties of the selected feature(s) are traversed and resolved if possible. A value of "1" indicates that one linking element locator attribute (href) XLink will be traversed and the referenced element returned if possible, but nested property XLink linking element locator attribute (href) XLinks in the returned element are not traversed. A value of "*" indicates that all nested property XLink linking element locator attribute (href) XLinks will be traversed and the referenced elements returned if possible. The range of valid values for this attribute consists of positive integers plus "*".

The **tarverseXlinkExpiry** attribute is specified in minutes. It indicates how long a Web Feature Service should wait to receive a response to a nested **GetGmlObject** request. If no **traverseXlinkExpiry** attribute is present for a **GetGmlObject** request, the WFS wait time is implementation dependent.

The value of each <**wfs:PropertyName**> element is a namespace-qualified name (XML Schema type QName, e.g. *myns:address*) whose value must match the name of one of the

property elements in the GML representation of the relevant feature. The relevant feature is of the type indicated as the value of the **typeName** attribute of the parent <**Query**> element. The names of the property elements may be discovered from the schema description of this feature type, being the names of the immediate child elements of the feature element. The schema description may be obtained as the result of a **DescribeFeatureType** request [clause 8].

There is typically some flexibility in the structure of the description of a feature type of interest, especially concerning the optional or mandatory nature of each property. The W3C XML Schema language used to define GML Application Schemas uses certain notations to indicate whether property elements are mandatory or optional or how many occurrences are valid. Thus, the GML representation of a feature may be schema-valid with only a subset of the possible properties present. Every feature representation must include the mandatory subset for the feature type, and then may include a selection of the other properties, according to the schema description. The <**wfs:PropertyName**> elements are used to enumerate which of the non-mandatory properties should be included in the response to a **GetFeature** request.

The response to a **GetFeature** request must be valid according to the structure described by the XML Schema description of the feature type. Thus the WFS must report all the mandatory properties of each feature, as well any properties requested through the <**wfs:PropertyName**> element. 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 mandatory property names. A WFS client must thus be prepared to deal with a situation where it receives more property values than it requests through <**wfs:PropertyName**> elements.

A **<wfs:XlinkPropertyName>** element may be used instead of the **<wfs:PropertyName>** element to enumerate feature properties that should be selected during a query and whose values should be processed by a **GetGmlObject** operation as specified by the **traverseXlinkDepth** and **traverseXlinkExpiry** attributes to traverse and resolve nested property XLink linking element locator attribute (href) XLinks that those values contain. The **traverseXlinkDepth** and **traverseXlinkExpiry** attributes on the **<wfs:XlinkPropertyName>** element apply only to the value of the named property, and override the values of the **traverseXlinkDepth** and **traverseXlinkExpiry** attributes, if any, on the **<GetFeature>** element.

If no **<wfs:PropertyName>** elements or **<wfs:XlinkPropertyName>** 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 feature instances that are selected; presumably to update the features in a subsequent operation.

Note that **<wfs:XlinkPropertyName>** elements shall only be used in the **<GetFeature>** element in this version of the WFS specification.

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 the MIME type **text/xml; subtype=gml/3.1.1.** 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 3.1.1 [2], and more specifically, the output must validate againt the GML application schema generated by the **DescribeFeatureType** operation [sec. 8].

The root element of the response to a **GetFeature** operation must be the <**wfs:FeatureCollection>** element which is defined by the following XML-Schema fragment:

```
<xsd:element name="FeatureCollection"
             type="wfs:FeatureCollectionType"
             substitutionGroup="gml:_FeatureCollection"/>
<xsd:complexType name="FeatureCollectionType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureCollectionType">
      <xsd:attribute name="lockId" type="xsd:string" use="optional"/>
      <xsd:attribute name="timeStamp" type="xsd:dateTime" use="optional"/>
      <xsd:attribute name="numberOfFeatures"
                     type="xsd:nonNegativeInteger"
                     use="optional"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The content of the <**wfs:FeatureCollection>** element is controlled by the value of the **resultType** attribute on the <**GetFeature>** element which is described in clause 9.2. If the specified value of the **resultType** attribute is **results** (the default value) then a web feature service must generate a complete response as the content of the <**wfs:FeatureCollection**> element. If, however, the value of the **resultType** attribute is specified as **hits**, a web feature service must generate a <**wfs:FeatureCollection**> element with no content (i.e. empty) but must populate the values of the **timeStamp** attribute and the **numberOfFeatures** attribute. In this way a client may obtain a count of the number of features that a query would return without having to incur the cost of transmitting the entire result set.

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. The lock action of the **GetFeatureWithLock** request is to attempt to lock all identified feature instances. If all identified feature instances cannot be locked, then an exception report should be generated.

The optional **timeStamp** attribute may be used by a web feature service to indicate the time and date when a response was generated.

The optional **numberOfFeatures** attribute is used to indicate the number of features that are in the response document. The count should only include feature type instances of the feature type names specified in the **typeName** attribute of the <**Query**> element (i.e. GetFeature/Query/@typeName) used to generate the response.

9.3.1 Use of the schemaLocation attribute

Any GML document generated by a WFS implementation, in response to a query where the **outputFormat** is the MIME type **text/xml; subtype=gml/3.1.1**, 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 indicates 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.someserver.com/myns"
   xmlns:myns="http://www.someserver.com/myns"
   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.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 the responsibility of each WFS implementation to arrange that the GML output makes the appropriate **schemaLocation** reference(s) such that the output can be validated.

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 subclause 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="1.1.3"
   outputFormat="text/xml; subtype=gml/3.1.1"
   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/1.1.3/wfs.xsd">
   <wfs:Query typeName="myns:InWaterA_1M">
      <ogc:Filter>
         <ogc:GmlObjectId gml:id="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="1.1.3"
   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/1.1.3/wfs.xsd">
   <wfs:Query typeName="myns:InWaterA_1M">
      <wfs:PropertyName>myns:wkbGeom</wfs:PropertyName>
      <wfs:PropertyName>myns:tileId</wfs:PropertyName>
      <wfs:PropertyName>myns:facId</wfs:PropertyName>
      <ogc:Filter>
         <ogc:GmlObjectId gml:id="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 **<GmlObjectId>** element is used to identify each feature to be fetched.

```
<?xml version="1.0" ?>
<GetFeature
   version="1.1.3"
   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/1.1.3/wfs.xsd">
   <Query typeName="myns:InWaterA_1M">
     <ogc:Filter>
        <ogc:GmlObjectId gml:id="InWaterA_1M.1013"/>
        <ogc:GmlObjectId gml:id="InWaterA_1M.1014"/>
        <ogc:GmlObjectId gml:id="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="1.1.3"
   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/1.1.3/wfs.xsd">
   <Query typeName="myns:InWaterA_1M">
      <wfs:PropertyName>myns:wkbGeom</wfs:PropertyName>
      <wfs:PropertyName>myns:tileId</wfs:PropertyName>
      <ogc:Filter>
         <ogc:GmlObjectId gml:id="InWaterA_1M.1013"/>
         <ogc:GmlObjectId gml:id="InWaterA_1M.1014"/>
         <ogc:GmlObjectId gml:id="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="1.1.3"
   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/1.1.3/wfs.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="1.1.3"
   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/1.1.3/wfs.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="1.1.3"
   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/1.1.3/wfs.xsd">
   <Query typeName="myns:Hydrography">
      <wfs:PropertyName>myns:geoTemp</wfs:PropertyName>
      <wfs:PropertyName>myns:depth</wfs:PropertyName>
      <ogc:Filter>
         <ogc:Not>
            <ogc:Disjoint>
               <ogc:PropertyName>myns:geoTemp</ogc:PropertyName>
               <gml:Envelope srsName="EPSG:63266405">
                  <gml:lowerCorner>-57.9118 46.2023<gml:lowerCorner>
                  <gml:upperCorner>-46.6873 51.8145</gml:upperCorner>
               </gml:Envelope>
            </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/1.1.3/wfs.xsd">
```

```
      <gml:boundedBy>
         <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
            <gml:lowerCorner>10 10</gml:lowerCorner>
            <gml:upperCorner>20 20</gml:upperCorner>
         </gml:Envelope>
      </gml:boundedBy>
      <gml:featureMember>
         <HydrographyHydrography gml:id="HydrographyHydrography.450">
            <geoTemp>
               <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
                  <gml:pos>10 10</gml:pos>
               </gml:Point>
            </geoTemp>
            <depth>565</depth>
         </HydrographyHydrography>
      </gml:featureMember>
      <gml:featureMember>
         <HydrographyHydrography gml:id="HydrographyHydrography.450">
            <geoTemp>
               <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
                  <gml:pos>10 11</gml:pos>
               </gml:Point>
            </geoTemp>
            <depth>566</depth>
         </HydrographyHydrography>
      </gml:featureMember>
      <!—
      .
      . … more HydrographyHydrography 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="1.1.3"
      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/1.1.3/wfs.xsd">
      <Query typeName="myns:ROADS">
         <wfs:PropertyName>myns:path</wfs:PropertyName>
         <wfs:PropertyName>myns:lanes</wfs:PropertyName>
         <wfs:PropertyName>myns:surfaceType</wfs:PropertyName>
         <ogc:Filter>
            <ogc:Within>
               <ogc:PropertyName>myns:path</ogc:PropertyName>
               <gml:Envelope srsName="EPSG:63266405">
                  <gml:lowerCorner>50 40</gml:lowerCorner>
                  <gml:upperCorner>100 60</gml:upperCorner>
               </gml:Envelope>
            </ogc:Within>
         </ogc:Filter>
      </Query>
      <Query typeName="myns:Rails">
         <wfs:PropertyName>myns:track</wfs:PropertyName>
         <wfs:PropertyName>myns:gauge</wfs:PropertyName>
         <ogc:Filter>
            <ogc:Within>
               <ogc:PropertyName>myns:track</ogc:PropertyName>
```

```
            <gml:Envelope srsName="…">
               <gml:lowerCorner>50 40</gml:lowerCorner>
               <gml:upperCorner>100 60</gml:upperCorner>
            </gml:Envelope>
         </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/1.1.3/wfs.xsd
                       http://www.someserver.com/myns ROADSRAILS.xsd">
   <gml:boundedBy>
      <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
         <gml:lowerCorner>0 0</gml:lowerCorner>
         <gml:upperCorner>180 360</gml:upperCorner>
      </gml:Envelope>
   </gml:boundedBy>
   <gml:featureMember>
      <Roads gml:id="Roads.100">
         <path>
            <gml:LineString gid="1"
                 srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
               <gml:posList>10 10 10 11 10 12 10 13</gml:posList>
            </gml:LineString>
         </path>
         <surfaceType>ASPHALT</surfaceType>
         <nLanes>4</nLanes>
      </Roads>
   </gml:featureMember>
   <gml:featureMember>
      <Roads gml:id="Roads.105">
         <path>
            <gml:LineString gid="2"
                 srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
               <gml:posList>10 10 10 11 10 12</gml:posList>
            </gml:LineString>
         </path>
         <surfaceType>GRAVEL</surfaceType>
         <nLanes>2</nLanes>
      </Roads>
   </gml:featureMember>
   <!--
   ... more Roads features ....
   -->
   <gml:featureMember>
      <Rails gml:id="Rails.119">
         <track>
            <gml:LineString gid="n"
                 srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
               <gml:posList>15 10 16 11 17 12</gml:posList>
            </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.someserver.com/myns"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   version="1.0">

   <import namespace="http://www.opengis.net/gml"
           schemaLocation="../gml/3.1.1/base/gml.xsd"/>

   <element name="Person" type="myns:PersonType"
            substitutionGroup="gml:_Feature"/>
   <complexType name="PersonType">
      <complexContent>
         <extension base="gml:AbstractFeatureType">
            <sequence>
               <element name="lastName" nillable="true">
                  <simpleType>
                     <restriction base="string">
                        <maxLength value="30"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="firstName" nillable="true">
                  <simpleType>
                     <restriction base="string">
                        <maxLength value="10"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="age" type="integer" nillable="true"/>
               <element name="sex" type="string"/>
               <element name="spouse">
                  <complexType>
                     <attribute name="SIN" type="xsd:anyURI" use="required" />
                  </complexType>
               </element>
               <element name="location"
                        type="gml:PointPropertyType"
                        nillable="true"/>
               <element name="mailAddress"
                        type="myns:AddressPropertyType" nillable="true"/>
               <element name="salary" type="positiveInteger" nillable="true"/>
            </sequence>
            <attribute name="SIN" type="xsd:anyURI" use="required"/>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="AddressPropertyType">
      <sequence>
         <element name="Address"
                  type="myns:AddressType" minOccurs="0" />
      </sequence>
   </complexType>
   <complexType name="AddressType">
      <sequence>
         <element name="streetName" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
               </restriction>
            </simpleType>
         </element>
         <element name="streetNumber" nillable="true">
```

```
            <simpleType>
               <restriction base="string">
                  <maxLength value="10"/>
               </restriction>
            </simpleType>
         </element>
         <element name="city" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
               </restriction>
            </simpleType>
         </element>
         <element name="province" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
               </restriction>
            </simpleType>
         </element>
         <element name="postalCode" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="15"/>
               </restriction>
            </simpleType>
         </element>
         <element name="country" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
               </restriction>
            </simpleType>
         </element>
      </sequence>
   </complexType>
</schema>
```

The *mailAddress* property is a complex property.

The following example fetches the last name of all the people who live on the 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="1.1.3"
   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/1.1.3/wfs.xsd
                       http://www.someserver.com/myns Person.xsd">
   <Query typeName="Person">
      <wfs:PropertyName>myns:Person/myns:lastName</wfs:PropertyName>
      <ogc:Filter>
         <ogc:And>
            <ogc:And>
               <ogc:PropertyIsGreaterThanOrEqualTo>
<ogc:PropertyName>myns:Person/myns:mailAddress/myns:Address/myns:streetNumber</ogc:Pr
opertyName>
                  <ogc:Literal>10000</ogc:Literal>
               </ogc:PropertyIsGreaterThanOrEqualTo>
               <ogc:PropertyIsLessThanOrEqualTo>
```

```
    <ogc:PropertyName>myns:Person/myns:mailAddress/myns:Address/myns:streetNumber</ogc:Pr
    opertyName>
                        <ogc:Literal>10999</ogc:Literal>
                     </ogc:PropertyIsLessThanOrEqualTo>
                  </ogc:And>
                  <ogc:And>
                     <ogc:PropertyIsEqualTo>
    <ogc:PropertyName>myns:Person/myns:mailAddress/myns:Address/myns:streetName</ogc:Prop
    ertyName>
                        <ogc:Literal>Main St.</ogc:Literal>
                     </ogc:PropertyIsEqualTo>
                     <ogc:PropertyIsEqualTo>
    <ogc:PropertyName>myns:Person/myns:mailAddress/myns:Address/myns:city</ogc:PropertyNa
    me>
                        <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>
```

