Copyright

by

James Adam Seppi

2010

The Thesis Committee for James Adam Seppi
Certifies that this is the approved version of the following thesis:

A Services Stack Architectural Model for the CUAHSI-HIS

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

David R. Maidment

Ben R. Hodges

A Services Stack Architectural Model for the CUAHSI-HIS

by

James Adam Seppi, B.S., B.A.

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

December 2010

Acknowledgements

I would like to thank my advisor, David R. Maidment, for his guidance and vision, as well as for the opportunity to work with the great team at the Center for Research in Water Resources. Thanks also to Timothy L. Whiteaker for being a knowledgeable resource for the discussion and critique of ideas.

I would also like to acknowledge the Consortium of Universities for the Advancement of Hydrologic Science, Inc. for its support of my work through the Hydrologic Information System project.

Finally, I would like to thank my wife, Taylor Cook, for her encouragement, love, and understanding throughout my graduate studies.

December 2010

Abstract

A Services Stack Architectural Model for the CUAHSI-HIS

James Adam Seppi, M.S.E.

The University of Texas at Austin, 2010

Supervisor: David R. Maidment

The Hydrologic Information System Project of the Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) has successfully created a large-scale prototype Hydrologic Information System (HIS). This system catalogs and provides access to over 23 million time series of hydrologic data, which are distributed across the United States at various academic, research, and governmental data providers. The service-oriented architecture that enables the HIS comprises distributed hydrologic data servers, a centralized series catalog, and various client software applications, and is supported by WaterML, a standardized language for transmission of hydrologic data.

The current architectural model, termed the Network-Observations Model, of the HIS relies on a searchable central catalog of series metadata. Harvesting series metadata from large federal data providers, such as the USGS, EPA, and NCDC, has proven a laborious undertaking and involves custom database migration tools. This time-consuming harvesting task, coupled with a multitude of custom-coded solutions at the central series catalog has led to concerns with the long-term sustainability of the current architectural model.

A new architectural model, termed the Services Stack Model, is proposed in this thesis. In the proposed model, a catalog of *services* metadata, rather than of *series* metadata is used to connect hydrologic data consumers with data providers. Internationally-recognized web service and data encoding standards, including the upcoming WaterML2.0 specification, from the Open Geospatial Consortium are used as the backbone of the new model. The proposed model will hopefully lead to greater acceptance of the CUAHSI-HIS, and result in increased sustainability and reduced maintenance of the system in the long-term.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Motivation

As global populations grow, stresses on the environment and natural resources will also increase. Demands on one of our most fundamental resources, water, will likely see unprecedented growth to meet drinking, agriculture, and power production requirements worldwide. In order to effectively plan and manage water resources we need to more fully understand them. Understanding water resources involves not only knowing how much water exists and how and where it is flowing, but also its quality and the environmental and biological impacts of its abstraction. However, due to the complex nature of the hydrologic cycle, wherein water moves in different phases over widely varying periods of time and distances, this is not an easy task. In order to most effectively and efficiently manage water resources, planners, scientists, and engineers need to synthesize information about our water environment. This synthesis of information requires that observations data about the hydrologic cycle be accessible and in a usable format.

The Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) Hydrologic Information System (HIS) project has created a large-scale prototype HIS to make integrated hydrologic data accessibility a reality. As the grant period of the HIS project comes to an end, the project must ensure that its legacy is sustainable, supportable, and deployable. To this end, lessons learned from the creation of the prototype HIS need to be used to propose a new, more robust, standards-based HIS architecture that will ensure future sustainability.

This chapter describes what hydrologic data are, why they are so complex, and in broad terms what the CUAHSI-HIS project is.

1.2 Hydrologic Data

Hydrologic data are data that describe the water environment. Observational hydrologic data includes the physical properties; chemical constituents; atmospheric conditions; and biological movement, processes, and life that comprise this environment. These data can be observed *in situ*, such as with stream flow gages, or *ex situ*, such as when a water sample is collected and analyzed in a lab. Hydrologic data may be collected at point-locations, such as with pan evaporation, or may cover wide swaths of the land surface, such as with remotely-sensed precipitation data. Additional hydrologic data are created via models or through other derived products.

Hydrologic observations data have three fundamental characteristics: location, date and time, and value (Tarboton, Horsburgh and Maidment 2007). The *data cube* (see Figure 1) is a conceptual way of describing hydrologic data along three primary axes: location (where), time (when), and variable (what) (Whiteaker 2010). At the intersection of selected values along these three axes is what could be an *observation*: a numeric value corresponding to a specific location in space, at a specific time for a specific variable. Data observation is not perfect, so for any observation system recorded observation values do not exist along the continuum of location, time, and variable. For example, the United States Geological Survey (USGS) National Water Information System measures streamflow at 15-minute intervals and at various locations within the surface water system of the United States.

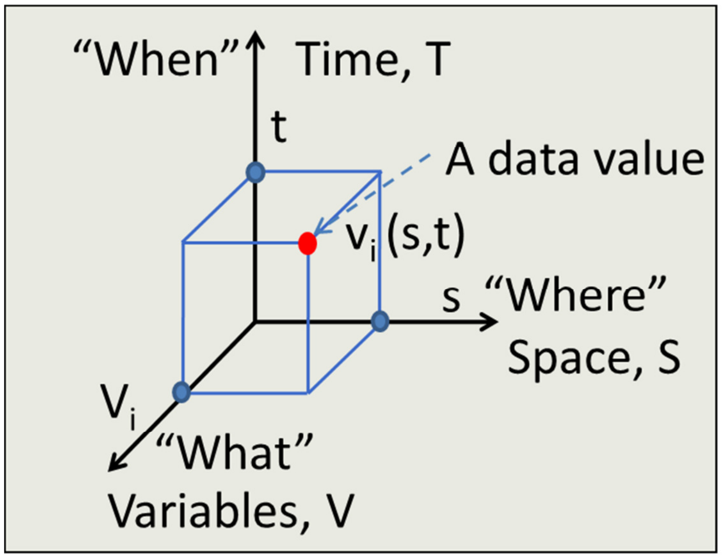


Figure 1: The "data cube" (Whiteaker 2010)

In addition to the diversity in the types of hydrologic data, there is also great variety in who collects the data and how the data are stored. In the United States, hydrologic data are collected and managed at many different levels of governmental, scientific, and academic agencies. The scales of these agencies' observations networks range from individual catchments to the nation's largest waterways and upwards to the global atmosphere. For instance, the USGS maintains a nationwide network of almost 1.5 million sites to measure surface water levels and flow, groundwater levels, and water quality. Smaller agencies include those like the Lower Colorado River Authority (LCRA) in Texas, which collects hydrologic data such as precipitation and streamflow for the Lower Colorado River.

The assortment of agencies collecting hydrologic data leads to both syntactic and semantic heterogeneity, as defined by Horsburgh, et al (2009, 881):

"Syntactic heterogeneity refers to a difference in how data and metadata are organized (e.g., rows vs. columns) and encoded (e.g., text files verses Excel spreadsheets), while semantic heterogeneity refers to the variety in language and terminology used to describe observations."

The diversity in types of hydrologic data along with the syntactic and semantic heterogeneity of the data make bringing together different observations both difficult and time consuming. Fierro (2007, 1) argues that "scientists spend 80% of their time managing the data and 20% analyzing and interpreting." The same issues with observational data also extend to *metadata*: the supplementary data that describes the observational data. With predictions on the vast amount of data that will be produced in the future, creating a framework by which all these differences in water data can be reconciled would be a boon to the scientific understanding and resource management capabilities in hydrologic disciplines.

1.3 CUAHSI-HIS PROJECT

The modern World Wide Web architecture, along with increases in data bandwidth and server speeds has led to the development of standards for sharing information via machine-to-machine interactions called *web services*. Many national-level hydrologic data collectors in the U.S., such as the USGS and Environmental Protection Agency (EPA), are mandated to make their data available to the public. Smaller data collectors, while often not mandated to share data, often do so anyway to aid the scientific and academic communities. The development of web services has undoubtedly made the dissemination of data easier than in the days of hard-copy printed data tables.

Unfortunately, nearly all agencies and data collectors have their own data formats, protocols, and nomenclature so that even though there are web services to access data, there is still great difficulty in integrating the data from different sources.

The CUAHSI-HIS project has been created to help solve these hydrologic data integration problems. CUAHSI is an international, U.S. National Science Foundation-funded consortium comprised of over 120 member universities and research groups, the majority of which are U.S.-based institutions (see Figure 2). Part of CUAHSI's overarching mission is "to enhance hydrologic science by facilitating user access to more and better data for testing hypotheses and analyzing hydrologic processes" (CUAHSI 2010) In support of this mission, CUAHSI directs the aforementioned HIS project, "a national cyber-information system for sharing hydrologic data" (CUAHSI-HIS 2010). Member teams of the CUAHS-HIS project are at the Center for Research in Water Resources (CRWR) at the University of Texas at Austin, the San Diego Supercomputer Center (SDSC), Utah State University, Idaho State University, and the University of South Carolina.

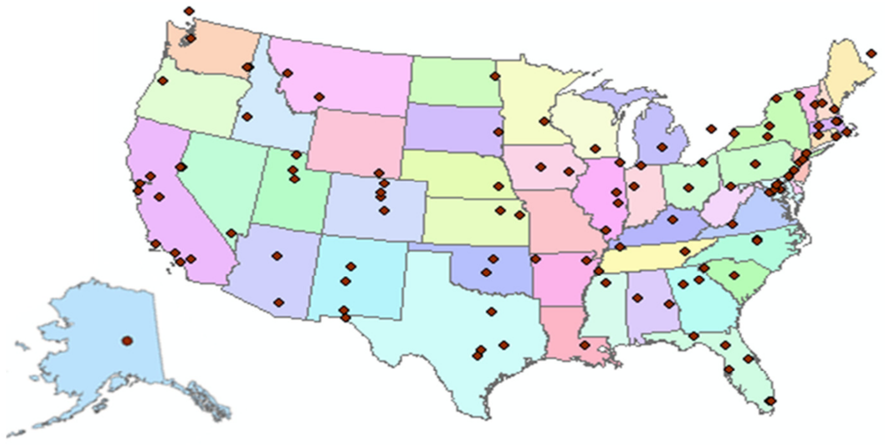


Figure 2: CUAHSI U.S. member institution locations

This CUAHSI-HIS comprises hydrologic data servers distributed throughout the U.S., a central cataloging system at the San Diego Supercomputer Center, and clients that use this system. The central catalog (HIS Central) has 62 public services registered at the time of writing this thesis. Collectively, these services provide access to nearly 23 million data series containing approximately 5.1 billion data values measured at nearly 2 million sites (Tarboton, Maidment, et al. 2010). Figure 3 shows some of these sites from the largest hydrologic data providers.

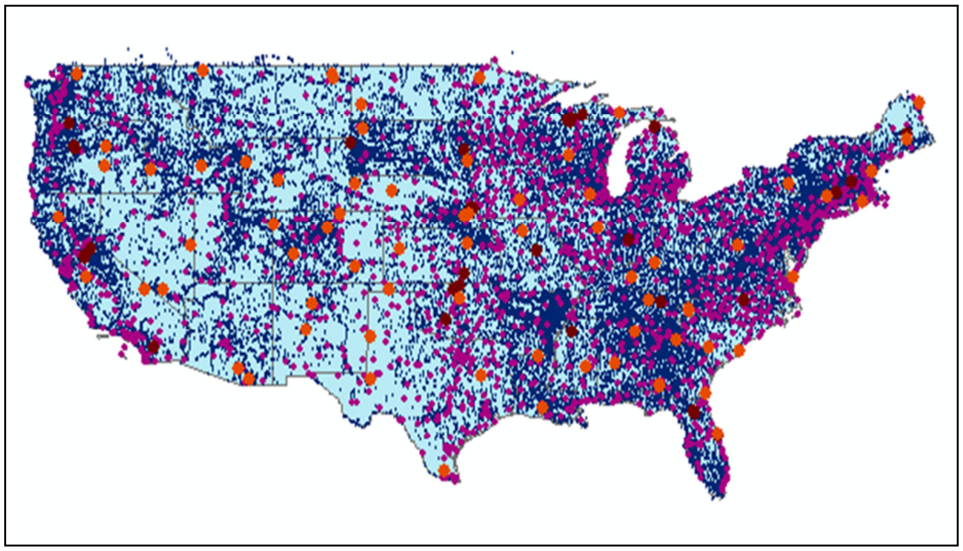


Figure 3: Registered hydrologic data sites at HIS Central

This system provides the means of water data publication, sharing, discovery, analysis, and visualization. The HIS project has created and published several key technologic products in support of this system. These products include the Observations Data Model, WaterOneFlow web services, Water Markup Language, HydroExcel, and HydroDesktop.

1.4 Objectives and Chapter Outline

The objectives of this thesis are to answer the following questions:

- What are the components of the current HIS architecture and how do these pieces work together?
- What are the operating models for hydrologic data access that have been followed within the current HIS architecture?
- Can a new, sustainable architectural model that leverages international standards and a deployable catalog component be proposed?

- What areas of future research are necessary to move toward this new architectural model?

Chapter two of this thesis is a literature and technology review for topics and technologies relating to the CUAHSI-HIS project. This chapter covers technologies both currently leveraged by the project as well as those toward which the project is likely headed.

Chapter three describes the current architecture of the CUAHSI-HIS. Each component of the system is elaborated. The client-centric operating models of the current HIS are also defined and described.

Chapter four proposes a new architecture for the CUAHSI-HIS that utilizes a suite of standard web services from the Open Geospatial Consortium. A proof-of-concept client application for this model is also presented to illustrate how the proposed architectural system will function.

Chapter five presents conclusions and areas of future investigation.

Chapter 2: Literature and Technology Review

2.1 Hydrologic Information Systems

2.1.1 Definitions

To understand what a hydrologic information system is and the needs it meets, a foundational definition for generic information systems should first be identified. Langefors (1973, 195) provides a definition of an *information system* as "a system of information sets needed for decision and signaling in a larger system (of which it is a subsystem) containing subsystems for collecting, storing, processing, distributing information sets." In this definition, *information* refers to "any kind of knowledge or message that can be used to improve or make possible a decision or action" (Langefors 1973, 319). The term *data*, then, refers to the digital representation of information (Langefors 1973).

