Quick Guide for CGDI Service Compliance Testing and Performance Optimization

May 12, 2009

Correspondence can be sent to:
GeoConnections
Natural Resources Canada
615 Booth Street
Ottawa, Ontario
K1A 0E9

Telephone: 613-947-8947
Toll-free: 1-877-221-6213
Fax: 613-947-2410

Email: info@geoconnections.org

Website: http://www.geoconnections.org/

Any correspondence regarding this publication should include the document date:
May 2009

This document is available online at the above web address.

GeoConnections is a national partnership program led by Natural Resources Canada to evolve and expand the Canadian Geospatial Data Infrastructure (CGDI). The CGDI provides Canadians with on-demand access to geospatial information (e.g., maps, satellite images) and related services and applications in support of sound decision making.

QUICK GUIDE FOR CGDI SERVICE COMPLIANCE TESTING AND PERFORMANCE OPTIMIZATION

Preface

The past decade has seen the emergence of spatial data infrastructure programs resulting in what some are calling the 'GeoWeb'. While Geographic Information Systems (GIS) are well established technologies dating back to the 1960s, accessing and using geographic information over the Internet is a relatively new development. Recent developments in mass market Geomatics from companies such as Google, Microsoft and Yahoo have raised awareness of the potential application of geospatial information within a wide variety of Web applications. Additionally, advances in Web development technologies (e.g. AJAX - Asynchronous JavaScript And eXtensible Markup Language) are providing a platform for the development of highly integrated, high performance, Web-based applications. Ensuring that these applications can be accessed seamlessly and reliably with an adequate level of performance requires that users and developers validate and optimize the Web Services used in application development.

Intended Audience

The intended audiences for this document are Web Service developers, data analysts and service providers, hereafter, these audiences will usually be referred to as users.

Scope

This document aims to provide application end users and developers with a concise text that identifies and describes resources available for Open Geospatial Consortium (OGC) Web Services (OWS) testing, and presents best practices for optimizing OWS performance. Due to the variety of OWS implementation technologies available, the domain of OWS compliance testing, validation and optimization is broad. This document is intended as a starting point for those interested in the topics addressed. The OWS in scope for discussion are Web Map Service (WMS), Web Feature Service (WFS), z39.50 distributed server used as part of an OGC Catalogue Services for Web implementation (CSW), GeoRSS and KML based services.

1 Introduction and background

The vision for the Canadian Geospatial Data Infrastructure (CGDI) is to enable access to the authoritative and comprehensive sources of Canadian geospatial information to support decision making. The premise that distributed networked access makes data more readily available is the driving force behind the CGDI. Web Services (e.g. OWS) based on open standards provide the basis for interaction across the Internet and allow users to contribute, access, and exchange geospatial data.

Many CGDI applications depend on Web Services, and OWS in particular, for data and processing services. Thus, ensuring a reliable application with an adequate level of performance requires that users and developers validate and optimize the OWS used in application development. This document provides an introduction to two key aspects of developing reliable and useable OWS.

(1) Compliance and interoperability testing: ensures that services are being invoked by and are providing data according to endorsed specifications.

(2) Service performance optimization: improves response time and efficiency of services. This is important for providing a useable application and minimizing load on service providers.

Section 1 provides an overview of OWS and selected client applications that can be used to access these services. Section 2 provides a guide to compliance and interoperability testing. In Section 3, you will learn about service performance optimization.

1.1 Web Services Overview

Currently, OWS provide the basis for interactions across the Internet that allow users to contribute, access and exchange geospatial data. This section provides reference to documents detailing a Web Services architecture approach within the context of the Canadian Geospatial Data Infrastructure (CGDI). For a more detailed treatment of these topics, you are directed to the following documents:
CGDI Vision; CGDI Roadmap; CGDI Architecture; CGDI Online Training; CGDI Developer's Corner
Open Geospatial Consortium Web Site; List of OGC compliant products

1.2 Web Service Clients

Before moving forward with compliance testing or service optimization, you will need to establish a method for accessing OWS. There are many client applications available that are specifically designed to access a particular service or services (e.g. WMS, WFS). Some examples are discussed in Section 1.2.2. However, before looking at service-specific clients, it is important to note that most CGDI compliant Web Services can be accessed using an ordinary, up-to-date, standards-compliant Web browser.

