The Naval Research Laboratory's Ongoing Implementation of the Open Geospatial Consortium's Catalogue Services Specification

Frank P. McCreedy, David B. Marks
Room D-9E, D-9F
1005 Balch Blvd
Stennis Space Center, MS 39529 USA

***Abstract*- The Naval Research Laboratory's (NRL) Digital Mapping, Charting and Geodesy Analysis Program (DMAP) team, located at Stennis Space Center, conducts research involving the online storage and manipulation of geospatial data. One aspect of this research concerns metadata. Metadata is "data about data" or simply put, various bits of information that provide an understanding of certain aspects of the data such as source, resolution, date created, geographic location and many others. Standardized metadata formats allow data to be easily categorized and thereby easily searched. Most geospatial data producers have realized or are starting to realize that creating standardized metadata along with their data is crucial for allowing their data to be utilized by standards-conforming Geographic Information Systems (GIS). But having the metadata in standard formats is only part of the solution. How can this metadata be stored, updated and queried in a standard way?**

To this end, the Open Geospatial Consortium (OGC) has created an open specification for cataloging metadata. This specification is named "Catalogue Services" and defines a base metadata and interface model to allow querying and updating a metadata catalog. The base model is an abstraction that is actualized by a "protocol binding". A protocol binding defines the mapping between the interfaces of that binding and those of the base specification. The Catalogue Services specification defines CORBA, Z39.50 and HTTP protocol bindings. The HTTP protocol binding is also known as "Catalogue Services for the Web" (CSW). The Catalogue Services specification also defines the concept of "application profiles" which allows the base model to be extended to support a particular user community's needs. The base catalog specification, protocol bindings, and application profiles together provide a very flexible and powerful metadata cataloging architecture.

The DMAP team has decided to investigate implementing the CSW component of Catalogue Services to complement its geospatial work. CSW was chosen due to the popularity of web services and the DMAP team's familiarity with them. CSW defines seven operations which map to operations in the base model. Four of these operations are required to be implemented and three are optional. These operations are required to be implemented using either HTTP POST or HTTP GET requests. Request parameters are required to be sent using either Key-Value-Pair (KVP) or Extensible Markup Language (XML) documents. Responses to these requests are formatted as XML documents. SOAP (a way to standardize web service communication) can optionally be used if the implementation supports it.

Part of the task of creating a CSW implementation involves the storage of metadata records. In this case, the metadata records are in the form of Extensible Markup Language documents that conform to certain geospatial metadata schemas (declared by the Catalogue Services base model or by an application profile). Currently, there are two popular types of databases for XML document storage. These are "XML-Enabled" and "XML-Native". XML-Enabled is an extension to tradition relational database software that provides some XML-specific functions. In this scheme, the data from an XML document is extracted and stored in relational tables. XML-Native is a newer database concept designed from the beginning to store XML files and uses the XML document as the base unit of storage. There are several commercial and open source XML database applications to choose from. However, some of these do not provide needed functionality and others have little or no current development activity. Finding one that solves your particular needs can be a challenge. DMAP has chosen the open source XML-Native database project eXist for initial CSW prototyping due to high developer activity level, open source code, ease of use and no cost.

This paper will further discuss the Catalog Services specification, XML databases, XML query languages, geospatial metadata schemas, implementation challenges and DMAP's current state of CSW development.

I. INTRODUCTION

The Naval Research Laboratory's (NRL) Digital Mapping, Charting and Geodesy Analysis Program (DMAP) team, located at Stennis Space Center, has been developing software for years that is targeted at storing, searching, and displaying digital geographic data. One of the aspects of this work involves metadata. Metadata is "data about data." For instance, a satellite image might have another file associated with it containing information about the image such as the date it was collected, what piece of equipment collected it, classification level, etc. Without this information, the data loses a large amount of its usefulness.

0-933957-38-1 ©2009 MTS

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **JUN 2010**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**

4. TITLE AND SUBTITLE **The Naval Research Laboratorys Ongoing Implementation of the Open Geospatial Consortiums Catalogue Services Specification**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Research Laboratory Code 7180 Stennis Space Center, MS 39529 USA**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM202806. Proceedings of the Oceans 2009 MTS/IEEE Conference held in Biloxi, Mississippi on 26-29 October 2009. U.S. Government or Federal Purpose Rights License, The original document contains color images.

