All Fields marked with * are mandatory.

Change Request #:	**157**
Assigned OGC Document #:	**11-080**
Name:	*Panagiotis (Peter) A. Vretanos
Organization:	*CubeWerx Inc.
Email:	*pvretano@cubewerx.com
Document Name/Version:	*OpenGIS Web Feature Service 2.0 Interface Standard (also ISO 19142) / 2.0
OGC Project Document:	*09-025r1

If this is a revision of a previous submission and you have a Change Request Number, then check here: ☐
Enter the CR number here:
Enter the Revsion Number that you are revising here:

Title:	*A REST binding for WFS 2.0
Source:	*Self
Work item code:	
Category:	* B (Addition of feature)
Reason for change:	* The WFS specification already defines an OGC XML-POST binding, an OGC KVP-GET binding and a SOAP binding. The other popular service binding, which the WFS does not currently support, is REST. This change proposal attempts to address this omission.
Summary of change:	* See attached Notes.
Consequences if not approved:	

Clauses affected:	* Clause 8.3 plus the addition of a new ANNEX.
Additional Documents affected:	OWS Common 2.0
Supporting Documentation:	
Comments:	
Status:	Assigned
Assigned To:	WFS WG
Disposition:	Referred and Posted

ANNEX X - REST binding for WFS

Table of contents

X.1 Introduction
X.2 Conformance classes
X.3 Basic service elements
X.3.1 Service root
X.3.2 Service parameter
X.3.3 Version negotiation
X.3.4 Feature representation
X.3.5 Content negotiation
X.3.6 Exceptions
X.3.7 Authentication and Access Control
X.3.8 Summary of resources
X.3.9 Content types
X.3.10 A word about examples
X.4 Service metadata
X.5 Feature type schemas
X.6 Feature access and management
X.7 Advanced queries
X.7.1 Ad-hoc queries
X.7.2 Stored queries
X.8 Complex Transactions
X.9 Capabilities document example

X.1 Introduction

- this annex describes a REST binding for WFS 2.0
  - it is anticipated that this binding will establish a pattern for other OGC data access services (e.g. CSW)
- the characteristics of a RESTful service are:
  - -> URIs -- every URI designates exactly one resource
  - -> addressability -- all the interesting aspects of a service are exposed as resources (e.g. features)
  - -> statelessness -- every HTTP request includes all necessary information for the server to fulfill the request without relying on information from previous requests
  - -> representation -- useful information about the state of a resource
  - -> connectedness -- resource should link to each other and their representations
  - -> uniform interface -- GET/HEAD to retrieve resources
    POST to create a resource
    PUT tp modify a resource
    DELETE to remove a resource
    OPTIONS to get available options for the resource
    (e.g. which methods can be used)
  - -> safety and idempotence -- safe means to client-requested side effects; idempotent means making one request is the same as making a dozen (e.g. whether you delete a resource one or a dozen times the effect is the same - the resource is gone); GET, HEAD are safe; GET, HEAD, PUT, DELETE are idempotent
- since REST is resource oriented and the standard OGC web architecture is service oriented some adaptation of the standard OGC service model is required
  - however, rather than throw the baby out with the bath water, this ANNEX describes an evolutionary approach to developing the REST binding
  - the characteristics of WFS that are preserved in this binding are:
    - -> service metadata (i.e. capabilities document)
      - all OGC services provide service metadata describing:
        - -> the type of service being offered
        - -> information about the service provider
        - -> the set of parameter and server constraints
        - -> a list of feature type that are available at the service
        - -> the kinds of query predicates the service can support
      - although the capabilities document is much maligned it is ironic to note that there are examples of RESTful services that use a similar concept; ATOMPUB uses a service document to perform much the same function as the OGC capability document
    - -> get the structure or schema of a feature type
    - -> access to individual feature instances
    - -> simple query capability
      - the ability to dynamically define collections of feature types of a single type using simple predicates (e.g. bbox)
    - -> advanced query capability
      - the ability to dynamically define a collection of features of one or more types using combinations of simple and/or complex predicates that can include scalar, spatial and temporal operators (e.g. joins, spatial joins, temporal joins, etc...)
      - stored query capability
        - the ability to create and invoke predefined, stored queries
    - -> simple transaction capability
      - the ability to create/modify/delete single feature type instances
    - -> advanced transaction capability
      - the ability to create/modify/delete multiple instances of features of one or more types and do so atomically

X.2 Conformance classes

-the following table maps resources and methods to the WFS conformance classes (see X.3.8 for a summary of resources)

Conformance Class	WFS Operations	Implemented by	Clause
Simple WFS	GetCapabilities	GET /	(4)
	DescribeFeatureType	GET schema[/...]	(5)
	ListStoredQueries	OPTIONS query	(7)
	GetFeature (with stored queries only)	GET query/{Query Id}	(7)
	GetFeatureById	GET FeatureTypes/{Feature Type}/{Feature Id}	(6)
Basic WFS	GetFeature	GET FeatureTypes/{Feature Type}[/...]	(6)
		GET query[/...]	(7)
	GetPropertyValue	GET FeatureTypes/{Feature Type}/{FeatureId}/Properties/{Property Name}	(6)
Transactional WFS	Transaction	POST FeatureTypes/{Feature Type}	(6)
		PUT FeatureTypes/{Feature Type}	(6)
		DELETE FeatureTypes{Feature Type}	(6)
		POST/PUT/DELETE transaction[/...]	(8)
Locking WFS	N/A		
HTTP GET	N/A		
HTTP POST	N/A		
SOAP	N/A		
Inheritance	schema-element()	PUT query/{Query Id}/typeNames	(7)
Remote Response		resolve query parameters	(6)
Response Paging		include prev/next parameters in response	(6)
Standard joins		PUT query/{Query Id}/typeNames	(7)
		PUT query/{Query Id}/filter	(7)
Spatial joins		PUT query/{Query Id}/typeNames	(7)
		PUT query/{Query Id}/filter	(7)
Temporal joins		PUT query/{Query Id}/typeNames	(7)
		PUT query/{Query Id}/filter	(7)
Feature version	N/A		
Manage Stored Qry	CreateStoredQuery	PUT query/{Query Id}	(7)
	DropStoredQuery	DELETE query/{Query Id}	(7)

NOTES:
- the notation {...} is used to indicate a component of a URI that is implementation or run-time specific
- the notation [...] or [/...] means "and all sub-resources"

X.3 Basic service elements

X.3.1 Service root

- this binding assumes that all resources that a WFS offers are accessible through some parent resource called the "service root"
- the response to accessing the service root with the GET method shall be a WFS capabilities document
- if an implementation supports multiple versions of the WFS specification, and hence multiple versions of the WFS capabilities document, each one of these shall have a its own service root
- a specific format or pattern for the service root URL is not specified by this standard

X.3.2 Service parameter

- the service parameter is not required since the service root URL, whatever it is, will point the client to the correct parent resource

X.3.3 Version negotiation

- for interaction with a RESTful WFS to commence, a client needs to discover the service root(s) of a WFS
  - -> in the other bindings version negotiation, using the ACCEPT_VERSIONS and VERSIONS parameter, is used to connect a client a the version of the WFS they understand
  - -> in the REST binding, where connectedness is a concern, this approach is not very satisfying since it forces the client to construct a URL and we are trying, as much as possible, to make all interesting aspects of the service directly accessible through links
- for the REST binding we assume the existence of a "service roots" document whose URL the client knows
  - -> interaction between the client and the service commences by accessing this "service roots" document with the GET method
  - -> the response shall be a wfs:ServiceRoots document that simply contains a list of versioned WFS service root links
- the schema of the wfs:ServiceRoots element shall be

```
<xsd:element name="ServiceRoots" type="wfs:ServiceRootsType"/>
<xsd:complexType name="ServceRootsType">
   <xsd:sequence>
```

```
            <xsd:element name="Link" maxOccurs="unbounded">
               <xsd:complexType>
                  <xsd:attribute name="href"
                                 type="xsd:anyURI"
                                 use="required"/>
                  <xsd:attribute name="wfsVersion"
                                 type="xsd:string"
                                 use="required"/>
               </xsd:complexType>
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
```

- each service root link resolves to a WFS capabilities document that corresponds to the version asserted by the wfsVersion attribute
- all sub-resources that the service offers are accessible through this service root link

Example:

Example: Get the service roots document

```
CLIENT                                                    SERVER
  |                                                         |
  |   GET / HTTP/1.1                                        |
  |   Host: wfs.someserver.com                              |
  |-------------------------------------------------------->|
  |                                                         |
  |   HTTP/1.1 200 OK                                       |
  |   Content-Type: text/xml                                |
  |                                                         |
  |   <?xml version="1.0"?>                                 |
  |   <wfs:ServiceRoots>                                    |
  |     <Link wfsVersion="1.0.0"                            |
  |           serviceRoot="http://wfs.somserver.com/1.0"/>  |
  |     <Link wfsVersion="1.1.0"                            |
  |           serviceRoot="http://wfs.somserver.com/1.1"/>  |
  |     <Link wfsVersion="2.0.0"                            |
  |           serviceRoot="http://wfs.somserver.com/2.0"/>  |
  |   </wfs:ServiceLinks>                                   |
  |<--------------------------------------------------------|
  |                                                         |
  |   GET /1.1 HTTP/1.1                                     |
  |   Host: wfs.someserver.com                              |
  |-------------------------------------------------------->|
  |                                                         |
  |   HTTP/1.1 200 OK                                       |
  |   Content-Type: text/xml                                |
  |                                                         |
  |   <?xml version="1.0"?>                                 |
  |   <wfs:Capabilities version="1.1.0">                    |
  |      ...                                                |
  |   </wfs:Capabilities>                                   |
  |<--------------------------------------------------------|
```

X.3.4 Feature representation

- the canonical representation of features for this binding shall be GML 3.2
- other representations (e.g. JSON) are allowed
- a strong recommendation
  -> for server that support JSON this standard strongly recommends that GeoJSON (http://www.geojson.org/geojson-spec.html) be used to represent features and JSON-schema (http://tools.ietf.org/html/draft-zyp-json-schema-03) be used for describing the schema

X.3.5 Content negotiation

- content negotiation shall proceed in the standard HTTP way using the Accept header
- when submitting an HTTP request to a service the client shall set the Accept header with a list of desired content types if order of preference
- the server shall respond with the most preferred content type
- the OPTIONS methods may be used on a resource to determine the representation that the service offers