Figure 1.1 Response to Web Map Service request as rendered by a Web browser

1.2.1 Web Browser as Client

The service interface implementation specifications for many OWS support the use of an ordinary Web browser as a client application. Using a URL, a Web browser can instantiate a request to a given OGC Web Service, affect the delivery of the request using the HTTP protocol, and receive a response from the server using the same HTTP protocol. The following are examples of service requests invoked using a URL:

WMS Request:
http://wms.geobase.ca/wms-bin/cubeserv.cgi?request=GetMap
&service=WMS&version=1.1.1& \
width=205&height=205&layers=DNEC_250K:ELEVATION/ELEVATION&styles=Dnec_CITS& \
format=image/png&srs=EPSG:42304&transparent=false&bbox=450000,2100000,550000,2200000

Results of the response are displayed in Figure 1.1. Note that parameters (separated by &) are used to define particular aspects of the returned map, including: *width* and *height* of output image, *layers* to be displayed, output *format*, spatial reference system (*srs)*, image transparency, and the bounding box (*bbox)* in *srs* coordinates.

WFS Request:
http://map.ns.ec.gc.ca/MapServer/mapserv.exe?map=/mapserver/services/envdat/config.map\
&service=WFS&version=1.0.0& request=GetFeature&TypeName=envirodat

By default, a request to a WFS will return a Geography Markup Language (GML) document. This eXtensible Markup Language (XML) document contains geospatial data and metadata rather than the graphic representation of geospatial data returned by a WMS. The WFS request example presented is used to obtain data from water quality stations in Newfoundland and Labrador. A browser rendering of the response result is displayed in Figure 1.2. Note that the GML document combines geospatial data in the form of x,y coordinates with metadata describing key attributes of the data. For example, the tag <gml:Box srsName="EPSG:4326"> uses an 'EPSG' code to establish that the nested spatial coordinates are expressed in decimal degrees using the WGS 84 datum and ellipsoid.

```
- <wfs:FeatureCollection xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.0.0/WFS-basic.xsd http://www.ec.gc.ca/envirodat
  /atlantic http://map.ns.ec.gc.ca/envdat/map.aspx?SERVICE=WFS&VERSION=1.0.0&REQUEST=DescribeFeatureType&TYPENAME=envirodat&
  OUTPUTFORMAT=XMLSCHEMA">
  - <gml:boundedBy>
    - <gml:Box srsName="EPSG:4326">
        <gml:coordinates>-63.234200,46.845300 -52.691400,54.661700</gml:coordinates>
      </gml:Box>
    </gml:boundedBy>
  - <gml:featureMember>
    - <envdat:envirodat>
      - <gml:boundedBy>
        - <gml:Box srsName="EPSG:4326">
            <gml:coordinates>-61.322500,52.228300 -61.322500,52.228300</gml:coordinates>
          </gml:Box>
        </gml:boundedBy>
      - <envdat:msGeometry>
        - <gml:Point srsName="EPSG:4326">
            <gml:coordinates>-61.322500,52.228300</gml:coordinates>
          </gml:Point>
        </envdat:msGeometry>
        <envdat:Station_ID>NF02XA0001</envdat:Station_ID>
        <envdat:lat>52.2283</envdat:lat>
      - <envdat:url_station_data>
          http://map.ns.ec.gc.ca/sp/profile.aspx?NL_ENV_SER_ID=NF02XA0001
        </envdat:url_station_data>
        <envdat:lon>-61.3225</envdat:lon>
```


Figure 1.2 A browser rendering of a WFS response

Although a Web browser is a simple client application, its simplicity makes it a valuable tool when testing and optimizing OWS. A browser provides a relatively transparent, low-overhead method for accessing OWS.

1.2.2 Desktop Applications

While a Web browser will provide access to CGDI compliant OWS, many applications require more extensive client features. For example, visualizing a WFS response is a typical use case. There are many OGC compliant client applications available. The screen capture presented in Figure 1.3 demonstrates how an open source, desktop tool called uDig can be used to access WMS and WFS services.