Example 10

This example illustrates a **GetFeature** request using **<wfs:PropertyName>** elements that serves as a base for comparison with Examples 11 and 12 . The request:

```
<wfs:GetFeature
   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/1.1.3/wfs.xsd"
   version="1.1.3"
   service="WFS"
   outputFormat="text/xml; subtype=gml/3.1.1">
   <wfs:Query typeName="Town">
      <wfs:PropertyName>gml:name</wfs:PropertyName>
      <wfs:PropertyName>gml:directedNode</wfs:PropertyName>
      <ogc:Filter>
         <ogc:GmlObjectId gml:id="t1"/>
      </ogc:Filter>
   </wfs:Query>
</wfs:GetFeature>
```

The response contains an xlink:href, but it is returned as-is:

```
<?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/1.1.3/wfs.xsd">
   <gml:boundedBy>
      <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
         <gml:lowerCorner>10 10</gml:lowerCorner>
         <gml:upperCorner>20,20</gml:upperCorner>
      </gml:Envelope>
   </gml:boundedBy>
```

```
   <gml:featureMember>
      <Town gml:id="t1">
         <gml:name>Bedford</gml:name>
         <gml:directedNode orientation="+" xlink:href="#n1"/>
      </Town>
   </gml:featureMember>
</wfs:FeatureCollection>
```

Example 11

This example illustrates the use of the **traverseXlinkDepth** and **traverseXlinkExpiry** attributes on the GetFeature request from Example 10. The request:

```
<wfs:GetFeature
   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/1.1.3/wfs.xsd"
   version="1.1.3"
   service="WFS"
   outputFormat="text/xml; subtype=gml/3.1.1"
   traverseXlinkDepth="1"
   traverseXlinkExpiry="1">
   <wfs:Query typeName="Town">
      <wfs:PropertyName>gml:name</wfs:PropertyName>
      <wfs:PropertyName>gml:directedNode</wfs:PropertyName>
      <ogc:Filter>
         <ogc:GmlObjectId gml:id="t1"/>
      </ogc:Filter>
   </wfs:Query>
</wfs:GetFeature>
```

In the response, the first level xlink:href has been traversed, and its contents returned as described for the **GetGmlObject** operation:

```
<?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/1.1.3/wfs.xsd">
   <gml:boundedBy>
      <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
         <gml:lowerCorner>10 10</gml:lowerCorner>
         <gml:upperCorner>20 20</gml:upperCorner>
      </gml:Envelope>
   </gml:boundedBy>
   <gml:featureMember>
      <Town gml:id="t1">
         <gml:name>Bedford</gml:name>
         <gml:directedNode orientation="+">
            <!-- xlink:href="#n1" →
            <gml:Node gml:id="n1">
               <gml:pointProperty
                     xlink:href="http://www.bedford.town.uk/civilworks/gps.gml#townHall"/>
            </gml:Node>
         </gml:directedNode>
      </Town>
   </gml:featureMember>
</wfs:FeatureCollection>
```

Example 12

This example illustrates the use of the **<wfs:XlinkPropertyName>** element with **traverseXlinkDepth** and **traverseXlinkExpiry** attributes that override those attributes on the **GetFeature** request from Example 11. The request:

```
<wfs:GetFeature
   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/1.1.3/wfs.xsd"
   version="1.1.3"
   service="WFS"
   outputFormat="text/xml; subtype=gml/3.1.1"
   traverseXlinkDepth="1"
   traveserXlinkExpiry="1">
   <wfs:Query typeName="Town">
      <wfs:PropertyName>gml:name</wfs:PropertyName>
      <wfs:XlinkPropertyName
           traverseXlinkDepth="2"
           traverseXlinkExpiry="2">gml:directedNode</wfs:XlinkPropertyName>
      <ogc:Filter>
         <ogc:GmlObjectId gml:id="t1"/>
      </ogc:Filter>
   </wfs:Query>
</wfs:GetFeature>
```

In the response, the first and second level xlink:hrefs in the gml:directedNode property have been traversed, and their contents returned as described for the GetGmlObject operation:

```
<?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/1.1.3/wfs.xsd">
   <gml:boundedBy>
      <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
         <gml:lowerCorner>10 10</gml:lowerCorner>
         <gml:upperCorner>20 20</gml:upperCorner>
      </gml:Envelope>
   </gml:boundedBy>
   <gml:featureMember>
      <Town gml:id="t1">
         <gml:name>Bedford</gml:name>
         <gml:directedNode orientation="+"> <!-- xlink:href="#n1" →
            <gml:Node gml:id="n1">
               <gml:pointProperty>
<!--  xlink:href="http://www.bedford.town.uk/civilworks/gps.gml#townHall" -->
                  <gml:Point gml:id="townHall" srsName="…">
                     <gml:pos>147 234</gml:pos>
                  </gml:Point>
               </gml:pointProperty>
            </gml:Node>
         </gml:directedNode>
      </Town>
   </gml:featureMember>
</wfs:FeatureCollection>
```

10 GetGmlObject operation

10.1 Introduction

The **GetGmlObject** operation allows retrieval of features and elements by ID from a web feature service. A **GetGmlObject** request is processed by a WFS, and an XML document **fragment**, containing the result set, is returned to the client. The **GetGmlObject** request provides the interface through which a WFS can be asked to traverse and resolve XLinks to the features and elements it serves.

The **GetGmlObject** operation is optional and a WFS implementation does not need to support it to conform to this specification as a *BasicWFS* or *Transactional WFS*. If the **GetGmlObject** operation is supported, for local resources, remote resources, or both, then this fact must be advertised on the capabilities document as described in clause 12. Request

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

```
<xsd:element name="GetGmlObject" type="wfs:GetGmlObjectType"/>
<xsd:complexType name="GetGmlObjectType">
   <xsd:complexContent>
      <xsd:extension base="wfs:BaseRequestType">
         <xsd:sequence>
            <xsd:element ref="ogc:GmlObjectId"/>
         </xsd:sequence>
         <xsd:attribute name="outputFormat"
                        type="xsd:string" use="optional" default="GML3"/>
         <xsd:attribute name="traverseXlinkDepth"
                        type="xsd:string" use="required"/>
         <xsd:attribute name="traverseXlinkExpiry"
                        type="xsd:positiveInteger"
                        use="optional"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

The **<GetGmlObject>** element is used to request that a web feature service return an element with a **gml:id** attribute value specified by an **ogc:GmlObjectId**. A **GetGmlObject** element contains exactly one **<GmlObjectId>**. The value of the **gml:id** attribute on that **<GmlObjectId>** is used as a unique key to retrieve the complex element with a **gml:id** attribute with the same value. The requested element could be any identified element of the GML data being served by a WFS, such as a feature, a geometry, a topology, or a complex attribute.

The **outputFormat** attribute defines the format to use to generate the result set. The default value is **text/xml; subtype=gml/3.1.1** 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 **traverseXlinkDepth** attribute indicates the depth to which nested property Xlink linking element locator attribute (href) XLinks are traversed and resolved if possible. A value of "1" indicates that one linking element locator attribute (href) XLink will be

traversed and the referenced element returned if possible, but nested property Xlink linking element locator attribute (href) XLinks in the returned element are not traversed. A value of "*" indicates that all nested property XLink linking element locator attribute (href) XLinks will be traversed and the referenced elements returned if possible. The range of valid values for this attribute consists of positive integers plus "*".

The **traverseXlinkExpiry** attribute is specified in minutes. It indicates how long a Web Feature Service should wait to receive a response to a nested GetGmlObject request. If no **traverseXlinkExpiry** attribute is present for a **GetGmlObject** request, the WFS wait time is implementation dependent.

10.2 Processing

10.2.1 URI Parsing

The locator attribute (href) URI may consist of an actual URI prefix, a "#" separator, and a fragment identifier suffix. If either the "#" separator or fragment identifier suffix is missing, the WFS shall raise an exception as described in subclause 7.7. If the actual URI prefix is a URL (presumably of a remote WFS) with no trailing "?" query string, then the request shall be redirected as specified in 10.3.4 Redirect. If the actual URI prefix is missing, then the fragment identifier suffix becomes the request ID that shall be used as specified in subclause 10.3.2. Otherwise, the WFS shall raise an exception as described in subclause 7.7.

10.2.2 TopLevel

If the **GetGmlObject** operation is not supported for local resources, the WFS shall raise an exception as described in subclause 7.7. The WFS feature store is searched for an element with an identifier equal to the request ID. If no such element is found, the WFS shall raise an exception as described in subclause 7.7. If the value of the **traverseXlinkDepth** attribute is "1", the element with an identifier equal to the request ID shall be returned as specified in 10.4 Response. Otherwise, the element shall be processed as the current element as specified in 10.3.3 Nested, at nesting depth 2.

10.2.3 Nested

Note: Processing in the section is specified as if the current element were in GML3 form. The actual form of the element and the actual steps taken during WFS processing are implementation dependent and irrelevant as long as the outcomes are as specified here.

The current element is searched for Xlink linking elements, e.g. any element that in GML3 form has an xlink:href locator attribute. If no Xlink linking elements are found, the current element shall be returned as specified in 10.4 Response.

Otherwise, new response element is created. Each fragment of the current element up to and including the next XLink linking element, or to the end of the current element, is processed as follows:

1. The fragment of the current element preceeding an XLink linking element, or to the end of the current element if there are no more XLink linking elements, is copied to the new response element.

2. If the end of the current element has been reached and there are no more fragments to process, the new response element shall be returned as specified in 10.4 Response.

3. If the **traverseXlinkDepth** attribute value is "*" or greater than the nesting depth, then:

    a. The XLink linking element start tag shall be written to the new response element without the xlink:href locator attribute or closing tag suffix, followed by a comment containing the locator attribute. For example, the XLink linking element `<Town gml:id="t1"/>` would be written as `<Town> <!-- gml:id="t1" -->`.

    b. Recurse through subclause 10.3 Processing using the Xlinking element locator attribute URI at nesting depth = nesting depth + 1.

    c. Write the response from (b) to the new response element.

    d. If there were any child elements contained in the XLink linking element, write them to the new response element.

    e. Write the XLink linking element end tag to the new response element. For example, `</Town>`.

The new response element shall be returned as specified in clause 10.4 Response.

10.2.4 Redirect

If the **GetGmlObject** operation is not supported for remote resources, the WFS shall raise an exception as described in subclause 7.7. The WFS shall encode a **GetGmlObject** request as specified in clause 10 wherein the value of the **gml:id** attribute of the **ogc:GmlObjectId** element is the fragment identifier suffix, and the values of the **outputFormat** , **traverseXLinkDepth** and **traverseXlinkExpiry** attributes are set from the current **GetGmlObject** request.

The WFS shall then send the **GetGmlObject** request to the actual URI prefix and wait for a response. If the **traverseXlinkExpiry** attribute was provided for the current **GetGmlObject** request, and more than the specified number of minutes elapse before a response is received, the WFS shall raise an exception as described in subclause 7.7.

The WFS shall then return the response to its cascaded **GetGmlObject** request as specified in clause 10.3.

10.3 Response

The format of the response to a **GetGmlObject** request is controlled by the **outputFormat** attribute. The default value for the **outputFormat** attribute shall be **text/xml; subtype=gml/3.1.1.** This will indicate that a WFS must generate a GML document fragment of the result set that conforms to the OpenGIS® Geography Markup Language Implementation Specification, version 3.1.1 [2].

The response to a **GetGmlObject** request is the referenced GML element returned as an XML document fragment. This differs from the response to a **GetFeature** request, which returns a complete document containing a wfs:FeatureCollection.

If the content of the response to a **GetGmlObject** request contains nested property XLink linking element locator attribute (href) XLinks, the content of the response is affected by the value of the **traverseXlinkDepth** and **traverseXlinkExpiry** attributes of the **GetGmlObject** request, which control how the response element is rewritten to include the elements referenced by the XLinks. If any such nested property XLink linking element locator attribute value references a remote WFS, the response content is also affected by whether the local WFS has the capability to respond to a remote **GetGmlObject** requests.

10.4 Exceptions

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

10.5 Examples

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

Example 1

This example shows a GetGmlObject request element that fetches the feature with the identifier "InWaterA_1M.1234".

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:GetGmlObject
   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.opengis.net/wfs ../wfs/1.1.0/WFS.xsd"
   service="WFS"
   version="1.1.0"
   outputFormat="text/xml; subtype=gml/3.1.1"
   traverseXlinkDepth="1"
   traverseXlinkExpiry="1">
```

```
   <ogc:GmlObjectId gml:id="InWaterA_1M.1234"/>
</wfs:GetGmlObject>
```

Example 2

This example shows a GetGmlObject request element that fetches the feature with the identifier “t1” without performing any nested XLink traversal:

```
   <?xml version="1.0" encoding="UTF-8"?>
   <wfs:GetGmlObject
      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.opengis.net/wfs ../wfs/1.1.0/WFS.xsd"
      service="WFS"
      version="1.1.0"
      outputFormat="text/xml; subtype=gml/3.1.1"
      traverseXlinkDepth="1"
      traverseXlinkExpiry="1">
      <ogc:GmlObjectId gml:id="t1"/>
   </wfs:GetGmlObject>
```

and the response:

```
   <Town gml:id="t1">
      <gml:name>Bedford</gml:name>
      <gml:directedNode orientation="+" xlink:href="#n1"/>
   </Town>
```

Example 3

This example is the same as example 2 except that it requests nested XLink traversal at depth 2:

```
   <?xml version="1.0" encoding="UTF-8"?>
   <wfs:GetGmlObject
      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.opengis.net/wfs ../wfs/1.1.0/WFS.xsd"
      service="WFS"
      version="1.1.0"
      outputFormat="text/xml; subtype=gml/3.1.1"
      traverseXlinkDepth="2"
      traverseXlinkExpiry="1">
      <ogc:GmlObjectId gml:id="t1"/>
   </wfs:GetGmlObject>
```

and the response:

```
   <Town gml:id="t1">
      <gml:name>Bedford</gml:name>
         <gml:directedNode orientation="+"> <!-- xlink:href="#n1" →
            <gml:Node gml:id=”n1”>
               <gml:pointProperty
xlink:href=”http://www.bedford.town.uk/civilworks/gps.gml#townHall”/>
            </gml:Node>
      </gml:directedNode>
   </Town>
```

Example 4

This example is the same as example 3 except that it requests unlimited nested XLink traversal from a WFS that has a remote GetGmlObject capability:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:GetGmlObject
   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.opengis.net/wfs ../wfs/1.1.0/WFS.xsd"
   service="WFS"
   version="1.1.0"
   outputFormat=" text/xml; subtype=gml/3.1.1"
   traverseXlinkDepth="*"
   traverseXlinkExpiry="2">
   <ogc:GmlObjectId gml:id="t1"/>
</wfs:GetGmlObject>
```

and receives the remote response within the specified timeout period, so that it can be included in the response:

```
<Town gml:id="t1">
   <gml:name>Bedford</gml:name>
   <gml:directedNode orientation="+"> <!-- xlink:href="#n1" →
      <gml:Node gml:id="n1">
         <gml:pointProperty>
      <!-- xlink:href="http://www.bedford.town.uk/civilworks/gps.gml#townHall" -->
            <gml:Point gml:id="townHall" srsName="…">
               <gml:pos>147 234</gml:pos>
            </gml:Point>
         </gml:pointProperty>
      </gml:Node>
   </gml:directedNode>
</Town>
```

11 LockFeature operation

11.1 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. 13].

11.2 Request

11.2.1 Schema definition

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

```
<xsd:element name="LockFeature" type="wfs:LockFeatureType"/>
<xsd:complexType name="LockFeatureType">
   <xsd:complexContent>
      <xsd:extension base="wfs:BaseRequestType">
         <xsd:sequence>
            <xsd:element name="Lock" type="wfs:LockType"
                         maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="expiry"
                        type="xsd:positiveInteger"
                        use="optional" default="5"/>
         <xsd:attribute name="lockAction"
                        type="wfs:AllSomeType"
                        use="optional"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="AllSomeType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="ALL"/>
      <xsd:enumeration value="SOME"/>
   </xsd:restriction>
</xsd:simpleType>
<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 would release the lock. The *expiry* limit is specified in minutes. The lock timer should be started once the entire lock request has been processed and the lock response has been completely transmitted to the client. 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**. Subclause 10.2.2 presents a state machine for the **LockFeature** operation.