X.3.6 Exceptions

- exceptions shall be handled as described in clause 7.5
  - servers shall return an appropriate HTTP error code
    -> all exceptions that are the result of something the client request has wrong (where the client should have known better) should have an HTTP error code in the 4xx range (and in most if not all cases should be specifically "400 Bad Request")
    -> server-side issues should be given an HTTP error code in the 5xx range (and in most if not all cases should be specifically "500 Internal Server Error").
    -> Table 28 in the OWS Common 2.0 specification (OGC 06-121r9) is a bit wonky, because it messes up what (I think) should be returned for OperationNotSupported, OptionNotSupported and NoApplicableCode.
  - content of the exception shall be a OGC service exception report
- this specification does not define behaviour for all HTTP methods for all resources (e.g. the behaviour of the PUT method is not described for the capabilities resource "/")
  -> when a server is presented with an unsupported HTTP method for a particular resource it should be responded with the HTTP error code "405 Method Not Allowed" and should contain a OGC exception message in the body of the response

```
     -> Editors Note: implementation are free to implement (or not) whatever
                      behaviour they want in cases where behaviour is not
                      defined in this standard (e.g. for experimental purposes
                      or vendor extensions) but whatever they do will be
                      outside the scope of this document

X.3.7 Authentication and Access Control
   - all methods and resource described in this standard are subject to
     authentication and access control
     - e.g. someone who does not have authorization to create new features
            would not see the POST methods when interrogating a resource
            with the OPTIONS method
   - authentication and access control are envisioned to be layered ontop
     of a RESTful WFS but the description of how that would be done is
     beyond the scope of this standard

X.3.8 Summary of resources
 ------------------------------------------------------------------------------
 NAME          DESCRIPTION                  URIs (relative to service root)
 ------------------------------------------------------------------------------
 Capabilities  service metadata document    /
               that includes a number of    ServiceIdentification
               sections and includes links  ServiceProvider
               to the available collections FeatureTypeList
               of features                  FilterCapabilities
 ------------------------------------------------------------------------------
 Schema        a document describing the    schema
               structure of one or more     schema/{Feature Type}
               feature type offered by a
               service
 ------------------------------------------------------------------------------
 FeatureTypes  a collection of features     FeatureTypes/{Feature Type}
               of a specific type

               a specific feature           FeatureTypes/{Feature Type}/{Fea
                                            ture Id}

               a property of a feature      FeatureTypes/{Feature Type}/{Pro
               of a specific type           perty Name}

               the property of a specific   FeatureTypes/{Feature Type}/{Fea
               feature                      ture Id}/Properties/{Property Name}
 ------------------------------------------------------------------------------
 Query         a resource used to define    query
               an advanced query

               a specific instance of an    query/{Query Id}
               advanced query

               the projections clause of    query/{Query Id}/properties
               a specific query

               the selection clause of      query/{Query Id}/filter
               a specific query

               the features types being     query/{Query Id}/typeNames
               queried by a specific
               query

               the sorting clause of a      query/{Query Id}/sort
               specific query
 ------------------------------------------------------------------------------
 Transaction   a resource to support        transaction
               atomic transactions on
               multiple feature types

               a specific transaction       transaction/{Transaction Id}

               a collection of features     transaction/{Transaction Id}/{Fe
               being operated on by a       ature Type}
               specific transaction

               a specific feature being     transaction/{Transaction Id}/{Fe
               operated on by a specific    ature Type}/{Feature Id}
               transaction

               a specific property of a     transaction/{Transaction Id}/{Fe
               feature being operated on    ature Type}/{Feature Id}/{Property
               by a specific transaction    Name}
 ------------------------------------------------------------------------------

X.3.9 Content types
     GML instance doc:
        -> application/gml+xml; version=X.X
     XML-Encoded OGC Filter:
        -> urn:ogc:def:query Language:OGC-FES:Filter
     Common Query Language:
        -> urn:ogc:def:query Language:OGC-CSW:Cql
     WFS query expression:
        -> urn:ogc:def:queryLanguage:OGC-WFS::WFS_QueryExpression
     GML/GMLSF application schema:
        -> text/xml; gmlver=X.X [,gmlsflev=X]

X.3.10 A word about examples
```

```
  - many of the concepts discussed in this binding are accompanied by
    examples in the form of sequence diagrams showing the HTTP interaction
    between a client and server
  - the examples in this document are presented using a fictitious
    service root: http://wfs.someserver.com/2.0
    -> this means that that the "/2.0" component of the paths in the
       examples is not part of the canonical resource URLs but simply
       part of this example service root
  - in a number of examples, text that is supposed to be on a single line
    is wrapped across multiple lines for the sake of clarity

X.4 Service metadata
   Resources (relative to the service root URL):
     (the service root URL itself)
     ServiceIdentification
     ServiceProvider
     FeatureTypeList
     FilterCapabilities

   Representations:
      - the canonical representation for service metadata shall
        be the XML-encoded OGC capabilities document as define in
        WFS clause 8.3 (with changed proposed is Discussion)
      - other representations (e.g. JSON, HTML) are allow but are
        not described in this specification

   Methods:
      OPTIONS -> gets the available methods and representations
      GET     -> gets the complete capabilities document or a section
                 of the capabilities document depending upon the
                 resource being retrieved
      POST    -> not specified by this standard (see X.3.6)
      PUT     -> not specified by this standard (see X.3.6)
      DELETE  -> not specified by this standard (see X.3.6)

   Discussion:
      - the following changes need to be made to the schema of the
        capabilities document to accommodate the REST binding
        - need to add a ServiceRoot element to the capabilities doc
        - the definition of the OperationsMetadata in ows common
          needs to be changed to minOccurs="0"
          -> it is currently set to minOccurs=2 to accommodate
             the GetCapabilities operation and one other operation
          -> the problem is that  REST services have a uniform
             API consisting of the HTTP methods OPTIONS, GET, POST,
             PUT and DELETE; so, no operations need to be specified
             in the capabilities document for REST implementations
        - the definition of ows:Operation needs to be changed to make
          the cardinality of ows:DCP, minOccurs="0"
          -> REST services do not need DCP urls
        - the wfs:FeatureTypeList element should be modified to
          allow zero or more Link elements
          -> this will allow the server to provide a link to
             the complete application schema plus links to any
             stored queries
        - the wfs:Feature element should be modified to allow zero
          or more Link elements
          -> this will allow the server to a provide links to
             the cannonical represetation of each feature type
             (i.e. GML 3.2), links to any alternative representations,
             links to the schema, etc ...

   Examples:
      Example: Get available representations
      CLIENT                                                    SERVER
        |                                                         |
        |   OPTIONS /2.0  HTTP/1.1                                |
        |   Host: wfs.someserver.com                              |
        |-------------------------------------------------------->|
        |                                                         |
        |   HTTP/1.1 200 OK                                       |
        |   Allow: OPTIONS, GET                                   |
        |   Accept: text/xml, text/html                           |
        |<--------------------------------------------------------|

      Example: Get the XML capabilities document
      CLIENT                                                    SERVER
        |                                                         |
        |   GET /2.0  HTTP/1.1                                    |
        |   Host: wfs.someserver.com                              |
        |   Accept: text/xml                                      |
        |-------------------------------------------------------->|
        |                                                         |
        |   HTTP/1.1 200 OK                                       |
        |   Content-Type: text/xml                                |
        |                                                         |
        |   <?xml version="1.0"?>                                 |
        |   <wfs:Capabilities version="2.0.0">                    |
        |     ... see X.9 for a complete example ...              |
        |   </wfs:Capabilities>                                   |
        |<--------------------------------------------------------|

      Example: Get the feature type list (i.e. resource list) for a WFS
      CLIENT                                                    SERVER
```

```
|                                                                |
|   GET /2.0/FeatureTypeList HTTP/1.1                            |
|   Host: wfs.someserver.com                                     |
|   Accept: test/xml                                             |
|--------------------------------------------------------------->|
|                                                                |
|   HTTP/1.1 200 OK                                              |
|   Content-Type: text/xml                                       |
|                                                                |
|   <?xml version="1.0"?>                                        |
|   <FeatureTypeList...>                                         |
|      <Link rel="describedby"                                   |
|            type="text/xml; gmlver=3.2"                         |
|            href="http://www.BlueOx.org/2.0/schema"/>           |
|      <FeatureType xmlns:bo="http://www.BlueOx.org/BlueOx"      |
|         <Link rel="self"                                       |
|               type="application/gml+xml; version=3.2"          |
|               href="http://www.BlueOx.org/2.0/FeatureTypes/Woods"/>
|         <Link rel="alternate"                                  |
|               type="application/json"                          |
|               href="http://www.BlueOx.org/2.0/FeatureTypes/Woods"/>
|         <Link rel="describedby"                                |
|               type="text/xml; gmlver=3.2"                      |
|               href="http://www.BlueOx.org/2.0/schema/Woods"/>
|         <Name>bo:Woods</Name>                                  |
|         <Title>The Great Northern Forest</Title>               |
|         <Abstract>                                             |
|            Describes the arborial diversity of the             |
|            Great Northern Forest.                              |
|         </Abstract>                                            |
|         <ows:Keywords>                                         |
|            <ows:Keyword>forest</ows:Keyword>                   |
|            <ows:Keyword>north</ows:Keyword>                    |
|            <ows:Keyword>woods</ows:Keyword>                    |
|            <ows:Keyword>arborial</ows:Keyword>                 |
|            <ows:Keyword>diversity</ows:Keyword>                |
|         </ows:Keywords>                                        |
|         <DefaultCRS>urn:ogc:def:crs:EPSG::6269</DefaultCRS>
|         <OtherCRS>urn:ogc:def:crs:EPSG::32615</OtherCRS>       |
|         <OtherCRS>urn:ogc:def:crs:EPSG::32616</OtherCRS>       |
|         <OtherCRS>urn:ogc:def:crs:EPSG::32617</OtherCRS>       |
|         <OtherCRS>urn:ogc:def:crs:EPSG::32618</OtherCRS>       |
|         <ows:WGS84BoundingBox>                                 |
|            <ows:LowerCorner>-180 -90</ows:LowerCorner>         |
|            <ows:UpperCorner>180 90</ows:UpperCorner>           |
|         </ows:WGS84BoundingBox>                                |
|      </FeatureType>                                            |
|      <FeatureType                                              |
|         <Link rel="self"                                       |
|               type="application/gml+xml; version=3.2"          |
|               href="http://www.BlueOx.org/2.0/FeatureTypes/Lakes"/>
|         <Link rel="alternate"                                  |
|               type="application/json"                          |
|               href="http://www.BlueOx.org/2.0/FeatureTypes/Lakes"/>
|         <Link rel="describedby"                                |
|               type="text/xml; gmlver=3.2"                      |
|               href="http://www.BlueOx.org/2.0/schema/Lakes"/>
|         <Name>bo:Lakes</Name>                                  |
|         <Title>The Great Northern Lakes</Title>                |
|         <Abstract>                                             |
|            Lake boundaries for all lakes in the                |
|            Great Northern Forest.                              |
|         </Abstract>                                            |
|         <ows:Keywords>                                         |
|            <ows:Keyword>lakes</ows:Keyword>                    |
|            <ows:Keyword>boundaries</ows:Keyword>               |
|            <ows:Keyword>water</ows:Keyword>                    |
|            <ows:Keyword>hydro</ows:Keyword>                    |
|         </ows:Keywords>                                        |
|         <DefaultCRS>urn:ogc:def:crs:EPSG::6269</DefaultCRS>
|         <OtherCRS>urn:ogc:def:crs:EPSG::32615</OtherCRS>       |
|         <OtherCRS>urn:ogc:def:crs:EPSG::32616</OtherCRS>       |
|         <OtherCRS>urn:ogc:def:crs:EPSG::32617</OtherCRS>       |
|         <OtherCRS>urn:ogc:def:crs:EPSG::32618</OtherCRS>       |
|         <ows:WGS84BoundingBox>                                 |
|            <ows:LowerCorner>-180 -90</ows:LowerCorner>         |
|            <ows:UpperCorner>180 90</ows:UpperCorner>           |
|         </ows:WGS84BoundingBox>                                |
|      </FeatureType>                                            |
|   </FeatureTypeList>                                           |
|<---------------------------------------------------------------|
```