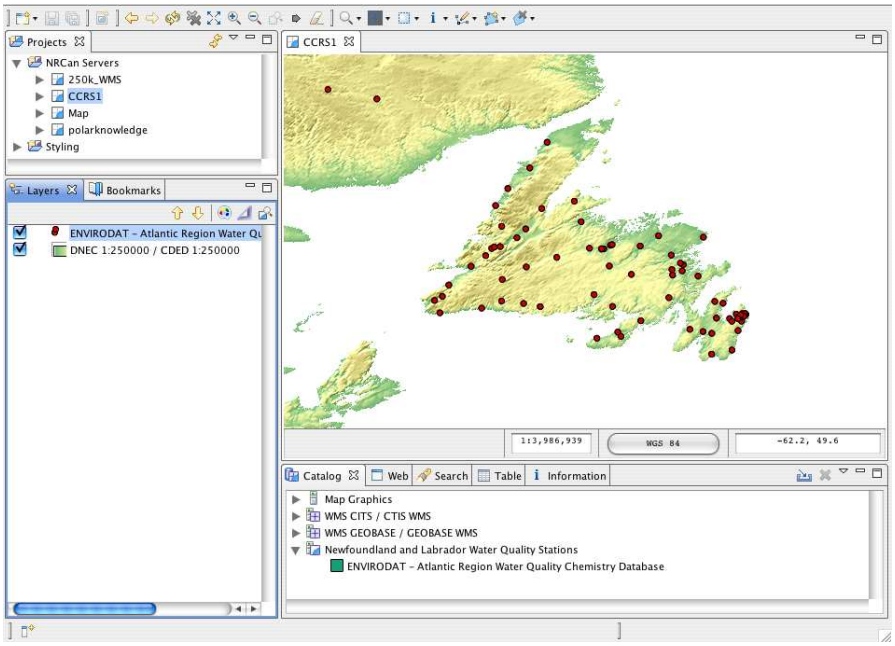

Figure 1.3 The uDig application renders data from WMS and WFS resources

Desktop software applications like uDig can be very useful tools when working with OWS. They allow a user to quickly familiarize themselves with the data offered by service providers, and evaluate service performance in a desktop environment.

1.2.3 Using other Tools to Access OGC Web Services

We have seen how Web browsers and desktop applications can be used to access OWS. These client applications provide powerful tools for accessing OWS data, however, many real world applications require the development of very specific functionality. A variety of tools can be used to support custom application development. Many popular programming and scripting languages such as C, C++, Java, Python, Perl, PHP, JavaScript etc. can be used to access OWS. Various languages provide access to OWS in different ways, but all provide some mechanism for instantiating a request and receiving a response. Using a programming or scripting language provides the most powerful, flexible, and complex method for working with OWS. Increasingly, we are seeing the emergence of application development toolkits that abstract much of the complexity from OWS. A good example of such a toolkit is the OpenLayers JavaScript library.

Although more complex than using a Web browser or desktop application, programming or scripting environments may be the most appropriate tool for more advanced service validation and optimization tasks. These environments provide a developer with the ability to customize requests, precisely manipulate the data provided by a server, trap errors and calculate important metrics such as response time. Specific methods are discussed in more detail in subsequent sections.

2 Compliance and Interoperability Testing

2.1 Compliance in the Context of the Canadian Geospatial Data Infrastructure

The guiding principles of the CGDI (see CGDI Vision) establish a Web Services model that is 'based on open and interoperable standards and specifications' and allows 'users to access data and service seamlessly, despite any complexities of the underlying technology'. To sustain these principles requires that CGDI services maintain a level of compliance to standards and specifications endorsed by the CGDI. A minimum level of compliance is important to ensure everyone can discover, understand and share data without the need to develop service-specific access tools.

While not a CGDI requirement, service compliance and interoperability testing/evaluation is recommended and strongly encouraged to ensure that OWS conform to CGDI endorsed standards and specifications. This section provides an introduction to service compliance and interoperability testing. As a vendor or a member of an OWS development community, you are encouraged to confirm your compliance testing activities through the OGC Compliance Testing Program.

2.2 XML, XML Schema, XSLT

OWS have as their core a fundamental set of functionality based on XML. XML provides well-defined data representation and a well-defined set of validity rules. XML Schema is a specification that allows machines to carry out rules encoded by people. XML Schema is an important element of service validation and testing and it is recommended that you become familiar with XML Schema and related tools. Similarly, the eXtensible Stylesheet Language Transformations (XSLT) language family is used to apply tests that may not be included in the XML validation process.

All levels of compliance testing will benefit from a sound understanding of XML, XML Schema, XLST and other related languages. Such an understanding is necessary if you plan to develop new compliance tests for OWS specifications, or if you are planning to develop or use profiles and/or application schemas. Profiles and application schemas define a subset of an OGC specification (i.e. GML), and model information elements specific to a particular application community respectively.

2.3 Using Compliance Testing and Validation Tools

This section introduces key programs and resources focused on compliance and interoperability testing and evaluation and provides an overview of how to establish a testing environment using reference applications.

2.3.1 The OGC CITE Program and Related Tools

The principal resource currently established for OWS compliance testing is the OGC Compliance & Interoperability Testing & Evaluation Initiative (CITE). The CITE program is an ongoing effort of the OGC that develops compliance tests for OGC standards. The compliance testing tools can be used as part of a vendor's product certification process or to allow service providers and users to test services.

In a typical OWS compliance testing situation, several conditions must be satisfied to consider the service compliant. First, an XML document result produced by an OWS implementation must validate against one or more XML Schema documents using a validating parser (e.g. http://xerces.apache.org/). Second, specific testing assertions established by the test author must be verified. XML Schema is central to satisfying the first requirement. The XML Schema documents used for validation can be those produced by the OGC as part of the standards development process or *Application Schema* documents produced by individuals or organizations for a specific application. XSLT is used in the test scripting language demonstrating the importance of these XML languages to the testing process.