14. ABSTRACT
The Naval Research Laboratorys (NRL) Digital Mapping, Charting and Geodesy Analysis Program (DMAP) team, located at Stennis Space Center, conducts research involving the online storage and manipulation of geospatial data. One aspect of this research concerns metadata. Metadata is data about data or simply put, various bits of information that provide an understanding of certain aspects of the data such as source, resolution, date created, geographic location and many others. Standardized metadata formats allow data to be easily categorized and thereby easily searched. Most geospatial data producers have realized or are starting to realize that creating standardized metadata along with their data is crucial for allowing their data to be utilized by standardsconforming Geographic Information Systems (GIS). But having the metadata in standard formats is only part of the solution. How can this metadata be stored, updated and queried in a standard way? To this end, the Open Geospatial Consortium (OGC) has created an open specification for cataloging metadata. This specification is named Catalogue Services and defines a base metadata and interface model to allow querying and updating a metadata catalog. The base model is an abstraction that is actualized by a protocol binding. A protocol binding defines the mapping between the interfaces of that binding and those of the base specification. The Catalogue Services specification defines CORBA, Z39.50 and HTTP protocol bindings. The HTTP protocol binding is also known as Catalogue Services for the Web (CSW). The Catalogue Services specification also defines the concept of application profiles which allows the base model to be extended to support a particular user communitys needs. The base catalog specification, protocol bindings, and application profiles together provide a very flexible and powerful metadata cataloging architecture.

15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**SAR**	**7**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

For decades, many geographic data producers have created large amounts of data which were stored in many different digital formats. These formats would generally have their own way of storing metadata (if they stored it at all). This wasn't as much of a problem in the past before the ubiquity of the internet. Data sharing wasn't a very easy thing to do. Systems would often be specifically created by the producer to just handle manipulation of their own data. In this case, it wasn't that big of an issue to support just that particular data's metadata. Modern Geographic Information Systems (GIS) however are made to support as many types of geographic data as they possibly can in order to appeal to as many customers as possible. Data sharing is now common. Finding data that you are interested in though can be challenging. If metadata can be standardized then large online metadata databases can be established allowing searching to become easier.

The Open Geospatial Consortium (OGC) creates open standards pertaining to geospatial data. Many are probably familiar with its popular Web Map Service (WMS) specification which allows easy online publication and viewing of geospatial images using simple URL-based parameters (these are parameters that are appended to URLs such as ?imageType=png). OGC has also created a standard [1] for searching, retrieving and updating (or "cataloging") metadata known as "Catalogue Services."

II. Catalogue Services

Catalogue Services is a specification designed to allow metadata to be searched, retrieved, and updated. The base specification is abstract and defines a general operational model that is then implemented by "protocol bindings." Protocol bindings exist for CORBA (Common Object Request Broker Architecture), Z39.50 (commonly used by libraries to search for and retrieve information) and Hyper Text Transfer Protocol (HTTP – commonly used to serve web pages). The HTTP protocol binding has been named "Catalogue Services for the Web" (CSW). CSW has been selected as the main focus of DMAP's work towards catalog implementation. Web services are fairly easy to implement and DMAP has considerable experience constructing them. Table I gives a brief description of the catalog interface methods defined by CSW.

Table I
CSW Method Descriptions

Method Name	Brief Description	Required
GetCapabilities	Defines abilities and limits of a particular CSW implementation	Yes
DescribeRecord	Provides definition of metadata record types that the CSW contains	Yes
GetRecords	Query for records based on particular metadata field(s)	Yes
GetRecordsByID	Retrieve a specific metadata record	Yes
GetDomain	List possible values for metadata fields that are supported	No
Harvest	Set up a periodic pull of metadata records into a catalog	No
Transaction	Add, delete and update a catalog's metadata records	No

The catalog specification has a concept of "core queryables" and "core returnables." These are basically metadata fields that must be supported by any catalog implementation. This allows a catalog implementation to have a minimum level of generic functionality regardless of what kind of metadata records it actually holds. Table II lists the core queryables and returnables defined in the specification. The returnables are mapped to an XML document schema that will be referred to as CSWRecord throughout the rest of this document. Results of metadata searches must minimally support returning the catalog metadata records in the form of CSWRecord with three levels of verbosity: "brief", "summary", and "full." "Brief" and "summary" leave out some of the metadata fields that "full" records would have.

