uDig WFS Test Plan

June 18, 2004

Refractions RESEARCH INC.

Submitted To: Program Manager
GeoConnections
Victoria, BC, Canada

Submitted By: David Zwiers
Refractions Research Inc.
Suite 400 – 1207 Douglas Street
Victoria, BC V8W 2E7
E-mail: dzwiers@refractions.net
Phone: (250) 383-3022
Fax: (250) 383-2140

TABLE OF CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

This document illustrates our testing plan for the Web Feature Server (WFS) data access component of the uDig project.

This document involves three main components:

- Definition of what data set will be used for testing;
- List of Web Feature Servers to test;
- Examples of expected Request and Response

The WFS component will be implemented as a GeoTools DataStore. GeoTools is an open-source library of GIS functionality. This library allows the uDig project to seamlessly change between data sources.

We intend to strictly adhere to the WFS specification provided by the Open GIS Consortium (OGC 02-058). By adhering to the specification we will be able to test against multiple WFS data sources.

Where applicable we will test our code with both live and static information. This means that test cases will be executed against both static XML files, and live WFS servers.

For each of these data sources, we plan to test the WFS DataStore's ability to initialize itself, generate feature types, read features and complete transactions.

2 GEOTOOLS DATASTORE API

The GeoTools DataStore API project strives to support as many geographical data formats as possible. Creating a WFS DataStore will allow users to gain access to a vast suite of tools through the GeoTools API. In order to transform a WFS instance into the GeoTools feature representation we will write an implementation of the DataStore interface.

Once a DataStore implementation is written for the uDig project, WFS data will be available for both GeoTools users, and other projects using the GeoTools DataStore API, such as GeoServer. The modular nature of GeoTools will allow the new WFS DataStores to quickly become part of the next GeoTools release.

The DataStore interface borrows most of its concepts and some of its syntax from the OpenGIS Consortium (OGC) Web Feature Server Specification:

- Feature - atomic unit of geographic information
- FeatureType - keeps track of which attributes each Feature can hold
- FeatureID - a unique id associated with each Feature (must start with a non-numeric character)

An extensive explanation of the API can be found at http://geotools.sourceforge.net/gt2docs/api/org/geotools/data/package-summary.html.

3 WEB FEATURE SERVICE

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

The requirements for a Web Feature Service are:

1. The interfaces must be defined in XML.

2. GML must be used to express features within the interface.

3. At a minimum a WFS must be able to present features using GML.

4. The predicate or filter language will be defined in XML and be derived from Catalog Query Language as defined in the OpenGIS Catalogue Interface Implementation Specification.

5. The datastore used to store geographic features should be opaque to client applications and their only view of the data should be through the WFS interface.

6. A WFS should use a subset of XPath expressions for referencing element properties.

The Web Feature Service interface is illustrated in Figure 1.

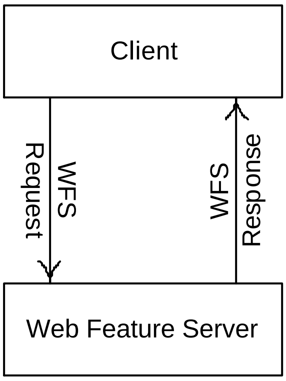

Figure 1 - Web Feature Service Interface

4 DATA SOURCES

Our testing plan includes performing two sets of tests. The goal of the first set is to ensure that the DataStore is performing as expected. This will be completed through the use of extensive jUnit tests using well-known data sets. These tests will include the use of XML files, GeoServer and CadCorp SIS.

The second group of tests will be used to test the WFS DataStore's level of interoperability. These tests will include a set of basic interactions to be performed on the following additional data sources:

- ESRI
- CARIS
- CubeWerx
- Galdos

4.1 XML Files

Our plan includes a large set of static files to test basic parsing functionality. The following is a list of WFS documents, which will be used to statically test the parsers.

WFS Test Documents:

- GetCapabilities
- DescribeFeatureType
- GetFeature
- LockFeature
- Transaction

For the most part these files will help complete low-level testing for portions of the WFS DataStore.