Currently, the CITE portal is able to test WMS, WFS, WCS and CSW for compliance to the latest approved specifications using the Test Evaluation And Measurement (TEAM) Engine. The CITE portal also provides resources for validating GeoRSS, GML, and Web Map Context documents in a non-certification capacity (OGC 2009a). In this document, the term *testing* is used when referring to the use of the TEAM Engine to apply test suites while *validation* is used when referring to the use of a tool that validates an XML document against an XML Schema.

2.3.2 Overview of testing using the TEAM Engine

Two things must be in place to begin testing with the TEAM Engine

- Where applicable, the service provider must load required test data sets. Links to test data sets can be found on the pages describing specific tests (i.e. CSW, WMS, WFS, and various Beta tests)
- A service provider or user must register for an account on the TEAM Engine and then create a new test session by choosing *'Start Testing'* (see Figure 2.1).

It is important to note that many compliance tests require that a test data set be installed to support testing. If you are an application developer or data analyst without administrative privileges for the service, you will need to work with your service provider to carry out compliance testing. Additionally, in some cases, OWS may have authentication (i.e. login) and security (i.e. secure connection) layers enabled by a service provider. Although authentication and security are not part of the OGC specifications, some services implement this functionality 'on top of' OWS. In this case, a given test script would need to be modified to account for additional steps related to the authentication and security layers.

2.3.3 Reference Implementations

The OGC has established a series of *reference implementations* which are defined as " a fully functional implementation of a specification in reference to which other implementations can be evaluated." The OGC provides open source reference implementations to ensure maximum transparency of its specifications for both vendors and customers (OGC 2009b). Reference implementations are useful tools for experimentation if you do not have administrative access to an existing OGC Web Service. These service implementations are deployed using servlet containers such as the popular Apache Tomcat. Additionally, both GeoServer, the WFS reference implementation, and GeoNetwork, the CSW reference implementation can be deployed locally as a standalone application.

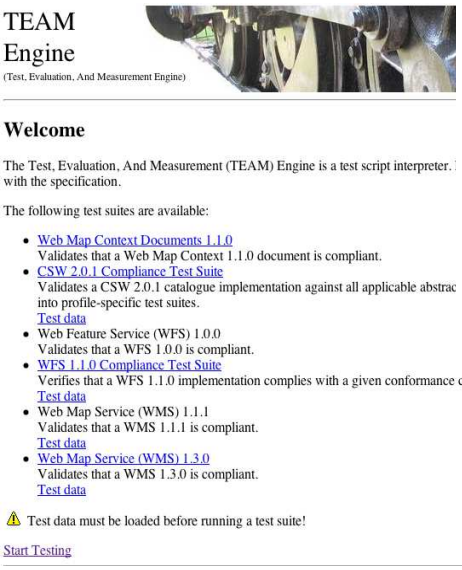

Figure 2.1 The TEAM Engine Welcome page found at http://cite.opengeospatial.org/teamengine/ provides links to service testing documentation and allows a user to "Start Testing"

Using an OGC reference implementation, a compliance and interoperability testing example is presented in section 2.3.5. To facilitate discussion related to these examples, a brief of overview of Compliance Test Language is provided in section 2.3.4.

2.3.4 Overview of the Compliance Test Language.

The TEAM Engine is a software product that executes scripts written in an XML language named *Compliance Test Language* (CTL). Although a detailed discussion of CTL is beyond the scope of this guide, key points are addressed here to support subsequent discussions. The summary presented here is based on the Compliance Test Language (CTL) Discussion Paper. Please refer to the discussion paper for more details.

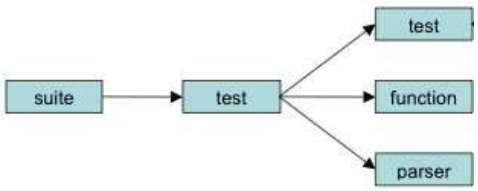

Figure 2.2 Compliance Test Language object model

A CTL document contains a number of elements including *suite, test, function*, *request* and *parser* (see Figure 2.2). The <**suite**> element is a container object. The <**test**> element establishes an individual test that includes an assertion statement and program code to test the assertion. An assertion is a statement derived from an OGC specification being tested which is true for a valid implementation. The program code used to verify the assertion is written in XML and can contain CTL and/or XSL instructions. To streamline program code development, script authors can also use custom <**function**> elements used to verify the assertion and establish program flow.

The <**parser**> element declares a parser for use in a test and links it to its external Java[tm] implementation. Parsers are used in the <r**equest**> instruction to convert the response from an OWS into well-formed XML for processing.