11.2.2 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 3.

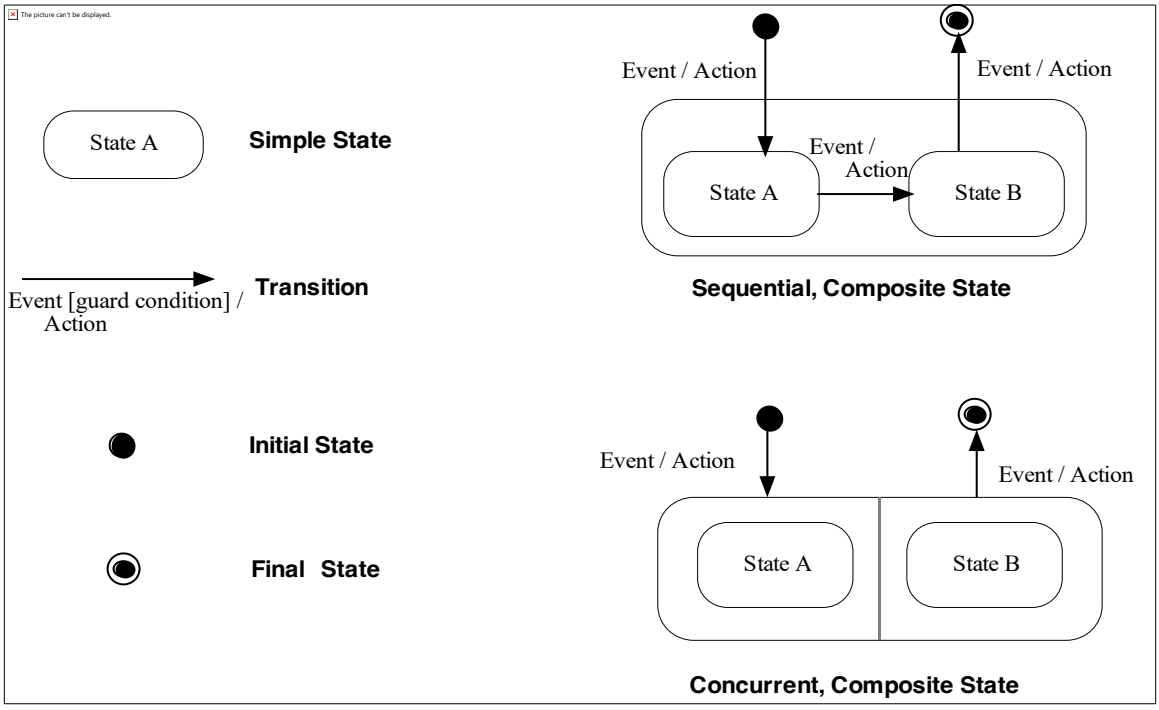

Figure 3 – Summary of UML state diagram notation

11.2.3 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 is 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.

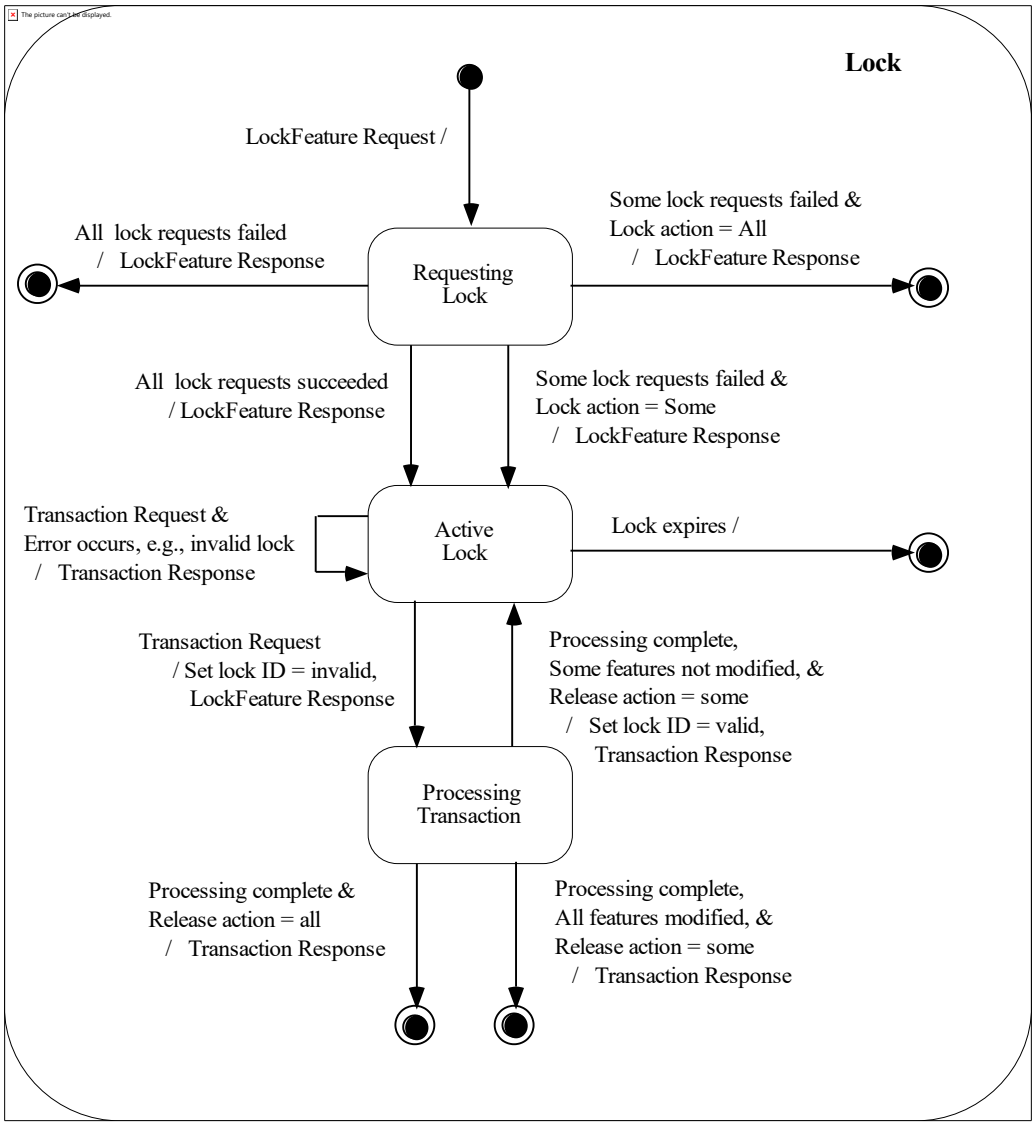

Figure 4 –State diagram for a WFS lock

11.3 Response

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

```
<xsd:element name="LockFeatureResponse"
             type="wfs:LockFeatureResponseType"/>
<xsd:complexType name="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 **ALL** and all identified feature instances can be locked, a WFS must respond with a <**WFS_LockFeatureResponse**> element that contains the <**FeaturesLocked**> element and no <**FeatureNotLocked**> element (since either all identified feature instances can be locked or none at all). If some or all feature instances cannot be locked, a WFS must response with an exception indicating that the lock request failed because some or all feature instances are locked by other clients.

If the lock action is specified as **SOME**, then the <**WFS_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.

If a lock request results in no features being locked, then a WFS should respond with **<WFS_LockFeatureResponse>** document that contains a value for the **lockId** attribute but that contains neither a <**FeaturesLocked**> element nor a <**FeatureNotLocked**> element. In other words, an empty response. After the request is completed, the **lockId** should be immediately released since no resources were locked. If that same **lockId** is used in a subsequent transaction, an exception should be raised since it no longer exists.

11.4 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 subclause 7.7.

11.5 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="1.1.0"
   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/1.1.0/WFS.xsd">
   <Lock typeName="myns:InWaterA_1M">
      <ogc:Filter>
         <ogc:GmlObjectId gml:id="InWaterA_1M.1013"/>
         <ogc:GmlObjectId gml:id="InWaterA_1M.1014"/>
         <ogc:GmlObjectId gml:id="InWaterA_1M.1015"/>
         <ogc:GmlObjectId gml:id="InWaterA_1M.1016"/>
         <ogc:GmlObjectId gml:id="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/1.1.0/WFS.xsd">
   <LockId>1</LockId>
   <FeaturesLocked>
      <ogc:GmlObjectId gml:id="InWaterA_1M.1013"/>
      <ogc:GmlObjectId gml:id="InWaterA_1M.1014"/>
      <ogc:GmlObjectId gml:id="InWaterA_1M.1016"/>
      <ogc:GmlObjectId gml:id="InWaterA_1M.1017"/>
   </FeaturesLocked>
   <FeaturesNotLocked>
      <ogc:GmlObjectId gml:id="InWaterA_1M.1015"/>
   </FeaturesNotLocked>
</WFS_LockFeatureResponse>
```

Example 2

Lock all the feature instances of type InWaterA_1M.

Request:

```
<?xml version="1.0" ?>
<LockFeature
   version="1.1.0"
   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/1.1.0/WFS.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/1.1.0/WFS.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="1.1.0"
   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/1.1.0/WFS.xsd">
   <Lock handle="Lock1" typeName="myns:InWaterA_1M">
      <ogc:Filter>
         <ogc:Within>
            <ogc:PropertyName>myns:wkbGeom</ogc:PropertyName>
            <gml:Polygon
                 srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
               <gml:exterior>
                  <gml:LinearRing>
                  <gml:posList>-95.7 38.1 -97.8 38.2 ...</gml:posList>
                  </gml:LinearRing>
               </gml:exterior>
            </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/1.1.0/WFS.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="1.1.0"
   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/1.1.0/WFS.xsd">
   <Lock handle="LOCK1" typeName="myns:BuiltUpA_1M">
      <ogc:Filter>
         <ogc:Within>
            <ogc:PropertyName>BuiltUpA_1M/wkbGeom</ogc:PropertyName>
            <gml:Polygon gid="1"
               srsName="http://www.opengis.net/gml/epsg.xml#63266405">
               <gml:exterior>
                  <gml:LinearRing>
                  <gml:posList>-95.7 38.1 -97.8 38.2 ...</gml:posList>
                  </gml:LinearRing>
               </gml:exterior>
            </gml:Polygon>
         </ogc:Within>
      </ogc:Filter>
   </Lock>
   <Lock handle="LOCK2" typeName="myns:InWaterA_1M">
      <ogc:Filter>
         <ogc:GmlObjectId gml:id="InWaterA_1M.1212"/>
         <ogc:GmlObjectId gml:id="InWaterA_1M.1213"/>
         <ogc:GmlObjectId gml:id="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/1.1.0/WFS.xsd">
   <LockId>LOCK1A</LockId>
   <FeaturesLocked>
      <ogc:GmlObjectId gml:id="BuiltUpA_1M.1" />
      <ogc:GmlObjectId gml:id="BuiltUpA_1M.10" />
      <ogc:GmlObjectId gml:id="BuiltUpA_1M.34" />
      <ogc:GmlObjectId gml:id="BuiltUpA_1M.786" />
      <ogc:GmlObjectId gml:id="BuiltUpA_1M.3" />
      <ogc:GmlObjectId gml:id="BuiltUpA_1M.13" />
      <ogc:GmlObjectId gml:id="BuiltUpA_1M.47563" />
      <ogc:GmlObjectId gml:id="InWaterA_1M.1212" />
      <ogc:GmlObjectId gml:id="InWaterA_1M.1213" />
      <ogc:GmlObjectId gml:id="InWaterA_1M.10" />
   </FeaturesLocked>
</WFS_LockFeatureResponse>
```

12 Transaction operation

12.1 Introduction

The **Transaction** operation is used to describe data transformation operations that are to be applied to web accessible feature instances. When the transaction has been completed, a web feature service will generate an XML response document indicating the completion status of the transaction.

NOTE: 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.

NOTE: 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 described in clause 13.