4.2 GeoServer

The GeoServer project is a full transactional Java (J2EE) implementation of the OpenGIS Consortium's Web Feature Server specification, with an integrated WMS. It is free software, available under the GPL 2.0 license. Our team has extensive experience with the WFS portion of GeoServer and intends to use GeoServer during our testing of the WFS DataStore. More information about GeoServer can be found at http://geoserver.sourceforge.net/.

4.3 CadCorp

The CadCorp SpatialInformation System (SIS) includes a WFS component. CadCorp markets their product as being compliant with the OGC WFS specification. At this time we intend to complete the same suite of tests with CadCorp's WFS as GeoServer. More information about CadCorp can be found at http://www.cadcorp.com/.

4.4 ESRI

ESRI ArcIMS provides the foundation for distributing high-end geographic information systems and mapping services via the Internet. ArcIMS software enables users to integrate local data sources with Internet data sources for display, query, and analysis in an easy-to-use web browser.

There are WMS and WFS connectors available for ArcIMS, which are Java web applications providing support for OGC WMS or WFS services. These connectors support read-only requests.

We intend to complete interoperability testing for the read-only portion of the WFS DataStore with ArcIMS. More information about ERSI's ArcIMS is available at http://www.esri.com/software/arcims/.

4.5 CARIS

CARIS Spatial Fusion provides the ability to read multiple data sources in their native formats. Spatial Fusion provides support for version 1.1.1 of the OGC WMS Specification Version 1.0.0 of the OGC WFS Specification is also available.

At the moment CARIS does not specify whether they support WFS Transactions. Therefore we intend to complete read-only testing, optionally including read/write interoperability testing if supported. More information about Spatial Fusion is available at http://www.caris.com/products/overview.cfm?productID=19.

4.6 CubeWerx

CubeSERV Web Feature Server builds on the CubeSTOR Spatial Data Warehouse solution by proving a web-based service for querying and performing transactional operations including database locking.

CubeSERV claims to be OGC WFS Version 1.0.0 compliant. We intend to complete a full range of interoperability tests against the CubeWerx WFS server. More information about CubeSERV is available at http://www.cubewerx.com/main/products/CubeSERV_WFS.html.

4.7 Galdos

Cartalinea is Galdos' geospatial web-services application. Cartalinea has implemented the OGC standards to improve their interoperability. Besides the WFS specification, Cartalinea also implemented the Web Catalog Service (WCS) and Simple Features SQL (SCS) specifications.

It is unclear if Catalinea supports WFS transactions. Therefore we intend to complete read-only testing, optionally including read/write interoperability testing if supported. More information about Catalinea is available at http://www.galdosinc.com/.

5 DATA STORE TESTS

5.1 Set-up

The data store initialization phase is of particular importance, because errors in this portion of development will affect the remaining operations. This portion will involve testing:

- GetCapabilities
  - Parser
  - Request Generation (Post + Get)

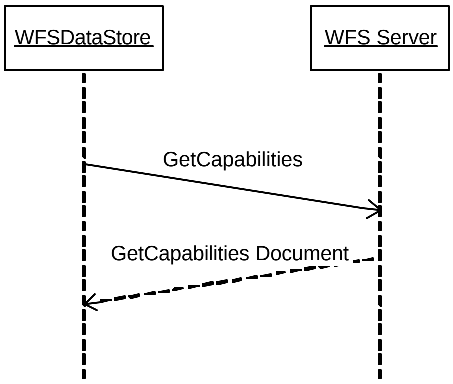

Figure 2 - GetCapabilities Interactions

- DescribeFeatureType
  - Parser
  - Request Generation (Post + Get)