X.5 Feature type schemas (= DescribeFeatureType)

Resources (relative to the service root URL):

- schema
- schema/{Feature Type}

Representations:

- the canonical representation of the schema of features shall be a GML 3.2 application schema (see WFS, clause 9.3)
- other representations (e.g. JSON-Schema) are allowed but are not described in this standard

Methods:

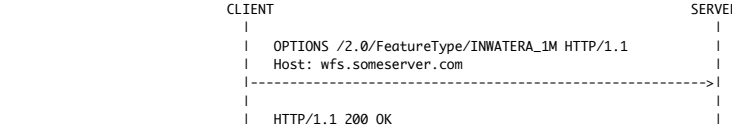
```
    OPTIONS -> gets the supported methods and representations for
               the schema resource via the Allow and Accept HTTP
               headers
    GET     -> gets the schemas of all feature types or the specified
               feature type id the /schema/{Feature Type} resource is
               specified
    POST    -> creates the feature types described in the schema
              (future item from OWS8)
    PUT     -> not specified by this standard (see X.3.6)
    DELETE  -> not specified by this standard (see X.3.6)

  Description:
    - for GML application schemas, the variables "gmlver" and "gmlsflev"
      may be used to specify the version(s) of GML that the server
      supports

  Examples:
    Example: Get all available schema representations.
    CLIENT                                                    SERVER
      |                                                         |
      |   OPTIONS /2.0/schema HTTP/1.1                          |
      |   Host: wfs.someserver.com                              |
      |---------------------------------------------------------->|
      |                                                         |
      |   HTTP/1.1 200 OK                                       |
      |   Allow: OPTIONS, GET, PUT                              |
      |   Accept: text/xml; gmlver=3.2,                         |
      |           text/xml; gmlver=3.1,                         |
      |           text/xml; gmlver=2.1,application/json         |
      |<----------------------------------------------------------|

      In this case the server support JSON and

    Example: Get the XML schema of all available feature types
             as a GML 2.1 application schema
    CLIENT                                                    SERVER
      |                                                         |
      |   GET /2.0/schema HTTP/1.1                              |
      |   Host: wfs.someserver.com                              |
      |   Content-Type: text/xml; gmlver=2.1                    |
      |---------------------------------------------------------->|
      |                                                         |
      |   HTTP/1.1 200 OK                                       |
      |   Content-Type: text/xml; gmlver=2.1                    |
      |                                                         |
      |   <?xml version="1.0"?>                                 |
      |   <xs:schema> ... </xs:schema>                          |
      |<----------------------------------------------------------|

    Example: Get the XML schema of INWATERA_1M
    CLIENT                                                    SERVER
      |                                                         |
      |   GET /2.0/schema/INWATERA_1M HTTP/1.1                  |
      |   Host: wfs.someserver.com                              |
      |   Content-Type: text/xml; gmlver=3.2                    |
      |---------------------------------------------------------->|
      |                                                         |
      |   HTTP/1.1 200 OK                                       |
      |   Content-Type: text/xml; gmlver=3.2                    |
      |                                                         |
      |   <?xml version="1.0"?>                                 |
      |   <xs:schema ...>                                       |
      |      ...                                                |
      |      <xs:element name="INWATERA_1M" ...>                |
      |      ...                                                |
      |   </xs:schema>                                          |
      |<----------------------------------------------------------|

X.6 Feature access and management (= GetPropertyValue/GetFeature/Transaction)
  Resources (relative to the service root URL):
    (a) FeatureTypes/{Feature Type}
    (b) FeatureTypes/{Feature Type}/{Feature Id}
    (c) FeatureTypes/{Feature Type}/Properties/{Property Name}
    (d) FeatureTypes/{Feature Type}/{Feature Id}/Properties/{Property Name}

  Representations:
    - the canonical representation of features shall be GML 3.2
    - the canonical representation of property values shall be plain
      text for scalar values and XML (possibly GML) for composite
      or complex values (e.g. the value of geometry property)
    - other representations (e.g. JSON) are allowed but are
      not described in this standard

  Methods:
    - the following table describes the actions the server should take when
      the specified method is applied to the specified resource
      -> the resources from the "Resource:" section above are reference
         by letter (a),(b),(c),(d) for the sake of brevity

    ------------------------------------------------------------------------
    RESOURCE   METHOD   ACTION
    ------------------------------------------------------------------------
      all      OPTIONS  - gets list of supported representations
                          and methods for the feature type via the
```

		Accept and Allow HTTP headers
(a)	GET	- gets a collection of features of the specified featured type
(b)	GET	- gets the feature with the specified id
(c)	GET	- gets a list of values for the specified property for a collection of features of the specified type
(d)	GET	- gets the value of the specified property for the feature with the specified id
(a)	POST	- creates a new instance of the specified feature type; a represented of the new feature is supplied in the body of the request
(b)	POST	- not specified by this standard (see X.3.6)
(c)	POST	- not specified by this standard (see X.3.6)
(d)	POST	- not specified by this standard (see X.3.6)
(a)	PUT	- replaced all features or a subset of features (if query parameters are supplied on the request -- e.g. bbox) of the specified type; a representation of the new feature is supplied as the body of the request (actually, should probably now allow this operation and server should throw an exception)
(b)	PUT	- replaces the existing feature; a representation of the new features is supplied as the body of the request
(c)	PUT	- replaces the value of the specified property for all features or a subset of features (if query parameters are supplied on the request -- e.g. bbox)
(d)	PUT	- replaced the value of the specified property for the feature with the specified id
(a)	DELETE	- deletes all instances of the specified feature type (should probably now allow this!)
(b)	DELETE	- deletes the feature with the specified id
(c)	DELETE	- sets the value of the specified property to NULL for all features or a subset of features if query parameters (e.g. bbox) are supplied on the request
(d)	DELETE	- sets the value of the specified property to NULL for the feature with the specified id

Description:
- the {Feature Id} shall, like the other bindings, be supplied by the server when the resource is created

- a resource URI can also include query parameters to identify subsets of features to be operated upon; here is the list of query parameters for WFS:
  -> count (see WFS Table 5)
  -> startIndex (see WFS Table 5)
  -> resolve (see WFS Table 6)
  -> resolveDepth (see WFS Table 6)
  -> resolveTimeout (see WFS Table 6)
  -> nameSpaces (see WFS Table 7)
  -> srsName (see WFS Table 8)
  -> filter_Language (see WFS Table 8)
  -> filter (see WFS Table 8)
  -> bbox (see WFS Table 8)
  -> sortBy (see WFS Table 8)
  -> propertyName (see WFS Table 9)
- here is the list of NEW query parameters:
  -> geometry - WKT-encoded geometry
  -> crs - CRS for geometry
  -> spatialOp - one of: Equals, Disjoint, Touches, Within, Overlaps, Crosses, Intersects (d), Contains
  -> time - ISO8601 time instance or interval
  -> temporalOp - one of After, Before, Begins, During(d) EndedBy, Ends, TEquals, Meets, MetBy