The aforementioned elements are combined with other elements to create a test script. Figure 2.3 depicts a rendered section of the WFS 1.0.0 test script. <**function**> and <**parser**> elements are initialized, followed by the definition of <**suite**> and <**tes**t> elements. Each test is applied, and warnings and errors are reported to the user. The TEAM Engine developed as part of the CITE Initiative provides a relatively user-friendly way to

```
+ <function name="citef:validate-gml-schema"></function>
+ <function name="wfs:sleep"></function>
+ <parser name="parsers:HeadersParser"></parser>
+ <parser name="parsers:XMLValidatingParser"></parser>
+ <parser name="example:ImageStats"></parser>
+ <parser name="example:StringLength"></parser>
+ <parser name="example:TransformXML"></parser>
+ <parser name="wfs:GMLValidatingParser"></parser>
+ <test name="wfs:main"></test>
+ <suite name="wfs:main_wfs"></suite>
+ <test name="wfs:test1.0.0-basic-describefeaturetype-get-1"></test>
+ <test name="wfs:test1.0.0-basic-describefeaturetype-get-2"></test>
```

Figure 2.3 Rendering of top level elements of WFS 1.0.0 CTL script

access and manage compliance testing tools. However, successfully completing compliance testing will be facilitated by familiarizing yourself with the TEAM Engine and its underlying technologies

2.3.5 Testing an OGC Web Service

As stated, the TEAM Engine can test several OWS. This example uses the WFS specification as a testing example. The testing process for other specifications (e.g. WMS, CSW) is very similar to that presented here. The TEAM engine can test the WFS 1.0.0 and WFS 1.1.0 specifications. Each suite tests basic (GetCapabilities, DescribeFeatureType, GetFeature), transactional (Transaction), and locking (LockFeature, GetFeatureWithLock) WFS requests.

With user registration complete and test data installed, you are now ready to start a testing session. From the TEAM Engine home page (http://cite.opengeospatial.org/teamengine ; see Figure 2.1), the user clicks the "Start Testing" link and is then presented with a login screen. If this is the first login, a test suite must be selected (i.e. WFS 1.0.0 or WFS 1.1.1) from the list of options. A description is entered to identify the session. The user is then presented with a form containing options relevant to WFS tests. The options are different depending on which version of the specification is being tested. Common to both tests is the need to enter a 'Capabilities URL' in the form:

http://hostname:port/path?request=GetCapabilities&service=WFS&version=x.x.x

Providing this information allows the test script parser to retrieve the GetCapabilities document from the server being tested. The GetCapabilities document is then used by the test script to apply the appropriate series of tests. For example, if the GetCapabilities document reveals that the server is not transactional, then transactional tests will not be applied. Once desired options are selected, the test is started using the "Start" button. The extensive nature of compliance testing can result in a relatively lengthy testing process with test durations of 5-7 minutes not uncommon for WFS testing. To ensure that the test is continuing as expected, it is useful to monitor the testing process using the method most appropriate to your browser.

Upon completion of the testing process, the user is presented with a test log containing test results. The log contains a copy of the service GetCapabilities document followed by the results of each test and overall result for the entire testing process. Full compliance requires that all tests evaluate to a 'Passed' result. Figure 2.4 provides an illustration of successful test results from tests run against an instance of the GeoServer reference implementation.

```
Test wfs:main (s0068)

Assertion: This WFS is valid

Form w6ac25b3b1b1_1:
   VAR_WFS_CAPABILITIES_URL=http://87.7.38.17:8080/geoserver/ows?service=WFS&request=GetCapabilities&version=1.0.0
   all=all

Request w6ac25b3b3b1_1:
   Method: get
   URL: http://87.7.38.17:8080/geoserver/ows?service=WFS&request=GetCapabilities&version=1.0.0
   Response from parser ::
      <WFS_Capabilities xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xmlns="http://www.opengis.net/wfs"
                  xmlns:ogc="http://www.opengis.net/ogc"

…
 Subtest s0068/w6ac25b3b9d179b1_1 Passed

 Subtest s0068/w6ac25b3b9d181b1_1 Passed

 Subtest s0068/w6ac25b3b9d183b1_1 Passed

...
```

```
Subtest s0068/w6ac25b3b9e1101b1_1 Passed

Result: Passed
```

Figure 2.4 A subset of output from a successful test session

2.3.6 Developing new compliance tests

Compliance tests have not been developed for all OWS specifications. Testing of these services will require the development of a testing suite. The same is true if a service using a profile or Application Schema is to be tested for interoperability with other systems.

The scope of OGC standards and specifications is extensive and, as a result, there are many XML Schema documents used to define OWS XML languages. These XML Schema documents can be accessed at: http://schemas.opengis.net/. Using these Schema documents, CTL can be used to develop new compliance tests. Existing executable test scripts are a useful reference when creating new scripts.

2.3.7 Other Validation Tools

Currently, the TEAM Engine supports the application of an extensive set of tests to selected OGC services. In addition to the TEAM Engine, the CITE initiative provides user-friendly tools to validate various forms of XML-based geospatial data.