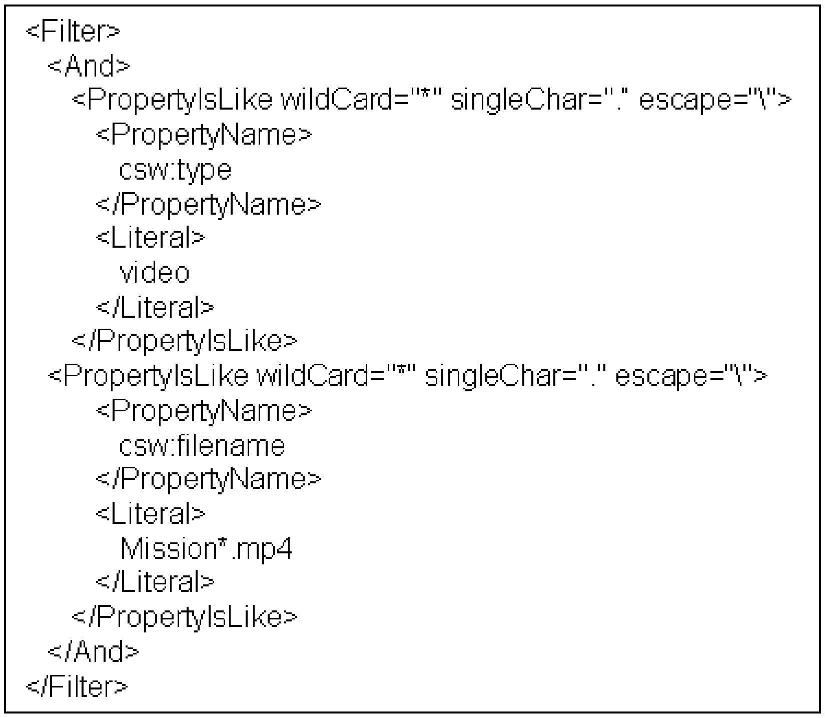
Table II
Core Metadata Properties

Queryables	Returnables
Subject	Title
Title	Creator
Abstract	Subject
AnyText	Description
Format	Publisher
Identifier	Contributor
Modified	Date
Type	Type
BoundingBox	Format
CRS	Identifier
Association	Source
	Language
	Relation
	Coverage
	Rights

The base specification is extended into different areas of interest by a concept known as “Application Profiles.” Application profiles define their own metadata record schema, provide a mapping between this schema and the base specification’s schema, and select a particular protocol binding. An application profile, “ISO Metadata Application Profile” [2], has been created by OGC to support ISO 19115/19119 metadata [3]. The metadata record defined by this profile greatly extends the amount of metadata provided by the base specification and is focused on describing geographic data.

CSW relies upon HTTP for interactions with the metadata catalog. HTTP defines many different types of requests for interacting with a web server. The most commonly used are GET and POST. GET requests a document and can optionally send parameters as part of the URL. POST is most often used in the case of passing very large parameters or a large number of parameters such that it isn’t convenient to place them into the URL. Request parameters are passed to the CSW using one of three approaches. “Key-Value Pair” (KVP) is used to describe sending request parameters as part of the URL. KVP can be sent using HTTP GET and encoding the parameters are part of the URL. KVP can also be embedded as headers in an HTTP POST request. KVP is not really appropriate for complex requests such as a complex search filter which would be implemented as an XML document. To support these more complex requests, CSW also supports sending a request in the form of an XML document using HTTP POST. Responses from the CSW server come in the form of an XML document.

Queries for metadata documents can be very simple, such as querying by document ID. However more advanced types of queries need a defining structure. OGC_Filter [4] as well as OGC_Common Query Language are used to provide this ability. OGC_Common Query Language is defined in the Catalogue Services specification. OGC_Filter is an XML schema defined by OGC that allows complex queries using such things as logical operators and nesting. Initial catalog development has focused on OGC_Filter implementation. Fig. 1 shows an example OGC_Filter document. This example will select any documents that have a “type” field set to “video” and have a “filename” field that starts with “Mission” and ends with “.mp4”. The first element after the “Filter” element is an “And” element, which causes the logical operator “And” to be applied to the two “PropertyIsLike” sub-elements. The “PropertyIsLike” element denotes that a metadata field is being matched as text. “wildCard” defines a “wild card” character that will match to zero or more of any character whereas the “singleChar” matches to any single character.



```
<Filter>
  <And>
    <PropertyIsLike wildCard="*" singleChar="." escape="\">
      <PropertyName>
        csw:type
      </PropertyName>
      <Literal>
        video
      </Literal>
    </PropertyIsLike>
  <PropertyIsLike wildCard="*" singleChar="." escape="\">
      <PropertyName>
        csw:filename
      </PropertyName>
      <Literal>
        Mission*.mp4
      </Literal>
    </PropertyIsLike>
  </And>
</Filter>
```