Examples:
Example: Get the available representations for INWATERA_1M



```
  |   Allow: OPTIONS, GET, POST, PUT, DELETE                  |
  |   Accept: application/gml+xml; version=3.2,               |
  |           application/gml+xml; version=3.1,               |
  |           application/json                                |
  |<----------------------------------------------------------|

Example: Get the available representations of the property F_CODE
CLIENT                                                      SERVER
  |                                                           |
  |   OPTIONS /2.0/FeatureTypes/INWATERA_1M/Properties/F_CODE HTTP/1.1
  |   Host: wfs.someserver.com                                |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |   Allow: OPTIONS, GET, PUT                                |
  |   Accept: text/plain                                      |
  |<----------------------------------------------------------|

Example: Get a collection of INWATERA_1M features
CLIENT                                                      SERVER
  |                                                           |
  |   GET /2.0/FeatureTypes/INWATERA_1M?count=10 HTTP/1.1     |
  |   Host: wfs.someserver.com                                |
  |   Accept: application/gml+xml; version=3.2                |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |   Content-Type: application/gml+xml; version=3.2          |
  |                                                           |
  |   <?xml version="1.0"?>                                   |
  |   <wfs:FeatureCollection next="http://..."> ...           |
  |   </wfs:FeatureCollection>                                |
  |<----------------------------------------------------------|

  NOTES:
    -> the response would contain at most 10 features since
       the default value for the "count" parameter is 10
    -> the "next" link points to the next 10 set of features

Example: Get a specific feature instance
CLIENT                                                      SERVER
  |                                                           |
  |   GET /2.0/FeatureTypes/INWATERA_1M/1013 HTTP/1.1         |
  |   Host: wfs.someserver.com                                |
  |   Accept: application/gml+xml; version=3.2                |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |   Content-Type: application/gml+xml; version=3.2          |
  |                                                           |
  |   <?xml version="1.0"?>                                   |
  |   <tns:INWATERA_1M gml:id="1013"> ... </tns:INWATERA_1M>  |
  |<----------------------------------------------------------|

  NOTES:
     -> getting a specific instance of a feature does not
        return a collection but rather the XML fragment that
        represents the feature

Example: Get a collection of features inside a specific BBOX
CLIENT                                                      SERVER
  |                                                           |
  |   GET /2.0/FeatureTypes/INWATERA_1M?bbox=10,10,20,20 HTTP/1.1
  |   Host: wfs.someserver.com                                |
  |   Accept: application/gml+xml; version=3.2                |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |   Content-Type: application/gml+xml; version=3.2          |
  |                                                           |
  |   <?xml version="1.0"?>                                   |
  |   <wfs:FeatureCollection> ... </wfs:FeatureCollection>    |
  |<----------------------------------------------------------|

Example: Get a collection of features based on a CQL filter
CLIENT                                                      SERVER
  |                                                           |
  |   GET /2.0/FeatureTypes/INWATERA_1M?filter_Language=CQL&  |
  |            filter=depth+between+10+and+20 HTTP/1.1        |
  |   Host: wfs.someserver.com                                |
  |   Accept: application/gml+xml; version=3.2                |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |   Content-Type: application/gml+xml; version=3.2          |
  |                                                           |
  |   <?xml version="1.0"?>                                   |
  |   <wfs:FeatureCollection> ... </wfs:FeatureCollection>    |
  |<----------------------------------------------------------|

Example: Get the value of the F_CODE property for the INWATERA_1M
         feature with id 1013
CLIENT                                                      SERVER
```

```
  |                                                          |
  |   GET /2.0/FeatureTypes/INWATERA_1M/1013/Properties/F_CODE HTTP/1.1
  |   Host: wfs.someserver.com                               |
  |   Accept: text/plain                                     |
  |--------------------------------------------------------->|
  |                                                          |
  |   HTTP/1.1 200 OK                                        |
  |                                                          |
  |   Content-Type: text/plain                               |
  |   GH301                                                  |
  |<---------------------------------------------------------|

Example: Get the list of depth values for features that lie
         within a specific bbox
CLIENT                                                     SERVER
  |                                                          |
  |   GET /2.0/FeatureTypes/INWATERA_1M/Properties/F_CODE?bbox=10,10,20,20 HTTP/1.1
  |   Host: wfs.someserver.com                               |
  |   Accept: text/plain                                     |
  |--------------------------------------------------------->|
  |                                                          |
  |   HTTP/1.1 200 OK                                        |
  |   Content-Type: text/plain                               |
  |                                                          |
  |   201                                                    |
  |   207                                                    |
  |   211                                                    |
  |   222                                                    |
  |   223                                                    |
  |   224                                                    |
  |   225                                                    |
  |   225                                                    |
  |   225                                                    |
  |<---------------------------------------------------------|

Example: Create a new feature instance (INSERT) represented as GML
CLIENT                                                     SERVER
  |                                                          |
  |   POST /2.0/FeatureTypes/INWATERA_1M HTTP/1.1            |
  |   Host: wfs.someserver.com                               |
  |   Content-Type: application/gml+xml; version=3.2         |
  |                                                          |
  |   <?xml version="1.0"?>                                  |
  |   <myns:INWATERA_1M> ... </myns:INWATERA_1M>             |
  |--------------------------------------------------------->|
  |                                                          |
  |   HTTP/1.1 201 Created                                   |
  |   Location: /2.0/FeatureTypes/INWATERA_1M/1013           |
  |<---------------------------------------------------------|

Example: Create a new feature instance represented as geo JSON
CLIENT                                                     SERVER
  |                                                          |
  |   POST /2.0/FeatureTypes/INWATERA_1M HTTP/1.1            |
  |   Host: wfs.someserver.com                               |
  |   Content-Type: application/json                         |
  |                                                          |
  |   { "type": "Feature",                                   |
  |     "geometry": {                                        |
  |       "type": "LineString",                              |
  |       "coordinates": [                                   |
  |         [102.0, 0.0],[103.0,1.0],[104.0,0.0],[105.0,1.0] |
  |         ]                                                |
  |       },                                                 |
  |     "properties": {                                      |
  |       "prop0": "value0",                                 |
  |       "prop1": 0.0                                       |
  |       }                                                  |
  |   }                                                      |
  |--------------------------------------------------------->|
  |                                                          |
  |   HTTP/1.1 201 Created                                   |
  |   Location: /2.0/FeatureTypes/INWATERA_1M/1014           |
  |<---------------------------------------------------------|

Example: Update an existing feature (UPDATE)
CLIENT                                                     SERVER
  |                                                          |
  |   PUT /2.0/FeatureTypes/INWATERA_1M/1013 HTTP/1.1        |
  |   Host: wfs.someserver.com                               |
  |   Content-Type: application/gml+xml; version=3.2         |
  |                                                          |
  |   <?xml version="1.0"?>                                  |
  |   <myns:INWATERA_1M> ... </myns:INWATERA_1M>             |
  |--------------------------------------------------------->|
  |                                                          |
  |   HTTP/1.1 200 OK                                        |
  |   Location: /2.0/FeatureTypes/INWATERA_1M/1013           |
  |<---------------------------------------------------------|

Example: Can I update the F_CODE property of a feature?
CLIENT                                                     SERVER
  |                                                          |
  |   OPTIONS /2.0/FeatureTypes/INWATERA_1M/1013/Properties/F_CODE HTTP/1.1
```

```
      |   Host: wfs.someserver.com                                  |
      |------------------------------------------------------------>|
      |                                                             |
      |   HTTP/1.1 200 OK                                           |
      |   Allow: OPTIONS, GET, PUT                                  |
      |   Accept: text/plain                                        |
      |<------------------------------------------------------------|

    Example: Change the value of the F_CODE property.
    CLIENT                                                    SERVER
      |                                                         |
      |   PUT /2.0/FeatureTypes/INWATERA_1M/1013/Properties/F_CODE HTTP/1.1
      |   Host: wfs.someserver.com                              |
      |   Content-Type: text/plain                              |
      |                                                         |
      |   302                                                   |
      |------------------------------------------------------------>|
      |                                                         |
      |   HTTP/1.1 200 OK                                       |
      |<------------------------------------------------------------|

    Example: Change a geometric property of a feature
    CLIENT                                                    SERVER
      |                                                         |
      |   PUT /2.0/FeatureTypes/INWATERA_1M/1013/extent HTTP/1.1  |
      |   Host: wfs.someserver.com                              |
      |   Content-Type: application/gml+xml; version=3.2        |
      |                                                         |
      |   <?xml version="1.0"?>                                 |
      |   <gml:Polygon> ... </gml:Polygon>                      |
      |------------------------------------------------------------>|
      |                                                         |
      |   HTTP/1.1 200 OK                                       |
      |<------------------------------------------------------------|

    Example: Delete a feature
    CLIENT                                                    SERVER
      |                                                         |
      |   DELETE /2.0/FeatureTypes/INWATERA_1M/1013 HTTP/1.1    |
      |   Host: wfs.someserver.com                              |
      |------------------------------------------------------------>|
      |                                                         |
      |   HTTP/1.1 200 OK                                       |
      |<------------------------------------------------------------|

X.7 Advanced queries (=GetPropertyValue/GetFeature)
X.7.1 Ad-hoc queries
   Resources (relative to the service root URL):
      (a)  query
      (b)  query/{Query Id}
      (c)  query/{Query Id}/properties
      (d)  query/{Query Id}/filter
      (e)  query/{Query Id}/typeNames
      (f)  query/{Query Id}/aliases
      (g)  query/{Query Id}/sort

   Representations:
      - there is no canonical representation for (a)
      - the canonical representation for (b) is text/xml in the form
        of a wfs:FeatureCollection document that contains the collection
        of features that satisfy the query
        -> other representations are allowed (e.g. JSON) but are not described
           in this standard
      - the canonical representation for (c) is text/plain in the form
        of a space-separated list of property names; the names may be
        qualified in which case the NAMESPACE query parameter must be
        specified
      - the canonical representation for (d) is the XML-encoding for
        the OGC Filter
        -> other representations (i.e. filter languages) are allowed (e.g. CQL)
           but are not described in this standard
      - the canonical representation for (e) is text/plain in the form of
        a space-separated list of feature type names; the names may be
        qualified in which case the NAMESPACE query parameter must be
        specified
      - the canonical representation for (f) is text/plain in the form of
        a space-separated list of aliases; the names may be qualified in
        which case the NAMESPACE query parameter must be specified
      - the canonical representation for (g) is XML-encoding for a
        sort clause as specified in the OGC filter specification
        -> other representations are allowed but are not described in this
           standard

   Methods:
     - the following table describes the actions the server should take when
       the specified method is applied to the specified resource
       -> the resources from the "Resource:" section above are reference
          by letter (a),(b),(c),(d),(e),(f),(g) for the sake of brevity

     ---------------------------------------------------------------------------
     RESOURCE   METHOD   ACTION
     ---------------------------------------------------------------------------
       all      OPTIONS  - gets list of supported representations
                           and methods for the feature type via the
```

		Accept and Allow HTTP headers
(a)	GET	- not specified by this standard (see X.3.6)
(b)	GET	- gets the collection of features that satisfy the query
(c)	GET	- gets a list of properties that the query is retrieving
(d)	GET	- gets the text of the query predicate for the feature with the specified id; the only representation that shall available is the one provided when the resource was created -> in other words the server is not expected to convert an OGC filter into CQL or any other query language it supports
(e)	GET	- gets list of feature type names that are being queried
(f)	GET	- gets the list of feature types alias names (should align 1:1 with the list from (e))
(g)	GET	- gets the text of the sort clause
(a)	POST	- creates a new query resource that can be used to define an ad-hoc query
(b)	POST	- creates a new query resource that can be used to define a stored query; the client must supply the {Query Id}; the body of the request may contain the text of the stored query (see X.7.2)
(c)	POST	- not specified by this standard (see X.3.6)
(d)	POST	- not specified by this standard (see X.3.6)
(d)	POST	- not specified by this standard (see X.3.6)
(d)	POST	- not specified by this standard (see X.3.6)
(g)	POST	- not specified by this standard (see X.3.6)
(a)	PUT	- not specified by this standard (see X.3.6)
(b)	PUT	- allows additional query expressions to be added to a stored query (see X.7.2)
(c)	PUT	- updates the specified query to add or replace a list of optional properties to present in the response document
(d)	PUT	- update the specified query to add or replace a query predicate used to identified a subset of features to present in the response document
(e)	PUT	- update the specified query to add or replace a list of one or more feature type names to be queried; multiple type names indicates that a join is being performed
(f)	PUT	- update the specified query to add or replace a list of alternative names or aliases for the feature type names in the (e) resource; this is mostly to support self-joins
(g)	PUT	- update the specified query to add or replace a sort clause that defines the order in which features should be presented in the response document
(a)	DELETE	- not specified by this standard (see X.3.6)
(b)	DELETE	- deletes the specified query
(c)	DELETE	- not specified by this standard (see X.3.6)
(d)	DELETE	- not specified by this standard (see X.3.6)
(e)	DELETE	- not specified by this standard (see X.3.6)
(f)	DELETE	- not specified by this standard (see X.3.6)
(g)	DELETE	- not specified by this standard (see X.3.6)

Description:
- the query resource and its sub-resources allow servers to support advanced query capabilities such as stored queries, joins, complex predicates, etc...