GeoRSS is a simple specification for tagging 'really simple syndication' or RSS feeds with location information. GeoRSS is an XML-based format that has both 'simple' and 'GML' encodings. The GML encoding supports a greater range of features than the simple encoding. As both encodings are XML-based, GeoRSS documents can be validated against associated XML Schema documents. The CITE Initiative portal provides a tool for validating Simple or GML GeoRSS. Box 2.1 provides an example of a GML/GeoRSS document which can be saved and validated using the CITE GeoRSS validation tool.
OWS validation tools exist beyond the CITE Initiative.

KML is an XML language used for visualization of geographic information including map labeling and the rendering of associated multimedia content. KML originally became established as the format used by the popular Google Earth visualization tool. As of Version 2.2, KML has been adopted an an OGC implementation standard. As an XML language, KML can be validated against associated XML Schema documents using a variety of validating parsers or an online tool such as the KML Validator by Galdos Systems Inc. The KML Validator validates KML documents against the OGC KML specification schema. Although a KML file may work in a variety of software tools, it may not be fully compliant against the specification. The KML Validator provides an error report and suggestions on how to meet compliance requirements.

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
        xmlns:georss="http://www.georss.org/georss"
        xmlns:gml="http://www.opengis.net/gml">
        <title>Features</title>
        <subtitle>International earthquake observation labs</subtitle>
        <link href="http://example.org/"/>
        <updated>2005-12-13T18:30:02Z</updated>
        <author>
                <name>John Doe</name>
                <email>jjd@jddomain.com</email>
        </author>
        <id>urn:uuid:60a76c80-d888-11d9-b93C-0003939e0af6</id>
        <entry>
                <title>Feature data</title>
                <link href="http://example.org/2008/08/08/atom01"/>
                <id>urn:uuid:1225c695-cfb8-4add-aaaa-80da344efa6a</id>
                <updated>2005-08-17T07:02:32Z</updated>
                <summary>Interesting feature.</summary>
                <georss:where>
                        <gml:Point>
                                <gml:pos>45.256 -71.92</gml:pos>
                        </gml:Point>
                </georss:where>
        </entry>
</feed>
```

Box 2.1 Sample GML/GeoRSS document

2.4 Summary: Validation and Specification Compliance

This section established the importance of compliance and interoperability testing in the context of the CGDI. The importance of XML Schema for testing and validation was stated. The remainder of the section provided guidance on establishing a testing environment and demonstrating and illustrating the testing process for various OGC Web Services.

3. Service Performance Optimization

This section provides an overview of service performance optimization strategies and techniques. OWS are often implemented as a 'stack' of physical and logical technology layers or levels. This section presents topics important for performance optimization at five common levels in the technology stack: database; server; OWS; network; and application.

Optimizing Web application performance in general, and OWS performance in particular, often depends on minimizing the volume of data required to support an application. Thus, many of the suggestions here are focused on reducing data volume. While this section uses particular examples for illustration, a good knowledge of your own application environment will be necessary to apply these strategies.

3.1 Database Level Optimization

Representing the complexities of the geographic world using contemporary computing tools can result in large databases. Large scale databases that include highly detailed accounts of the environment such as urban maps or high resolution satellite imagery may benefit from the techniques outlined here.

3.1.1 Data Generalization: Applying the cartographic generalization practices of selection and simplification can be a useful strategy for optimizing OWS performance at the database level. Selection involves choosing the data most important for an application. Simplification involves modifying the form or shape of features, usually when data is transformed from a large scale to small scale use case. Unlike a paper map, digital spatial data are typically stored without an inherent scale. It is important, though, to recognize that the precision, accuracy and detail of a given data set are most appropriate for a particular scale of analysis and/or display. To effectively apply generalization techniques requires that you become familiar with your dataset and its intended uses so that you can achieve a balance between performance optimization, and maintaining necessary information content (e.g. feature shape, classification).

GIS and spatial databases provide tools to support generalization including the ability to classify using aggregate functions, simplify using built-in algorithms (e.g. Douglas-Peucker), and selecting using queries (e.g. SQL SELECT). A detailed discussion of applying generalization techniques is beyond the scope of this documents, and the reader is directed to GIS texts and the documentation for your chosen software.

3.1.2 Use of a spatial database management system for large and/or complex databases: While spatial data file formats such as those made popular by large GIS vendors provide a fast, convenient, and portable method for storing spatial data, file-based systems can present limitations. File-based systems often have size limitations and support fewer inherent data management and modeling functions. Applications with large data tables (i.e. > 1GB depending on the hardware) or complex data models can benefit from the use of a spatially-enabled relational data management system, hereafter referred to as a spatial database. There are many spatial databases now available, including **Oracle Spatial**, **ArcGIS**, **DB2 Spatial Extender** and the **PostGIS** extension to the well established open source **PostgreSQL**. Through physical and logical database management strategies, spatial databases can effectively handle very large databases. Moreover, the use of standard query languages like SQL allows users to embed foundational data models and logic at the database (rather than application) level. If you are working with a large or complex database, consider using a spatial database to improve performance and efficiency in application maintenance.