The preceding definition of an information system can be specialized and extended to refer to information systems of particular information domains. For instance, Marble (1984) defines *geographic information systems* (GIS) as containing four subsystems for geospatial data: (1) input, (2) storage and retrieval, (3) manipulation and analysis, and (4) reporting through maps or tables. Connections between the subsystems that define an information system laid forth by Langefors and those that define a GIS can be made, as depicted in Figure 4.

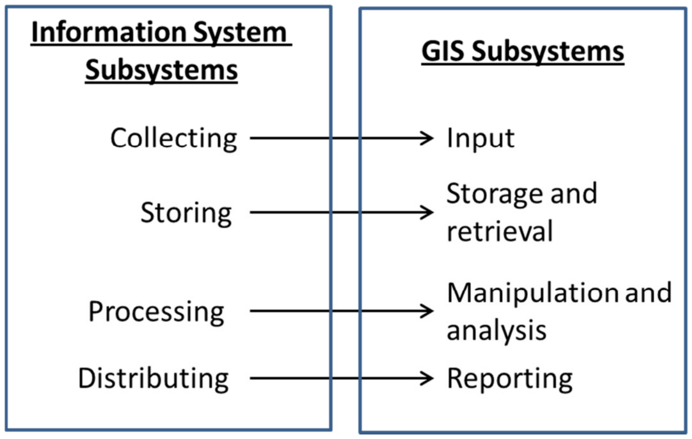


Figure 4: Information system subsystems mapped to GIS subsystems

Following this pattern of information system definitions, a *hydrologic information system* (HIS) can thus be defined as an information system for the domain of hydrologic data. A HIS should then contain subsystems for hydrologic data (discussed in Chapter 1) collection, storage/retrieval, processing/analysis, and distributing/reporting. Tarboton, et al. (2010, 1) offer a definition of HIS "1) as a way of publishing hydrologic data in a uniform way; 2) as a way of discovering and accessing remote water information archives in a uniform way; and 3) as a way of displaying, synthesizing and analyzing water information and exporting it to other analysis and modeling systems." This definition of HIS in terms of capabilities generally fits within the framework of information systems set forth by Langefors. Figure 5 shows how the capabilities of a HIS map to the basic information systems subsystems. The storage capability in the HIS definition from Tarboton, et al. (2010) is implicit in mentioning "water information archives."

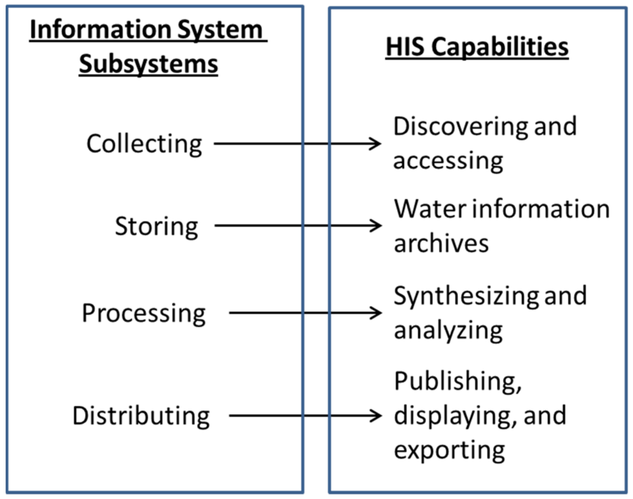

Information System Subsystems
Collecting
Storing
Processing
Distributing
HIS Capabilities
Discovering and accessing
Water information archives
Synthesizing and analyzing
Publishing, displaying, and exporting

Figure 5: Information system subsystems mapped to HIS subsystems

Maidment (2009, 2) defines the components of a HIS as "software applications that store, access and index hydrologic information." Further pointing out that a HIS can work in conjunction with a GIS, a key difference between GIS and HIS is that hydrologic data vary greatly with time, while geospatial data are usually static and have little time variation (Maidment 2009).

2.1.2 Web Services

The World Wide Web Consortium (W3C) defines a generic *web service* as "a software system designed to support interoperable machine-to-machine interaction over a network" (W3C 2004). Web services are accessed through *endpoints*, which are typically addressed via uniform resource identifiers (URIs) (Endrei, et al. 2004). A single web service usually comprises several *methods* or *operations* that, according to the input parameters, act upon the resources available through the service. Web services are loosely-coupled to their clients and other interacting services, and two-way communication is accomplished through messages (Endrei, et al. 2004). A message to a web service is a *request* and the message back from the service is a *response*. The published means of interacting with a particular web service, including all of its methods and the possible inputs to those methods, is an *application programming interface* (API).

At the base level, web services, as their name implies, function over the World Wide Web (WWW) which is made possible by the Internet. The WWW uses Hypertext Transfer Protocol version 1.1 (HTTP/1.1) as its communication protocol over the Internet. HTTP/1.1 has four main operations on web resources: *GET*, *POST*, *PUT*, and *DELETE* (W3C 1999). Generally speaking, the GET operation is for reading data, POST is for creating new data, PUT is for creating or updating data, and DELETE is for deleting data (W3C 1999). Because web services are implemented over the WWW, at the most basic level, all web service methods eventually map to these HTTP operations. RESTful web services and SOAP web services are two strategies of web service implementation (Pautasso, Zimmerman and Leymann 2008).

RESTful web services are based on the architectural idea of REpresentational State Transfer (REST), a concept developed by Fielding (2000) in his doctoral dissertation. REST is actually the guiding architecture to the HTTP/1.1 specification, of which Fielding was a principal designer. Web services that follow the REST architectural guidelines are known as *RESTful*. In RESTful systems, communication is stateless "such that each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server" (Fielding 2000, 78-79). This lack of session state tracking on the server increases scalability because servers do not have to keep track of session information, though it decreases network performance since data may be repeated in requests (Fielding 2000).

Rather than defining new method interfaces, RESTful web services take advantage of some or all the HTTP/1.1 operations to perform their actions (Pautasso, Zimmerman and Leymann 2008). For instance, HTTP GET requests are used to request data from a RESTful web service. Parameters to GET requests are sent via simple URI-encoded calls to the web server. Thus, a request to get information about a particular book with a given identification number from an online bookstore's web service might look like the following (where the service endpoint URI is highlighted in red and the request parameters are in blue):

```
http://www.bookstore.com/bookService?request=GetInfo&bookId=123
```

Because REST is an architectural ideal rather than a specifically defined protocol, there is no agreed-upon standard for RESTful web services. Rather, the endpoints, operations, parameters, and other information of a RESTful web service are usually defined in an API by the service provider, and client software must be programmed to interact with that API in particular.

SOAP, by comparison, is a completely-specified, standardized protocol that describes the message formats, encoding rules, and transport mechanism for web services (Endrei, et al. 2004). The SOAP (which formerly stood for Simple Object Access Protocol) protocol is developed and maintained by the W3C (W3C 2007). The SOAP message format (both requests and response) is based on eXtensible Markup Language (XML). A SOAP message comprises a wrapping *SOAP envelope* that holds an optional *SOAP header* component and a mandatory *SOAP body* component (W3C 2007). The SOAP header is usually used for authentication and session state management, while the SOAP body contains the actual request/response payload. Although technically any transport protocol could be used with SOAP, HTTP is currently the only such transport protocol accepted by the SOAP specification (Endrei, et al. 2004).

SOAP web service providers usually publish a Web Services Description Language (WSDL) file on their servers. WSDL files provide the “operational characteristics of a Web service using an XML document” (Endrei, et al. 2004, 123). These operational characteristics include what the web service is about, where it resides (the endpoint URI), and what is needed to invoke the service (Endrei, et al. 2004).

2.1.3 Service-Oriented Architectures

Service-Oriented Architectures (SOA) are system architectures built around services connected together to achieve higher-level processes and solutions (Rosen, Lublinsky and Smith 2008). There are three core concepts that comprise SOA: services, interoperability, and loose-coupling (Josuttis 2007). Endrei, et al. (2004, 27-28) describe the web services that empower SOA as having several key characteristics:

> *“Services are self-contained and modular. Services support interoperability. Services are loosely coupled. Services are location-transparent. Services are composite modules, comprised of components.”*

Based these characteristics and the definition of SOA, web services are an appropriate class of services upon which a larger system architecture can be built.

In a SOA, there are two fundamental components: (1) service providers and (2) service consumers (Nickul 2007). Service consumers connect directly with service providers to request and receive data (Nickul 2007). A third component, called a service registry can also be included in a SOA (Endrei, et al. 2004). The interactions among these three components are displayed in Figure 6. Service providers publish web services to make data available and register their services at a registry. Service registries (i.e., catalogs) allow consumers to search for desired services based on some criteria. The registry may also provide the consumer with the interface or endpoint to matching services. Service consumers (i.e., clients) invoke services to request data (Endrei, et al. 2004).

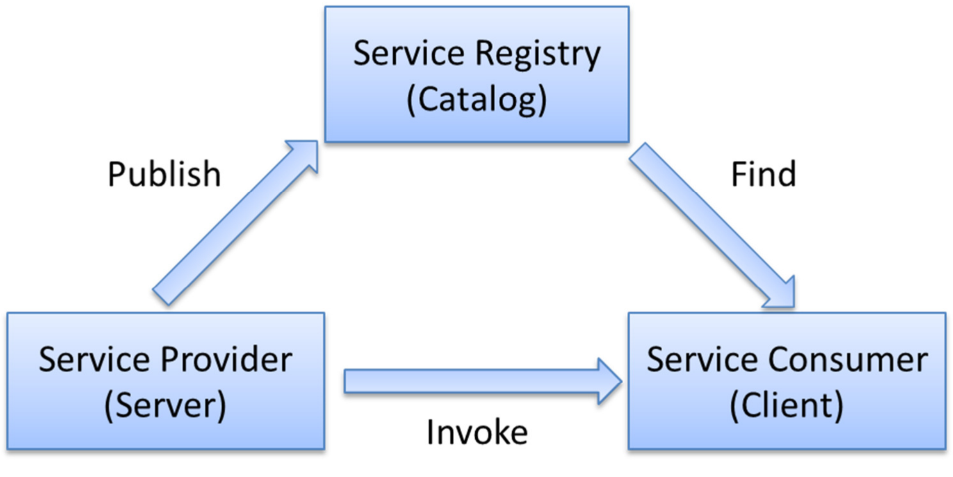

Service Registry (Catalog)
Publish
Find
Service Provider (Server)
Service Consumer (Client)
Invoke

Figure 6: Component interactions in SOA

2.2 THE CUAHSI-HIS SERVICE-ORIENTED ARCHITECTURE

2.2.1 Background

The CUAHSI-HIS project has created a prototype SOA for hydrologic information. Figure 7 is a conceptual diagram of this system. As with the generic SOA described in the previous section, the HIS SOA has three basic components: service providers called *HydroServers*, a service registry called *HIS Central*, and service consumers such as *HydroDesktop*. The enabling technologies behind all these components are *Water Markup Language* and *WaterOneFlow* web services. The following sections contain descriptions of the WaterML and WOF specifications.

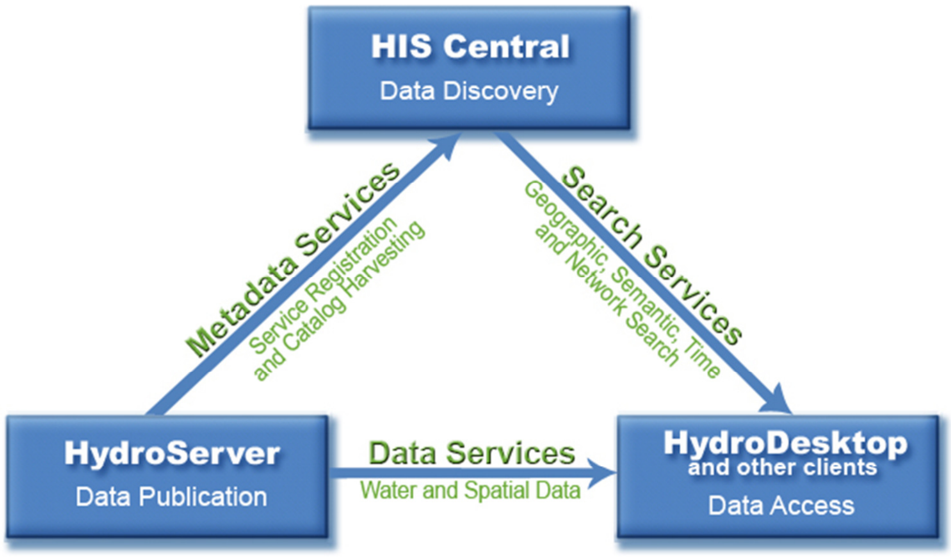


Figure 7: CUAHSI-HIS SOA (*image from http://his.cuahsi.org/*)

2.2.2 WaterML

Water Markup Language (WaterML) is CUAHSI's standardized encoding for transmission of hydrologic observations data via the Internet, and specifically via WOF services. WaterML can accommodate time series of observations with different time support (such as hourly, daily, or monthly) and time representations, and includes structures specifically for the SOAP protocol (Zaslavsky, Valentine and Whiteaker 2007). In addition to time series, WaterML provides elements for the description of sampling sites, methods, observed variables, and other metadata relating to hydrologic observations.

The WaterML schema is XML-based and was originally developed in support of the WOF web service standard. It thus contains four main elements specifically for describing the responses from each of the WOF methods described in the following section (Zaslavsky, Valentine and Whiteaker 2007). These response elements are called *GetSitesResponse*, *GetSiteInfoResponse*, *GetVariableInfoResponse*, and

GetValuesResponse, corresponding to the GetSites, GetSiteInfo, GetVariableInfo, and GetValues operations of WOF, respectively. Example excerpts from each of these response elements are shown in section 2.2.3 of this thesis.