Examples:
   Example: Perform a Join Query - this is a hypothetical sequence designed to show how the various resource and HTTP

```
         methods interact to perform an ad-hoc query.  In
         practice it is unlikely that a client would need to
         perform this many steps to execute a query

STEP 1: Does the server support advanced queries?
CLIENT                                                      SERVER
  |                                                           |
  |   OPTIONS /2.0/query HTTP/1.1                             |
  |   Host: wfs.someserver.com                                |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |   Allow: OPTIONS, POST                                    |
  |<----------------------------------------------------------|

  NOTE: if the server supported stored queries there would be an
        Accept header indicating the supported representations and
        the PUT method would be included in the Allow list

STEP 2: Create the query resource (no content)
CLIENT                                                      SERVER
  |                                                           |
  |   POST /2.0/query HTTP/1.1                                |
  |   Host: wfs.someserver.com                                |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 201 Created                                    |
  |   Location: /2.0/query/7f26634c                           |
  |<----------------------------------------------------------|

STEP 3: Using the OPTION method a client can discover the query
        capabilities of the server.

-> Does the server support ad-hoc queries?
CLIENT                                                      SERVER
  |                                                           |
  |   OPTIONS /2.0/query/7f26634c/filter HTTP/1.1             |
  |   Host: wfs.someserver.com                                |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |   Allow: OPTIONS, GET, PUT                                |
  |   Accept: urn:ogc:def:query Language:OGC-FES:Filter,      |
  |           urn:ogc:def:query Language:OGC-CSW:Cql          |
  |<----------------------------------------------------------|
  |                                                           |
  |   OPTIONS /2.0/query/7f26634c/typeNames HTTP/1.1          |
  |   Host: wfs.someserver.com                                |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |   Allow: OPTIONS, GET, PUT                                |
  |<----------------------------------------------------------|

  NOTE: Since the PUT method is allowed for these resources
        ad-hoc queries can be supported.  We also see that the
        server support predicates encoded using OGC Filter or
        CQL. The other resources that comprise an ad-hoc query
        can be similarly interrogated to determine the server's
        capabilities.

-> What query response representations does the server provide
CLIENT                                                      SERVER
  |                                                           |
  |   OPTIONS /2.0/query/7f26634c HTTP/1.1                    |
  |   Host: wfs.someserver.com                                |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |   Allow: OPTIONS, GET, DELETE                             |
  |   Accept: application/gml+xml; version=3.2,               |
  |           application/gml+xml; version=3.2,               |
  |           application/gml+xml; version=3.2,               |
  |           application/json                                |
  |<----------------------------------------------------------|

STEP 4: Edit the query resource to add feature type names to the
        ad-hoc query.
CLIENT                                                      SERVER
  |                                                           |
  |   PUT /2.0/query/7f26634c/typeNames HTTP/1.1              |
  |   Host: wfs.someserver.com                                |
  |   Content-Type: text/plain                                |
  |                                                           |
  |   PARKL_1M ROADL_1M                                       |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |<----------------------------------------------------------|

  NOTES:
  -> encoded just like the typeNames parameter for the XML
     encoding (i.e. a space-separated list of values on a
     single line)
```

```
      -> the "aliases" resource can be used to associate aliases
         with each feature type; the two lists must match up 1:1

    STEP 5: Edit the query resource to add a filter
    CLIENT                                                    SERVER
      |                                                         |
      |   PUT /2.0/query/7f26634c/filter HTTP/1.1               |
      |   Host: wfs.someserver.com                              |
      |   Content-Type: urn:ogc:def:query Language:OGC-FES:Filter |
      |                                                         |
      |   <?xml version="1.0"?>                                 |
      |   <fes:Filter> ... </fes:Filter>                        |
      |-------------------------------------------------------->|
      |                                                         |
      |   HTTP/1.1 200 OK                                       |
      |<--------------------------------------------------------|

    STEP 6: Get the collection of features that satisfy the query.
    CLIENT                                                    SERVER
      |                                                         |
      |   GET /2.0/query/7f26634c HTTP/1.1                      |
      |   Host: wfs.someserver.com                              |
      |   Accept: application/gml+xml; version=3.2              |
      |-------------------------------------------------------->|
      |                                                         |
      |   HTTP/1.1 200 OK                                       |
      |   Content-Type: text/xml                                |
      |                                                         |
      |   <?xml version="1.0"?>                                 |
      |   <wfs:FeatureCollection> ... </wfs:FeatureCollection>  |
      |<--------------------------------------------------------|

    STEP 7: Delete the query resource (or it could just expire)
    CLIENT                                                    SERVER
      |                                                         |
      |   DELETE /2.0/query/7f26634c                            |
      |   Host: wfs.someserver.com                              |
      |-------------------------------------------------------->|
      |                                                         |
      |   HTTP/1.1 200 OK                                       |
      |<--------------------------------------------------------|

X.7.2 Stored queries (=GetFeature)
   Resources (relative to the service root URL):
      - see X.7.1

   Methods:
      - see X.7.1

   Description:
   - stored queries are create and behave like ad-hoc queries with
     the following exceptions:
     -> the {Query Id} shall be assigned by the client
     -> the response to an OPTIONS request on the query resource
        shall include an Accept header indicating the supported
        query languages
   - server that support stored queries shall support stored
     queries defined using the sub-resources (c), (d), (e), (f),
     (g) (see X.7.1) and using a WFS query expression indicated by the
     content type urn:ogc:def:queryLanguage:OGC-WFS::WFS_QueryExpression
   - other query languages may be supported (e.g. SQL) but these are
     not described in this standard
   - the notation {variable-name} is used in the text of the query
     to indicate parameter substitution; the parameters appear as
     query parameters

   Examples:
      Example: Does the server support stored queries?
      CLIENT                                                  SERVER
        |                                                       |
        |   OPTIONS /2.0/query HTTP/1.1                         |
        |   Host: wfs.someserver.com                            |
        |------------------------------------------------------>|
        |                                                       |
        |   HTTP/1.1 200 OK                                     |
        |   Allow: OPTIONS, POST, PUT                           |
        |   Accept: urn:ogc:def:queryLanguage:OGC-WFS::WFS_QueryExpression,
        |           application/x-sql                           |
        |<------------------------------------------------------|

        NOTES:
        -> the server supports stored queries expressed as WFS query
           expressions encoded in XML or as SQL queries

      Example: Create a stored query using the /typeNames and
               /filter sub-resources
      STEP 1: Create an empty stored query
      CLIENT                                                  SERVER
        |                                                       |
        |   PUT /2.0/query/ParksInPolygon HTTP/1.1              |
        |   Host: wfs.someserver.com                            |
        |------------------------------------------------------>|
        |                                                       |
        |   HTTP/1.1 201 Created                                |
```

```
  |   Location:/2.0/query/FeaturesInPolygon                    |
  |<-----------------------------------------------------------|

  NOTES:
  -> if the specified query name already exists, the server shall
     return a "409 Conflict" HTTP code

STEP 2: Add the PARKL_1M feature type
CLIENT                                                     SERVER
  |                                                           |
  |   PUT /2.0/query/ParksInPolygon/typeNames HTTP/1.1        |
  |   Host: wfs.someserver.com                                |
  |   Content-Type: text/plain                                |
  |                                                           |
  |   PARKL_1M                                                |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |<----------------------------------------------------------|

STEP 3: Add a filter to the stored query
CLIENT                                                     SERVER
  |                                                           |
  |   PUT /2.0/query/ParksInPolygon/filter HTTP/1.1           |
  |   Host: wfs.someserver.com                                |
  |   Content-Type: urn:ogc:def:query Language:OGC-FES:Filter |
  |                                                           |
  |   <?xml version="1.0"?>                                   |
  |   <fes:Filter>                                            |
  |      <fes:Within>                                         |
  |         <fes:ValueReference>geometry</fes:ValueReference> |
  |         ${AreaOfInterest}                                 |
  |      </fes:Within>                                        |
  |   </fes:Filter>                                           |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |<----------------------------------------------------------|

STEP 4: Get the collection of features that satisfy the query
        for a specified value of {AreaOfInterest}.
CLIENT                                                     SERVER
  |                                                           |
  |   GET /2.0/query/ParksInPolygon?AreaOfInterest=.. HTTP/1.1|
  |   Host: wfs.someserver.com                                |
  |   Accept: application/gml+xml; version=3.2                |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |   Content-Type: application/gml+xml; version=3.2          |
  |                                                           |
  |   <?xml version="1.0"?>                                   |
  |   <wfs:FeatureCollection> ... </wfs:FeatureCollection>    |
  |<----------------------------------------------------------|

Example: Create the same query as in the previous example using
         a WFS query expression.
CLIENT                                                     SERVER
  |                                                           |
  |   PUT /2.0/query/ParksInPolygon HTTP/1.1                  |
  |   Host: wfs.someserver.com                                |
  |   ContentType: urn:ogc:def:queryLanguage:OGC-WFS::WFS_QueryExpression
  |                                                           |
  |   <?xml version="1.0"?>                                   |
  |   <wfs:Query typeNames="myns:Parks">                      |
  |      <fes:Filter>                                         |
  |         <fes:Within>                                      |
  |         <fes:ValueReference>geometry</fes:ValueReference> |
  |         ${AreaOfInterest}                                 |
  |         </fes:Within>                                     |
  |      </fes:Filter>                                        |
  |   </wfs:Query>                                            |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 201 Created                                    |
  |   Location: /2.0/query/ParksInPolygon                     |
  |<----------------------------------------------------------|

CLIENT                                                     SERVER
  |                                                           |
  |   GET /2.0/query/ParksInPolygon?AreaOfInterest=.. HTTP/1.1|
  |   Host: wfs.someserver.com                                |
  |   Accept: application/gml+xml; version=3.2                |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |   Content-Type: application/gml+xml; version=3.2          |
  |                                                           |
  |   <?xml version="1.0"?>                                   |
  |   <wfs:FeatureCollection> ... </wfs:FeatureCollection>    |
  |<----------------------------------------------------------|


Example: Create a more complex stored query.  In this example
```