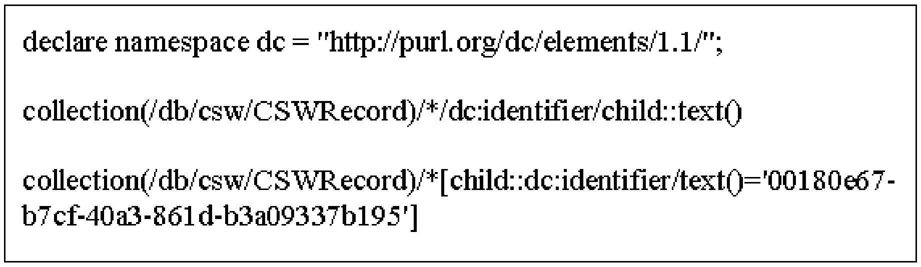
Fig. 1 Example OGC_Filter document selecting all records that describe video files that have names starting with "Mission."

III. XML Databases

The catalog contains metadata records that are in the form of XML documents. Traditional database systems did not offer any targeted support for storing XML documents. However, given the popularity of XML it became obvious that database systems would have to support storing these kind of documents. Two forms of database systems have evolved to support storing XML documents – "XML-Enabled" and "XML-Native." XML-Enabled databases are a layer on top of traditional relational database software. In this scenario, an XML document is sliced up into its component data segments and these segments are stored in traditional relational tables. "XML-Native" stores the XML document in its native form. The base unit of storage in an XML-Native database is an XML document. XML-native has the advantage that the document can be retrieved in a form identical to that in which is was stored. The XML-Enabled database must recreate the document from tables. This can have the effect of losing data ordering and also losing comments and XML processing instructions. XML-Enabled has the advantage of leveraging an existing operational relational database system in cases where they already exist. DMAP doesn't maintain a commercial database instance and did not want to incur the expense and labor of installing one, so it was decided to go the route of using an open-source XML-Native database software named "eXist."

eXist organizes its stored documents in a tree structure. The root of this tree is named "db". Documents are referenced by their path in the tree. So, for instance, to reference the document "users.xml" that is stored in the root of the tree, the path would be "/db/users.xml." The database to hold catalog metadata records needed to hold three types of documents: base specification records (CSWRecord), ISO Metadata Application Profile records (MD_Metadata), and ISO Metadata Application Profile records that have been translated into base specification records. The translated MD_Metadata records are kept in a collection instead of being created on the fly because the CSWRecord is small enough that the storage cost is very small. To support storing these three different types of records in the database, three tree paths were created: "/db/csw/CSWRecords", "/db/csw/MD_MetadataRecords", and "/db/csw/MD_MetadataAsCSWRecords." A parent named "csw" was included to keep the database organized in case some other project also needed to use the database.

The traditional way to query a database (SQL) is not really a good match for XML documents. SQL statements target columns found in tables defined in a relational database schema. XML documents don't have as rigid of a structure. So, a new query language has been created named "XPath." XPath is an easy-to-use query language that operates on the paths founds in a XML document. XPath was originally geared to just finding data in a single XML document but has been extended to operate on collections of documents. XQuery adds to XPath and gives it more power via typical programming constructs. Example XPath statements are shown in Fig. 2. The first statement defines the context for the “identifier” field. The second statement selects the identifier field from all of the documents that are in the CSWRecord collection. The third statement selects any document that has a identifier field of “00180e67-b7cf-40a3-861d-b3a09337b195”. Queries can operate against a single document, a collection of documents, or the entire document database.



```
declare namespace dc = "http://purl.org/dc/elements/1.1/";

collection(/db/csw/CSWRecord)/*/dc:identifier/child::text()

collection(/db/csw/CSWRecord)/*[child::dc:identifier/text()='00180e67-
b7cf-40a3-861d-b3a09337b195']
```

Fig. 2 Example XPath statements.