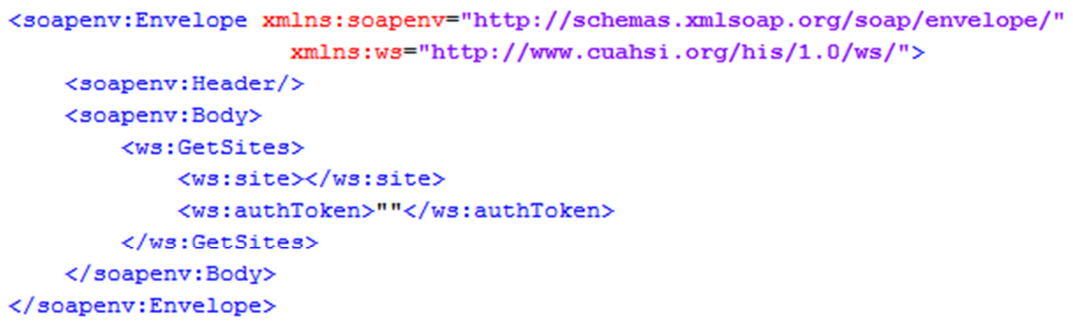
There are currently two versions of CUAHSI's WaterML schema: a stable 1.0 version and an experimental 1.1 version. By using this standard format for hydrologic data representation, a client application that understands the schema can interpret data from any compliant web service.

2.2.3 WaterOneFlow Web Services

WaterOneFlow (WOF) is CUAHSI's web service specification for transferring hydrologic observations data and the metadata that describe them. WOF services use the SOAP protocol, though there has been some movement within the HIS project toward supporting a RESTful interface as well. Corresponding to WaterML, there are two versions of the WOF specification: the original, stable 1.0 version and the new, though experimental 1.1 version. Most (45 of the total 62) of the WOF services that have been registered at HIS Central are using the 1.0 version. Except for where explicitly mentioned, the information in this section refers to the stable 1.0 version of the WOF specification.

There are four main methods available from WOF services: *GetSites, GetSiteInfo, GetVariableInfo,* and *GetValues*. This section describes all four of these methods based on the CUAHSI WaterOneFlow Workbook (Whiteaker 2010). The SOAP message responses from WOF web services all contain WaterML-formatted payloads. All four of the WOF methods can accept an optional "authCode" parameter, which in the future will be used to provide only authenticated access to the data served from the WOF service instance. Currently this parameter is always set to an empty string value ("").

The GetSites operation is used to obtain metadata describing the sampling sites represented through a WOF service. This method takes an optional "site" parameter that specifies the codes for sites about which metadata is desired. If the "site" parameter is left blank, metadata about all of the sites that the WOF service instance contains will be returned. Figure 8 is an example SOAP request to the GetSites operation with a blank "site" parameter.



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:ws="http://www.cuahsi.org/his/1.0/ws/">
    <soapenv:Header/>
    <soapenv:Body>
        <ws:GetSites>
            <ws:site></ws:site>
            <ws:authToken>""</ws:authToken>
        </ws:GetSites>
    </soapenv:Body>
</soapenv:Envelope>
```

Figure 8: Example SOAP WOF GetSites request

The metadata returned from GetSites includes site names, identification codes, and geographic coordinates, and is returned as a WaterML GetSitesReponse. An excerpt of site metadata from a GetSitesResponse is shown in Figure 9.

```
<site>
   <siteInfo>
      <siteName>Lydia Ann Channel</siteName>
      <siteCode network="TWDBQuality" siteID="50">Aransas95_2</siteCode>
      <geoLocation>
         <geogLocation xsi:type="LatLonPointType" srs="EPSG:4269">
            <latitude>27.85416667</latitude>
            <longitude>-97.05583333</longitude>
         </geogLocation>
      </geoLocation>
      <note title="State">Texas</note>
   </siteInfo>
</site>
```

Figure 9: Example site metadata from GetSitesResponse

The GetSiteInfo operation is for retrieving metadata that describe the time series of hydrologic observations available through the service. This metadata is referred to as a *series catalog* and includes value counts, time range, source, and other information about each series of data for each of the variables measured at the requested sites. In version 1.0 of WOF, the GetSiteInfo method takes a “site” parameter that specifies the site codes for which sites a series catalog is desired. In version 1.1 of WOF, the GetSiteInfo method instead takes a “location” parameter that can contain specific site codes or rectangular geographic extents for which series catalogs are desired. Figure 10 is an example GetSiteInfo SOAP request for the series catalog of the “TWDB:Aransas95_2” site.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                    xmlns:ws="http://www.cuahsi.org/his/1.0/ws/">
   <soapenv:Header/>
   <soapenv:Body>
      <ws:GetSiteInfoObject>
         <ws:site>TWDBQuality:Aransas95_2</ws:site>
         <ws:authToken>""</ws:authToken>
      </ws:GetSiteInfoObject>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 10: Example SOAP WOF GetSiteInfo request

The WaterML response from this request is the GetSiteInfoResponse. An excerpt of a single series from the series catalog in this response is shown in Figure 11. As can be seen in the figure, this particular series shows is for the “specific conductance” variable and has 30 values in a time series that starts on 9/29/1995 and ends on 9/30/1995.

```
<series>
   <variable>
      <variableCode vocabulary="TWDBQuality" default="true" variableID="1">Cond</variableCode>
      <variableName>Specific conductance</variableName>
      <valueType>Field Observation</valueType>
      <dataType>Continuous</dataType>
      <generalCategory>Water Quality</generalCategory>
      <sampleMedium>Surface Water</sampleMedium>
      <units unitsAbbreviation="mS/cm" unitsCode="269">millisiemens per centimeter</units>
      <NoDataValue>-9.99</NoDataValue>
      <timeSupport isRegular="false"/>
   </variable>
   <valueCount>30</valueCount>
   <variableTimeInterval xsi:type="TimeIntervalType">
      <beginDateTime>1995-09-29T13:05:00</beginDateTime>
      <endDateTime>1995-09-30T06:48:00</endDateTime>
   </variableTimeInterval>
   <Method methodID="0">
      <MethodDescription>No method specified</MethodDescription>
   </Method>
   <Source sourceID="1">
      <Organization>Texas Water Development Board</Organization>
      <SourceDescription>TWDB Coastal Water Quality Data</SourceDescription>
   </Source>
   <QualityControlLevel QualityControlLevelID="0"/>
</series>
```

Figure 11: Example series metadata from GetSiteInfoResponse

The GetVariableInfo operation is for retrieving metadata describing hydrologic variables available through the web service. This method takes an optional “variable code” parameter that specifies for which variables information is wanted. If no variable codes are specified, then metadata about all of the variables represented in the WOF service instance will be returned. Figure 12 is an example SOAP request for information about the “TWDBQuality:Cond” variable.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                    xmlns:ws="http://www.cuahsi.org/his/1.0/ws/">
   <soapenv:Header/>
   <soapenv:Body>
      <ws:GetVariableInfoObject>
         <ws:variable>TWDBQuality:Cond</ws:variable>
         <ws:authToken>""</ws:authToken>
      </ws:GetVariableInfoObject>
   </soapenv:Body>
</soapenv:Envelope>
```

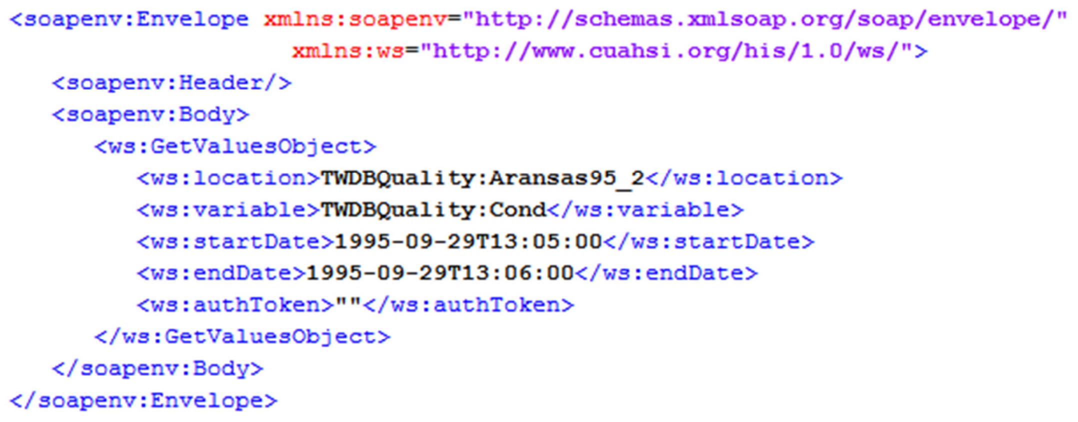
Figure 12: Example SOAP WOF GetVariableInfo request

The WaterML GetVariableInfoResponse returned from the GetVariableInfo method includes variable name, code, sample medium, data type, units and other related metadata. An excerpt from this response for the preceding "TWDBQuality:Cond" GetVariableInfo request is shown in Figure 13.

```
<variable>
    <variableCode vocabulary="TWDBQuality" default="true" variableID="1">Cond</variableCode>
    <variableName>Specific conductance</variableName>
    <valueType>Field Observation</valueType>
    <dataType>Continuous</dataType>
    <generalCategory>Water Quality</generalCategory>
    <sampleMedium>Surface Water</sampleMedium>
    <units unitsAbbreviation="mS/cm" unitsCode="269">millisiemens per centimeter</units>
    <NoDataValue>-9.99</NoDataValue>
    <timeSupport isRegular="false"/>
</variable>
```

Figure 13: Example variable metadata from WOF GetVariableInfoResponse

The GetValues method is the core data retrieval method of WOF. The GetValues operation usually has a single required parameter, "location", for specifying the site code from which time series data is desired. In some WOF instances the variable code for the desired series must also be supplied via the "variable" parameter. Optionally, the "startDate" and "endDate" parameters may be used to specify the time extent of the desired time series data. Figure 14 is a request for the "TWDBQuality:Cond" variable at the “TWDBQuality:Aransas95_2” site for the time period of 5:00 to 6:00AM on 9/29/1995.



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                    xmlns:ws="http://www.cuahsi.org/his/1.0/ws/">
   <soapenv:Header/>
   <soapenv:Body>
      <ws:GetValuesObject>
         <ws:location>TWDBQuality:Aransas95_2</ws:location>
         <ws:variable>TWDBQuality:Cond</ws:variable>
         <ws:startDate>1995-09-29T13:05:00</ws:startDate>
         <ws:endDate>1995-09-29T13:06:00</ws:endDate>
         <ws:authToken>""</ws:authToken>
      </ws:GetValuesObject>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 14: Example SOAP WOF GetValues request

The response from the GetValues method is the WaterML GetValuesResponse, which contains a time series of observations data along with values-level metadata (such as quality control level) for the requested sampling site. Figure 15 contains the values element from the response to the preceding sample request. This response contains a series of three values for the variable, site, and time period specified in the request.

```xml
<values unitsAbbreviation="mS/cm" unitsCode="269" count="3">
      <value censorCode="nc" dateTime="1995-09-29T13:05:00"
                qualityControlLevel="Raw data" methodID="0" sourceID="1"
                offsetValue="13" offsetTypeID="1"
                offsetUnitsAbbreviation="ft" offsetUnitsCode="48">46.6</value>
      <value censorCode="nc" dateTime="1995-09-29T13:05:00"
                qualityControlLevel="Raw data" methodID="0" sourceID="1"
                offsetValue="18" offsetTypeID="1"
                offsetUnitsAbbreviation="ft" offsetUnitsCode="48">46.8</value>
      <value censorCode="nc" dateTime="1995-09-29T13:05:00"
                qualityControlLevel="Raw data" methodID="0" sourceID="1"
                offsetValue="6.5" offsetTypeID="1"
                offsetUnitsAbbreviation="ft" offsetUnitsCode="48">47.6</value>
      <method methodID="0">
         <MethodDescription>EPA #120.1</MethodDescription>
      </method>
      <source sourceID="1">
         <Organization>Texas Water Development Board</Organization>
         <SourceDescription>TWDB Coastal Water Quality Data</SourceDescription>
         <ContactInformation>
            <ContactName>Tom Smith</ContactName>
            <TypeOfContact>main</TypeOfContact>
            <Phone>512-555-4198</Phone>
            <Email>tom.smith@example.com</Email>
            <Address xsi:type="xsd:string">1234 Elm St.,Austin, TX 78751</Address>
         </ContactInformation>
         <SourceLink>http://www.twdb.state.tx.us/</SourceLink>
      </source>
      <offset offsetTypeID="1">
         <offsetDescription>Feet below the water surface the measurement was taken</offsetDescription>
         <units unitsAbbreviation="ft" unitsCode="48">international foot</units>
      </offset>
</values>
```

Figure 15: Example time series data from WOF GetValuesResponse

2.3 Open Geospatial Consortium Standards

2.3.1 Background

The *Open Geospatial Consortium (OGC)* is an international standards consortium comprising various governmental, private, and research institutions around the world. The main goal main goal of the OGC is to "geo-enable the Web" through development and publication of its OpenGIS standards. The *OpenGIS* standards include schemas and specification documents for geospatial web services and encodings of the data served through these services. In October 2010, the Federal Geographic Data Committee

(FGDC) formally endorsed several of the OGC's OpenGIS standards for use by U.S. agencies (OGC 2010).

The Hydrology Domain Working Group (HDWG), a joint working group of the OGC and World Meteorological Organization, is investigating the use of OGC standards for hydrologic data. The HDWG's main activities include hosting interoperability experiments in the areas of both surface and groundwater data, and creating the new *Water Markup Language 2.0* specification. The purpose of these interoperability experiments is to determine best practices for the use of OGC standards and to identify any gaps that these standards have in the realm of hydrologic data.