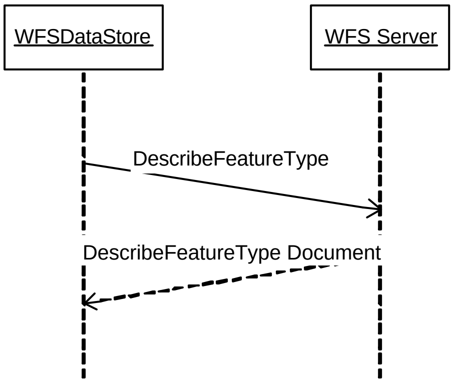

Figure 3 - DescribeFeatureType Interactions

- WFSDataStore internal data storage and retrieval

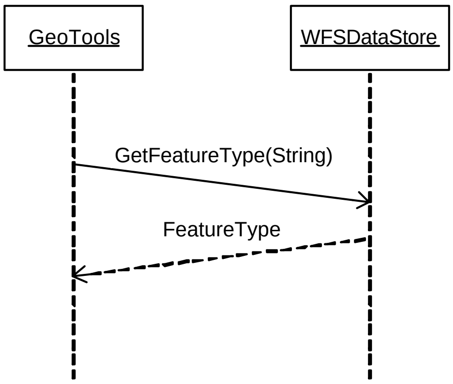

Figure 4 - GeoTools Interactions

The GetCapabilities parser will be tested using live servers or files containing responses, such as the following one generated by GeoServer.