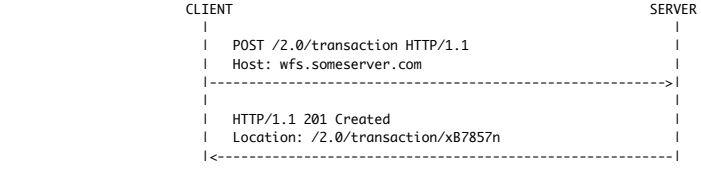
```
            the stored query actually executes three queries. WFS
            query expressions are used to encode each of the
            member queries.
      CLIENT                                                  SERVER
        |                                                       |
        |   PUT /2.0/query/FeaturesInPolygon HTTP/1.1           |
        |   Host: wfs.someserver.com                            |
        |   ContentType: urn:ogc:def:queryLanguage:OGC-WFS::WFS_QueryExpression
        |                                                       |
        |   <?xml version="1.0"?>                               |
        |   <wfs:Query typeNames="myns:Parks">                  |
        |      <fes:Filter>                                     |
        |         <fes:Within>                                  |
        |         <fes:ValueReference>geometry</fes:ValueReference> |
        |         ${AreaOfInterest}                             |
        |         </fes:Within>                                 |
        |      </fes:Filter>                                    |
        |   </wfs:Query>                                        |
        |------------------------------------------------------>|
        |                                                       |
        |   HTTP/1.1 201 Created                                |
        |   Location: /2.0/query/FeaturesInPolygon              |
        |<------------------------------------------------------|

      CLIENT                                                  SERVER
        |                                                       |
        |   PUT /2.0/query/FeaturesInPolygon HTTP/1.1           |
        |   Host: wfs.someserver.com                            |
        |   ContentType: urn:ogc:def:queryLanguage:OGC-WFS::WFS_QueryExpression
        |                                                       |
        |   <?xml version="1.0"?>                               |
        |   <wfs:Query typeNames="myns:Lakes">                  |
        |      <fes:Filter>                                     |
        |         <fes:Within>                                  |
        |            <fes:ValueReference>region</fes:ValueReference>|
        |            ${AreaOfInterest}                          |
        |         </fes:Within>                                 |
        |      </fes:Filter>                                    |
        |   </wfs:Query>                                        |
        |------------------------------------------------------>|
        |                                                       |
        |   HTTP/1.1 200 OK                                     |
        |<------------------------------------------------------|

      CLIENT                                                  SERVER
        |                                                       |
        |   PUT /2.0/query/FeaturesInPolygon HTTP/1.1           |
        |   Host: wfs.someserver.com                            |
        |   ContentType: urn:ogc:def:queryLanguage:OGC-WFS::WFS_QueryExpression
        |                                                       |
        |   <?xml version="1.0"?>                               |
        |   <wfs:Query typeNames="myns:Rivers">                 |
        |      <fes:Filter>                                     |
        |         <fes:Within>                                  |
        |            <fes:ValueReference>region</fes:ValueReference>|
        |            ${AreaOfInterest}                          |
        |         </fes:Within>                                 |
        |      </fes:Filter>                                    |
        |   </wfs:Query>                                        |
        |------------------------------------------------------>|
        |                                                       |
        |   HTTP/1.1 200 OK                                     |
        |<------------------------------------------------------|

      Example: Create a stored query using SQL.
      CLIENT                                                  SERVER
        |                                                       |
        |   PUT /2.0/query/Neighbours HTTP/1.1                  |
        |   Host: wfs.someserver.com                            |
        |   ContentType: application/x-sql                      |
        |                                                       |
        |   select c1.cntry_name                                |
        |     from country c1, country c2                       |
        |    where touches(c1.the_geom,c2.the_geom)='T'         |
        |      and c2.cntry_name='{country}'                    |
        |------------------------------------------------------>|
        |                                                       |
        |   HTTP/1.1 201 Created                                |
        |<------------------------------------------------------|

X.8 Complex Transactions (=Transaction)
   Resources (relative to the service root URL):
      (a) transaction
      (b) transaction/{Transaction Id}
      (c) transaction/{Transaction Id}/{Feature Type}
      (d) transaction/{Transaction Id}/{Feature Type}/{Feature Id}
      (e) transaction/{Transaction Id}/{Feature Type}/{Feature Id}/Properties/{Property Name}

   Methods:
     ---------------------------------------------------------------------------
     RESOURCE   METHOD   ACTION
     ---------------------------------------------------------------------------
       all      OPTIONS  - gets list of supported representations
                           and methods for the transaction and sub-resources
```

		via the Accept and Allow HTTP headers
(a)	GET	- not specified by this standard (see X.3.5)
(b)	GET	- not specified by this standard (see X.3.6)
(c)	GET	- not specified by this standard (see X.3.6)
(d)	GET	- gets a representation of the feature type with the specified feature id from the transaction resource; this representation of the feature is only valid in the context of the transaction and does not permanently become part of the server's datastore until the transaction is committed
(e)	GET	- get the value of the specified property name from the transaction resource; this value is only valid in the context of the transaction and does not permanently become part of the server's datastore until the transaction is committed
(a)	POST	- creates an new transaction resource
(b)	POST	- not specified by this standard (see X.3.6)
(c)	POST	- creates a new instance of the specified feature type within the context of the transaction; a representation of the new feature is provided in body of the request
(d)	POST	- not specified by this standard (see X.3.6)
(e)	POST	- not specified by this standard (see X.3.6)
(a)	PUT	- not specified by this standard (see X.3.6)
(b)	PUT	- used to commit the transaction
(c)	PUT	- not specified by this standard (see X.3.6)
(d)	PUT	- replaces the existing feature with the specified identifier; a representation of the replacement feature is provided in the body of the request
(e)	PUT	- replace the existing value of the specified property; a representation of the replacement value is provided in the body of the request
(a)	DELETE	- not specified by this standard (see X.3.6)
(b)	DELETE	- deletes the transaction and rolls back any pending changes
(c)	DELETE	- deletes all instances of the specified feature type (should probably not allow this to happen)
(d)	DELETE	- deletes the feature with the specified identifier
(e)	DELETE	- sets the value of the specified property to NULL

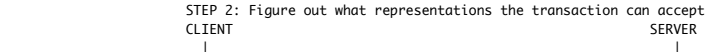
Description:
- supports complex transactions such as transactions on multiple features that need to happen atomically (as opposed to modifying a single features as in clause 6)
- POST, PUT and DELETE methods are used to perform insert, update and delete actions within the transaction
- the Native action is not supported in the REST binding
  - extended Transaction operators are not supported either
- all actions performed on the transaction (i.e. POST, PUT, DELETE) are not committed to the server's datastore until the token "commit" is sent to the server with the PUT method
  -> the response to a transaction, upon commit shall be the standard wfs:TransactionResponse

- a transaction can be rolled back by deleting it without sending a commit token; the response is simply a "200 OK" HTTP code

Examples:
Example: Insert, update and delete features atomically
STEP 1: Create a transaction resource