The OpenGIS web service specifications that are most applicable to the HIS project are *Web Map Service* (WMS), *Web Feature Service* (WFS), *Sensor Observation Service* (SOS), and *Catalogue Services for the Web* (CSW). Each of these service specifications has its own set of methods and data encoding specifications, though there is much similarity across them. This section discusses each of these service specifications with attention to their primary methods and the inputs and outputs from these methods. In support of these service standards are several data encoding standards, *Filter Encoding Standard (FES)*, *Geographic Markup Language (GML), Observations & Measurement (O&M),* and *Water Markup Language 2.0 (WaterML2.0).* These encoding standards are also discussed in this section.

2.3.2 Web Map Service

Web Map Service (WMS) is the OGC service standard for requesting and transmitting geospatially-referenced map images (OGC 2006). The information in this section is an overview of the WMS Implementation Specification from the OGC (2006). WMS implementations must always accept Key-Value Pair (KVP) requests via HTTP

GET, and can optionally accept XML-based requests via HTTP POST. The information in this section refers specifically to HTTP GET requests with KVP parameter encoding, though it is generally applicable to the HTTP POST format as well.

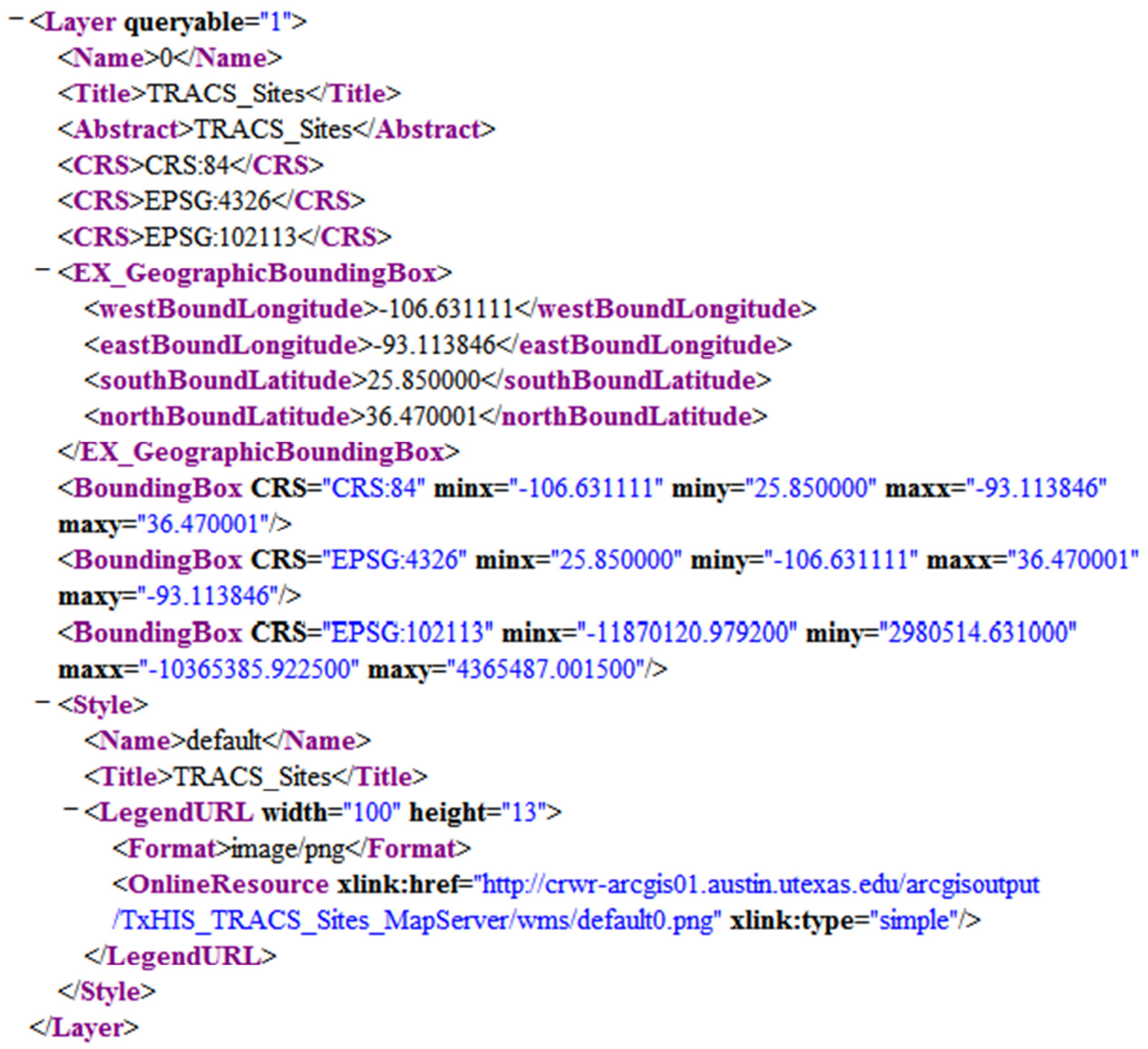
The primary, required methods of a WMS are *GetCapabilities* and *GetMap*. In addition, there is the optional *GetFeatureInfo* method. As with the other OGC service standards, the GetCapabilities operation is for retrieving service-level metadata. GetCapabilities returns an XML document whose schema is specified in the Web Map Server Implementation Specification (OGC 2006). In WMS, this metadata includes (but is not limited to):

- which of the WMS operations are supported by the WMS endpoint;
- the abstract, author, keywords, contact information, fees, and authorized use of the service; and
- the names, titles, Styles, geographic bounds, coordinate reference system, and other metadata about the Layers represented.

WMS GetCapabilities requests have two required parameters, "request" which specifies the method name "GetCapabilities" and "service" which specifies the service name "WMS". An example GetCapabilities request to the CRWR TRACS_Sites WMS endpoint request via HTTP GET is shown below, with the URL highlighted in red and the KVP parameters in blue:

```
http://crwr-arcgis01.austin.utexas.edu/arcgis/services/TxHIS/TRAC
      S_Sites/MapServer/WMSServer?
      request=GetCapabilities&
      service=WMS
```

Figure 16 shows an excerpt containing the TRACS_Sites Layer element from the response to this request.



```
- <Layer queryable="1">
    <Name>0</Name>
    <Title>TRACS_Sites</Title>
    <Abstract>TRACS_Sites</Abstract>
    <CRS>CRS:84</CRS>
    <CRS>EPSG:4326</CRS>
    <CRS>EPSG:102113</CRS>
  - <EX_GeographicBoundingBox>
      <westBoundLongitude>-106.631111</westBoundLongitude>
      <eastBoundLongitude>-93.113846</eastBoundLongitude>
      <southBoundLatitude>25.850000</southBoundLatitude>
      <northBoundLatitude>36.470001</northBoundLatitude>
    </EX_GeographicBoundingBox>
    <BoundingBox CRS="CRS:84" minx="-106.631111" miny="25.850000" maxx="-93.113846"
    maxy="36.470001"/>
    <BoundingBox CRS="EPSG:4326" minx="25.850000" miny="-106.631111" maxx="36.470001"
    maxy="-93.113846"/>
    <BoundingBox CRS="EPSG:102113" minx="-11870120.979200" miny="2980514.631000"
    maxx="-10365385.922500" maxy="4365487.001500"/>
  - <Style>
      <Name>default</Name>
      <Title>TRACS_Sites</Title>
    - <LegendURL width="100" height="13">
        <Format>image/png</Format>
        <OnlineResource xlink:href="http://crwr-arcgis01.austin.utexas.edu/arcgisoutput
        /TxHIS_TRACS_Sites_MapServer/wms/default0.png" xlink:type="simple"/>
      </LegendURL>
    </Style>
  </Layer>
```

Figure 16: Excerpt containing Layer element from WMS GetCapabilities response

The core functionality of a WMS is exposed through the GetMap method. This operation retrieves map images based on criteria specified in the request parameters. The images can be in a variety of formats, including PNG, JPG, SVG, and others, depending on the specific WMS implementation. Images from WMS can be requested to have transparent backgrounds so that they may be overlaid onto one another to make composite maps.

Maps served through WMS are divided into *layers*. For example, a WMS for a Texas surface water hydrology map could contain 3 distinct layers: streams, water bodies, and sampling points. Each layer has one or more predefined *styles*, which principally refer to different symbologies. For example, the sampling point layer could have two styles: one in which all the sampling points are blue circles, and another in which the sampling points are red squares. Layers can also be arranged hierarchically, in which case only the leaf nodes in the hierarchy would be layers of visual data. The parent layers would be for organizational purposes. For example, a WMS for Texas water quality could have parent layers called "Surface Water Features" (with child layers for streams and water bodies), and "Sampling Points" (with child layers for salinity sampling points and nutrient sampling points).

The GetMap method has several required and optional parameters, which are described in section 7.3 of the OGC WMS Implementation Specification (OGC 2006). The required parameters are "version" (1.3.0), "request" (GetMap), "layers", "styles", "crs" (coordinate reference system), "bbox" (geographic bounding box), "width" (pixel width of the image), "height" (pixel height of the image), and "format" (file format of the image). The combination of these parameters defines a geographic extent in a given coordinate system for which an image containing the specified Layers of the map symbolized by the specified Styles is requested. There are additional optional parameters for specifying the background color or transparency of the map. An example GetMap request to CRWR's TRACS_Sites WMS is shown below:

```
http://crwr-arcgis01.austin.utexas.edu/arcgis/services/TxHIS/TRAC
      S_Sites/MapServer/WMSServer?
      request=GetMap&
      version=1.3.0&
      styles=default&
```

```
format=image/png&
layers=0&
crs=EPSG:4326&
width=700&
height=600&
bbox=25.850000,-106.631111,36.470001,-93.113846
```

The image response from this request is shown in Figure 17.

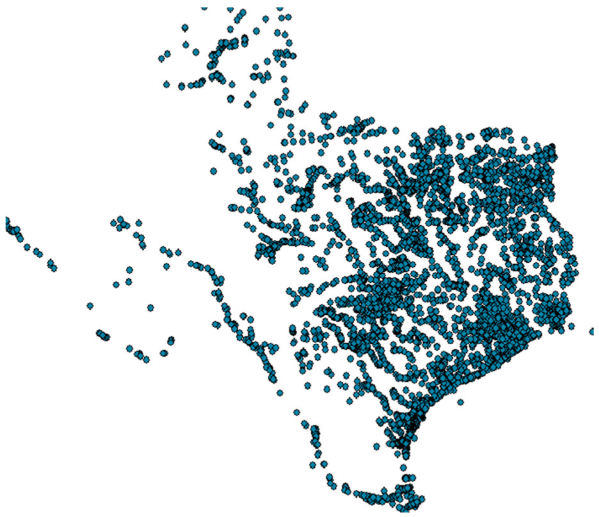


Figure 17: Response from GetMap request on TCEQ_Tracs WMS Service

The optional WMS method, GetFeatureInfo, provides additional information about a selected point on a map provided by the GetMap operation. The parameters for this method are described in section 7.4.2 of the WMS Implementation Specification (OGC 2006). The response method is very loosely defined and is largely left up to the specific WMS implementation.

2.3.3 Web Feature Service

OGC's Web Feature Service (WFS) specification provides "…interfaces for data access and manipulation operations on geographic features…" and is described in detail by the WFS Implementation Specification (OGC 2005, 5). The information in this section is a brief overview from the OGC's WFS Implementation Specification (2005). As with WMS, WFS requests can be made through HTTP GET with KVP URL encoding or through HTTP POST with an XML-formatted request. Additionally, WFS can be implemented to accept and respond to SOAP messages. Again, the information in this section refers specifically to HTTP GET requests with KVP parameter encoding, though it is generally applicable to the HTTP POST and SOAP formats as well.

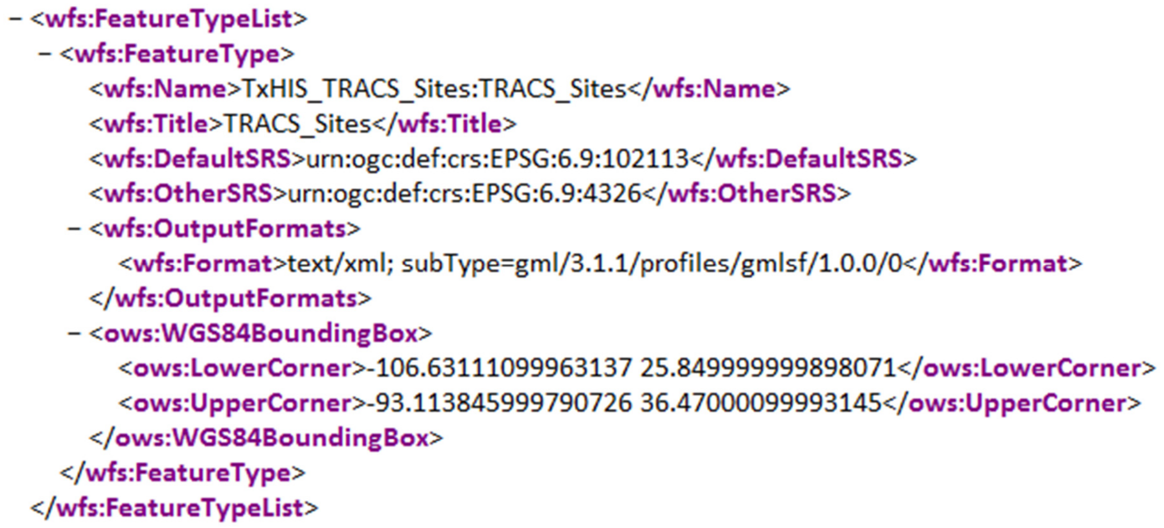
The core concept of WFS is the *geographic feature*. A feature can be almost anything of interest, and is represented by a collection of attributes. Each attribute has a name, a type (such as "double" or "string"), and a value. Geographic features are features with a geometric property.

The methods of WFS are *GetCapabilities, DescribeFeatureType, GetFeature, GetGmlObject, Transaction, and LockFeature*. However, for a basic WFS implementation, only GetCapabilities, DescribeFeatureType, and GetFeature are required. Only these three methods are described in this section.