3.1.3 Data sorting and indexing: Whether using a file-based system or a spatial database, data sorting and indexing can dramatically improve data query response time. Data sorting can be effective for improving access and display times if an application requires that features be consistently accessed in a particular order. For example, the open source MapServer application provides a tool called *sortshp* for sorting an ESRI Shapefile using a particular attribute.

Although sorting can be useful, many applications require more flexibility with respect to how data are queried and accessed. In this case, a variety of different indexing strategies can be applied. Standard database indexing methods can be used to improve performance when using non-spatial attribute values to define queries. To improve the speed of searching a database using spatial queries requires the use of a spatial index. There are many spatial indexing methods with the R-tree method being favoured by many spatial database applications, including the PostGIS extension. Additionally, file-based spatial data can be indexed, with an example being the use of the *shptree* program to index an ESRI Shapefile.

3.1.4 Raster data optimization: Raster or ‘grid’ methods for data representation can provide many advantages, including: effective representation of continuous surfaces, fast map algebra processing, fast display of surface data, and effective representation of image data. However, raster data representations can result in very large data sets in comparison to relatively homogeneous vector datasets representing similar phenomena. However, some phenomena are not effectively represented using the vector model and so raster data are typically part of contemporary OWS implementations.

If you are using large raster data sets (i.e. high resolution satellite imagery, or Digital Elevation Models), there are a number of strategies that can be used to optimize performance. The two primary techniques are: a) the use of image pyramids, and b) image compression including wavelet compression. Open source tools such as those included in the GDAL distribution can be used to generate image pyramids and translate uncompressed imagery to a compressed format. Wavelet compression is typically achieved using proprietary software packages.

Most software application communities will provide specific instruction on how to optimize raster data and the database level (e.g. MapServer, GeoServer).

3.2 Server Level Optimization

Most OWS technology 'stacks' involve at least two servers, an OWS specific server (e.g. WMS, WFS) and a Web server through which http requests and responses are managed. There may also be a database server used to support the application. This section addresses Web server and database server optimization, while the subsequent section discusses OWS server optimization.

As stated, a Web server will typically be used to manage http requests and responses in relation to the OWS service hosted by you or used in your application. If you are a service user, we recommend that you establish a dialogue around Web server performance issues with your service provider. If you are hosting the service, then researching best practices for optimizing the performance of your Web server is recommended. There are many books and Web-based resources available to provide Web server optimization advice. For example, the widely used open source Apache Web server provides a performance tuning document as part of its standard documentation package.

Database servers can be tuned in many ways to optimize performance. The use of indexing at the database level has already been discussed. There are also database server-level optimizations that can be performed. These range from optimizing the configuration of storage devices (e.g. partitioning of hard disks), to effectively establishing the 'page size' used for storing data (e.g. PostgreSQL method), to efficient memory management, and to? regular database maintenance tasks. Each major spatial database will have its own specific approaches that will be documented by your vendor and the user and publishing community. For example, the PostgreSQL project provides server configuration document that discusses many server optimization issues.

It is important to note that server level optimization is dependent upon optimization of software and hardware. Spending time and effort optimizing through software configuration may be wasted if the server is running on sub-standard hardware (e.g. insufficient RAM, slow CPU or hard drive). If you are using a service in your application, it may be useful to discuss these issues with your service provider. Similarly, if you are hosting a service, either directly or through a hosting service, understanding the server level capacity of your system may assist in evaluating overall performance.

3.3 OWS Level Optimization

As with Web servers and database servers, OWS server applications can be optimized in a variety of ways. Optimization techniques will be specific to particular software implementations and the OWS being served, however, there are several general approaches that will be useful in many instances.

3.3.1 Using Profiles: In the context of OWS, a profile is a restricted subset of an XML language (e.g. GML) that limits the number of object types that can appear in a compliant schema. This results in XML document instances that are easier to process. A well known profile within the OGC community is the Geography Markup Language Simple Feature Profile (OGC 2006). Using such a profile can improve the performance of a service (by reducing complexity), and potentially the volume of response data due to the use of simpler data structures.