12.2 Request

12.2.1 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:complexContent>
      <xsd:extension base="ows:GetCapabilitiesType">
         <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="releaseAction"
                        type="wfs:AllSomeType" use="optional"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:element name="LockId" type="xsd:string"/>
<xsd:element name="Insert" type="wfs:InsertElementType"/>
<xsd:complexType name="InsertElementType">
   <xsd:choice>
      <xsd:element ref="gml:_FeatureCollection" />
      <xsd:sequence>
         <xsd:element ref="gml:_Feature" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:choice>
   <xsd:attribute name="idgen"
                  type="wfs:IdentifierGenerationOptionType"
                  use="optional" default="GenerateNew"/>
   <xsd:attribute name="handle" type="xsd:string" use="optional"/>
   <xsd:attribute name="inputFormat" type="xsd:string"
                  use="optional" default="text/xml; subversion=gml/3.1.1"/>
   <xsd:attribute name="srsName" type="xsd:anyURI" use="optional"/>
</xsd:complexType>
<xsd:simpleType name="IdentifierGenerationOptionType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="UseExisting"/>
```

```
      <xsd:enumeration value="ReplaceDuplicate"/>
      <xsd:enumeration value="GenerateNew"/>
   </xsd:restriction>
</xsd:simpleType>
<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:attribute name="inputFormat" type="xsd:string"
                  use="optional" default="text/xml; subversion=gml/3.1.1"/>
   <xsd:attribute name="srsName" type="xsd:anyURI" use="optional"/>
</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" minOccurs="0"/>
   </xsd:sequence>
</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="Native" type="wfs:NativeType"/>
<xsd:complexType name="NativeType">
   <xsd:attribute name="vendorId" type="xsd:string" use="required"/>
   <xsd:attribute name="safeToIgnore" type="xsd:boolean" use="required"/>
</xsd:complexType>
```

12.2.2 Attribute descriptions

As described in clause 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. This is the default action if no value is specified for the **releaseAction** attribute. A value of **SOME** indicates that only the locks on feature instances modified by the transaction 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.

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.

12.2.3 <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. A web feature service must process <**Insert**>, <**Update**> and <**Delete**> elements in the order in which they are presented in the transaction request. Subsequent update and delete actions, in a transaction request, may operate on feature instances created by previous insert actions in the same transaction request[2]. An empty <**Transaction**> element 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. Clause 11 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 clause 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 clause 7.5.

12.2.4 <Insert> element

The **<Insert>** element is used to create new feature instances. By default, the initial state of a feature to be created is expressed using GML3 and must validate relative to a GML3 application schema generated by the **DescribeFeatureType** operation [sec. 8]. However, the defined **inputFormat** attribute may be used to support older versions of GML. Table 2c defines the possible values for the **inputFormat** attribute:

[2] NOTE: It is not be possible to identify feature instances created in the same transaction request, to be updated or deleted, with a predicate that uses the <**GmlObjectId**> (or <**FeatureId**>) element if those feature instances where create with the value of the **idgen** attribute set to **GenerateNew**. With the **idgen** attribute set to **GenerateNew**, the feature identifiers assigned to those feature instances will not be known until the transaction response is generated.

Table 7 – Possible values for the inputFormat attribute

inputFormat Value	Description
application/gml+xml; version=2.1	This value is specified for backward compatability and indicates that the input features is a GML2 feature instance that validates against a GML2 application schema.
application/gml+xml; version=3.1	This value indicates that the input feauture instance is a GML3 feature that validates against a GML3 application schema. This is the default value for the **inputFormat** attribute if it is not specified on the <Insert> or <Update> element.

Multiple **<Insert>** elements can be enclosed in a single **Transaction** request and multiple feature instances can be created using a single **<Insert>** element.

The optional **srsName** attribute is used to assert the SRS of the incoming feature data, which can be useful if the incoming feature data does not have the SRS declared for each geometry. If the **srsName** attribute exists on an <**Insert**> element, its value must be equivalent to the value of <**DefaultSRS**> or any of the <**OtherSRS**> of the relevant feature types. If, however, the SRS is not supported, the WFS shall raise an exception as described in subclause 7.7. If the **srsName** is not specified on the <**Insert**> element, the WFS shall interpret that the feature data is given in the <**DefaultSRS**>, except where an SRS is specified on an included feature geometry. In this case, if the SRS for such a geometry is one of the <**DefaultSRS**> or <**OtherSRS**> values for the respective feature types, it will be transformed as required before insertion. However, if the aforesaid SRS is not supported for the respective feature type, the entire transaction must fail and the WFS shall raise an exception as described in subclause 7.7. If atomic transactions are not supported by the underlying DBMS, the WFS shall skip any feature with an unsupported SRS and continue.

The **srsName** attribute on the <**Insert**> element cannot be specified if the features types being inserted have no spatial properties, which is indicated via the <**NoSRS**> element in the capabilities document.

In response to an <**Insert**> operation, a web feature service shall generate a list of identifiers assigned to the new feature instances.

Feature identifiers may be generated by a web feature service or specified by the client upon input using **gml:id** attribute values on inserted features and elements. A specific web feature service implementation must support one of these methods of assigning features identifiers to new feature instances and may support all methods. In either case, the capability of the web feature service must be advertised in the capabilities document as described in clause 12.

The **idgen** attribute defined on the <**Insert**> element may be used to indicate which method of assigning feature identifiers is to be used. Table 3 defines the possible values for the attribute and their meaning:

Table 3 – idgen Attribute Values

idgen Value	Action
GenerateNew (default)	The web feature service shall generate unique identifiers for all newly inserted feature instances.
UseExisting	In response to an **<Insert>** operation, a web feature service shall use the gml:id attribute values on inserted features and elements. If any of those IDs duplicates the ID of a feature or element already stored in the WFS, the WFS shall raise an exception.
ReplaceDuplicate	A WFS client can request that the WFS generate IDs to replace the input values of gml:id attributes of feature elements that duplicate the ID of a feature or element already stored in the WFS, instead of raising an exception, by setting the idgen attribute of the InsertElementType to the value "ReplaceDuplicate".

The feature identifiers for successfully inserted features must be presented in the order in which the **<Insert>** operations were encountered in the **Transaction** request. These feature identifiers will have either originated from the input data or have been generated by the WFS according to the identifier policy described above.

Example

The following transaction creates two instances of feature type InWaterA_1M.

```
<?xml version="1.0"?>
<wfs:Transaction
   version="1.1.0"
   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/1.1.0/WFS.xsd">
   <wfs:Insert idgen="UseExisting">
      <InWaterA_1M gml:id="INW1">
         <wkbGeom>
            <gml:Polygon gml:id="P1"
               srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
               <gml:exterior>
                  <gml:LinearRing>
                     <gml:posList>-98.54 24.26 ...</gml:posList>
                  </gml:LinearRing>
               </gml:exterior>
            </gml:Polygon>
         </wkbGeom>
         <id>150</id>
         <f_code>ABCDE</f_code>
         <hyc>152</hyc>
         <tileId>250</tileId>
         <facId>111</facId>
      </InWaterA_1M>
```

```
        <InWaterA_1M gml:id="INW2">
           <wkbGeom>
              <gml:Polygon gml:id="P2"
                 srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
                 <gml:exterior>
                    <gml:LinearRing>
                       <gml:posList>-99.99 22.22 ...</gml:posList>
                    </gml:LinearRing>
                 </gml:exterior>
              </gml:Polygon>
           </wkbGeom>
           <id>111</id>
           <f_code>FGHIJ</f_code>
           <hyc>222</hyc>
           <tileId>333</tileId>
           <facId>444</facId>
        </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.

12.2.5 <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 the NULL value is not allowed.

Using this operation, it is possible to update geometry properties of features. The treatment of **srsName** is identical to that of the same attributed on the **<Insert>** element

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. In the case where the <**Filter**> element does not identify any feature instances to operate on, the update action will simply not have any effect. This is not an exception condition.

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="1.1.0"
   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/1.1.0/WFS.xsd">
   <wfs:Update typeName="BuiltUpA_1M">
      <wfs:Property>
         <wfs:Name>population</wfs:Name>
         <wfs:Value>4070000</wfs:Value>
      </wfs:Property>
      <ogc:Filter>
         <ogc:GmlObjectId gml:id="BuiltUpA_1M.10131"/>
      </ogc:Filter>
   </wfs:Update>
</wfs:Transaction>
```

Example

Update the *populationType* 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 *populationType* attribute set to the value "CITY".

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.1.0"
   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/1.1.0/WFS.xsd">
   <wfs:Update typeName="BuiltUpA_1M">
      <wfs:Property>
         <wfs:Name>populationType</wfs:Name>
         <wfs:Value>CITY</wfs:Value>
      </wfs:Property>
      <ogc:Filter>
         <ogc:GmlObjectId gml:id="BuiltUpA_1M.1013"/>
         <ogc:GmlObjectId gml:id="BuiltUpA_1M.34"/>
         <ogc:GmlObjectId gml:id="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 *tileId* property to values greater than 1000.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.1.0"
   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/1.1.0/WFS.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:GmlObjectId gml:id="BuiltUpA_1M.1013"/>
         <ogc:GmlObjectId gml:id="BuiltUpA_1M.34"/>
         <ogc:GmlObjectId gml:id="BuiltUpA_1M.24256"/>
      </ogc:Filter>
   </wfs:Update>
   <wfs:Update typeName="myns:BuiltUpA_1M">
      <wfs:Property>
         <wfs:Name>myns:facId</wfs:Name>
         <wfs:Value>100</wfs:Value>
      </wfs:Property>
      <ogc:Filter>
         <ogc:PropertyIsGreaterThan>
            <ogc:PropertyName>BuiltUpA_1M/tileId</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_1MTreesA_1M. All features of OceansA_1M with a depth greater than 2400m are updated and feature TreesA_1MTreesA_1M.1010 is also updated.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.1.0"
   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/1.1.0/WFS.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_1MTreesA_1M">
      <wfs:Property>
         <wfs:Name>myns:treeType</wfs:Name>
         <wfs:Value>CONIFEROUS</wfs:Value>
      </wfs:Property>
```

```
      <ogc:Filter>
         <ogc:GmlObjectId gml:id="TreesA_1MTreesA_1M.1010"/>
      </ogc:Filter>
   </wfs:Update>
</wfs:Transaction>
```

12.2.6 <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]**.** In the event that the <**Filter**> element does not identify any feature instances to delete, the delete action will simply have not effect. This is not an exception condition.

Example

Delete a single feature.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.1.0"
   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/1.1.0/WFS.xsd">
   <wfs:Delete typeName="InWaterA_1M">
      <ogc:Filter>
         <ogc:GmlObjectId gml:id="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="1.1.0"
   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/1.1.0/WFS.xsd">
   <wfs:Delete typeName="myns:InWaterA_1M">
      <ogc:Filter>
         <ogc:GmlObjectId gml:id="InWaterA_1M.1013"/>
         <ogc:GmlObjectId gml:id="InWaterA_1M.10"/>
         <ogc:GmlObjectId gml:id="InWaterA_1M.13"/>
         <ogc:GmlObjectId gml:id="InWaterA_1M.140"/>
         <ogc:GmlObjectId gml:id="InWaterA_1M.5001"/>
         <ogc:GmlObjectId gml:id="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="1.1.0"
   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/1.1.0/WFS.xsd">
   <wfs:Delete typeName="InWaterA_1M">
      <ogc:Filter>
         <ogc:Within>
            <ogc:PropertyName>wkbGeom</ogc:PropertyName>
            <gml:Polygon gid="pp9"
               srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
               <gml:exterior>
                  <gml:LinearRing>
                  <gml:posList>-95.7 38.1 -97.8 38.2 ...</gml:posList>
                  </gml:LinearRing>
               </gml:exterior>
            </gml:Polygon>
         </ogc:Within>
      </ogc:Filter>
   </wfs:Delete>
</wfs:Transaction>
```

12.3 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.

The following XML Schema fragment defines the XML encoding of the WFS transaction response:

```
<xsd:element name="TransactionResponse"
             type="wfs:TransactionResponseType"/>
<xsd:complexType name="TransactionResponseType">
   <xsd:sequence>
      <xsd:element name="TransactionSummary"
                   type="wfs:TransactionSummaryType"/>
      <xsd:element name="TransactionResults"
                   type="wfs:TransactionResultsType" minOccurs="0"/>
      <xsd:element name="InsertResults"
                   type="wfs:InsertResultsType" minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="version"
                  type="xsd:string" use="required" fixed="1.1.0"/>
</xsd:complexType>
<xsd:complexType name="TransactionSummaryType">
   <xsd:sequence>
      <xsd:element name="totalInserted"
                   type="xsd:nonNegativeInteger"
                   minOccurs="0"/>
      <xsd:element name="totalUpdated"
                   type="xsd:nonNegativeInteger"
                   minOccurs="0"/>
      <xsd:element name="totalDeleted"
                   type="xsd:nonNegativeInteger"
```

```
                  minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TransactionResultsType">
   <xsd:sequence>
      <xsd:element name="Action" type="wfs:ActionType"
                  minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ActionType">
   <xsd:sequence>
      <xsd:element name="Message" type="xsd:string"
                  minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
   <xsd:attribute name="locator" type="xsd:string" use="required"/>
   <xsd:attribute name="code" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:complexType name="InsertResultsType">
   <xsd:sequence>
      <xsd:element name="Feature"
                  type="wfs:InsertedFeatureType"
                  maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="InsertedFeatureType">
   <xsd:sequence>
      <xsd:element ref="ogc:FeatureId" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="handle" type="xsd:string" use="optional"/>
</xsd:complexType>
```

The **<TransactionResponse>** element contains a <**TransactionSummary**> element, an optional <**TransactionResult**> element and an optional <**InsertResults**> element.

The overall results of a transaction request are summarized using the <**TransactionSummary**> element. The <**TransactionSummary**> element contains the <**totalInserted**>, <**totalUpdated**> and <**totalDeleted**> elements. The <**totalInserted**> element contains a count of the number of new feature instances created. The <**totalUpdated**> element contains a count of the total number of features updated and the <**totalDeleted**> elements contains a count of the number of feature instances deleted.

The optional <**TransactionResult**> element is included to support systems that do not support atomic transactions[3] raising the possibility of partially successful transactions. The contents of the element indicate which actions of a transaction request failed to complete successfully. Not including a <**TransactionResult**> element indicates that all actions of a transaction request completed successfully. The content of the <**TransactionResult**> element is one or more <**Action**> elements. Each <**Action**> element corresponds to an action of a transaction request that failed to complete successfully. The <**Action**> element is correlated with an action of a transaction request using the mandatory **locator** attribute. The value of the **locator** attribute may be equal to the value of the **handle** attribute for a corresponding action (i.e. <**Insert**>, <**Update**> or <**Delete**> element) of a transaction request. If a **handle** value is not specified for an action of a transaction request, then a web feature service may use another method of locating the action (e.g. line numbers, XPath expressions, etc.). The optional **code**

[3] Web Feature Services that implement atomic transaction should raise an exception if any action of a transaction request failed. The exception response format is describe is clause 7.7.

attribute may be used to indicate and error code and the optional <**Message**> element may be used to specify an exception message.

In a transaction request contains insert actions then the <**InsertResults**> element must be specified in the transaction response. The **<InsertResults>** element contains one or more <**Feature**> elements that indicate the feature identifiers of newly created feature instances. One **<Feature>** element is reported for each newly created feature instance. In addition, the <**Feature**> elements must be presented in the order in which the insert actions were encountered in the **<Transaction>** element. Additionally, <**Feature**> elements may be correlated to insert actions using the **handle** attribute. If a value was specified for the **handle** attribute on an <**Insert**> element that created a feature, that same handle value may be specified as the value of the **handle** attribute on the <**Feature**> element to indicate which insert action generated which feature instance.

Example

Consider a transaction request 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="1.1.0"
   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/1.1.0/WFS.xsd">
   <wfs:TransactionSummary>
      <wfs:totalInserted>5</wfs:totalInserted>
   </wfs:TransactionSummary>
   <wfs:InsertResults>
      <wfs:Feature handle="STMT1">
         <ogc:FeatureId fid="SomeFeature.4567"/>
      </wfs:Feature>
      <wfs:Feature handle="STMT1">
         <ogc:FeatureId fid="SomeFeature.4568"/>
      </wfs:Feature>
      <wfs:Feature handle="STMT1">
         <ogc:FeatureId fid="SomeFeature.4569"/>
      </wfs:Feature>
      <wfs:Feature handle="STMT2">
         <ogc:FeatureId fid="Feature1.4569"/>
      </wfs:Feature>
      <wfs:Feature handle="STMT3">
         <ogc:FeatureId fid="Feature2.389345"/>
      </wfs:Feature>
   </wfs:InsertResults>
</wfs:TransactionResponse>
```

12.4 Exceptions

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

If a web feature service supports atomic transactions and encounters an error while processing a particular action contained in a **Transaction** request, then the service shall raise an exception as described in clause 7.7.