As with all OGC service specifications, the GetCapabilities operation of WFS is for retrieving service-level metadata, such as title, abstract, keywords, and access constraints, and listing which methods of the specification are available. For WFS, GetCapabilities also lists all the feature types for which data are available and the types of filters that can be used for the GetFeature request. Filters are discussed more in 2.3.6 of this thesis. There are only two required parameters for the WFS GetCapabilities method, "request", whose value is always "GetCapabilities" and "service", whose value is always "WFS". The response from GetCapabilities is an XML document whose schema is defined in the WFS Implementation Specification. An example GetCapabilities request to CRWR's TRACS_Sites WFS service is shown below:

```
http://crwr-arcgis01.austin.utexas.edu/arcgis/services/TxHIS/TRAC
      S_Sites/MapServer/WFSServer?
      request=GetCapabilities&
      service=WFS
```

An excerpt showing the FeatureTypeList in the XML response to this request is shown in Figure 18. For this WFS, there is only a single feature type available, called TRACS_Sites as can be seen in the figure.



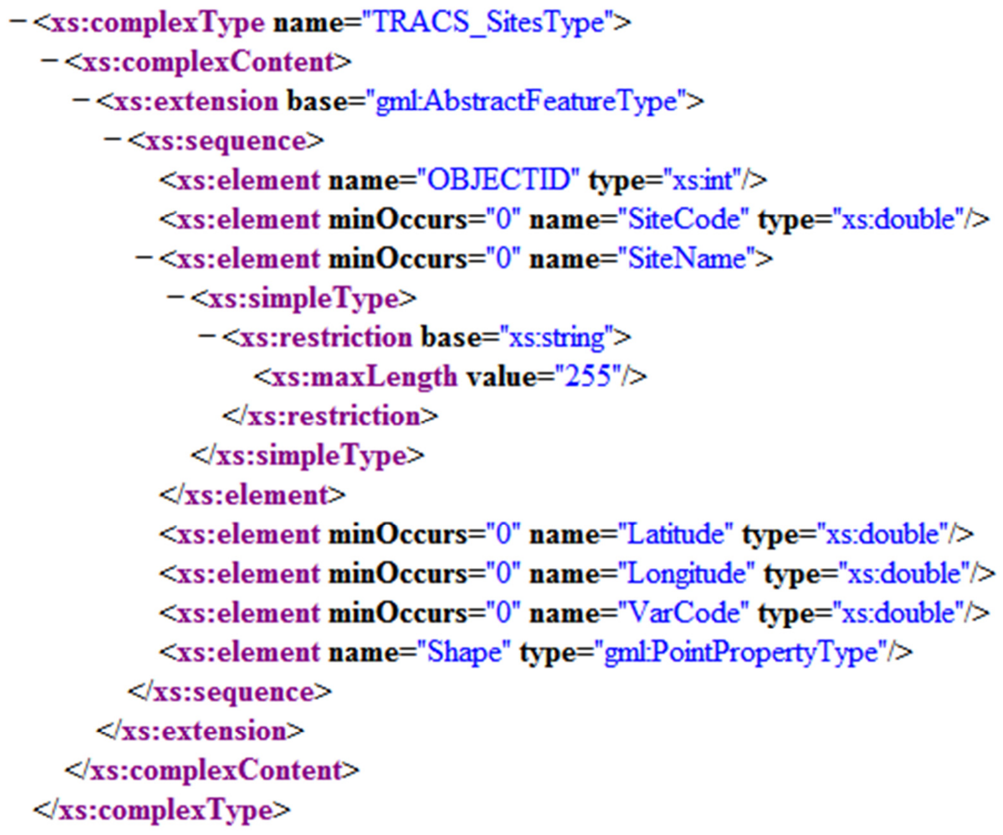
```
- <wfs:FeatureTypeList>
  - <wfs:FeatureType>
      <wfs:Name>TxHIS_TRACS_Sites:TRACS_Sites</wfs:Name>
      <wfs:Title>TRACS_Sites</wfs:Title>
      <wfs:DefaultSRS>urn:ogc:def:crs:EPSG:6.9:102113</wfs:DefaultSRS>
      <wfs:OtherSRS>urn:ogc:def:crs:EPSG:6.9:4326</wfs:OtherSRS>
    - <wfs:OutputFormats>
        <wfs:Format>text/xml; subType=gml/3.1.1/profiles/gmlsf/1.0.0/0</wfs:Format>
      </wfs:OutputFormats>
    - <ows:WGS84BoundingBox>
        <ows:LowerCorner>-106.63111099963137 25.849999999898071</ows:LowerCorner>
        <ows:UpperCorner>-93.113845999790726 36.47000099993145</ows:UpperCorner>
      </ows:WGS84BoundingBox>
    </wfs:FeatureType>
  </wfs:FeatureTypeList>
```

Figure 18: FeatureTypeList excerpt from GetCapabilities response on TCEQ_Tracs WFS

The DescribeFeatureType operation describes the attributes of the features served by the WFS. The only required parameter to this method is “request”, whose value is always “DescribeFeatureType”. By default, DescribeFeatureType returns a GML document with the structure and attributes of each type of feature served by the WFS. The optional “typename” parameter can be used to specify the feature types for which a description is desired. An example DescribeFeatureType request to the TRACS_Sites WFS at CRWR is shown below:

```
http://crwr-arcgis01.austin.utexas.edu/arcgis/services/TxHIS/TRAC
      S_Sites/MapServer/WFSServer?
      request=DescribeFeatureType
```

Figure 19 is an excerpt of the response from this DescribeFeatureType request. In the response, it can be seen that the TRACS_SitesType, the only feature type available from the TRACS_Sites WFS, has seven attributes: OBJECTID, SiteCode, SiteName, Latitude, Longitude, VarCode, and Shape. The Shape attribute is of type gml:PointPropertyType and is the attribute that makes the TRACS_SitesType a geographic feature. The other six attributes were the attributes in the shapefile from which this WFS was produced.



```
- <xs:complexType name="TRACS_SitesType">
  - <xs:complexContent>
    - <xs:extension base="gml:AbstractFeatureType">
      - <xs:sequence>
          <xs:element name="OBJECTID" type="xs:int"/>
          <xs:element minOccurs="0" name="SiteCode" type="xs:double"/>
        - <xs:element minOccurs="0" name="SiteName">
          - <xs:simpleType>
            - <xs:restriction base="xs:string">
                <xs:maxLength value="255"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:element minOccurs="0" name="Latitude" type="xs:double"/>
          <xs:element minOccurs="0" name="Longitude" type="xs:double"/>
          <xs:element minOccurs="0" name="VarCode" type="xs:double"/>
          <xs:element name="Shape" type="gml:PointPropertyType"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

Figure 19: TRACS_SitesType definition (GML) from DescribeFeatureType on TRACS_Sites WFS

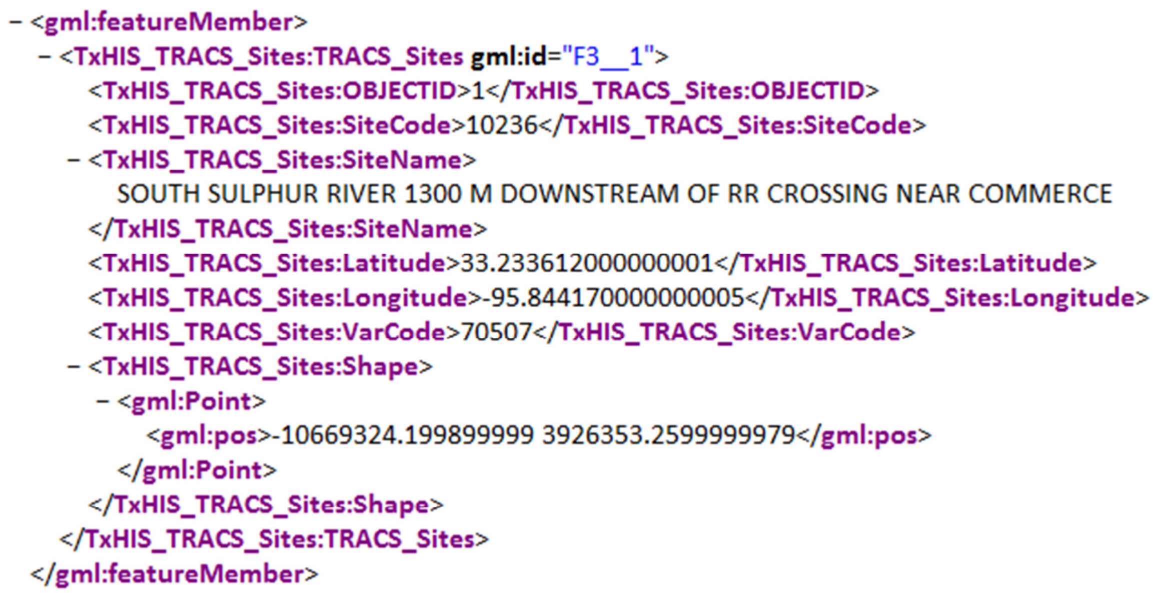
The GetFeature operation provides the main functionality of WFS. Through this method, the set of features (with their attributes) that match given criteria are returned as a GML document. If no criteria are specified, then all the features of a given type are returned. The required parameters for a GetFeature request are the "request" parameter whose value is "GetFeature" and "typeName" whose value is a comma-separated list of the types of features to return. As mentioned earlier, the feature types available from a WFS are listed in the GetCapabilities response. Criteria on which features can be matched are constructed as OGC Filters, and are included in the GetFeature request through the "filter" parameter. If the "resultType" parameter is set to a value of "hits", then an XML document containing the count of features that would be returned by the GetFeature request is returned instead of the actual listing of features.

Alternatively, instead of the "typeName" parameter, a GetFeature request can also use the "featureid" parameter whose value is a comma-separated list of identifiers for specific features. There are several other possible parameters available for crafting a GetFeature request explained in the WFS Implementation Specification (OGC 2005).

An example of a GetFeature request on the TRACS_Sites WFS service at CRWR is shown below:

```
http://crwr-arcgis01.austin.utexas.edu/arcgis/services/TxHIS/TRAC
      S_Sites/MapServer/WFSServer?
      request=GetFeature&
      typeName=TxHIS_TRACS_Sites:TRACS_Sites
```

This request returns 7138 features, the complete set of TRACS_Sites available from the service. A GML-excerpt containing a feature returned from this request is shown in Figure 20. Note that this example feature has values for all the attributes of the TRACS_Sites feature type as described in the DescribeFeatureType response from Figure 19.



```
- <gml:featureMember>
  - <TxHIS_TRACS_Sites:TRACS_Sites gml:id="F3__1">
      <TxHIS_TRACS_Sites:OBJECTID>1</TxHIS_TRACS_Sites:OBJECTID>
      <TxHIS_TRACS_Sites:SiteCode>10236</TxHIS_TRACS_Sites:SiteCode>
    - <TxHIS_TRACS_Sites:SiteName>
        SOUTH SULPHUR RIVER 1300 M DOWNSTREAM OF RR CROSSING NEAR COMMERCE
      </TxHIS_TRACS_Sites:SiteName>
      <TxHIS_TRACS_Sites:Latitude>33.233612000000001</TxHIS_TRACS_Sites:Latitude>
      <TxHIS_TRACS_Sites:Longitude>-95.844170000000005</TxHIS_TRACS_Sites:Longitude>
      <TxHIS_TRACS_Sites:VarCode>70507</TxHIS_TRACS_Sites:VarCode>
    - <TxHIS_TRACS_Sites:Shape>
      - <gml:Point>
          <gml:pos>-10669324.199899999 3926353.2599999979</gml:pos>
        </gml:Point>
      </TxHIS_TRACS_Sites:Shape>
    </TxHIS_TRACS_Sites:TRACS_Sites>
  </gml:featureMember>
```

Figure 20: Excerpt from GetFeature response on TCEQ_Tracs WFS

To handle large response sets, the WFS specification allows *paging* of results from the GetFeature method. If paging is enabled on a WFS instance, there is a maximum allowable number of results, called a *page*, that can be contained in a response. In the case that a greater number of features meet the query criteria, only a single page is returned along a URI specified in the "next" attribute to retrieve the next page. Clients can follow this URI to obtain the following page of results. Subsequent responses will contain URIs to subsequent response pages until no pages are left.

2.3.4 Catalogue Services for the Web

OGC Catalogue Services support registration and discovery of services (as well as data sets and other information sets) and allow searching for these registered objects via their metadata (OGC 2007). Catalogue Services for the Web (CSW) is an HTTP-bound interface to Catalogue Services and is described in the Catalogue Service Implementation Specification (OGC 2007). The information in this section is an overview of the CSW interface from this implementation specification. As with WFS, CSW requests can be made through HTTP GET via KVP encoding, POST via XML encoding, or through SOAP. The information in this section refers specifically to HTTP GET requests with KVP parameter encoding, though it is generally applicable to the HTTP POST and SOAP methods as well.

The Catalogue Service Implementation specification describes several possible operations for a CSW service: *GetCapabilities, DescribeRecord, GetDomain, GetRecords, GetRecordById, Transaction,* and *Harvest.* This section describes the four required operations: GetCapabilities, DescribeRecord, GetRecords, and GetRecordById. Every CSW operation has three required parameters: “request” whose value is the name of the desired operation (such as “GetRecords”), “service” whose value is always “CSW”, and “version” whose value is always “2.0.2”.

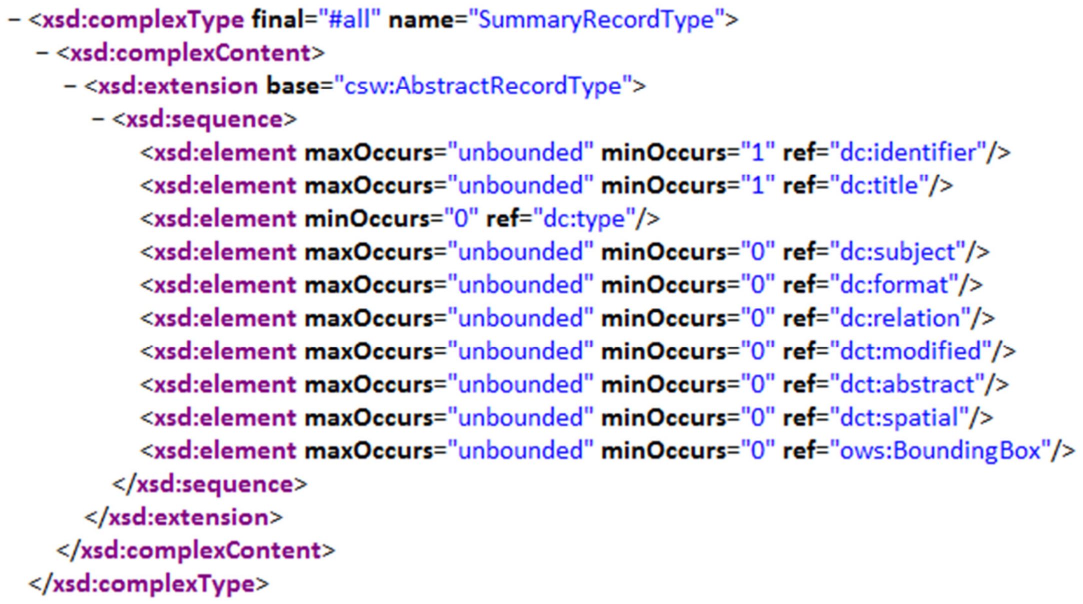
As with every OGC service, the GetCapabilities operation provides service-level metadata about the CSW implementation. This metadata includes the title, abstract, keywords, fees, contact information, and access constraints of the service, as well as which of the CSW operations are supported and the types of filters that may be used for the GetRecords operation. The response from a CSW GetCapabilities request is an XML document whose schema is described in the Catalogue Services Implementation Specification.

The core element of CSW is the *record*. A record contains information about a registered object. The Catalogue Services specification gives 11 core queryable properties for CSW records. These properties are “Subject”, “Title”, “Abstract”, “AnyText”, “Format”, “Identifier”, “Modified”, “Type”, “BoundingBox”, “CRS”, and “Association”. Record properties are extendable through specialized *Application Profiles* which may specify additional queryable properties. For example, the FGDC Content Standard for Digital Geospatial Metadata Application Profile candidate specification (OGC 2006) adds the “ThemeKeywords”, “BeginDate”, and “EndDate” queryable properties (among several others). It is up to the CSW implementation which properties are supported both from the core list and from any Application Profiles.

The DescribeRecord operation of CSW is similar to the GetFeatureInfo operation of WFS. This method gives the schema for records served from the target CSW endpoint. Records can be retrieved with different levels of detail, which are described by the schema from DescribeRecord. The typical levels are usually named “BriefRecord”, “SummaryRecord” and “Record”. The following is an example DescribeRecord request to the ESRI GeoPortal CSW endpoint hosted at CRWR:

```
https://hydroportal.crwr.utexas.edu/geoportal/csw/discovery?
      request=DescribeRecord&
      service=CSW&
      version=2.0.2
```

Figure 21 shows the SummaryRecord schema excerpted from the full response to the preceding request. The SummaryRecord from this CSW has 10 properties, as indicated by each of the “xsd:element” lines, in addition to those inherited from the base AbstractRecordType indicated by the “xsd:extension” line.



```
- <xsd:complexType final="#all" name="SummaryRecordType">
  - <xsd:complexContent>
    - <xsd:extension base="csw:AbstractRecordType">
      - <xsd:sequence>
          <xsd:element maxOccurs="unbounded" minOccurs="1" ref="dc:identifier"/>
          <xsd:element maxOccurs="unbounded" minOccurs="1" ref="dc:title"/>
          <xsd:element minOccurs="0" ref="dc:type"/>
          <xsd:element maxOccurs="unbounded" minOccurs="0" ref="dc:subject"/>
          <xsd:element maxOccurs="unbounded" minOccurs="0" ref="dc:format"/>
          <xsd:element maxOccurs="unbounded" minOccurs="0" ref="dc:relation"/>
          <xsd:element maxOccurs="unbounded" minOccurs="0" ref="dct:modified"/>
          <xsd:element maxOccurs="unbounded" minOccurs="0" ref="dct:abstract"/>
          <xsd:element maxOccurs="unbounded" minOccurs="0" ref="dct:spatial"/>
          <xsd:element maxOccurs="unbounded" minOccurs="0" ref="ows:BoundingBox"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```

Figure 21: Example SummaryRecord schema from DescribeRecord response

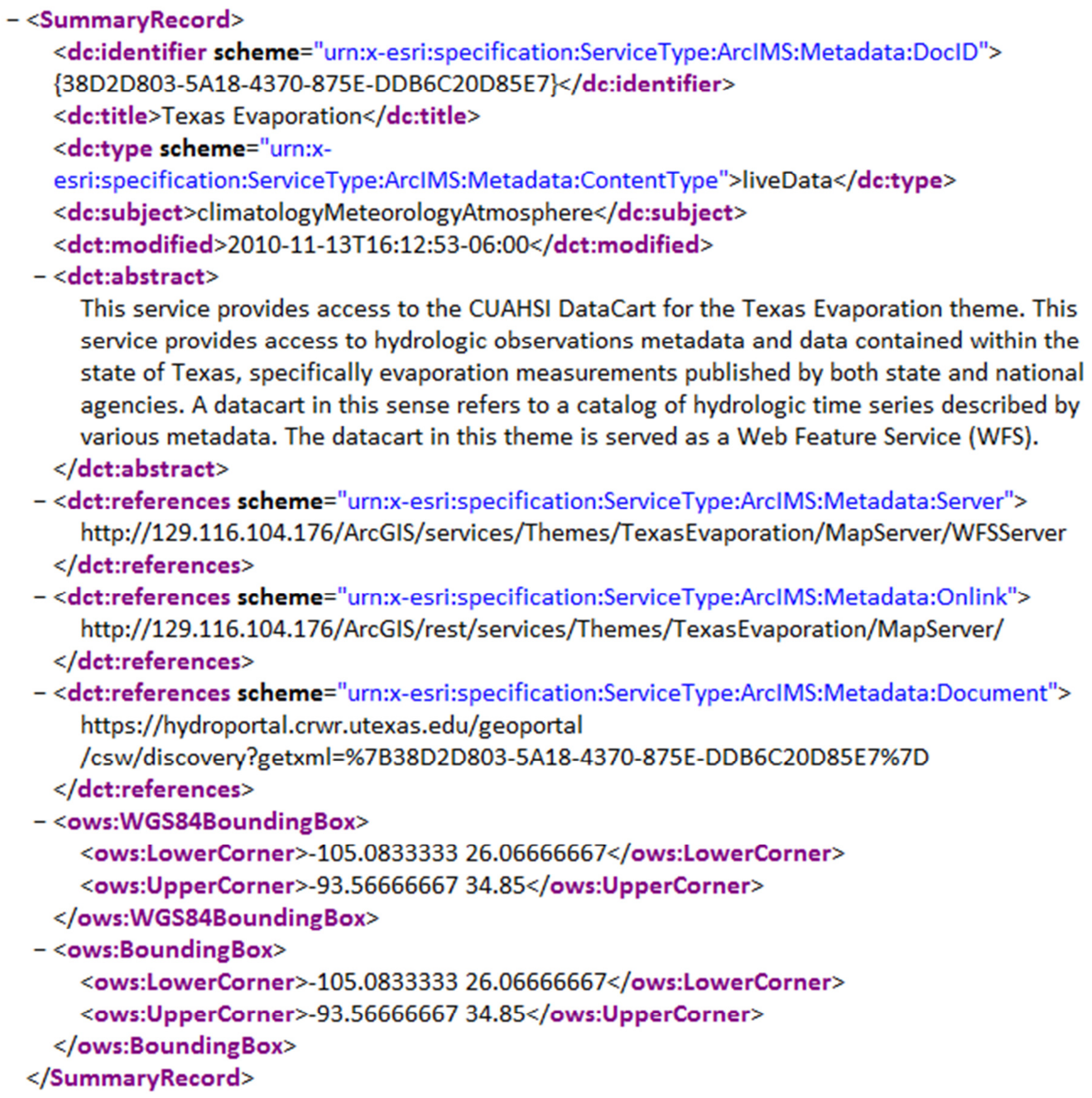
The GetRecords operation is the core method of CSW. This method returns all records from the CSW that match the criteria set by the request parameters. The response is an XML document whose base schema is defined in the Catalogue Services Implementation Specification, with the record schema extended by that provided from the DescribeRecord method. In addition to the three CSW-required parameters, there are 16 other possible parameters for the GetRecords operation described in the implementation specification. The most relevant parameters are “resultType”, “maxRecords”, “elementSetName”, “CONSTRAINTLANGUAGE”, and “constraint”. “resultType” can take the values “hits” which just returns the number of matching records, “results” which returns the result set, or “validate” which returns whether the request is valid. “maxRecords” takes a numeric value and is used to specify the maximum number of records that the response can contain. “elementSetName” is used to specify which level of detail of records should be returned: “brief”, “summary”, or “full”, with “summary” being the default value.

GetRecords queries can be constrained through either Common Query Language or through Filters. The value of the “CONSTRAINTLANGUAGE” parameter can be either “CQL_TEXT” or “FILTER” corresponding to the two possible types of constraint queries. The “constraint” parameter contains the actual query string in either CQL or FES (described in section 2.3.6 of this thesis).

The following is an example GetRecords request (with no constraints specified) to the ESRI GeoPortal CSW endpoint at CRWR:

```
https://hydroportal.crwr.utexas.edu/geoportal/csw/discovery?
      request=GetRecords&
      service=CSW&
      version=2.0.2&
      resultType=results
```

Figure 21 shows an example SummaryRecord from the result set returned by the above call.



```
- <SummaryRecord>
    <dc:identifier scheme="urn:x-esri:specification:ServiceType:ArcIMS:Metadata:DocID">
    {38D2D803-5A18-4370-875E-DDB6C20D85E7}</dc:identifier>
    <dc:title>Texas Evaporation</dc:title>
    <dc:type scheme="urn:x-
    esri:specification:ServiceType:ArcIMS:Metadata:ContentType">liveData</dc:type>
    <dc:subject>climatologyMeteorologyAtmosphere</dc:subject>
    <dct:modified>2010-11-13T16:12:53-06:00</dct:modified>
  - <dct:abstract>
      This service provides access to the CUAHSI DataCart for the Texas Evaporation theme. This
      service provides access to hydrologic observations metadata and data contained within the
      state of Texas, specifically evaporation measurements published by both state and national
      agencies. A datacart in this sense refers to a catalog of hydrologic time series described by
      various metadata. The datacart in this theme is served as a Web Feature Service (WFS).
    </dct:abstract>
  - <dct:references scheme="urn:x-esri:specification:ServiceType:ArcIMS:Metadata:Server">
      http://129.116.104.176/ArcGIS/services/Themes/TexasEvaporation/MapServer/WFSServer
    </dct:references>
  - <dct:references scheme="urn:x-esri:specification:ServiceType:ArcIMS:Metadata:Onlink">
      http://129.116.104.176/ArcGIS/rest/services/Themes/TexasEvaporation/MapServer/
    </dct:references>
  - <dct:references scheme="urn:x-esri:specification:ServiceType:ArcIMS:Metadata:Document">
      https://hydroportal.crwr.utexas.edu/geoportal
      /csw/discovery?getxml=%7B38D2D803-5A18-4370-875E-DDB6C20D85E7%7D
    </dct:references>
  - <ows:WGS84BoundingBox>
      <ows:LowerCorner>-105.0833333 26.06666667</ows:LowerCorner>
      <ows:UpperCorner>-93.56666667 34.85</ows:UpperCorner>
    </ows:WGS84BoundingBox>
  - <ows:BoundingBox>
      <ows:LowerCorner>-105.0833333 26.06666667</ows:LowerCorner>
      <ows:UpperCorner>-93.56666667 34.85</ows:UpperCorner>
    </ows:BoundingBox>
  </SummaryRecord>
```

Figure 22: SummaryRecord from GetRecords response

The CSW GetRecordById operation is for returning specific records referenced by their identifier string. A comma-separated list of identifier strings are passed to this operation via the "id" parameter. Additionally, as with GetRecords the "elementSetName" parameter can be used to specify which level of detail of records should be returned: “brief”, “summary”, or “full”. The following example GetRecordById request returns the same SummaryRecord as the example GetRecords request above:

```
https://hydroportal.crwr.utexas.edu/geoportal/csw/discovery?
      request=GetRecordById&
      service=CSW&
      version=2.0.2&
      id={38D2D803-5A18-4370-875E-DDB6C20D85E7}
```

2.3.5 Sensor Observation Service

The Sensor Observation Service (SOS) specification is one of the major components of the OGC’s Sensor Web Enablement (SWE) family of standards (OGC 2007). The SOS specification defines methods for accessing field-deployed sensors and retrieving sets of observations data from them. The full SOS specification is detailed in the OGC’s Sensor Observation Service Implementation Specification (2007), and the information in this section is an overview from that document. SOS requests can be made through HTTP GET via KVP encoding or through HTTP POST via XML encoding. The information in this section refers to HTTP GET requests with KVP parameter encoding, though it is generally applicable to the HTTP POST method as well.