IV. Software Architecture

DMAP’s current software architecture for its catalog server implementation is show in Fig. 3. Catalogue Services clients send their requests via HTTP to the Apache web server. Apache recognizes requests for Catalogue Services by examining the URL that the client requests. These requests are then forwarded to Tomcat for processing. Tomcat receives these requests and sends them to the NRL-created CSW servlet (a servlet is a Java-based server-side program) running inside it. The CSW servlet identifies which HTTP method is being used and whether KVP or XML is being used to send the request data. Once the CSW servlet identifies the parameter passing scheme it can extract the parameters from the request. It then identifies and dispatches the particular CSW method that the client wants to invoke, passing the extracted parameters to the selected method. This typically results in the CSW servlet storing, updating, or retrieving metadata records from the eXist database. The result of the client request is an XML document that is sent back to the client through Apache as a standard HTTP response. Optionally, “SOAP” can be used when interfacing with the CSW. SOAP is a standard for interfacing with web services and wraps an XML request and response in a SOAP “envelope.” This envelope is very lightweight and makes it easy for SOAP clients and servers to recognize SOAP messages and also provides an exception reporting capability in case something goes wrong during request processing.

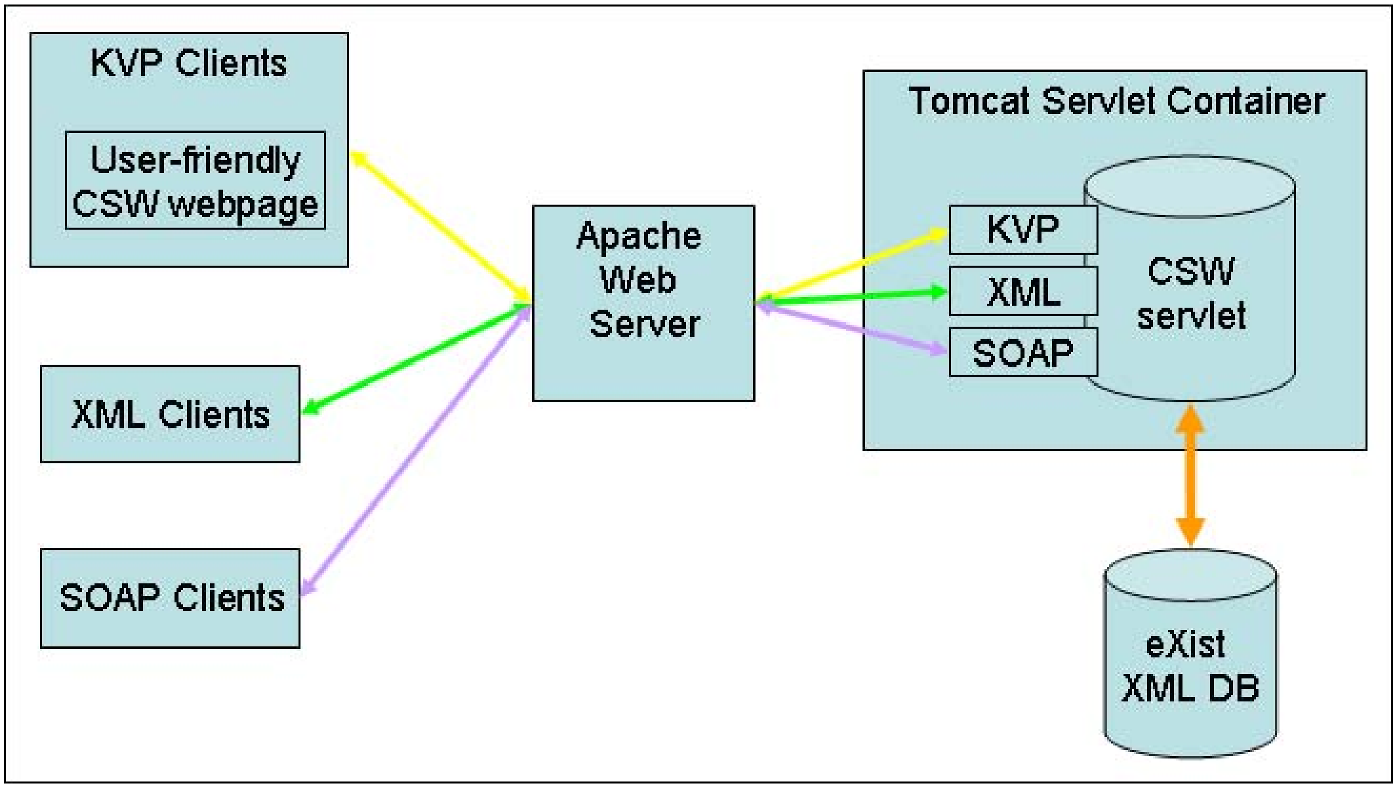


Fig. 3 Catalog Services server architecture diagram.