3.3.2 Varying Data Encoding: By default, OWS services such as WFS and CSW return XML in text form. While this is elegant in it simplicity, the resulting data streams can be very large due to the relatively verbose nature of XML. Rather than lose the expressive power of XML, compression can be used to reduce the size of XML data as it is being transmitted. This method is well documented for WFS in a 2005 study carried out by OGC members (see OGC 2005). If you are serving large volumes of XML data, and your server software supports alternate data encoding, this may improve performance.

The same logic can be applied to the use of WMS. A WMS can return a wide range of image formats including uncompressed formats such as TIFF. Unless there is a compelling reason to use uncompressed formats, it is suggested that a compressed format such as JPEG or PNG be served and/or requested to reduce bandwidth consumption and response time respectively.

3.3.3 WMS Caching and Tiling: In the case where data are relatively static, server-side caching of WMS tiles is a very effective way to improve service performance. Upon receiving a request, the server responds with a pre-rendered tile result, thus significantly reducing the processing time required to serve the request. A good example of a WMS caching and tiling application is GeoWebCache. Additionally, the OGC recently published an informative paper on the topic entitled *OpenGIS Tiled WMS Discussion Paper* (see OGC 2009c).

There are many other possible OWS server level optimization approaches. For example, Anderson & Deoliveira (2009) present very useful recommendations for optimizing MapServer and GeoServer. The GeoServer project site provides a comprehensive discussion of optimization techniques, as does the MapServer site. Other resources, such as the OGC report entitled *OGC Canadian Geospatial Data Infrastructure WFS and GML Best Practices* (see OGC 2008), present issues and recommendations for addressing challenges related to OWS implementation and performance. Reviewing the resources referenced here, as well as software documentation and other relevant resources, will provide you with a solid foundation for establishing an OWS service level optimization strategy.

3.4 Network Level

Latency can be defined as the duration of time between the instant a process is initiated, and the instant that one of its effects is perceived. Latency is an important concept in a Web services environment where the performance of an application is dependent upon requests and resulting responses transmitted across a communications network (i.e. the Internet). Optimizing an OWS at the database, server and OWS server levels without considering issues such as the average network bandwidth available, may still result in a service (and consequently dependent applications) exhibiting relatively poor performance. Necessary bandwidth will, of course, be specific to the application context, however, spatial data applications are typically bandwidth intensive. WMSs produce relatively large image data while WFS, CSW, GeoRSS, KML etc., produce verbose XML. In addition to optimizing at other levels of the stack, you will need to ensure that the bandwidth available at the network level is sufficient for the needs of the application. This is relevant for both service providers, application developers, and application end users.

3.5 Client-side Optimization

In *High Performance Web Sites* (Souders 2009), Steve Souders states that “only 10-20% of the end user response time is spent downloading the HTML document. The other 80-90% is spent downloading all of the components in the page.” (pg. 5). This is an important consideration for spatial Web applications that are typically component rich (e.g. map images) and are increasingly being developed using client-side development approaches (JavaScript, AJAX etc.). Thus, developers and users should seriously consider the impact of client-side design and optimization. In addition to reading the referenced book, you can find many of the Souders’s recommendations online. Note that recommendations such as ‘gzip components’ are already familiar to us. Considering the client-side component of overall performance is increasingly important as Web development evolves.

Summary: Tuning Service Performance

Improving service performance can be the result of one or two changes with major impact, and/or, the combination of many changes with small, cumulative, impacts. Some strategies are applicable across software systems, while others will be specific to a particular service software or hardware environment. Web Service users are encouraged to establish as much detail as possible about the services being used in a given application. Using this information as the basis for software-specific performance research can support the development of constructive feedback to a service provider.

References

OGC (2005*) OWS 3 GML Investigations - Performance Experiment by Galdos* Systems. Open Geospatial Consortium document 05-101, available for download at: http://www.opengeospatial.org/standards/gml. Last accessed on March 10th, 2009.

OGC (2006) *GML 3.1.1 simple features profile*. Open Geospatial Consortium document 06-049r1, available for download at: http://www.opengeospatial.org/standards/profile. Last accessed on March 10th, 2009.

Anderson & Deoliveira (2009) *WMS Performance: Mapserver vs. Geoserver*. Presentation made at the Free and Open Source Software for Geospatial 2009 Conference. Victoria, British Columbia, September 24-27, 2009. Available online at: http://www.foss4g2009.org/presentations/view.php?abstract_id=120. Last accessed on March 17th, 2009.

OGC (2009a) *OGC Compliance Testing Portal.* Online resource available at: http://cite.opengeospatial.org/instructions. Last accessed on February 12, 2009

OGC (2009b) *Reference Implementations*. Online resource available at: http://cite.opengeospatial.org/reference. Last accessed on February 12th, 2009

OGC (2009c) *OpenGIS Tiled WMS Discussion Paper*. Open Geospatial Consortium document 07-057r2, available for download at: http://www.opengeospatial.org/standards/dpd. Last accessed on March 10th, 2009.

Souders, S. (2009) *High Performance Web Sites: Essential Knowledge for Front-End Engineers*. Sebastapol, CA: O'Reilly Media Inc. 168pp.

OGC (2008) *OGC® Canadian Geospatial Data Infrastructure WFS and GML Best Practices*. Open Geospatial Consortium document document 08-002, available for download at: http://www.opengeospatial.org/standards/dpd. Last accessed on March 10th, 2009.