If a web feature service does not support atomic transaction and encounters errors while processing a actions contained in a <**Transaction**> element, then the service shall queue the exceptions and report the failures in a **<TransactionResults>** element [sec. 12.3].

Example

In this example, the second statement of a **Transaction** request has failed. A service that does not support atomic transaction might respond with:

```
<?xml version="1.0" ?>
<wfs:TransactionResponse
   version="1.1.0"
   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/1.1.0/WFS.xsd">
   <wfs:TransactionSummary>
      <wfs:totalInserted>4</wfs:totalInserted>
   </wfs:TransactionSummary>
   <wfs:TransactionResults>
      <wfs:Action locator="STMT2" code="9999">
         <wfs:Message>Insert Action Failed</wfs:Message>
      </wfs:Action>
   </wfs:TransactionResults>
   <wfs:InsertResults>
      <wfs:Feature handle="STMT1">
         <ogc:FeatureId fid="SomeFeature.4567"/>
      </wfs:Feature>
      <wfs:Feature handle="STMT1">
         <ogc:FeatureId fid="SomeFeature.4568"/>
      </wfs:Feature>
      <wfs:Feature handle="STMT1">
         <ogc:FeatureId fid="SomeFeature.4569"/>
      </wfs:Feature>
      <wfs:Feature handle="STMT3">
         <ogc:FeatureId fid="Feature2.389345"/>
      </wfs:Feature>
   </wfs:InsertResults>
</wfs:TransactionResponse>
```

12.5 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="1.1.0"
   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/1.1.0/WFS.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>
      <tileId>250</tileId>
      <end_id>111</end_id>
      <location>
         <gml:Point gid="e33"
            srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
            <gml:pos>-98.5485 24.2633</gml:pos>
         </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>
      <segement>
         <designation>SEG_A41</designation>
         <geometry>
            <gml:LineString gid="e3"
               srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
               <gml:posList>...</gml:posList>
            </gml:LineString>
         </geometry>
      </segment>
      <roadType>
         <surfaceType>Asphalt</surfaceType>
         <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#63266405">
            <gml:exterior>
               <gml:LinearRing>
               <gml:posList>-95.7 38.1 -97.8 -38.2 ...</gml:posList>
               </gml:LinearRing>
            </gml:exterior>
         </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#63266405">
            <gml:exterior>
               <gml:LinearRing>
               <gml:posList>-89.8 44.3 -89.9 44.4 ...</gml:posList>
               </gml:LinearRing>
            </gml:exterior>
         </gml:Polygon>
      </bndry>
   </BuiltUpA_1M>
</wfs:Insert>

<!-- Update the name property of the feature instance BuiltUpA_1M.1210 -->
<wfs:Update typeName="BuiltUpA_1M">
   <wfs:Property>
      <wfs:Name>name</wfs:Name>
      <wfs:Value>SMALLVILLE</wfs:Value>
   </wfs:Property>
   <ogc:Filter>
      <ogc:GmlObjectId gml:id="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#63266405">
            <gml:exterior>
               <gml:LinearRing>
               <gml:posList>-89.8 44.3 -89.9 44.4 ...</gml:posList>
               </gml:LinearRing>
            </gml:exterior>
         </gml:Polygon>
      </wfs:Value>
   </wfs:Property>
   <ogc:Filter>
      <ogc:GmlObjectId gml:id="BuiltUpA_1M.1725"/>
   </ogc:Filter>
</wfs:Update>

<!-- Delete the feature instance BuiltUpA_1M.1013. -->
<wfs:Delete typeName="BuiltUpA_1M">
   <ogc:Filter>
      <ogc:GmlObjectId gml:id="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#63266405">
                    <gml:exterior>
                       <gml:LinearRing>
                          <gml:posList>0,0 0,5 5,5 5,0 ...</gml:posList>
                       </gml:LinearRing>
                    </gml:exterior>
                 </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="1.1.0"
     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/1.1.0/WFS.xsd">
     <wfs:TransactionSummary>
        <wfs:totalInserted>2</wfs:totalInserted>
        <wfs:totalUpdated>4</wfs:totalUpdated>
        <wfs:totalDeleted>2</wfs:totalDeleted>
     </wfs:TransactionSummary>
     <wfs:InsertResults>
        <wfs:Feature handle="Statement 1">
           <ogc:FeatureId fid="ElevP_1M.1001"/>
        </wfs:Feature>
        <wfs:Feature handle="ComplexInsert">
           <ogc:FeatureId fid="RoadL_1M.1553"/>
        </wfs:Feature>
        <wfs:Feature handle="Statement 2">
           <ogc:FeatureId fid="BuiltUpA_1M.509876"/>
        </wfs:Feature>
        <wfs:Feature handle="Statement 2">
           <ogc:FeatureId fid="BuiltUpA_1M.509877"/>
        </wfs:Feature>
     </wfs:InsertResults>
  </wfs:TransactionResponse>
```

13 GetCapabilities operation

13.1 Introduction

Every OGC Web Service (OWS), including a Web Feature Service, must have the ability to describe its capabilities by returning service metadata in response to a GetCapabilities request. Specifically, every web feature service must support the KVP encoded form of

the GetCapabilities request over HTTP GET so that a client can always know how to obtain a capabilities document. The KVP encoded GetCapabilities request is described in clause 14.

This section defines the XML encoding for a GetCapabilities request and defines the service metadata generated in response as an XML document that a web feature service must generate to describe its capabilities.

13.2 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:complexContent>
    <xsd:extension base="ows:GetCapabilitiesType">
      <xsd:attribute name="service" type="ows:ServiceType"
                     use="optional" default="WFS"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, `ows:GetCapabilitiesType`, is defined in the OWS Common Implementation Specification [15].

13.3 Response

13.3.1 Response schema

The root element of the response to a **<GetCapabilities>** request is the **<WFS_Capabilities>** element.

It is partially defined be the following XML Schema fragment:

```
<xsd:element name="WFS_Capabilites"
             type="wfs:WFS_CapabilitiesType"
             substitutionGroup="ows:Capabilites"/>
<xsd:complexType name="WFS_CapabilitiesType">
   <xsd:complexContent>
      <xsd:extension base="ows:CapabilitiesBaseType">
         <xsd:sequence>
            <xsd:element ref="wfs:FeatureTypeList" minOccurs="0"/>
            <xsd:element ref="wfs:ServesGMLObjectTypeList" minOccurs="0"/>
            <xsd:element ref="wfs:SupportsGMLObjectTypeList"/>
            <xsd:element ref="ows:Filter_Capabilities"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

The base type, `ows:CapabilitiesBaseType`, is defined in the OWS Common Implementation Specification [15].

The entire GetCapabilities response schema is normatively defined in annex A.4.

13.3.2 Capabilities document

The capabilities response document contains the following sections:

1. **Service Identification section**
   The service identification section provides information about the WFS service itself, as defined in the OWS Common Implementation Specification clause 7.4.3 [15].

2. **Service Provider section**
   The service provider section provides metadata about the organization operating the WFS server as defined in the OWS Common Implementation Specification, clause 7.4.4. [15].

3. **Operation Metadata section**
   The operations metadata section provides metadata about the operations defined in this specification and implemented by a particular WFS server. The contents of the operation metadata section are defined in the OWS Common Implementation Specification, clause 7.4.5. This metadata includes the DCP, parameters and constraints for each operation.

4. **FeatureType list section**
   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 the default SRS, any other supported SRSs, or no SRS whatsoever (for non-spatial feature types), for WFS requests is provided for each feature type.

5. **ServesGMLObjectType list section**
   This section defines the list of GML Object types, not derived from **gml:AbstractFeatureType**, that are available from a web feature service that supports the GetGMLObject operation. These types may be defined in a base GML schema, or in an application schema using its own namespace.

6. **SupportsGMLObjectType list section**
   The Supports GML Object Type section defines the list of GML Object types that a WFS server would be capable of serving if it was deployed to serve data as described by an application schema that either used those GML Object types directly (for non-abstract types), or defined derived types based on those types.

7. **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].

13.3.3 FeatureTypeList section

The purpose of the **<FeatureTypeList>** element is to contain a list of feature types, derived from **gml:AbstractFeatureType**, that a WFS can service and defines the transaction and query operations that are supported on each feature type.

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

```
<xsd:element name="FeatureTypeList" type="wfs:FeatureTypeListType"/>
<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>
<xsd:complexType name="FeatureTypeType">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:QName"/>
      <xsd:element name="Title" type="xsd:string">
      <xsd:element name="Abstract" type="xsd:string" minOccurs="0"/>
      <xsd:element ref="ows:Keywords" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:choice>
         <xsd:sequence>
            <xsd:element name="DefaultSRS" type="xsd:anyURI"/>
            <xsd:element name="OtherSRS" type="xsd:anyURI"
                         minOccurs="0" maxOccurs="unbounded">
         </xsd:sequence>
         <xsd:element name="NoSRS">
           <xsd:complexType/>
         </xsd:element>
      </xsd:choice>
      <xsd:element name="Operations"
                   type="wfs:OperationsType"
                   minOccurs="0"/>
      <xsd:element name="OutputFormats"
                   type="wfs:OutputFormatListType"
                   minOccurs="0"/>
      <xsd:element ref="ows:WGS84BoundingBox"
                   minOccurs="1" maxOccurs="unbounded"/>
      <xsd:element name="MetadataURL"
                   type="wfs:MetadataURLType"
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="OperationsType">
   <xsd:sequence>
      <xsd:element name="Operation"
                   type="wfs:OperationType"
                   maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="OperationType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Insert"/>
      <xsd:enumeration value="Unsert"/>
      <xsd:enumeration value="Delete"/>
      <xsd:enumeration value="Query"/>
      <xsd:enumeration value="Lock"/>
      <xsd:enumeration value="GetGmlObject"/>
```

```
      </xsd:restriction>
   </xsd:simpleType>
   <xsd:complexType name="OutputFormatListType">
      <xsd:sequence maxOccurs="unbounded">
         <xsd:element name="Format" type="xsd:string"/>
      </xsd:sequence>
   </xsd:complexType>
   <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:enumeration value="19115"/>
                     <xsd:enumeration value="19139"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:attribute>
            <xsd:attribute name="format" use="required">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:NMTOKEN">
                     <xsd:enumeration value="text/xml"/>
                     <xsd:enumeration value="text/html"/>
                     <xsd:enumeration value="text/sgml"/>
                     <xsd:enumeration value="txt/plain"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:attribute>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
```

The <**FeatureTypeList**> element contains one <**Operations**> element that defines operations common to all feature types and one or more <**FeatureType**> elements describing each feature type that the service offers.

The possible transaction and query operations that may be specified in the feature type list using <**Operations**> elements are described in Table 5.

Table 5 – Transaction and Query Actions 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 operations 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 operations are inherited by every feature type contained in the **<FeatureTypeList>** element and can be augmented by specifying local transaction and query operations using the <**Operations**> element contained by each <**FeatureType**> element.. 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 actions are defined either globally for all feature types of locally for specified features type, then the default element **<Query>** indicating the query operation 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.
DefaultSRS	The <DefaultSRS> element indicates which spatial reference system shall be used by a WFS to express the state of a spatial feature if not otherwise explicitly identified within a query or transaction request. For example, if a <GetFeature> request specifies no SRS value for the <Query> 'srsName' attribute, any spatial properties of feature data satisfying the request shall be expressed using the <DefaultSRS/> value. The SRS may be indicated one of the supported format models for srsName values (see 9.2).The <DefaultSRS> shall not necessarily be the internal storage SRS used for the feature data, and therefore should not be interpreted as such. If the <DefaultSRS> is different from the internal storage SRS, then the WFS must support a transformation between the <DefaultSRS> and the internal storage SRS. The effects of such a transformation must be considered when determining and declaring the guaranteed data accuracy.
OtherSRS	The <OtherSRS> element is used to indicate other supported SRSs within query transaction and query requests. A 'supported SRS' means that the WFS supports the transformation of spatial properties between the <OtherSRS> and the internal storage SRS. The effects of such a transformation must be considered when determining and declaring the guaranteed data accuracy.
NoSRS	The <NoSRS> element is used for feature types that have no spatial properties, and therefore no SRS whatsoever. It is not a GML requirement for Features and FeatureCollections to have spatial properties. The <NoSRS> element shall never imply, and therefore cannot be used for, semantics of "Unknown SRS". This element is used as an identifying label only, and therefore has no element or attribute content.

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. Some operations may be excluded because they cannot be supported by a WFS implementation for a given storage SRS.
OutputFormats	This is a list of MIME types indicating the output formats that may be generated for a feature type. If this optional element is not specified, then all the result formats listed for the GetFeature operation are assumed to be supported.
WGS84BoundingBox	The <WGS84BoundingBox> element is used to indicate the edges of an enclosing rectangle in decimal degrees of latitude and longitude in WGS84. Its purpose is to facilitate geographic searches by indicating where instances of the particular feature type exist. Since multiple 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 feature 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. Three types are defined at present: 'IS019115' = ISO TC211 19115; 'ISO19139 = ISO TC211 ISO19139; 'FGDC' = FGDC CSDGM.