STEP 2: Figure out what representations the transaction can accept



```
  |   OPTIONS /2.0/transaction/xB7857n HTTP/1.1               |
  |   Host: wfs.someserver.com                                |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |   Allow: OPTIONS, POST, PUT, DELETE                       |
  |   Accept: application/gml+xml; version=3.2,               |
  |           application/gml+xml; version=3.1,               |
  |           application/json                                |
  |<----------------------------------------------------------|

STEP 3: Insert a new feature
CLIENT                                                      SERVER
  |                                                           |
  |   POST /2.0/transaction/xB7857n/INWATERA_1M HTTP/1.1      |
  |   Host: wfs.someserver.com                                |
  |   ContentType: application/gml+xml; version=3.2           |
  |                                                           |
  |   <?xml version="1.0"?>                                   |
  |   <INWATERA_1M>...</INWATERA_1M>                          |
  |---------------------------------------------------------->|
  |                                                           |
  |    HTTP/1.1 201 Created                                   |
  |    Location: /2.0/transaction/xxB7857n/INWATERA_1M/57634  |
  |<----------------------------------------------------------|

STEP 4: Replace a feature
CLIENT                                                      SERVER
  |                                                           |
  |    PUT /2.0/transaction/xB7857n/BUILTUPA_1M/12357 HTTP/1.1|
  |    Host: wfs.someserver.com                               |
  |    ContentType: application/gml+xml; version=3.2          |
  |                                                           |
  |    <?xml version="1.0"?>                                  |
  |    <BUILTUPA_1M>...</BUILTUPA_1M>                         |
  |---------------------------------------------------------->|
  |                                                           |
  |    HTTP/1.1 200 OK                                        |
  |<----------------------------------------------------------|

STEP 5: Update the property of a feature
CLIENT                                                      SERVER
  |                                                           |
  |    PUT /2.0/transaction/xB7857n/BUILTUPA_1M/12357/Properties/F_CODE HTTP/1.1
  |    Host: wfs.someserver.com                               |
  |    ContentType: text/plain                                |
  |                                                           |
  |    307                                                    |
  |---------------------------------------------------------->|
  |                                                           |
  |    HTTP/1.1 200 OK                                        |
  |<----------------------------------------------------------|

STEP 6: Delete a feature
CLIENT                                                      SERVER
  |                                                           |
  |    DELETE /2.0/transaction/xB7857n/WATERL_1M/329876 HTTP/1.1
  |    Host: wfs.someserver.com                               |
  |---------------------------------------------------------->|
  |                                                           |
  |    HTTP/1.1 200 OK                                        |
  |<----------------------------------------------------------|

STEP 7: Commit the transaction
CLIENT                                                      SERVER
  |                                                           |
  |    PUT /2.0/transaction/xB7857n HTTP/1.1                  |
  |    Host: wfs.someserver.com                               |
  |    ContentType: text/plain                                |
  |                                                           |
  |    commit                                                 |
  |---------------------------------------------------------->|
  |                                                           |
  |    HTTP/1.1 200 OK                                        |
  |    ContentType: text/xml                                  |
  |                                                           |
  |    <?xml version="1.0"?>                                  |
  |    <wfs:TransactionResponse> ..</wfs:TransactionResponse> |
  |<----------------------------------------------------------|

X.9 Capabilities document example
CLIENT                                                      SERVER
  |                                                           |
  |    GET /2.0  HTTP/1.1                                     |
  |    Host: wfs.someserver.com                               |
  |    Accept: text/xml                                       |
  |---------------------------------------------------------->|
  |                                                           |
  |    HTTP/1.1 200 OK                                        |
  |    Content-Type: text/xml                                 |
  |                                                           |
  |   <?xml version="1.0"?>                                   |
  |   <WFS_Capabilities                                       |
  |      version="2.0.0"                                      |
```

```
|      xmlns="http://www.opengis.net/wfs/2.0"               |
|      xmlns:gml="http://www.opengis.net/gml/3.2"           |
|      xmlns:fes="http://www.opengis.net/fes/2.0"           |
|      xmlns:xlink="http://www.w3.org/1999/xlink"           |
|      xmlns:ows="http://www.opengis.net/ows/1.1"           |
|      xmlns:xsd="http://www.w3.org/2001/XMLSchema"         |
|      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"|
|      xsi:schemaLocation="http://www.opengis.net/wfs/2.0   |
|          http://www.pvretano.com/schemas/wfs/2.0.0/wfs.xsd|
|          http://www.opengis.net/ows/1.1                   |
|          http://schemas.opengis/net/ows/1.1.0/owsAll.xsd">|
|      <ServiceRoot>http://www.BlueOx.org/2.0</ServceRoot>  |
|      <ows:ServiceIdentification>                          |
|         <ows:Title>OGC Member WFS</ows:Title>             |
|         <ows:Abstract>                                    |
|            Web Feature Service maintained by NSDI data    |
|            provider, serving FGDC framework layer XXX;    |
|            contact Paul.Bunyon@BlueOx.org                 |
|         </ows:Abstract>                                   |
|         <ows:Keywords>                                    |
|            <ows:Keyword>FGDC</ows:Keyword>                |
|            <ows:Keyword>NSDI</ows:Keyword>                |
|            <ows:Keyword>Framework Data Layer</ows:Keyword>|
|            <ows:Keyword>BlueOx</ows:Keyword>              |
|            <ows:Type>String</ows:Type>                    |
|         </ows:Keywords>                                   |
|         <ows:ServiceType>WFS</ows:ServiceType>            |
|         <ows:ServiceTypeVersion>2.0.0</ows:ServiceTypeVersion>
|         <ows:ServiceTypeVersion>1.1.0</ows:ServiceTypeVersion>
|         <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
|         <ows:AccessConstraints>NONE</ows:AccessConstraints>
|      </ows:ServiceIdentification>                         |
|      <ows:ServiceProvider>                                |
|         <ows:ProviderName>BlueOx Inc.</ows:ProviderName>  |
|         <ows:ProviderSite xlink:href="http://www.cubewerx.com"/>
|         <ows:ServiceContact>                              |
|            <ows:IndividualName>Paul Bunyon</ows:IndividualName>
|            <ows:PositionName>Mythology Manager</ows:PositionName>
|            <ows:ContactInfo>                              |
|               <ows:Phone>                                 |
|                  <ows:Voice>1.800.BIG.WOOD</ows:Voice>    |
|                  <ows:Facsimile>1.800.FAX.WOOD</ows:Facsimile>
|               </ows:Phone>                                |
|               <ows:Address>                               |
|                  <ows:DeliveryPoint>                      |
|                     North Country                         |
|                  </ows:DeliveryPoint>                     |
|                  <ows:City>Small Town</ows:City>          |
|                  <ows:AdministrativeArea>                 |
|                     Rural County                          |
|                  </ows:AdministrativeArea>                |
|                  <ows:PostalCode>12345</ows:PostalCode>   |
|                  <ows:Country>USA</ows:Country>           |
|                  <ows:ElectronicMailAddress>              |
|                     Paul.Bunyon@BlueOx.org                |
|                  </ows:ElectronicMailAddress>             |
|               </ows:Address>                              |
|               <ows:OnlineResource                         |
|               xlink:href="http://www.BlueOx.org/contactUs"/>
|               <ows:HoursOfService>24x7</ows:HoursOfService>
|               <ows:ContactInstructions>                   |
|                  eMail Paul with normal requsts; Phone    |
|                  Paul for emergency requests; if you get  |
|                  voice mail and your request can't wait,  |
|                  contact another mythological figure listed
|                  on the contactUs page of our web site.   |
|               </ows:ContactInstructions>                  |
|            </ows:ContactInfo>                             |
|            <ows:Role>PointOfContact</ows:Role>            |
|         </ows:ServiceContact>                             |
|      </ows:ServiceProvider>                               |
|      <ows:OperationsMetadata>                             |
|         <ows:Constraint name="AutomaticDataLocking">      |
|            <ows:NoValues/>                                |
|            <ows:DefaultValue>TRUE</ows:DefaultValue>      |
|         </ows:Constraint>                                 |
|         <ows:Constraint name="PreservesSiblingOrder">     |
|            <ows:NoValues/>                                |
|            <ows:DefaultValue>TRUE</ows:DefaultValue>      |
|         </ows:Constraint>                                 |
|         <!-- ************************************** -->  |
|         <!-- *   CONFORMANCE DECLARATION           * -->  |
|         <!-- ************************************** -->  |
|         <ows:Constraint name="ImplementsBasicWFS">        |
|            <ows:NoValues/>                                |
|            <ows:DefaultValue>TRUE</ows:DefaultValue>      |
|         </ows:Constraint>                                 |
|         <ows:Constraint name="ImplementsTransactionalWFS">|
|            <ows:NoValues/>                                |
|            <ows:DefaultValue>TRUE</ows:DefaultValue>      |
|         </ows:Constraint>                                 |
|         <ows:Constraint name="ImplementsLockingWFS">      |
|            <ows:NoValues/>                                |
|            <ows:DefaultValue>TRUE</ows:DefaultValue>      |
```

```
|         </ows:Constraint>                                    |
|         <ows:Constraint name="KVPEncoding">                  |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>TRUE</ows:DefaultValue>         |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="XMLEncoding">                  |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>TRUE</ows:DefaultValue>         |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="SOAPEncoding">                 |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>TRUE</ows:DefaultValue>         |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="ImplementsInheritance">        |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>TRUE</ows:DefaultValue>         |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="ImplementsRemoteResolve">      |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>TRUE</ows:DefaultValue>         |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="ImplementsResultPaging">       |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>TRUE</ows:DefaultValue>         |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="ImplementsStandardJoins">      |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>TRUE</ows:DefaultValue>         |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="ImplementsSpatialJoins">       |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>TRUE</ows:DefaultValue>         |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="ImplementsTemporalJoins">      |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>TRUE</ows:DefaultValue>         |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="ImplementsFeatureVersioning">
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>TRUE</ows:DefaultValue>         |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="ManageStoredQueries">          |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>TRUE</ows:DefaultValue>         |
|         </ows:Constraint>                                    |
|         <!-- ****************************************** -->|
|         <!-- *   CAPACITY CONSTRAINTS                * -->|
|         <!-- ****************************************** -->|
|         <ows:Constraint name="PagingIsTransactionSafe">      |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>FALSE</ows:DefaultValue>        |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="CountDefault">                 |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>1000</ows:DefaultValue>         |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="ResolveTimeoutDefault">        |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>300</ows:DefaultValue>          |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="SortLevelLimit">               |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>1</ows:DefaultValue>            |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="ResolveLocalScope">            |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>*</ows:DefaultValue>            |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="ResolveRemoteScope">           |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>5</ows:DefaultValue>            |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="ResponseCacheTimeout">         |
|            <ows:NoValues/>                                   |
|            <ows:DefaultValue>300</ows:DefaultValue>          |
|         </ows:Constraint>                                    |
|         <ows:Constraint name="QueryExpressions">             |
|            <ows:AllowedValues>                               |
|               <ows:Value>wfs:Query</ows:Value>               |