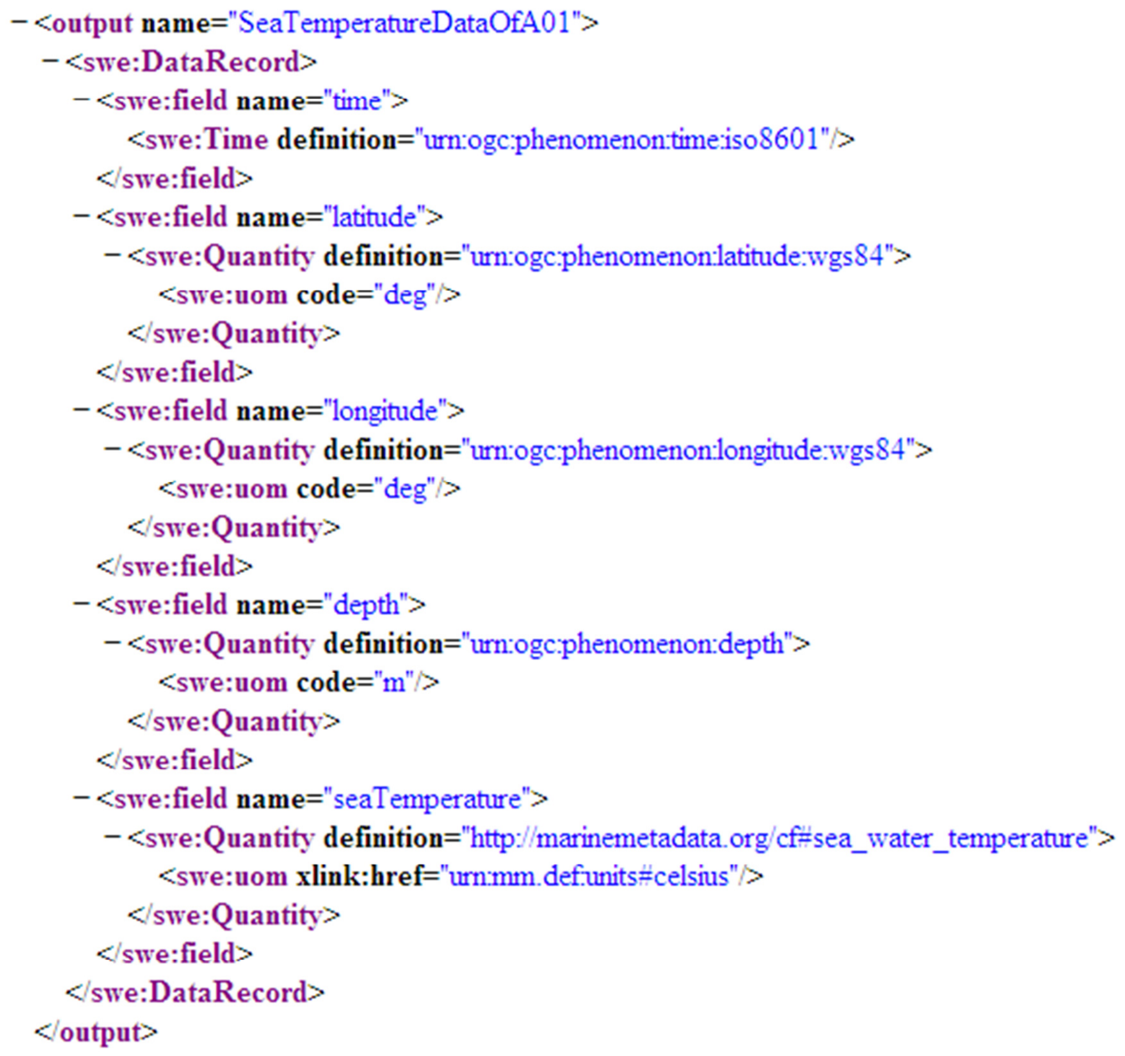
At a minimum, an SOS implementation must provide three operations: *GetCapabilities, DescribeSensor,* and *GetObservation*. Several other optional SOS operations are detailed in the implementation specification. Note that although SOS is for a sensor network, it is not a requirement for an SOS to actually refer to physical sensors. In that respect, any observations data source could be considered a “sensor.”

The OGC standard GetCapabilities operation returns service-level metadata about the SOS endpoint. The required parameters for the SOS GetCapabilities request are “service” whose value is “SOS” and “request” whose value is “GetCapabilities”. The response is an XML document whose schema is described in the SOS implementation specification. The metadata returned includes, as with the other service specifications, the service title, abstract, keywords, fees, usage constraints and other information. For SOS, the GetCapabilities response also describes which of the SOS operations are supported, the types of filters that may be used for the GetObservation operation, and the *observation offerings* of the service. Observation offerings are groupings of related observations, and their identification strings (returned from GetCapabilities) are used in the GetObservation operation.

The SOS DescribeSensor operation provides detailed metadata about the sensors represented in the SOS instance. Responses from this method are typically formatted as SensorML, an OGC customization of XML made to describe sensors and their capabilities. There are four parameters for a DescribeSensor request, all of which are required: “service” whose value is “SOS”, “request” whose value is “DescribeSensor”, “sensorId” whose value is an identification string of an observation offering, and “outputFormat” whose value describes the desired response format. The following is a DescribeFeature request to the SOS endpoint from the Gulf of Maine Ocean Observing System (GOMOOS):

```
http://www.gomoos.org/cgi-bin/sos/oostethys_sos.cgi?
      service=SOS&
      request=DescribeSensor&
      sensorId=A01&
      outputFormat=text/xml;subtype="sensorML/1.0.1"
```

Figure 23 shows an excerpt of the SensorML response from the preceding request containing metadata describing sea water temperature observations available from the “A01” sensor.



```
- <output name="SeaTemperatureDataOfA01">
  - <swe:DataRecord>
    - <swe:field name="time">
        <swe:Time definition="urn:ogc:phenomenon:time:iso8601"/>
      </swe:field>
    - <swe:field name="latitude">
      - <swe:Quantity definition="urn:ogc:phenomenon:latitude:wgs84">
          <swe:uom code="deg"/>
        </swe:Quantity>
      </swe:field>
    - <swe:field name="longitude">
      - <swe:Quantity definition="urn:ogc:phenomenon:longitude:wgs84">
          <swe:uom code="deg"/>
        </swe:Quantity>
      </swe:field>
    - <swe:field name="depth">
      - <swe:Quantity definition="urn:ogc:phenomenon:depth">
          <swe:uom code="m"/>
        </swe:Quantity>
      </swe:field>
    - <swe:field name="seaTemperature">
      - <swe:Quantity definition="http://marinemetadata.org/cf#sea_water_temperature">
          <swe:uom xlink:href="urn:mm.def:units#celsius"/>
        </swe:Quantity>
      </swe:field>
    </swe:DataRecord>
  </output>
```

Figure 23: SensorML excerpt from DescribeRecord request

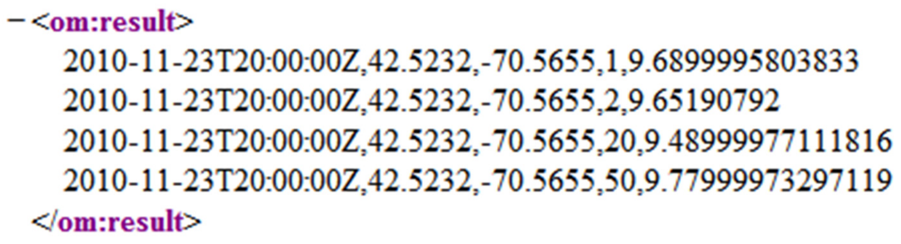
The core method from SOS is GetObservation. The response from this operation is an O&M-based document containing the requested observations data. The GetObservation method has five required parameters: "service" whose value is "SOS", "version" whose value is "1.0.0", "request" whose value is "GetObservation", "offering" whose value is an ID of one or more of the offerings obtained from GetCapabilities, "observedProperty" whose value is one or more of the properties obtained from

GetCapabilities, and "responseFormat" whose value is the desired encoding type of the response. Other parameters that can be used to constrain the observations values returned from GetObservation are described in the SOS implementation specification.

The following is an example GetObservation request to the GOMOOS SOS endpoint for sea water temperature from the "A01" offering:

```
http://www.gomoos.org/cgi-bin/sos/oostethys_sos.cgi?
        service=SOS&
        version=1.0.0&
        request=GetObservation&
        offering=A01&
        observedProperty=sea_water_temperature&
        responseFormat=text/xml;subtype="om/1.0.0"
```

Figure 24 shows the "result" element of the O&M-format response obtained from the preceding GetObservation request. This element contains a time series of observations data, with each entry containing a timestamp, latitude, longitude, depth, and temperature. The format of each entry, including units, is also part of the O&M-format response.



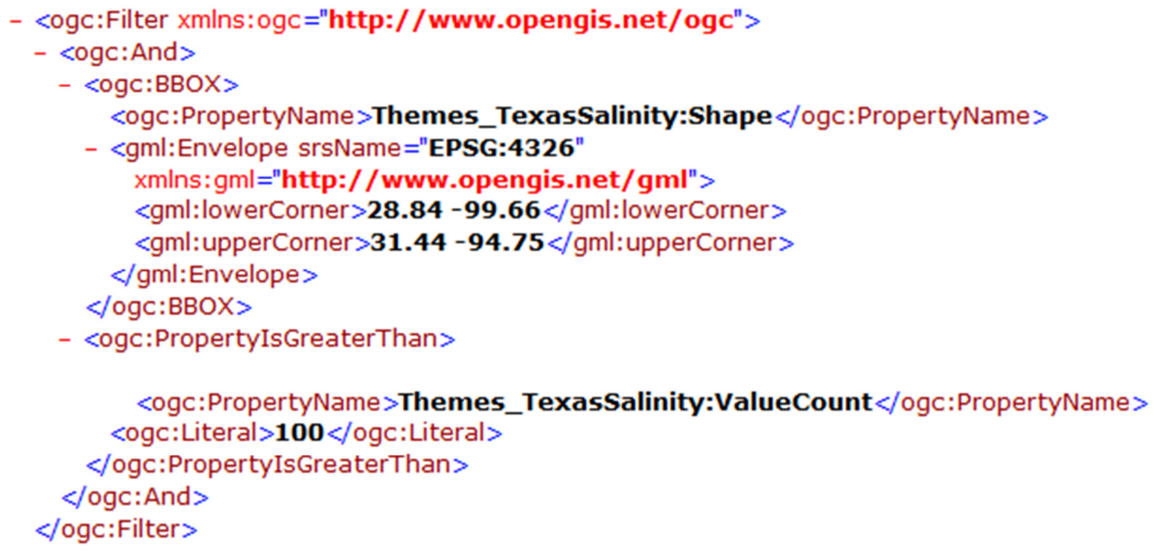
```
- <om:result>
    2010-11-23T20:00:00Z,42.5232,-70.5655,1,9.6899995803833
    2010-11-23T20:00:00Z,42.5232,-70.5655,2,9.65190792
    2010-11-23T20:00:00Z,42.5232,-70.5655,20,9.48999977111816
    2010-11-23T20:00:00Z,42.5232,-70.5655,50,9.77999973297119
  </om:result>
```

Figure 24: Result element from GetObservation request

2.3.6 Filter Encoding Standard

The Filter Encoding Implementation Specification describes an XML-based language called Filter Encoding Standard (FES) for adding constraints (i.e. *filters*) to the query methods of some OGC services (OGC 2005). The FES is an XML-based representation of OGC's Common Query Language, which is defined in the Catalog Services Implementation Specification. Filters can be used on the WFS GetFeature and CSW GetRecords operations to constrain the result sets from these operations.

The current (1.1.0) version of the FES describes four classes of filter operators: spatial, comparison, logical, and arithmetic. The upcoming 2.0 version of FES also includes support for temporal operators. Filter operators evaluate whether properties of the possible return set of the target operation meet the constraints set forth by the operator clauses. Spatial operators include BBOX for specifying a bounding box. Comparison operators include typical operations such as less-than, greater-than, and equal-to. The logical operators, such as "And" and "Or", are used to combine spatial and comparison operator clauses. An example filter with both a spatial operator ("BBOX" acting on the "Shape" property) and a comparison operator ("PropertyIsGreaterThan" acting on the "ValueCount" property) is shown in Figure 25.



```
- <ogc:Filter xmlns:ogc="http://www.opengis.net/ogc">
  - <ogc:And>
    - <ogc:BBOX>
        <ogc:PropertyName>Themes_TexasSalinity:Shape</ogc:PropertyName>
      - <gml:Envelope srsName="EPSG:4326"
          xmlns:gml="http://www.opengis.net/gml">
          <gml:lowerCorner>28.84 -99.66</gml:lowerCorner>
          <gml:upperCorner>31.44 -94.75</gml:upperCorner>
        </gml:Envelope>
      </ogc:BBOX>
    - <ogc:PropertyIsGreaterThan>

          <ogc:PropertyName>Themes_TexasSalinity:ValueCount</ogc:PropertyName>
        <ogc:Literal>100</ogc:Literal>
      </ogc:PropertyIsGreaterThan>
    </ogc:And>
  </ogc:Filter>
```

Figure 25: Filter example with spatial (BBOX) and comparison operators

2.3.7 WaterML2.0

The OGC has created XML-based standards for encoding the responses from its services. GML is used to describe features returned from the WFS GetFeature method. The elements in a GML document reflect the attributes of interest in a WFS and are specified by the provider of the service. The O&M format is used to return observations data from the SOS GetObservation method. These data encodings are not as strict as CUAHSI's WaterML specification, and may be extended or specialized to meet requirements of the service providers and consumers. Specializations of O&M are referred to as *profiles* and define agreed upon practices for encoding observations data for a particular domain.

The WaterML2.0 profile for O&M is currently being developed by the OGC and partner organizations, including CUAHSI. WaterML2.0 takes the framework and lessons-learned from CUAHSI's original WaterML specification and harmonizes the structure to be consistent with OGC standards. The WaterML2.0 specification is in the draft stage as of the writing of this thesis.

Chapter 3: The CUAHSI HIS Architecture

3.1 Overview

The current CUAHSI HIS architectural model, termed the Network-Observations Model, is a SOA with centralized metadata and distributed data. This system is empowered primarily by CUAHSI's WaterML data encoding and WOF web services. This chapter provides an overview of the intellectual basis of Network-Observations model. Each of the three major components of the HIS architecture are described in detail and their roles specified. The various client-centric operating models that have resulted from the Network-Observations Model are also defined and explained. Finally, the issues that have arisen from the current architecture are enumerated.