13.3.4 ServesGMLObjectTypeList section

The following XML Schema fragment defines the contents ServesGMLObjectTypeList section:

```
<xsd:element name="ServesGMLObjectTypeList"
             type="wfs:GMLObjectTypeListType"/>
<xsd:complexType name="GMLObjectTypeListType">
   <xsd:sequence>
      <xsd:element name="GMLObjectType"
                   type="wfs:GMLObjectTypeType"
                   maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="GMLObjectTypeType">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:QName"/>
      <xsd:element name="Title" type="xsd;string" minOccurs="0"/>
      <xsd:element name="Abstract" type="xsd:string" minOccurs="0"/>
      <xsd:element ref="ows:Keywords"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="OutputFormats"
                   type="wfs:OutputFormatListType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

The **ServesGMLObjectTypeList** element contains a list of GML object type names that a web feature service, that supports the **GetGMLObject** operation, offers that are not defined from **gml:AbstractFeatureType**. The elements used to define each offer are identical to those used to describe feature types and are described in detail in Table 6.

13.3.5 SupportsGMLObjectTypeList section

The following XML Schema fragments defines the <SupportsGMLObjectTypeList> element:

```
<xsd:element name="SupportsGMLObjectTypeList" type="wfs:GMLObjectTypeListType"/>
```

The <**SupportsGMLObjectTypeList**> element contains a list of GML object types that a WFS server would be capable of serving if it was deployed to serve data as described by an application schema that either used those GML object types directly (for non-abstract types), or defined derived types based on those types. A WFS server that is capable of serving any GML object types shall include just AbstractGMLType in this list. A WFS server with more restricted capabilities shall list only the base GML object types that it can serve. For example, a WFS server that supports only features with simple geometries might return a list containing **gml:AbstractFeatureType**, **gml:PointType**, **gml:LineStringType**, and **gml:PolygonType**.

13.3.6 Parameters domains and Constraints

The <**ows:Parameter**> and <**ows:Constraint**> elements are defined in the OWS Common Implementation Specification [3] and allow valid domain values and constraints to be defined globally for all operations or locally for specific operations that a web feature service offers.

The following table defines the parameter domains which may be defined in the capabilities document of a web feature service.

Table 6a – Operation parameters

Operation Name	Parameter Name
All operations (Globally)	SrsName
GetCapabilities	AcceptVersions
GetCapabilities	AcceptFormats
GetCapabilities	Sections
DescribeFeatureType	OutputFormat
GetFeature	ResultType
GetFeature	OutputFormat
GetFeatureWithLock	ResultType
GetFeatureWithLock	OutputFormat
GetGMLObject	OutputFormat
LockFeature	LockAction
Transaction	InputFormat
Transaction	Idgen
Transaction	ReleaseAction

In general the domain of a parameter is specific to a web feature service implementation. For example the allowed values for the **srsName** parameter are dictated by the particular transformations that a WFS supports. In some cases, however, a parameter domain is defined in this specification (e.g. **idgen**). In such cases a web feature service may only restrict the domain.

The following table defines constraints which may be specified be a web feature service in its capabilities document.

Table 7b – Operation constraints

Constraint Name	Description
SupportsSOAP	Specifies that the service can process requests using HTTP POST and presented in a SOAP envelope as described in subclause 6.7.
DefaultMaxFeatures	Specifies the default value for the **maxFeatures** attribute of the <GetFeature> element. If the constraint is not specified then there is not limit on the number of features that a GetFeature request may return.
LocalTraverseXlinkScope	Defines the minimum and maximum number of levels, when traversing local xlinks, that a WFS will try to resolve. The value * means to as many all levels. If the constrain is not specified then the WFS will attempt to resolve all levels.
RemoteTraverseXlinkScope	Defines the minimum and maximum number of levels, when traversing remote xlinks, that a WFS will try to resolve. The value * means to as many all levels. If the constrain is not specified then the WFS will attempt to resolve all levels.
DefaultLockExpiry	Define the default lock expiry time in minutes. If the constraint is not specified then locks will be held indefinitely and would require administrator intervention to clear them.

13.4 Exceptions

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

13.5 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/1.1.0/WFS.xsd"/>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:WFS_Capabilites
   xmlns:ows="http://www.opengis.net/ows"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wfs="http://www.opengis.net/wfs"
   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/wfs ../wfs.xsd"
   version="1.1.0"
   updateSequence="0">

   <!-- ============================================================== -->
   <!--    SERVICE IDENTIFICATION SECTION                              -->
   <!-- ============================================================== -->
   <ows:ServiceIdentification>
      <ows:ServiceType>WFS</ows:ServiceType>
      <ows:ServiceTypeVersion>1.1.0</ows:ServiceTypeVersion>
      <ows:Title>OGC Member WFS</ows:Title>
      <ows:Abstract>
         Web Feature Service maintained by NSDI data provider, serving
         FGDC framework layer XXX; contact Paul.Bunyon@BlueOx.org
      </ows:Abstract>
      <ows:Keywords>
         <ows:Keyword>FGDC</ows:Keyword>
         <ows:Keyword>NSDI</ows:Keyword>
         <ows:Keyword>Framework Data Layer</ows:Keyword>
         <ows:Keyword>BlueOx</ows:Keyword>
         <ows:Type>String</ows:Type>
      </ows:Keywords>
      <ows:Fees>None</ows:Fees>
      <ows:AccessConstraints>None</ows:AccessConstraints>
   </ows:ServiceIdentification>

   <!-- ============================================================== -->
   <!--    SERVICE PROVIDER SECTION                                    -->
   <!-- ============================================================== -->
   <ows:ServiceProvider>
      <ows:ProviderName>BlueOx</ows:ProviderName>
      <ows:ProviderSite/>
      <ows:ServiceContact>
         <ows:IndividualName>Paul Bunyon</ows:IndividualName>
         <ows:PositionName>Mythology Manager</ows:PositionName>
         <ows:ContactInfo>
            <ows:Phone>
               <ows:Voice>1.800.BIG.WOOD</ows:Voice>
               <ows:Facsimile>1.800.FAX.WOOD</ows:Facsimile>
            </ows:Phone>
            <ows:Address>
               <ows:DeliveryPoint>North Country</ows:DeliveryPoint>
               <ows:City>Small Town</ows:City>
               <ows:AdministrativeArea>Rural County</ows:AdministrativeArea>
               <ows:PostalCode>12345</ows:PostalCode>
               <ows:Country>USA</ows:Country>

<ows:ElectronicMailAddress>Paul.Bunyon@BlueOx.org</ows:ElectronicMailAddress>
            </ows:Address>
            <ows:OnlineResource xlink:href="http://www.BlueOx.org/contactUs"/>
            <ows:HoursOfService>24x7</ows:HoursOfService>
            <ows:ContactInstructions>
               eMail Paul with normal requsts; Phone Paul for emergency
```

```
            requests; if you get voice mail and your request can't wait,
            contact another mythological figure listed on the contactUs
            page of our web site.
         </ows:ContactInstructions>
      </ows:ContactInfo>
      <ows:Role>PointOfContact</ows:Role>
   </ows:ServiceContact>
</ows:ServiceProvider>

<!-- ================================================================== -->
<!--    OPERATIONS METADATA SECTION                                     -->
<!-- ================================================================== -->
<ows:OperationsMetadata>
   <ows:Operation name="GetCapabilities">
      <ows:DCP>
         <ows:HTTP>
            <ows:Get xlink:href="http://www.BlueOx.org/wfs"/>
         </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="AcceptVersions">
         <ows:Value>1.1.0</ows:Value>
         <ows:Value>1.0.0</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="AcceptFormats">
         <ows:Value>text/xml</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="Sections">
         <ows:Value>ServiceIdentification</ows:Value>
         <ows:Value>ServiceProvider</ows:Value>
         <ows:Value>OperationsMetadata</ows:Value>
         <ows:Value>FeatureTypeList</ows:Value>
         <ows:Value>ServesGMLObjectTypeList</ows:Value>
         <ows:Value>SupportsGMLObjectTypeList</ows:Value>
         <ows:Value>Filter_Capabilities</ows:Value>
      </ows:Parameter>
   </ows:Operation>
   <ows:Operation name="DescribeFeatureType">
      <ows:DCP>
         <ows:HTTP>
            <ows:Get  xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
            <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
         </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="outputFormat">
         <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
      </ows:Parameter>
   </ows:Operation>
   <ows:Operation name="GetFeature">
      <ows:DCP>
         <ows:HTTP>
            <ows:Get  xlink:href="http://www.BlueOx.org/wfs/wfs.cgi?"/>
            <ows:Post xlink:href="http://www.BlueOx.org/wfs/wfs.cgi"/>
         </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="resultType">
         <ows:Value>results</ows:Value>
         <ows:Value>hits</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="outputFormat">
         <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
      </ows:Parameter>
   </ows:Operation>
   <ows:Operation name="GetFeatureWithLock">
      <ows:DCP>
         <ows:HTTP>
            <ows:Post xlink:href="http://www.BlueOx.org/wfs"/>
         </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="resultType">
         <ows:Value>results</ows:Value>
         <ows:Value>hits</ows:Value>
```

```
      </ows:Parameter>
      <ows:Parameter name="outputFormat">
         <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
      </ows:Parameter>
   </ows:Operation>
   <ows:Operation name="GetGMLObject">
      <ows:DCP>
         <ows:HTTP>
            <ows:Post xlink:href="http://www.BlueOx.org/wfs"/>
         </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="outputFormat">
         <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
         <ows:Value>text/xhtml</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="LocalTraverseXLinkScope">
         <ows:Value>0</ows:Value>
         <ows:Value>*</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="RemoteTraverseXLinkScope">
         <ows:Value>0</ows:Value>
         <ows:Value>*</ows:Value>
      </ows:Parameter>
   </ows:Operation>
   <ows:Operation name="LockFeature">
      <ows:DCP>
         <ows:HTTP>
            <ows:Post xlink:href="http://www.BlueOx.org/wfs"/>
         </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="lockAction">
         <ows:Value>ALL</ows:Value>
         <ows:Value>SOME</ows:Value>
      </ows:Parameter>
   </ows:Operation>
   <ows:Operation name="Transaction">
      <ows:DCP>
         <ows:HTTP>
            <ows:Post xlink:href="http://www.BlueOx.org/wfs"/>
         </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="inputFormat">
         <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="idgen">
         <ows:Value>GenerateNew</ows:Value>
         <ows:Value>UseExisting</ows:Value>
         <ows:Value>ReplaceDuplicate</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="releaseAction">
         <ows:Value>ALL</ows:Value>
         <ows:Value>SOME</ows:Value>
      </ows:Parameter>
   </ows:Operation>
   <ows:Parameter name="srsName">
      <ows:Value>EPSG:4326</ows:Value>
      <ows:Value>EPSG:32100</ows:Value>
      <ows:Value>EPSG:32101</ows:Value>
      <ows:Value>EPSG:32102</ows:Value>
   </ows:Parameter>
   <ows:Constraint name="DefaultMaxFeatures">
      <ows:Value>10000</ows:Value>
   </ows:Constraint>
   <ows:Constraint name="LocalTraverseXLinkScope">
      <ows:Value>0</ows:Value>
      <ows:Value>*</ows:Value>
   </ows:Constraint>
   <ows:Constraint name="RemoteTraverseXLinkScope">
      <ows:Value>0</ows:Value>
      <ows:Value>*</ows:Value>
   </ows:Constraint>
```

```
      <ows:Constraint name="DefaultLockExpiry">
         <ows:Value>5</ows:Value>
      </ows:Constraint>
   </ows:OperationsMetadata>

   <!-- ================================================================== -->
   <!--    FEATURE TYPE LIST SECTION                                       -->
   <!-- ================================================================== -->
   <wfs:FeatureTypeList>
      <wfs:FeatureType xmlns:bo="http://www.BlueOx.org/BlueOx">
         <wfs:Name>bo:WoodsType</wfs:Name>
         <wfs:Title>The Great Northern Forest</wfs:Title>
         <wfs:Abstract>
            Describes the arborial diversity of the Great
            Northern Forest.
         </wfs:Abstract>
         <ows:Keywords>
            <ows:Keyword>forest</ows:Keyword>
            <ows:Keyword>north</ows:Keyword>
            <ows:Keyword>woods</ows:Keyword>
            <ows:Keyword>arborial</ows:Keyword>
            <ows:Keyword>diversity</ows:Keyword>
         </ows:Keywords>
         <wfs:DefaultSRS>EPSG:62696405</wfs:DefaultSRS>
         <wfs:OtherSRS>EPSG:32615</wfs:OtherSRS>
         <wfs:OtherSRS>EPSG:32616</wfs:OtherSRS>
         <wfs:OtherSRS>EPSG:32617</wfs:OtherSRS>
         <wfs:OtherSRS>EPSG:32618</wfs:OtherSRS>
         <wfs:OutputFormats>
            <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
         </wfs:OutputFormats>
         <ows:WGS84BoundingBox>
            <ows:LowerCorner>-180 -90</ows:LowerCorner>
            <ows:UpperCorner>180 90</ows:UpperCorner>
         </ows:WGS84BoundingBox>
         <wfs:MetadataURL type="FGDC"
format="text/xml">http://www.ogccatservice.com/csw.cgi?service=CSW&amp;version=2.0.0&
amp;request=GetRecords&amp;constraintlanguage=CQL&amp;constraint="recordid=urn:uuid:4
ee8b2d3-9409-4a1d-b26b-6782e4fa3d59"</wfs:MetadataURL>
      </wfs:FeatureType>
   </wfs:FeatureTypeList>

   <!-- ================================================================== -->
   <!--    SERVES GML OBJECT TYPE LIST SECTION                             -->
   <!-- ================================================================== -->
   <wfs:ServesGMLObjectTypeList>
      <wfs:GMLObjectType xmlns:bo="http://www.BlueOx.org/BlueOx">
         <wfs:Name>bo:OxType</wfs:Name>
         <wfs:Title>Babe's Lineage</wfs:Title>
         <wfs:OutputFormats>
            <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
            <wfs:Format>text/xhmtl</wfs:Format>
         </wfs:OutputFormats>
      </wfs:GMLObjectType>
   </wfs:ServesGMLObjectTypeList>

   <!-- ================================================================== -->
   <!--    SUPPORTS GML OBJECT TYPE LIST SECTION                           -->
   <!-- ================================================================== -->
   <wfs:SupportsGMLObjectTypeList>
      <wfs:GMLObjectType>
         <wfs:Name>gml:AbstractGMLFeatureType</wfs:Name>
         <wfs:OutputFormats>
            <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
            <wfs:Format>text/xhtml</wfs:Format>
         </wfs:OutputFormats>
      </wfs:GMLObjectType>
      <wfs:GMLObjectType>
         <wfs:Name>gml:PointType</wfs:Name>
         <wfs:OutputFormats>
            <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
```

```
            <wfs:Format>text/xhtml</wfs:Format>
         </wfs:OutputFormats>
      </wfs:GMLObjectType>
      <wfs:GMLObjectType>
         <wfs:Name>gml:LineStringType</wfs:Name>
         <wfs:OutputFormats>
            <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
            <wfs:Format>text/xhtml</wfs:Format>
         </wfs:OutputFormats>
      </wfs:GMLObjectType>
      <wfs:GMLObjectType>
         <wfs:Name>gml:PolygonType</wfs:Name>
         <wfs:OutputFormats>
            <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
            <wfs:Format>text/xhtml</wfs:Format>
         </wfs:OutputFormats>
      </wfs:GMLObjectType>
      <wfs:GMLObjectType>
         <wfs:Name>gml:MultiPointType</wfs:Name>
         <wfs:OutputFormats>
            <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
            <wfs:Format>text/xhtml</wfs:Format>
         </wfs:OutputFormats>
      </wfs:GMLObjectType>
      <wfs:GMLObjectType>
         <wfs:Name>gml:MultiCurveType</wfs:Name>
         <wfs:OutputFormats>
            <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
            <wfs:Format>text/xhtml</wfs:Format>
         </wfs:OutputFormats>
      </wfs:GMLObjectType>
      <wfs:GMLObjectType>
         <wfs:Name>gml:MultiSurfaceType</wfs:Name>
         <wfs:OutputFormats>
            <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
            <wfs:Format>text/xhtml</wfs:Format>
         </wfs:OutputFormats>
      </wfs:GMLObjectType>
      <wfs:GMLObjectType>
         <wfs:Name>gml:AbstractMetaDataType</wfs:Name>
         <wfs:OutputFormats>
            <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
            <wfs:Format>text/xhtml</wfs:Format>
         </wfs:OutputFormats>
      </wfs:GMLObjectType>
      <wfs:GMLObjectType>
         <wfs:Name>gml:AbstractTopologyType</wfs:Name>
         <wfs:OutputFormats>
            <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
            <wfs:Format>text/xhtml</wfs:Format>
         </wfs:OutputFormats>
      </wfs:GMLObjectType>
   </wfs:SupportsGMLObjectTypeList>

   <!-- ================================================================= -->
   <!--    FILTER CAPABILITIES SECTION                                    -->
   <!-- ================================================================= -->
   <ogc:Filter_Capabilities>
      <ogc:Spatial_Capabilities>
         <ogc:GeometryOperands>
            <ogc:GeometryOperand>gml:Envelope</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Point</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:LineString</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Polygon</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:ArcByCenterPoint</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:CircleByCenterPoint</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Arc</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Circle</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:ArcByBulge</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Bezier</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Clothoid</ogc:GeometryOperand>
```