|               <ows:Value>wfs:StoredQuery</ows:Value>         |
|            </ows:AllowedValues>                              |
|         </ows:Constraint>                                    |
|         <!-- ****************************************** -->|
|      </ows:OperationsMetadata>                               |
|      <FeatureTypeList>                                       |
|         <Link rel="describedby"                              |
|               type="text/xml; gmlver=3.2"                    |
|               href="http://www.BlueOx.org/2.0/schema"/>      |
|         <FeatureType xmlns:bo="http://www.BlueOx.org/BlueOx"
|            <Link rel="self"                                  |
|                  type="application/gml+xml; version=3.2"     |
|                  href="http://www.BlueOx.org/2.0/FeatureTypes/Woods"/>
|            <Link rel="alternate"                             |
|                  type="application/json"                     |
```

```
|                href="http://www.BlueOx.org/2.0/FeatureTypes/Woods"/>
|            <Link rel="describedby"                          |
|                  type="text/xml; gmlver=3.2"               |
|                  href="http://www.BlueOx.org/2.0/schema/Woods"/>
|            <Name>bo:Woods</Name>                            |
|            <Title>The Great Northern Forest</Title>         |
|            <Abstract>                                       |
|               Describes the arborial diversity of the       |
|               Great Northern Forest.                        |
|            </Abstract>                                      |
|            <ows:Keywords>                                   |
|               <ows:Keyword>forest</ows:Keyword>             |
|               <ows:Keyword>north</ows:Keyword>              |
|               <ows:Keyword>woods</ows:Keyword>              |
|               <ows:Keyword>arborial</ows:Keyword>           |
|               <ows:Keyword>diversity</ows:Keyword>          |
|            </ows:Keywords>                                  |
|            <DefaultCRS>urn:ogc:def:crs:EPSG::6269</DefaultCRS>
|            <OtherCRS>urn:ogc:def:crs:EPSG::32615</OtherCRS>
|            <OtherCRS>urn:ogc:def:crs:EPSG::32616</OtherCRS>
|            <OtherCRS>urn:ogc:def:crs:EPSG::32617</OtherCRS>
|            <OtherCRS>urn:ogc:def:crs:EPSG::32618</OtherCRS>
|            <ows:WGS84BoundingBox>                           |
|               <ows:LowerCorner>-180 -90</ows:LowerCorner>   |
|               <ows:UpperCorner>180 90</ows:UpperCorner>     |
|            </ows:WGS84BoundingBox>                          |
|         </FeatureType>                                      |
|         <FeatureType                                        |
|            <Link rel="self"                                 |
|                  type="application/gml+xml; version=3.2"    |
|                  href="http://www.BlueOx.org/2.0/FeatureTypes/Lakes"/>
|            <Link rel="alternate"                            |
|                  type="application/json"                    |
|                  href="http://www.BlueOx.org/2.0/FeatureTypes/Lakes"/>
|            <Link rel="describedby"                          |
|                  type="text/xml; gmlver=3.2"               |
|                  href="http://www.BlueOx.org/2.0/schema/Lakes"/>
|            <Name>bo:Lakes</Name>                            |
|            <Title>The Great Northern Lakes</Title>          |
|            <Abstract>                                       |
|               Lake boundaries for all lakes in the          |
|               Great Northern Forest.                        |
|            </Abstract>                                      |
|            <ows:Keywords>                                   |
|               <ows:Keyword>lakes</ows:Keyword>              |
|               <ows:Keyword>boundaries</ows:Keyword>         |
|               <ows:Keyword>water</ows:Keyword>              |
|               <ows:Keyword>hydro</ows:Keyword>              |
|            </ows:Keywords>                                  |
|            <DefaultCRS>urn:ogc:def:crs:EPSG::6269</DefaultCRS>
|            <OtherCRS>urn:ogc:def:crs:EPSG::32615</OtherCRS>
|            <OtherCRS>urn:ogc:def:crs:EPSG::32616</OtherCRS>
|            <OtherCRS>urn:ogc:def:crs:EPSG::32617</OtherCRS>
|            <OtherCRS>urn:ogc:def:crs:EPSG::32618</OtherCRS>
|            <ows:WGS84BoundingBox>                           |
|               <ows:LowerCorner>-180 -90</ows:LowerCorner>   |
|               <ows:UpperCorner>180 90</ows:UpperCorner>     |
|            </ows:WGS84BoundingBox>                          |
|         </FeatureType>                                      |
|      </FeatureTypeList>                                     |
|      <fes:Filter_Capabilities>                              |
|         <fes:Conformance>                                   |
|            <fes:Constraint name="ImplementsQuery">         |
|               <ows:NoValues/>                               |
|               <ows:DefaultValue>TRUE</ows:DefaultValue>     |
|            </fes:Constraint>                                |
|            <fes:Constraint name="ImplementsAdHocQuery">    |
|               <ows:NoValues/>                               |
|               <ows:DefaultValue>TRUE</ows:DefaultValue>     |
|            </fes:Constraint>                                |
|            <fes:Constraint name="ImplementsFunctions">     |
|               <ows:NoValues/>                               |
|               <ows:DefaultValue>TRUE</ows:DefaultValue>     |
|            </fes:Constraint>                                |
|            <fes:Constraint name="ImplementsMinStandardFilter">
|               <ows:NoValues/>                               |
|               <ows:DefaultValue>TRUE</ows:DefaultValue>     |
|            </fes:Constraint>                                |
|            <fes:Constraint name="ImplementsStandardFilter">
|               <ows:NoValues/>                               |
|               <ows:DefaultValue>TRUE</ows:DefaultValue>     |
|            </fes:Constraint>                                |
|            <fes:Constraint name="ImplementsMinSpatialFilter">
|               <ows:NoValues/>                               |
|               <ows:DefaultValue>TRUE</ows:DefaultValue>     |
|            </fes:Constraint>                                |
|            <fes:Constraint name="ImplementsSpatialFilter">|
|               <ows:NoValues/>                               |
|               <ows:DefaultValue>TRUE</ows:DefaultValue>     |
|            </fes:Constraint>                                |
|            <fes:Constraint name="ImplementsMinTemporalFilter">
|               <ows:NoValues/>                               |
|               <ows:DefaultValue>TRUE</ows:DefaultValue>     |
|            </fes:Constraint>                                |
```

```
|            <fes:Constraint name="ImplementsTemporalFilter">
|               <ows:NoValues/>                              |
|               <ows:DefaultValue>TRUE</ows:DefaultValue>    |
|            </fes:Constraint>                               |
|            <fes:Constraint name="ImplementsVersionNav">    |
|               <ows:NoValues/>                              |
|               <ows:DefaultValue>FALSE</ows:DefaultValue>   |
|            </fes:Constraint>                               |
|            <fes:Constraint name="ImplementsSorting">       |
|               <ows:NoValues/>                              |
|               <ows:DefaultValue>FALSE</ows:DefaultValue>   |
|            </fes:Constraint>                               |
|            <fes:Constraint name="ImplementsExtendedOperators">
|               <ows:NoValues/>                              |
|               <ows:DefaultValue>FALSE</ows:DefaultValue>   |
|            </fes:Constraint>                               |
|         </fes:Conformance>                                 |
|         <fes:Id_Capabilities>                              |
|            <fes:ResourceIdentifier name="fes:ResourceId"/>|
|         </fes:Id_Capabilities>                             |
|         <fes:Scalar_Capabilities>                          |
|            <fes:LogicalOperators/>                         |
|            <fes:ComparisonOperators>                       |
|               <fes:ComparisonOperator                      |
|                    name="PropertyIsLessThan"/>             |
|               <fes:ComparisonOperator                      |
|                    name="PropertyIsGreaterThan"/>          |
|               <fes:ComparisonOperator                      |
|                    name="PropertyIsLessThanOrEqualTo"/>    |
|               <fes:ComparisonOperator                      |
|                    name="PropertyIsGreaterThanOrEqualTo"/>|
|               <fes:ComparisonOperator                      |
|                    name="PropertyIsEqualTo"/>              |
|               <fes:ComparisonOperator                      |
|                    name="PropertyIsNotEqualTo"/>           |
|               <fes:ComparisonOperator                      |
|                    name="PropertyIsLike"/>                 |
|               <fes:ComparisonOperator                      |
|                    name="PropertyIsBetween"/>              |
|               <fes:ComparisonOperator                      |
|                    name="PropertyIsNull"/>                 |
|               <fes:ComparisonOperator                      |
|                    name="PropertyIsNil"/>                  |
|            </fes:ComparisonOperators>                      |
|         </fes:Scalar_Capabilities>                         |
|         <fes:Spatial_Capabilities>                         |
|            <fes:GeometryOperands>                          |
|               <fes:GeometryOperand name="gml:Point"/>      |
|               <fes:GeometryOperand name="gml:MultiPoint"/>|
|               <fes:GeometryOperand name="gml:LineString"/>|
|               <fes:GeometryOperand name="gml:MultiLineString"/>
|               <fes:GeometryOperand name="gml:Curve"/>      |
|               <fes:GeometryOperand name="gml:MultiCurve"/>|
|               <fes:GeometryOperand name="gml:Polygon"/>    |
|               <fes:GeometryOperand name="gml:MultiPolygon"/>
|               <fes:GeometryOperand name="gml:Surface"/>    |
|               <fes:GeometryOperand name="gml:MultiSurface"/>
|               <fes:GeometryOperand name="gml:MultiGeometry"/>
|               <fes:GeometryOperand name="gml:Box"/>        |
|               <fes:GeometryOperand name="gml:Envelope"/>   |
|            </fes:GeometryOperands>                         |
|            <fes:SpatialOperators>                          |
|               <fes:SpatialOperator name="BBOX"/>           |
|               <fes:SpatialOperator name="Equals"/>         |
|               <fes:SpatialOperator name="Disjoint"/>       |
|               <fes:SpatialOperator name="Intersects"/>     |
|               <fes:SpatialOperator name="Touches"/>        |
|               <fes:SpatialOperator name="Crosses"/>        |
|               <fes:SpatialOperator name="Within"/>         |
|               <fes:SpatialOperator name="Contains"/>       |
|               <fes:SpatialOperator name="Overlaps"/>       |
|               <fes:SpatialOperator name="Beyond"/>         |
|               <fes:SpatialOperator name="DWithin"/>        |
|            </fes:SpatialOperators>                         |
|         </fes:Spatial_Capabilities>                        |
|         <fes:Temporal_Capabilities>                        |
|            <fes:TemporalOperands>                          |
|               <fes:TemporalOperand name="gml:validTime"/>  |
|               <fes:TemporalOperand name="gml:TimeInstant"/>
|               <fes:TemporalOperand name="gml:TimePeriod"/>|
|               <fes:TemporalOperand name="gml:timePosition"/>
|               <fes:TemporalOperand name="gml:timeInterval"/>
|               <fes:TemporalOperand name="gml:duration"/>   |
|            </fes:TemporalOperands>                         |
|            <fes:TemporalOperators>                         |
|               <fes:TemporalOperator name="After"/>         |
|               <fes:TemporalOperator name="Before"/>        |
|               <fes:TemporalOperator name="Begins"/>        |
|               <fes:TemporalOperator name="BegunBy"/>       |
|               <fes:TemporalOperator name="TContains"/>     |
|               <fes:TemporalOperator name="During"/>        |
|               <fes:TemporalOperator name="TEquals"/>       |
|               <fes:TemporalOperator name="TOverlaps"/>     |
|               <fes:TemporalOperator name="Meets"/>         |
```

```
|               <fes:TemporalOperator name="OverlappedBy"/> |
|               <fes:TemporalOperator name="MetBy"/>        |
|               <fes:TemporalOperator name="EndedBy"/>      |
|            </fes:TemporalOperators>                       |
|         </fes:Temporal_Capabilities>                      |
|         <fes:Functions>                                   |
|            <fes:Function name="min">                      |
|               <fes:Returns>xsd:double</fes:Returns>       |
|               <fes:Arguments>                             |
|                  <fes:Argument name="value1">             |
|                     <fes:Type>xsd:double</fes:Type>       |
|                  </fes:Argument>                          |
|                  <fes:Argument name="value2">             |
|                     <fes:Type>xsd:double</fes:Type>       |
|                  </fes:Argument>                          |
|               </fes:Arguments>                            |
|            </fes:Function>                                |
|            <fes:Function name="max">                      |
|               <fes:Returns>xsd:double</fes:Returns>       |
|               <fes:Arguments>                             |
|                  <fes:Argument name="value1">             |
|                     <fes:Type>xsd:double</fes:Type>       |
|                  </fes:Argument>                          |
|                  <fes:Argument name="value2">             |
|                     <fes:Type>xsd:double</fes:Type>       |
|                  </fes:Argument>                          |
|               </fes:Arguments>                            |
|            </fes:Function>                                |
|         </fes:Functions>                                  |
|      </fes:Filter_Capabilities>                           |
|   </WFS_Capabilities>                                     |
|<----------------------------------------------------------|
```