5.1.1 GetCapabilities Document

```
<?xml version="1.0" encoding="UTF-8"?>
<WFS_Capabilities version="1.0.0" xmlns="http://www.opengis.net/wfs"
xmlns:gd="http://www.refractions.net" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs
http://www.refractions.net:8080/geoserver/data/capabilities/wfs/1.0.0/WFS-
capabilities.xsd">
  <Service>
    <Name>My GeoServer WFS</Name>
    <Title>My GeoServer WFS</Title>
    <Abstract>
This is a description of your Web Feature Server.

The GeoServer is a full transactional Web Feature Server, you may wish to
limit
GeoServer to a Basic service level to prevent modificaiton of your geographic
data.
     </Abstract>
    <Keywords>WFS, WMS, GEOSERVER</Keywords>
<OnlineResource>http://geoserver.sourceforge.net/html/index.php</OnlineResourc
e>
  <Fees>NONE</Fees>
  <AccessConstraints>NONE</AccessConstraints></Service>
  <Capability>
    <Request>
      <GetCapabilities>
        <DCPType><HTTP><Get
onlineResource="http://www.refractions.net:8080/geoserver/wfs/GetCapabilities?
"/></HTTP></DCPType>
        <DCPType><HTTP><Post
onlineResource="http://www.refractions.net:8080/geoserver/wfs/GetCapabilities"
/></HTTP></DCPType>
      </GetCapabilities>

      <DescribeFeatureType>
        <SchemaDescriptionLanguage><XMLSCHEMA/></SchemaDescriptionLanguage>
        <DCPType><HTTP><Get
onlineResource="http://www.refractions.net:8080/geoserver/wfs/DescribeFeatureT
ype?"/></HTTP></DCPType>
        <DCPType><HTTP><Post
onlineResource="http://www.refractions.net:8080/geoserver/wfs/DescribeFeatureT
ype"/></HTTP></DCPType>
      </DescribeFeatureType>

      <GetFeature>
        <ResultFormat><GML2/></ResultFormat>
        <DCPType><HTTP><Get
onlineResource="http://www.refractions.net:8080/geoserver/wfs/GetFeature?"/></
HTTP></DCPType>
        <DCPType><HTTP><Post
onlineResource="http://www.refractions.net:8080/geoserver/wfs/GetFeature"/></H
TTP></DCPType>
      </GetFeature>

      <Transaction>
        <DCPType><HTTP><Get
```

```
onlineResource="http://www.refractions.net:8080/geoserver/wfs/Transaction?"/><
/HTTP></DCPType>
        <DCPType><HTTP><Post
onlineResource="http://www.refractions.net:8080/geoserver/wfs/Transaction"/></
HTTP></DCPType>
      </Transaction>

      <LockFeature>
        <DCPType><HTTP><Get
onlineResource="http://www.refractions.net:8080/geoserver/wfs/LockFeature?"/><
/HTTP></DCPType>
        <DCPType><HTTP><Post
onlineResource="http://www.refractions.net:8080/geoserver/wfs/LockFeature"/></
HTTP></DCPType>
      </LockFeature>

      <GetFeatureWithLock>
        <ResultFormat><GML2/></ResultFormat>
        <DCPType><HTTP><Get
onlineResource="http://www.refractions.net:8080/geoserver/wfs/GetFeatureWithLo
ck?"/></HTTP></DCPType>
        <DCPType><HTTP><Post
onlineResource="http://www.refractions.net:8080/geoserver/wfs/GetFeatureWithLo
ck"/></HTTP></DCPType>
      </GetFeatureWithLock>
    </Request>
  </Capability><FeatureTypeList>
    <Operations>
      <Query/><Insert/><Update/><Delete/><Lock/>
    </Operations><FeatureType>
      <Name>gd:roads</Name>
      <Title>roads_Type</Title>
      <Abstract>Generated from roads</Abstract>
      <Keywords>roads</Keywords>

    <SRS>EPSG:26910</SRS>
    <LatLongBoundingBox minx="-122.248437678166" miny="47.2480730062538"
maxx="-122.123411805448" maxy="47.37217444602"/>
  </FeatureType>
  <FeatureType>
    <Name>gd:swamps</Name>
    <Title>swamps_Type</Title>
    <Abstract>Generated from swamps</Abstract>
    <Keywords>swamps</Keywords>

  <SRS>EPSG:26910</SRS>
  <LatLongBoundingBox minx="-122.248437678166" miny="47.2480730062538" maxx="-
122.123411805448" maxy="47.37217444602"/>
</FeatureType>
<FeatureType>
  <Name>gd:lakes</Name>
  <Title>lakes_Type</Title>
  <Abstract>Generated from lakes</Abstract>
  <Keywords>lakes</Keywords>

<SRS>EPSG:26910</SRS>
<LatLongBoundingBox minx="-122.248437678166" miny="47.2480730062538" maxx="-
122.123411805448" maxy="47.37217444602"/>
</FeatureType>
<FeatureType>
```

```
  <Name>gd:streams</Name>
  <Title>streams_Type</Title>
  <Abstract>Generated from streams</Abstract>
  <Keywords>streams</Keywords>

<SRS>EPSG:26910</SRS>
<LatLongBoundingBox minx="-122.248437678166" miny="47.2480730062538" maxx="-
122.123411805448" maxy="47.37217444602"/>
</FeatureType>
<FeatureType>
  <Name>gd:rivers</Name>
  <Title>rivers_Type</Title>
  <Abstract>Generated from rivers</Abstract>
  <Keywords>rivers</Keywords>

<SRS>EPSG:26910</SRS>
<LatLongBoundingBox minx="-122.248437678166" miny="47.2480730062538" maxx="-
122.123411805448" maxy="47.37217444602"/>
</FeatureType>
<FeatureType>
  <Name>gd:buildings</Name>
  <Title>buildings_Type</Title>
  <Abstract>Generated from buildings</Abstract>
  <Keywords>buildings</Keywords>

<SRS>EPSG:26910</SRS>
<LatLongBoundingBox minx="-122.248437678166" miny="47.2480730062538" maxx="-
122.123411805448" maxy="47.37217444602"/>
</FeatureType>
</FeatureTypeList>
  <ogc:Filter_Capabilities>
    <ogc:Spatial_Capabilities>
      <ogc:Spatial_Operators>
      <ogc:Disjoint/>
      <ogc:Equals/>
      <ogc:DWithin/>
      <ogc:Beyond/>
      <ogc:Intersect/>
      <ogc:Touches/>
      <ogc:Crosses/>
      <ogc:Within/>
      <ogc:Contains/>
      <ogc:Overlaps/>
      <ogc:BBOX/>
    </ogc:Spatial_Operators>
  </ogc:Spatial_Capabilities>
  <ogc:Scalar_Capabilities>
    <ogc:Logical_Operators/>
      <ogc:Comparison_Operators>
        <ogc:Simple_Comparisons/>
        <ogc:Between/>
        <ogc:Like/>
        <ogc:NullCheck/>
      </ogc:Comparison_Operators>
      <ogc:Arithmetic_Operators>
        <ogc:Simple_Arithmetic/>
      </ogc:Arithmetic_Operators>
    </ogc:Scalar_Capabilities>
  </ogc:Filter_Capabilities>
</WFS_Capabilities>
```