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

14 Keyword-value pair encoding

14.1 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.

14.1.1 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/wkbGeom<PropertyName>
<gml:Envelope><gml:lowerCorner>10 10</gml:lowerCorner><gml:upperCorner>20
20</gml:upperCorner></gml:Envelope></Within></Filter>
```

should more correctly be specified as:

```
FILTER=<Filter xmlns="http://www.opengis.net/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/1.1.0/filter.xsd
                                   http://www.opengis.net/gml
                                   ../gml/3.1.1/base/gml.xsd">
       <Within><PropertyName>InWaterA_1M/wkbGeom<PropertyName>
       <gml:Envelope><gml:lowerCorner>10 10</lowerCorner>
       <gml:upperCorner>20,20</gml:upperCorner></gml:Envelope>
       </Within></Filter>
```

In order to not obfuscate the essential information, however, the namespace and schema location attribute tags have been omitted from the examples in this 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.

14.2 Request parameter rules

14.2.1 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 web feature service **must** be prepared to encounter parameters that are not part of this specification. In terms of producing results per this specification, a web feature service **shall** ignore such parameters.

14.2.2 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; "(", ")". The characters ",", "(" and ")" may be escaped using the "\" character.

Example 1

This example shows a list of items.

```
parameter=item1,item2,item3,item4a\,item4b
```

This list consists of 4 values: item1, item2, item3 and the value "item4a,item4b". Notice that the last comma has been escaped using the "\" character which means that it is actually part of the value and not a delimiter.

Example 2

This example shows 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>)
```

14.3 Common request parameters

14.3.1 Version parameter

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

14.3.2 Request parameter

The **REQUEST** parameter indicates which service operation is being invoked. The value *operation_name* **must** be one of those offered by the web feature service.

14.3.3 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 KVP encoding for a bounding box is defined in subclause 10.2.3 of normative reference [15]. The general form of the parameter is:

BBOX=lcc1,lcc2,…,lccN,ucc1,ucc2,…uccN[,crsuri]

where *lcc* means Lower Corner Coordinate, *ucc* means Upper Corner Coordinate and *crsuri* means the URI reference to the coordinate system being used. This encoding allows N coordinates for each corner listed in the order of the optional *crsuri*. If the *crsuri* is not specified then the 2-D coordinates shall be specified using decimal degrees and WGS84 as described in [15].

14.3.4 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.

14.4 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[4]	DEFAULT	Description
http:/server_address/path/script	M		URL prefix of web feature service
VERSION	M[5]	1.1.3	Request version.

4 O = Optional, M=Mandatory

5 VERSION is mandatory for all operations **except** the GetCapabilities operation.

SERVICE	M	WFS	Service type.
REQUEST	M		Name of WFS request.
NAMESPACE	O		Used to specify a namespace and its prefix. The format must be *xmlns(prefix=escaped_url)* where *escaped_url* is defined in [17]. If the prefix is not specified then this is the default namespace. More that one namespace may be bound by specifying a comma seperated list of *xmlns()* values.
Additional parameters			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 subclause 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.

The optional parameter **NAMESPACE** is included to support qualified feature and property names such as *myns:InWaterA_1M* where the prefix *myns* must be bound to a particular namespace. The format of the parameter values is:

xmlns(*prefix=EscapedNamespaceName*)

where *EscapedNamespaceName* is the escaped url of the namespace and is defined in [17]. The *prefix* may be omitted to denote the default namespace. The **NAMESPACE** parameter may contain a comma separated list of xmlns() values in order to bind all the namespaces being referenced in a KVP encoded request.

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 that 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.

14.5 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.

14.6 Exceptions

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

14.7 Operations

14.7.1 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.

14.7.2 DescribeFeatureType operation

14.7.2.1 Request

Table 14 – 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	application/gml+xml; version=3.1 Value "text/xml; subtype=gml/3.1.1" may also be used but us deprecated and support is only required for backward compatibility.	The output format to use to describe feature types. application/gml+xml; version=3.1 and text/xml; subtype=gml/3.1.1 must be supported. Other output formats, such as DTD are possible.

14.7.2.2 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=1.1.0&
       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=1.1.0&
       REQUEST=DescribeFeatureType&
       TYPENAME=TreesA_1M,BuiltUpA_1M
```

14.7.3 GetFeature & GetFeatureWithLock operation

14.7.3.1 Request

Table 9a – GetFeature & GetFeatureWithLock encoding

URL Component	O/M	DEFAULT	Description
REQUEST=[GetFeature | GetFeatureWithLock]	M	application/gml+xml; version=3.1	The name of the WFS request.
OUTPUTFORMAT	O	Value “text/xml; subtype=gml/3.1.1” may also be used but is deprecated and support is only required for backward compatibility.	The output format to use for the response. application/gml+xml; version=3.1 and **text/xml; subtype=gml/3.1.1** must be supported. Other output formats are possible as well as long as their MIME type is advertised in the capabilities document.
RESULTTYPE	O	results	The resulttype parameter is used to indicate whether a WFS should generate a complete response document of whether it should generate an empty response document indicating only the number of features that the query would return. A value of **results** indicates that a full response should be generated. A value of **hits** indicates that only a count of the number of features should be returned.
PROPERTYNAME	O		A list of properties may be specified for each feature type that is being queried. Refer to subclause 14.2.2 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.
EXPIRY=N	O		This parameter may only be specified if the request is GetFeatureWithLock. It indicates the length of time (in minutes) that a lock will be held on the features in the result set. If the parameter is not specified then the locks will be held indefinitely.
SRSNAME	O		This parameter is used to specify a WFS-supported SRS that should be used for returned feature geometries. The value may be the DefaultSRS or any of the OtherSRS values that a WFS declares it supports in the capabilities document. The SRS may be indicated using one of the supported format models for srsName values (see 9.2). If the parameter is not specified then the value of the DefaultSRS for the feature type being queried shall be used.
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 subclause 13.3.3.

SORTBY	O		The SORTBY parameter is used to specify a list of property names whose values should be used to order (upon presentation) the set of feature instances that satify the query. The value of the SORTBY parameter shall have the form "*PropertyName [A|D][,PropertyName [A|D],...]*" where the letter A is used to indicate an ascending sort and the letter D is used to indicate a descending sort. If neither A nor D are specified, the default sort order shall be ascending. An example value might be: *"SORTBY=Field1 D,Field2 D,Field3"*. In this case the results are sorted by Field 1 descending, Field2 descending and Field3 ascending.

Additional URL components may be used to control nested XLink traversal and resolution for the **GetFeature** encoding. Note that either the TRAVRSEXLINKDEPTH and TRAVERSEXLINKEXPIRY components should be used to set XLink traversal conditions that apply to all properties in all features in the request, or the PROPTRAVXLINKDEPTH and PROPTRAVXLINKEXPIRY components should be used to provide lists of XLink traversal conditions that match the lists of selected feature properties.

Table 9b – Additional GetFeature encoding

URL Component	O/M	DEFAULT	**Description**
TRAVERSEXLINKDEPTH	O		The depth to which nested property XLink linking element locator attribute (href) XLinks are traversed and resolved if possible for all properties, assuming a PROPTRAVXLINKDEPTH list is not specified. The range of valid values consists of non-negative[15] integers plus "*" for unlimited.
TRAVERSEXLINKEXPIRY	O		The number of minutes a WFS should wait to receive a response to a nested GetGmlObject operation performed for any property (if no PROPEXPIRY list is specified) because a TRAVERSEXLINKDEPTH component is present. If no value is specified then the period is implementation dependent.

PROPTRAVXLINKDEPTH	O		A list of XLink traversal depths may be specified for each feature type that is being queried. Refer to subclause 14.2.2 on how to form lists of parameters. There is a 1:1 mapping between each element in a FEATUREID or TYPENAME list and the PROPTRAVXLINKDEPTH list. The range of valid values consists of non-negative integers plus "*" for unlimited.
PROPTRAVXLINKEXPIRY	O		A list of XLink traversal expiry times in minutes may be specified for each feature type that is being queried. Refer to cubclause 14.2.2 on how to form lists of parameters. There is a 1:1 mapping between each element in a FEATUREID or TYPENAME list and the PROPTRAVXLINKEXPIRY list.

14.7.3.2 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/wkbGeom<PropertyName>
       <gml:Envelope><gml:lowerCorner>10 10</gml:lowerCorner>
       <gml:upperCorner>20 20</gml:upperCorner></gml:Envelope>
       </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/1.1.0/filter.xsd
                                   http://www.opengis.net/gml
                                   ../gml/3.1.1/base/gml.xsd">
       <Within><PropertyName>InWaterA_1M/wkbGeom<PropertyName>
       <gml:Envelope><gml:lowerCorner>10 10</gml:lowerCorner>
       <gml:upperCorner>20 20</gml:upperCorner></gml:Envelope>
       </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=1.1.0&
       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=1.1.0&
       REQUEST=GetFeature&
       PROPERTYNAME=InWaterA_1M/wkbGeom,InWaterA_1M/tileId&
       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=1.1.0&
       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=1.1.0&
       REQUEST=GetFeature&
       PROPERTYNAME=InWaterA_1M/wkbGeom,InWaterA_1M/tileId&
       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=1.1.0&
       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 wkbGeom and tileId properties are selected for each feature instance.

```
http://www.someserver.com/wfs.cgi&
     SERVICE=WFS&
     VERSION=1.1.0&
```

```
     REQUEST=GetFeature&
     PROPERTYNAME=(InWaterA_1M/wkbGeom,InWaterA_1M/tileId)
                  (InWaterA_1M/wkbGeom,InWaterA_1M/tileId)
                  (InWaterA_1M/wkbGeom,InWaterA_1M/tileId)&
     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=1.1.0&
       REQUEST=GetFeature&
       TYPENAME=InWaterA_1M&
       FILTER=<Filter><Within><PropertyName>InWaterA_1M/wkbGeom<PropertyName>
              <gml:Envelope><gml:lowerCorner>10 10<gml:lowerCorner>
              <gml:upperCorner>20 20</gml:upperCorner></gml:Envelope>
              </Within></Filter>
```

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=1.1.0&
       REQUEST=GetFeature&
       PROPERTYNAME=InWaterA_1M/wkbGeom,InWaterA_1M/tileId&
       TYPENAME=InWaterA_1M&
       FILTER=<Filter><Within><PropertyName>InWaterA_1M/wkbGeom<PropertyName>
       <gml:Envelope><gml:lowerCorner>10,10</gml:lowerCorner>
       <gml:upperCorner>20 20</gml:upperCorner></gml:Envelope></Within></Filter>
```

Example 9

Query all properties of a list of feature types in different namespaces:

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.1.0&
       REQUEST=GetFeature&
       NAMESPACE=xmlns(myns=http://www.someserver.com),
                 xmlns(yourns=http://www.someotherserver.com)
       TYPENAME=myns:InWaterA_1M,your:BuiltUpA_1M
```

Example 10

Query some properties of a list of feature types. In this case, the attributes wkbGeom and tileId are fetched for the InWaterA_1M feature type and all the attributes of feature type BuiltUpA_1M are fetched.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.1.0&
       REQUEST=GetFeature&
       PROPERTY=(InWaterA_1M/wkbGeom,InWaterA_1M/tileId)(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=1.1.0&
       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 wkbGeom and tileId attributes are fetched for feature instance "InWaterA_1M.1013". The attribute wkbGeom is fetched for feature instance "BuiltUpA_1M.3456".

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.1.0&
       REQUEST=GetFeature&
       PROPERTYNAME=InWaterA_1M/wkbGeom,InWaterA_1M/tileId,BuiltUpA_1M/wkbGeom)&
       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=1.1.0&
       REQUEST=GetFeature&
       TYPENAME=InWaterA_1M,BuiltUpA_1M&
       FILTER=(<Filter><Within><PropertyName>InWaterA_1M/wkbGeom
              <PropertyName><gml:Envelope><gml:lowerCorner>10 10</gml:lowerCorner>
              <gml:upperCorner>20 20</gml:upperCorner></gml:Envelope>
              </Within></Filter>)(<Filter><Within><PropertyName>
              BuiltUpA_1M/wkbGeom<PropertyName><gml:Envelope><gml:lowerCorner>10 10
              </gml:lowerCorner><gml:upperCorner>20 20</gml:upperCorner>
              </gml:Envelope></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=1.1.0&
       REQUEST=GetFeature&
       PROPERTYNAME=(InWaterA_1M/wkbGeom,InWaterA_1MtileId)(BuiltUpA_1M/wkbGeom)&
       TYPENAME=InWaterA_1M,BuiltUpA_1M&
       FILTER=(<Filter><Within><PropertyName>InWaterA_1M/wkbGeom|InWaterA_1M/wkbGeom
              <PropertyName><gml:Envelope><gml:lowerCorner>10 10</gml:lowerCorner>
              <gml:upperCorner>20,20</gml:upperCorner>
              </gml:Envelope></Within></Filter>)(<Filter><Within><PropertyName>
              InWaterA_1M/wkbGeom<PropertyName><gml:Envelope><gml:lowerCorner>10 10
              </gml:lowerCorner><gml:upperCorner>20 20</gml:upperCorner>
              </gml:Envelope></Within></Filter>)
```