V. Challenges

When working with XML documents inside of Java programs, it is useful to have some way to convert these documents into Java objects in order to manipulate them more easily. This can be done by using the Java Architecture for XML Binding (JAXB) schema-compiler. The schema-compiler processes the XML document schemas and creates matching Java class source files. These files are organized into the standard paradigm of Java “packages.” By default, the namespace name defined in the XML document schema is used to construct the package name.

The CSW schema references many other schemas, which in turn reference other schemas. All of these schemas have to be processed by the schema-compiler. Unfortunately, compiling schemas is a somewhat new concept and compilers are often not very tolerant of deviations from the norm. One of the issues that became apparent was that some of the schemas would define an element with a certain name. Later on in the schema, that same name would be used to name a different element. To differentiate the two elements, the schema author would prefix the second instance of the element name with an underscore character. The JAXB schema-compiler does not recognize underscores as being significant and would complain that the name is being used twice, by different elements. Another problem that occurred was that the referenced schemas would sometimes have several versions. Some schema files would reference one version while other schema files reference another version. The problem here is that since they used the same XML namespace name in both versions, the schema compiler would have to put both of these compiled versions into the same Java package. This isn’t possible, so the schema compiler complains and gives up (a simple fix for this problem is to include a version number in the namespace name when creating schemas). Another problem is that the schema author considered differing capitalization to be sufficient to make element names unique. The schema compiler does not and will complain about duplicate names (it is not case-sensitive).

The solution to these and other problems can usually be found by using what are known as “binding customizations.” This is simply a file with some extra instructions to the schema compiler as to how it is to function. Fig. 4 shows a sample of the binding customization file that was used. The first example solves the underscore problem and the second example explicitly defines the Java package that a particular schema should be compiled into. The third example highlights how it is possible to explicitly define the target class name for a particular element- allowing elements with duplicate names to be compiled into separate class files.

```
<jaxb:bindings>
   <jaxb:globalBindings underscoreBinding="asCharInWord" />
</jaxb:bindings>


<jaxb:bindings xmlns:gml="http://www.opengis.net/gml"
         schemaLocation="schemas/gml/3.1.1/base/geometryAggregates.xsd" node="/xsd:schema">
   <jaxb:schemaBindings>
      <jaxb:package name="net.opengis.gml.v_3_1_1"></jaxb:package>
   </jaxb:schemaBindings>
</jaxb:bindings>


<jaxb:bindings schemaLocation="schemas/iso/19139/20060504/gml/datums.xsd"
         node="/xsd:schema/xsd:element[@name='ellipsoid']">
   <jaxb:class name="ellipsoidCaseCollisionFix" />
</jaxb:bindings>
```

Fig. 4 Binding customizations.

VI. Status

At this point of development, several of the CSW methods have been implemented. These include Harvest, GetRecordByID, and GetCapabilities. The Transaction method has been partially completed – metadata record insertion is functional but record updating and deletion require OCG_Filter and OGC_Common Query Language support to be completed. Mapping MD_Metadata records to CSWRecords has been completed. The database instance is set up and working as is the web server and its connection to the servlet container. A rough but functional web page has been constructed to view and add metadata records to the catalog.

References

[1] Open Geospatial Consortium Inc., “OpenGIS® Catalogue Services Specification,” Version 2.0.2, Corrigendum 2, February 2007.
[2] Open Geospatial Consortium Inc., “OpenGIS® Catalogue Services Specification 2.0.2 – ISO Metadata Application Profile,” Version 1.0, July 2007.
[3] International Organization for Standardization, “ISO 19115 Geographic Information - Metadata,” First Edition, May 2003.
[4] Open Geospatial Consortium Inc., “OpenGIS® Filter Encoding Implementation Specification,” Version 1.1.0, May 2005.