5.1.2 GetCapabilities Requests

The GetCapabilities request generation portion will be tested using live servers or files containing requests, such as the following one sent to GeoServer using HTTP POST.

```
<wfs:GetCapabilities
  service="WFS"
  version="1.0.0"
  xmlns:wfs="http://www.opengis.net/wfs"
/>
```

Or this equivalent request sent using HTTP GET.

```
http://localhost:8080/geoserver/wfs?request=GetCapabilities
```

5.1.3 DescribeFeatureType Document

The DescribeFeatureType parser will be tested using live servers or files containing responses, such as the following one generated by GeoServer.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified" version="1.0">
  <xs:import namespace="http://www.opengis.net/cite/data"

schemaLocation="http://localhost:8080/geoserver/wfs/DescribeFeatureType?typeNa
me=cdf:Locks,cdf:Deletes,cdf:Fifteen,cdf:Other,cdf:Inserts,cdf:Updates,cdf:Nul
ls,cdf:Seven"/>
  <xs:import namespace="http://www.opengis.net/cite/geometry"

schemaLocation="http://localhost:8080/geoserver/wfs/DescribeFeatureType?typeNa
me=cgf:Lines,cgf:MPoints,cgf:MPolygons,cgf:Points,cgf:MLines,cgf:Polygons"/>
  <xs:import namespace="http://www.opengis.net/cite/bluelake"
schemaLocation="http://localhost:8080/geoserver/wfs/DescribeFeatureType?typeNa
me=cblf:Ponds,cblf:BuildingCenters,cblf:Streams,cblf:DividedRoutes,cblf:Bridge
s,cblf:RoadSegments,cblf:MapNeatline,cblf:Lakes,cblf:NamedPlaces,cblf:Building
s,cblf:Forests,cblf:BasicPolygons"/>
</xs:schema>
```

And the associated referenced schemas such as the following one:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="http://www.opengis.net/cite/bluelake"
  xmlns:cblf="http://www.opengis.net/cite/bluelake"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified" version="1.0">

<xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://localhost:8080/geoserver/data/capabilities/gml/2.1.2/fe
ature.xsd"/>

<xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema" name="Ponds_Type">
    <xs:complexContent>
        <xs:extension base="gml:AbstractFeatureType">
            <xs:sequence>
                <xs:element name="the_geom" minOccurs="0" nillable="true"
type="gml:MultiPolygonPropertyType"/>
                <xs:element name="FID" minOccurs="0" nillable="true">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:maxLength value="16"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="NAME" minOccurs="0" nillable="true">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:maxLength value="64"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="TYPE" minOccurs="0" nillable="true">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:maxLength value="64"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
  <xs:element name='Ponds' type='cblf:Ponds_Type'
substitutionGroup='gml:_Feature'/>
</xs:schema>
```

5.1.4 DescribeFeatureType Requests

The DescribeFeatureType request generation portion will be tested using live servers or files containing requests, such as the following one sent to GeoServer using HTTP POST.

```
<wfs:DescribeFeatureType
  service="WFS"
  version="1.0.0"
  xmlns:wfs="http://www.opengis.net/wfs"
/>
```

Or this equivalent request sent using HTTP GET.

```
http://localhost:8080/geoserver/wfs?request=DescribeFeatureType
```

The WFSDataStore internal data representation and data retrieval will be completed by comparing the outputs returned to a known set of inputs from both live and static sources.