Example 15

This example illustrates a **GetFeature** request using **PROPERTYNAME** components that serves as a base for comparison with Examples 16 and 17 . The request:

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.1.0&
       REQUEST=GetFeature&
       PROPERTYNAME=uk:Town/gml:name,uk:Town/gml:directedNode&
       TYPENAME=uk:Town&
       FEATUREID=t1
```

The response contains an **xlink:href**, but it is returned as-is:

```
<?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/1.1.0/WFS.xsd">
   <gml:boundedBy>
      <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
         <gml:lowerCorner>10 10</gml:lowerCorner>
         <gml:upperCorner>20 20</gml:upperCorner>
      </gml:Envelope>
   </gml:boundedBy>
   <gml:featureMember>
      <Town gml:id="t1">
         <gml:name>Bedford</gml:name>
         <gml:directedNode orientation="+" xlink:href="#n1"/>
      </Town>
   </gml:featureMember>
</wfs:FeatureCollection>
```

Example 16

This example illustrates the use of the **TRAVERSEXLINKDEPTH** and **TRAVERSEXLINKEXPIRY** components on the **GetFeature** request from Example 15.

The request:

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.1.0&
       REQUEST=GetFeature&
       PROPERTYNAME=uk:Town/gml:name,uk:Town/gml:directedNode&
       TRAVERSEXLINKDEPTH=1&
       EXPIRY=1&
       TYPENAME=uk:Town&
       FEATUREID=t1
```

In the response, the first level xlink:href has been traversed, and its contents returned as described for the **GetGmlObject** operation:

```
<?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/1.1.0/WFS.xsd">
   <gml:boundedBy>
      <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
         <gml:lowerCorner>10 10</gml:lowerCorner>
         <gml:upperCorner>20 20</gml:upperCorner>
      </gml:Envelope>
   </gml:boundedBy>
   <gml:featureMember>
      <Town gml:id="t1">
         <gml:name>Bedford</gml:name>
         <gml:directedNode orientation="+"> <!-- xlink:href="#n1" →
            <gml:Node gml:id=”n1”>
               <gml:pointProperty
                     xlink:href=”http://www.bedford.town.uk/civilworks/gps.gml#townHall”/>
            </gml:Node>
         </gml:directedNode>
      </Town>
   </gml:featureMember>
</wfs:FeatureCollection>
```

Example 17

This example illustrates the use of the **PROPTRAVXLINKDEPTH** and **PROPTRAVXLINKEXPIRY** components on the **GetFeature** request from Example 16.

The request:

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.1.0&
       REQUEST=GetFeature&
       PROPERTYNAME=uk:Town/gml:name,uk:Town/gml:directedNode&
       PROPTRAVXLINKDEPTH=0,2&
       PROPERTYEXPIRY=0,2&
       TYPENAME=uk:Town&
       FEATUREID=t1
```

In the response, the first and second level **xlink:hrefs** in the **gml:directedNode** property have been traversed, and their contents returned as described for the **GetGmlObject** operation:

```
<?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/1.1.0/WFS.xsd">
   <gml:boundedBy>
      <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
         <gml:lowerCorner>10 10</gml:lowerCorner>
         <gml:upperCorner>20 20</gml:upperCorner>
      </gml:Envelope>
   </gml:boundedBy>
   <gml:featureMember>
      <Town gml:id="t1">
         <gml:name>Bedford</gml:name>
         <gml:directedNode orientation="+"> <!-- xlink:href="#n1" →
            <gml:Node gml:id="n1">
               <gml:pointProperty>
<!--  xlink:href="http://www.bedford.town.uk/civilworks/gps.gml#townHall" -->
                  <gml:Point gml:id="townHall">
                     <gml:pos>147 234</gml:pos>
```

```
                </gml:Point>
              </gml:pointProperty>
            </gml:Node>
          </gml:directedNode>
        </Town>
    </gml:featureMember>
</wfs:FeatureCollection>
```

14.7.4 GetGmlObject operation

14.7.4.1 Request

Table 17 - GetGmlObject encoding

URL Component	O/M	DEFAULT	Description
REQUEST=[GetGmlObject]	M		The name of the WFS request.
TRAVERSEXLINKDEPTH	M		The depth to which nested property XLink linking element locator attribute (href) XLinks are traversed and resolved if possible. The range of valid values consists of non-negative integers plus "*" for unlimited.
TRAVERSEXLINKEXPIRY	O		The number of minutes a WFS should wait to receive a response to a nested GetGmlObject request.. If no value is specified then the period is implementation dependent.
GMLOBJECTID	M		The XML ID of the element to fetch.

14.7.4.2 Examples

Example 1

This example shows a GetGmlObject request element that fetches the feature with the identifier "InWaterA_1M.1234".

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.1.0&
       REQUEST=GetGmlObject&
       TRAVERSEXLINKDEPTH=1&
       EXPIRY=1&
       GMLOBJECTID=InWaterA_1M.1234
```

Example 2

This example shows a GetGmlObject request element that fetches the feature with the identifier "t1" without performing any nested XLink traversal:

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.1.0&
       REQUEST=GetGmlObject&
       TRAVERSEXLINKDEPTH=1&
       EXPIRY=1&
       GMLOBJECTID=t1
```

and the response:

```
<Town gml:id="t1">
   <gml:name>Bedford</gml:name>
   <gml:directedNode orientation="+" xlink:href="#n1"/>
</Town>
```

Example 3

This example is the same as example 2 except that it requests nested XLink traversal at depth 2:

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.1.0&
       REQUEST=GetGmlObject&
       TRAVERSEXLINKDEPTH=2&
       EXPIRY=1&
       GMLOBJECTID=t1
```

and the response:

```
<Town gm:id="t1">
   <gml:name>Bedford</gml:name>
   <gml:directedNode orientation="+"> <!-- xlink:href="#n1" →
      <gml:Node gml:id="n1">
         <gml:pointProperty
         xlink:href="http://www.bedford.town.uk/civilworks/gps.gml#townHall"/>
      </gml:Node>
   </gml:directedNode>
</Town>
```

Example 4

This example is the same as example 3 except that it requests unlimited nested XLink traversal from a WFS that has a remote GetGmlObject capability:

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.1.0&
       REQUEST=GetGmlObject&
       TRAVERSEXLINKDEPTH=*&
       EXPIRY=2&
       GMLOBJECTID=t1
```

and receives the remote response within the specified timeout period, so that it can be included in the response:

```
<Town gml:id="t1">
   <gml:name>Bedford</gml:name>
   <gml:directedNode orientation="+"> <!-- xlink:href="#n1" à
      <gml:Node gml:id="n1">
         <gml:pointProperty>
            <!-- xlink:href="http://www.bedford.town.uk/civilworks/gps.gml#townHall" -->
            <gml:Point gml:id="townHall">
               <gml:pos>147 234</gml:pos>
            </gml:Point>
         </gml:pointProperty>
      </gml:Node>
   </gml:directedNode>
</Town>
```

14.7.5 LockFeature operation

14.7.5.1 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 subclause 13.3.3.

14.7.5.2 Examples

Example 1

The following example locks all instances of feature type InWaterA_1M.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.1.0&
       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=1.1.0&
        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=1.1.0&
        REQUEST=LockFeature&
        TYPENAME=InWaterA_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=1.1.0&
     REQUEST=LockFeature&
     LOCKACTION=ALL&
     TYPENAME=INWATER_1M,BuiltUpA_1M&
     FILTER=(<Filter><Within><PropertyName>wkbGeom</PropertyName><gml:Envelope>
             <gml:lowerCorner>10 10</gml:lowerCorner>
             <gml:upperCorner>20 20</gml:upperCorner></gml:Envelope></Within>
             </Filter>)
              (<Filter><Within><PropertyName>wkbGeom</PropertyName><gml:Envelope>
             <gml:lowerCorner>10 10</gml:lowerCorner>
             <gml:upperCorner>20 20</gml:upperCorner></gml:Envelope></Within>
             </Filter>)
```

14.7.6 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.

14.7.6.1 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 subclause 13.3.3.

14.7.6.2 Examples

Example 1

Delete the feature instance identified by "RoadL_1M.1013".

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.1.0&
       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=1.1.0&
       REQUEST=Transaction&
       OPERATION=Delete&
       TYPENAME=InWaterA_1M,BuiltUpA_1M&
       FILTER=(<Filter><Within><PropertyName>InWaterA_1M/wkbGeom<PropertyName>
              <gml:Envelope>
              <gml:lowerCorner>10 10</gml:lowerCorner>
              <gml:upperCorner>20 20</gml:upperCorner></gml:Envelope></Within>
```

```
</Filter>)(< Filter><Within><PropertyName>BuiltUpA_1M/wkbGeom
<PropertyName><gml:Envelope><gml:lowerCorner>10 10</gml:lowerCorner>
<gml:upperCorner>20,20</gml:upperCorner>
</gml:Envelope></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=1.1.0&
       REQUEST=Transaction&
       OPERATION=Delete&
       TYPENAME=InWaterA_1M,BuiltUpA_1M&
       BBOX=10,10,20,20
```

14.7.7 GetCapabilities Operation

14.7.7.1 Request

Table 12 – GetCapabilities encoding

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

14.7.7.2 Examples

Example 1

Request the capabilities document from a WFS.

```
http://www.someserver.com/wfs.cgi?
       VERSION=1.1.0&
       SERVICE=WFS&
       REQUEST=GetCapabilities
```

Annex A - XML Schema definitions (Normative)

A Introduction

In order to keep this document to a reasonable length, the normative schemas are not included inline but are attached to the archive package that includes this document. Optionally, the schemas may be obtained at http://schemas.opengis.net/wfs/1.1.

The files that make up the WFS schemas are:

```
1.1.0/wfs.xsd
1.1.0/examples/WFS_Capabilities_Sample.xml
1.1.0/wsdl/dependencies.jpeg
1.1.0/wsdl/example-endpoints.wsdl
1.1.0/wsdl/example-GET-endpoints.wsdl
1.1.0/wsdl/example-POST-endpoints.wsdl
1.1.0/wsdl/example-SOAP-endpoints.wsdl
1.1.0/wsdl/readme.txt
1.1.0/wsdl/wfs-http-bindings.wsdl
1.1.0/wsdl/wfs-kvp-bindings.wsdl
1.1.0/wsdl/wfs-kvp-interfaces.wsdl
1.1.0/wsdl/wfs-kvp.xsd
1.1.0/wsdl/wfs-responses.wsdl
1.1.0/wsdl/wfs-soap-bindings.wsdl
1.1.0/wsdl/wfs-util.xsd
1.1.0/wsdl/wfs-xml-interfaces.wsdl
1.1.0/wsdl/WSDL2Java.bat
```

Annex B - Conformance Testing

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

Annex C - UML model (informative)

This annex provides a UML model of the WFS interface, using the OGC/ISO profile of UML summarized in Subclause 5.4.

Figure 5 is a draft UML diagram summarizing the WFS interface. This class diagram shows that the WebFeatureService class inherits the getCapabilities operation from the abstract OGCWebService class, which is common to all OGC Web Services. The WebFeatureService class adds the describeFeatureType, GetFeature, GetFeatureWithLock, “transaction”, and lockFeature operations. (The capitalization of class, operation, and data type names uses the OGC/ISO profile of UML.)

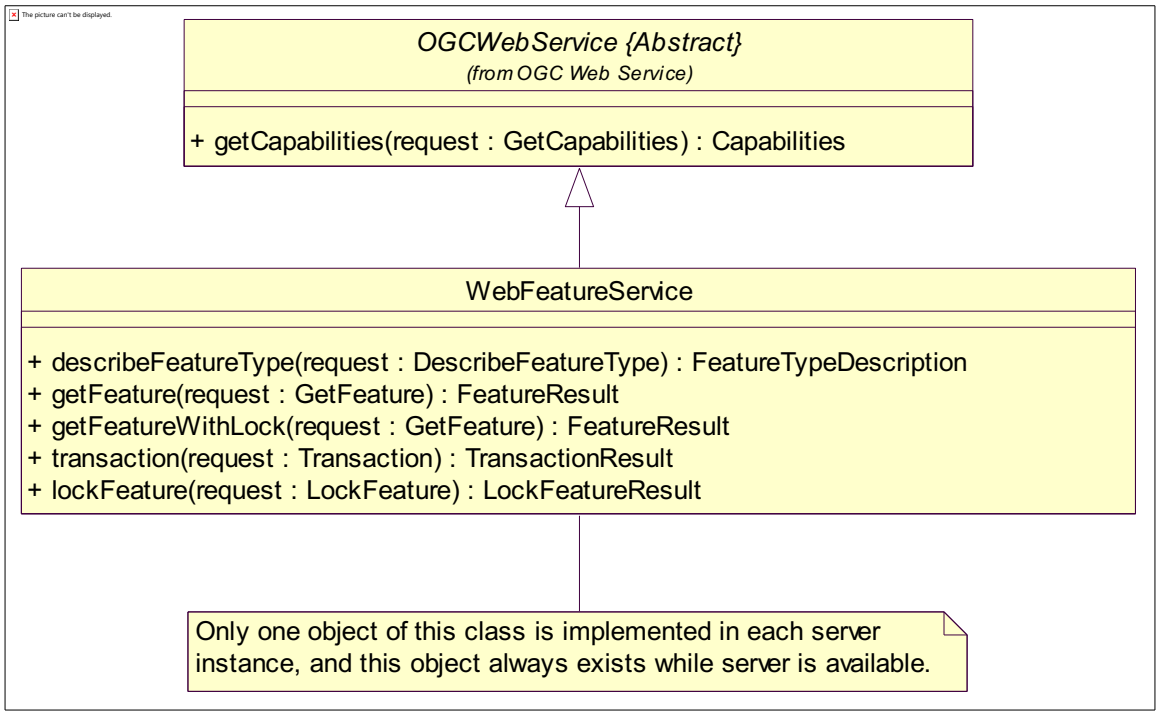

Figure 5 — WFS interface UML diagram

Each of the three operations uses a request and a response data type, each of which can also be defined by one or more additional UML classes in future more-complete class diagrams.

GLOSSARY

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