From the above examples we would ensure that the list of FeatureTypes was:

- gd:roads
- gd:swamps
- gd:lakes
- gd:streams
- gd:rivers
- gd:buildings

We would also ensure that the Attribute list for the cblf:Ponds FeatureType was:

- the_geom.
- FID
- NAME
- TYPE

5.2 Feature (Read Only)

Reading features from the WFS allows this DataStore to function. In almost every case, users will want to retrieve data to perform their daily tasks. Testing the feature parser and request generator will involve both static and live content. The live content will consist of both HTTP GET and HTTP POST requests.

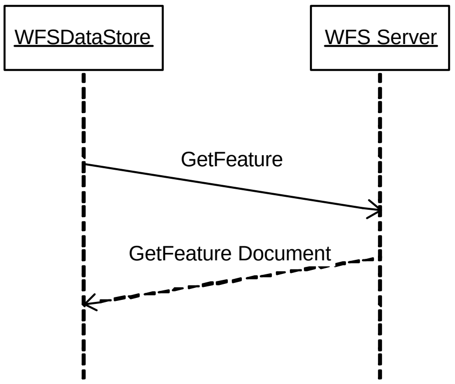

Figure 5 - GetFeature Interactions

5.2.1 GetFeature Requests

One Example HTTP GET request might look like this getFeature request.

```
http://localhost:8080/geoserver/wfs?request=GetFeature&typeName=cblf:Ponds
```

A simple HTTP POST request would look similar to this getFeature request.

```
<wfs:GetFeature
  service="WFS"
  version="1.0.0"
  outputFormat="GML2"
  xmlns:cdf="http://www.opengis.net/cite/data"
  xmlns:wfs="http://www.opengis.net/wfs"
  xmlns:ogc="http://www.opengis.net/ogc"
>
  <wfs:Query typeName="cblf:Ponds">
  </wfs:Query>
</wfs:GetFeature>
```

5.2.2 GetFeature Document

A sample response from GeoServer looks like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection xmlns:wfs="http://www.opengis.net/wfs"
xmlns:gml="http://www.opengis.net/gml"
xmlns:cblf="http://www.opengis.net/cite/bluelake"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/cite/bluelake
http://localhost:8080/geoserver/wfs/DescribeFeatureType?typeName=cblf:Ponds
http://www.opengis.net/wfs
http://localhost:8080/geoserver/data/capabilities/wfs/1.0.0/WFS-basic.xsd">
  <gml:boundedBy>
    <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
      <gml:coordinates decimal="." cs="," ts=" ">-0.002,0.0016 -
0.0014,0.002</gml:coordinates>
    </gml:Box>
  </gml:boundedBy>
  <gml:featureMember>
    <cblf:Ponds fid="Ponds.1">
      <cblf:the_geom>
        <gml:MultiPolygon
srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
          <gml:polygonMember>
            <gml:Polygon>
              <gml:outerBoundaryIs>
                <gml:LinearRing>
                  <gml:coordinates decimal="." cs="," ts=" ">-0.002,0.0018 -
0.0018,0.002 -0.0018,0.0016 -0.002,0.0018</gml:coordinates>
                </gml:LinearRing>
              </gml:outerBoundaryIs>
            </gml:Polygon>
          </gml:polygonMember>
          <gml:polygonMember>
            <gml:Polygon>
              <gml:outerBoundaryIs>
                <gml:LinearRing>
                  <gml:coordinates decimal="." cs="," ts=" ">-0.0016,0.0016 -
0.0016,0.002 -0.0014,0.0018 -0.0016,0.0016</gml:coordinates>
                </gml:LinearRing>
              </gml:outerBoundaryIs>
            </gml:Polygon>
          </gml:polygonMember>
        </gml:MultiPolygon>
      </cblf:the_geom>
      <cblf:FID>120</cblf:FID>
      <cblf:NAME> </cblf:NAME>
      <cblf:TYPE>Stock Pond</cblf:TYPE>
    </cblf:Ponds>
  </gml:featureMember>
</wfs:FeatureCollection>
```

5.3 Transaction (Write)

Transactions are an important part of the uDig project, and as such will emphasize a complete testing plan for this optional portion of the WFS specification. We intend to test WFS transactions using HTTP POST for all three types of transactions:

- Insert
- Update
- Delete

For each type of transaction we will test that the request is well-formed leaving the WFSDataStore, and that the response is correctly parsed and interpreted.

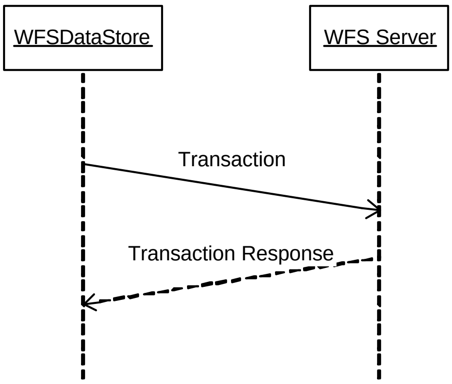

Figure 6 - Transaction Interactions

5.3.1 Insert Request

An example Insert request:

```
<wfs:Transaction
  service="WFS"
  version="1.0.0"
  xmlns:gd="http://www.refractions.net"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:wfs="http://www.opengis.net/wfs"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.refractions.net
http://localhost:8080/geoserver/wfs/DescribeFeatureType?typeName=buildings"
>
  <wfs:Insert>
  <gd:buildings fid="buildings.7">
    <gd:the_geom>
      <gml:MultiPolygon srsName="http://www.opengis.net/gml/srs/epsg.xml#0">
        <gml:polygonMember>
<gml:Polygon>
  <gml:outerBoundaryIs>
    <gml:LinearRing>
      <gml:coordinates decimal="." cs="," ts=" ">
560119.11,5246772.66 560119.16,5246826.31 560161.84,5246826.14
560162.4,5246772.5 560119.11,5246772.66
      </gml:coordinates>
    </gml:LinearRing>
    </gml:outerBoundaryIs>
</gml:Polygon>
        </gml:polygonMember>
      </gml:MultiPolygon>
    </gd:the_geom>
    <gd:POLY>PC01_15</gd:POLY>
    <gd:POLY_OBRP>PC</gd:POLY_OBRP>
    <gd:LABEL>2000400</gd:LABEL>
    <gd:TO_GROUND> </gd:TO_GROUND>
    <gd:STATUS> </gd:STATUS>
    <gd:PRODUCT> </gd:PRODUCT>
    <gd:COVERED> </gd:COVERED>
    <gd:TOWER_TYPE> </gd:TOWER_TYPE>
    <gd:UNINCORPOR> </gd:UNINCORPOR>
    <gd:POPULATION> </gd:POPULATION>
    <gd:CAPITAL> </gd:CAPITAL>
    <gd:CAPITAL0> </gd:CAPITAL0>
    <gd:SEAT> </gd:SEAT>
    <gd:CLASS>-9</gd:CLASS>
    <gd:PHOTOREVIS> </gd:PHOTOREVIS>
    <gd:WIDTH>-999</gd:WIDTH>
    <gd:ANGLE>-99</gd:ANGLE>
    <gd:ESTIMATE> </gd:ESTIMATE>
  </gd:buildings>
  </wfs:Insert>
</wfs:Transaction>
```

5.3.2 Update Request

An example Update request:

```
<wfs:Transaction
  service="WFS"
  version="1.0.0"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:wfs="http://www.opengis.net/wfs"
>
  <wfs:Update typeName="buildings">
    <wfs:Property>
      <wfs:Name>LABEL</wfs:Name>
      <wfs:Value></wfs:Value>
    </wfs:Property>
    <ogc:Filter>
      <ogc:FeatureId fid="buildings.1063"/>
    </ogc:Filter>
  </wfs:Update>
</wfs:Transaction>
```

5.3.3 Delete Request

An example Delete request:

```
<wfs:Transaction
  service="WFS"
  version="1.0.0"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:wfs="http://www.opengis.net/wfs"
>
<wfs:Delete typeName="gd:buildings">
  <ogc:Filter>
    <ogc:FeatureId fid="buildings.1063"/>
  </ogc:Filter>
</wfs:Delete>
</wfs:Transaction>
```

5.3.4 Transaction Reply

An example Transaction reply:

```
<wfs:WFS_TransactionResponse version="1.0.0"
xsi:schemaLocation="http://www.opengis.net/wfs
http://www.refractions.net:8080/geoserver/data/capabilities/wfs/1.0.0/WFS-
transaction.xsd">
  <wfs:InsertResult handle="null buildings 1">
    <ogc:FeatureId fid="buildings.1138"/>
  </wfs:InsertResult>
  <wfs:TransactionResult>
    <wfs:Status>
      <wfs:SUCCESS/>
    </wfs:Status>
  </wfs:TransactionResult>
</wfs:WFS_TransactionResponse>
```

5.3.5 Failed Transaction Reply

An example Failed Transaction reply:

```
<wfs:WFS_TransactionResponse version="1.0.0"
xsi:schemaLocation="http://www.opengis.net/wfs
http://schemas.opengis.net/wfs/1.0.0/WFS-transaction.xsd">
  <wfs:TransactionResult>
    <wfs:Status>
      <wfs:FAILED/>
    </wfs:Status>
    <wfs:Locator>null buildings 1</wfs:Locator>
    <wfs:Message>
buildings inconsistent with update:Could not find property named: LABEL in
schema: buildings
    </wfs:Message>
  </wfs:TransactionResult>
</wfs:WFS_TransactionResponse>
```

5.4 Locking

For distributed systems, transactions need to be ordered to avoid creating data inconsistencies. In some cases this is provided by the native data storage format implicitly or explicitly. In some rare cases locking must be included at the WFS level.

Whether the locking is completed at the WFS level or in the repository, if locking is not completed implicitly, then the WFS developer has three options:

- Provide implicit locking for WFS transactions;
- Allow the user to explicitly lock portions of the data repository;
- Both, but use implicit locking when not explicitly provided by the user.

We intend to test the ability of the WFSDataStore to request a lock, and release the lock using the HTTP POST pr otocol.

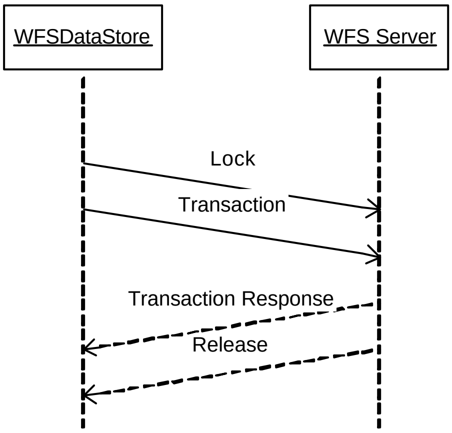

Figure 7 - Locking Interactions

5.4.1 Lock Request

An example Lock request:

```
<wfs:GetFeatureWithLock
  service="WFS"
  version="1.0.0"
  expiry="10"
  xmlns:cdf="http://www.opengis.net/cite/data"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:wfs="http://www.opengis.net/wfs"
>
  <wfs:Query typeName="cdf:Locks">
    <ogc:Filter>
      <ogc:FeatureId fid="Locks.892568684"/>
    </ogc:Filter>
  </wfs:Query>
</wfs:GetFeatureWithLock>
```

5.4.2 Lock Release

An example Release request:

```
<wfs:Transaction
  service="WFS"
  version="1.0.0"
  releaseAction="ALL"
  xmlns:cdf="http://www.opengis.net/cite/data"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:wfs="http://www.opengis.net/wfs"
>
  <wfs:LockId>GeoServer_3399aba632f3f4ad</wfs:LockId>
  <wfs:Update typeName="cdf:Locks">
    <wfs:Property>
      <wfs:Name>cdf:id</wfs:Name>
      <wfs:Value>gfwlra0001</wfs:Value>
    </wfs:Property>
    <ogc:Filter>
      <ogc:FeatureId fid="Locks.892568708"/>
    </ogc:Filter>
  </wfs:Update>
</wfs:Transaction>
```