3.2 The Network-Observations Model

The architectural model of the CUAHSI HIS is heavily influenced by the organization of the CUAHSI Observations Data Model. In the ODM, data values (observations) are at the center of the data model, and are surrounded by metadata that unambiguously describe them (Tarboton, Horsburgh and Maidment 2007). The metadata that describe the data values can be arranged into a *SeriesCatalog* view. Each series in the SeriesCatalog is a unique collection of time-indexed observations of a given variable at a specific site. HIS Central has implemented a centralized series catalog of similar structure to that of the ODM. The HIS Central metadata catalog contains the series metadata from all registered HydroServers.

The current model that the HIS architecture follows has been termed the *Network-Observations Model*. The fundamental piece of information in this model is a hydrologic observation, or value. *Observations* are differentiated by the metadata that describe them, such as site location, sample medium, or quality control level. These metadata come primarily from the other levels of the Network-Observations Model, *Network*, *Site*, and *Variable*, as shown in Figure 26. This data model is built upon and consistent with the principles behind the CUAHSI Observations Data Model.

.

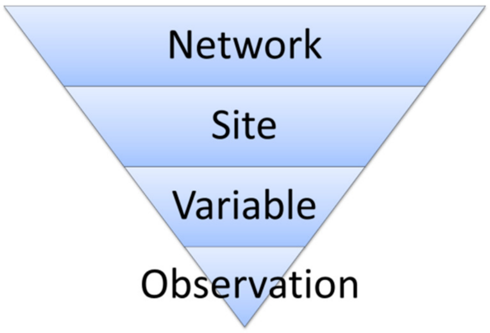


Figure 26: The Network-Observations Model

The Network-Observations Model with the USGS NWIS Daily Values service as an example is shown in Figure 27. Each HydroServer provides access to one or more Networks through WOF WSDL files. A Network contains many sites, accessed by the GetSites method. The series catalog of Variables measured at each Site is accessed via GetSiteInfo. Metadata about Variables is retrieved from GetVariableInfo. Finally, observation values are obtained through GetValues, with the site, variable, and time range specified via parameters to the service, the network identified through prefixes to those parameters, and the service specified by the address to the WOF service.

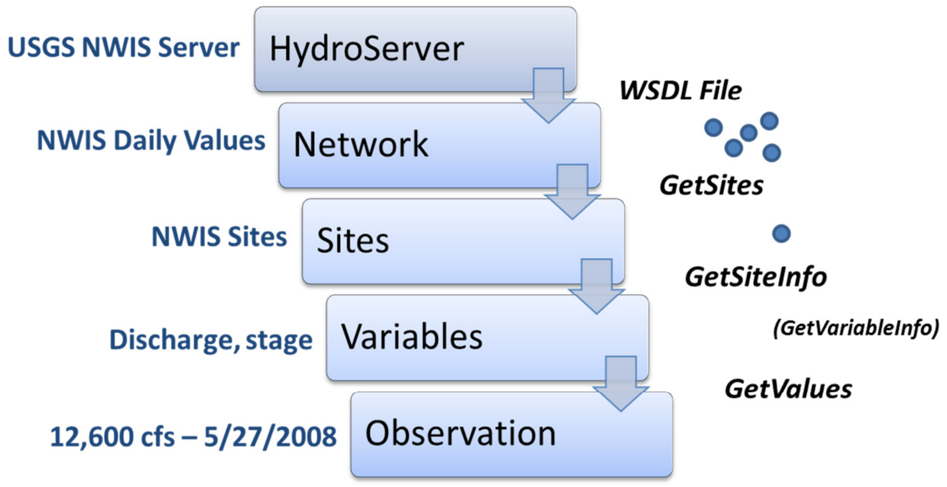


Figure 27: Network-Observations Model hierarchy with example

3.3 HIS Components

3.3.1 Introduction

HydroServers provide data access, HS Central provides data discovery, and the system is integrated with WaterML and web services.

The CUAHSI HIS architecture comprises three major components: HydroServers, HIS Central, and client applications. These components' primary interactions are shown in Figure 28. HIS Central harvests series metadata from HydroServers to create its searchable series catalog. Clients use the HIS Central search web services to find HydroServers with series that match given parameters. Clients download the actual hydrologic time series data from the corresponding HydroServers. Each of the three components and their interactions are described in detail in this section.

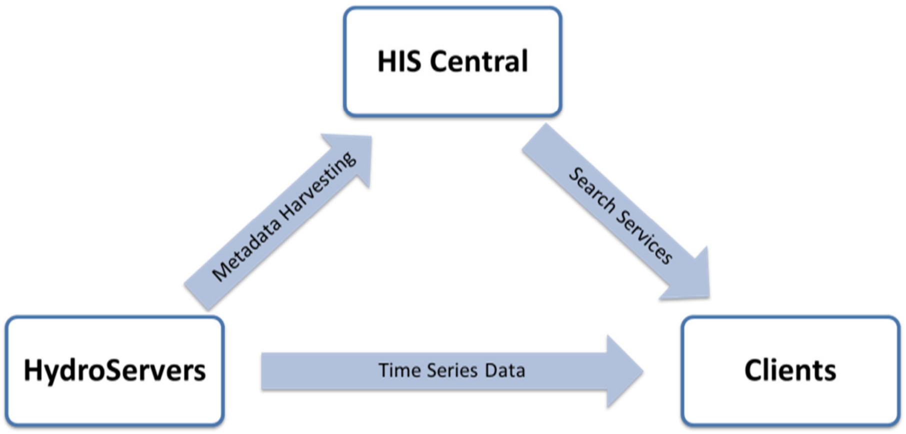


Figure 28: Network-Observations components

3.3.2 HydroServers

HydroServers are the data providers of the HIS SOA. They provide both the data and metadata about hydrologic observations. The conceptual definition of a HydroServer within the Network-Observations Model is simply a web server that provides access to WaterML-encoded hydrologic data and metadata through the WOF web services. A HydroServer can provide access to multiple WOF networks, which each have their own WSDL file address on the web server and network identifier code within the data source. For example, the USGS NWIS HydroServer has separate networks for its Daily Values, Ground Water, Instantaneous Irregular Data, and Unit Values services. The collection of web-accessible HydroServers around the country comprises one component of the CUAHSI HIS. Data providers register their HydroServers at HIS Central so that their data series can be discovered by clients of the HIS.

Generically, a HydroServer has four main components: a database containing the hydrologic data, a database server to interact with the database, a web server to provide web-based access to the service, and an implementation of the WOF specification. These components are illustrated in Figure 29. The box surrounding the generic HydroServer diagram indicates that only the network WSDL files are accessible from outside the HydroServer, with the underlying components essentially a "black-box" to clients.

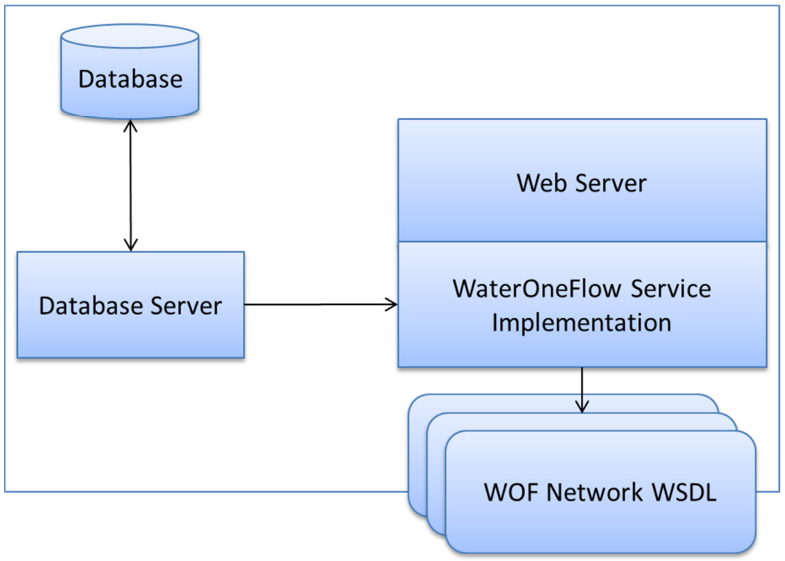


Figure 29: Generic HydroServer diagram

In practice, there have been multiple methods for how the HydroServer concept has been implemented. First, there is the *off-the-shelf ODM* method, which is the one most supported by the HIS project. This method of deploying a HydroServer uses a series of tools that the HIS project has developed to leverage commercial off-the-shelf software components along with an ODM database. This method has primarily been used by smaller data providers (typically for networks at local or regional scales) that do not already have any data access services online. Examples of services using this method are the Texas Instream Flow Lower Sabine service[1] and the Dry Creek Experimental Watershed ODMDCEW2 service[2], among several others.

---

[1] http://his.crwr.utexas.edu/SabineBio/cuahsi_1_0.asmx?WSDL
[2] http://icewater.boisestate.edu/dcew2dataservices/cuahsi_1_0.asmx?WSDL

A schematic representation of a HydroServer built from this method is depicted in Figure 30. In the off-the-shelf ODM HydroServer, hydrologic data are loaded into an ODM-schema database managed by a Microsoft SQL Server instance. The data loading can be accomplished with CUAHSI's ODM Data Loader program or with custom transformation scripts. The *ODM WaterOneFlow Services* application, which is an ASP .NET 2.0 implementation of WOF provided by the HIS project, is installed on a Windows Server with connection information for the SQL Server supplied in configuration settings. The ODM WaterOneFlow Services code supplies the WSDL files and WOF methods for accessing the data networks stored in the underlying ODM database.

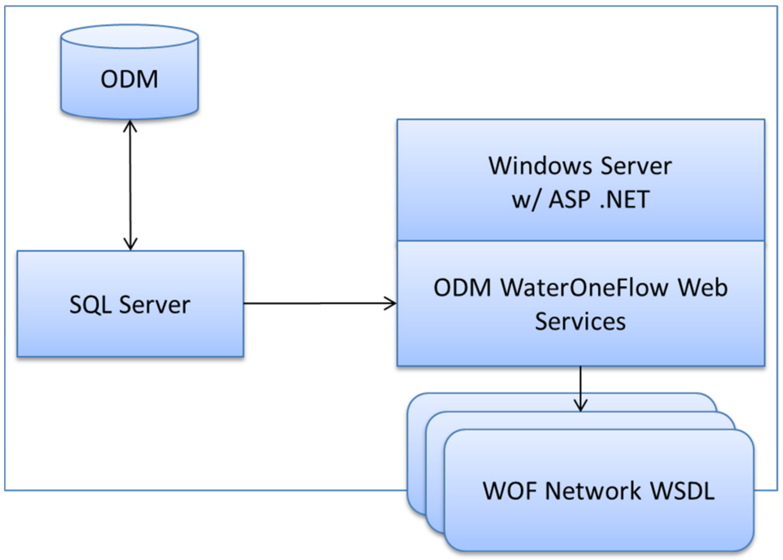


Figure 30: HydroServer from off-the-shelf components

A second method of implementing a HydroServer is the *ODM-View* method, depicted in Figure 31. This method is very similar to the off-the-shelf method, with the only difference being the underlying database schema. Rather than loading the hydrologic data into an ODM-schema database, a custom *view* is used instead. Essentially, a view virtually maps the tables and columns of a source database schema to mirror a different schema. In this case, a view mirroring the ODM schema is created over an agency's propriety database schema. This method has been used on the experimental TWDB Groundwater service hosted at CRWR.[3]

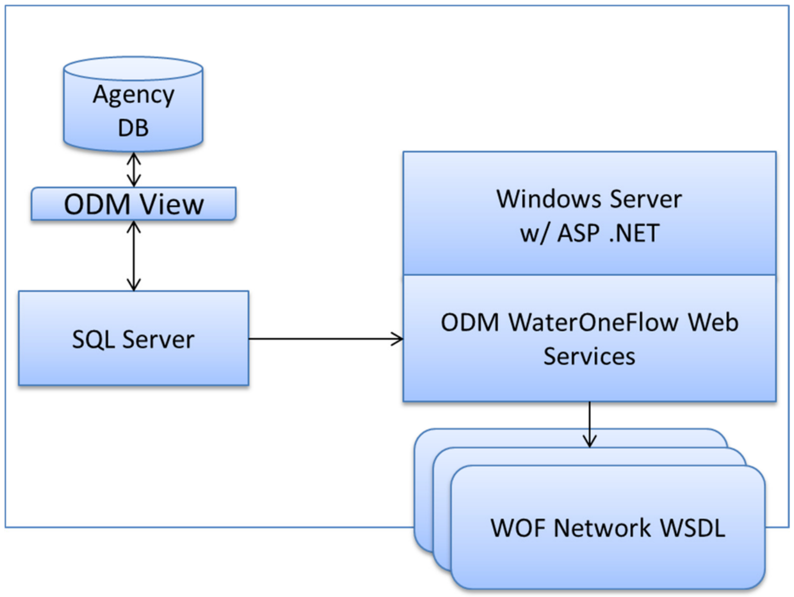


Figure 31: ODM-View HydroServer diagram

Implementing a HydroServer with completely different off-the-shelf components from the two previous methods is also an option. This method makes sense when a data provider either already has or prefers a data infrastructure using different components from those supported in the off-the-shelf ODM or ODM-view methods. For example, a data provider might want to use a MySQL database server with an Apache web server running PHP code. In this case, a custom implementation of the WOF specification

[3] Not publicly-accessible at this time.

would just need to expose the appropriate WSDL files and web service methods to outside clients. This method has been used by CRWR to publish hydrologic data from the Texas Coastal Ocean Observing Network (TCOON).[4]

For the three large federal hydrologic data providers (USGS, NCDC, and EPA), a quite complex implementation of the HydroServer server concept is used: the *hybrid* method. In these HydroServers, the data providers transmit copies of their metadata databases (or just the relevant tables) to the SDSC HIS team at intermittent intervals. The SDSC team migrates these databases from their native format (such as Oracle) into SQL Server, following the originally-provided schema. Once the SQL Server migration is complete, custom ODM views are placed on top of the native schema. A customized version of the ODM WaterOneFlow Web Service software installed over the ODM views to provide the three WOF metadata methods, GetSites, GetSiteInfo, and GetVariableInfo. The data method, GetValues, is implemented as a "pass-through" service. In the case of the USGS NWIS, for example, GetValues requests are translated by the custom WOF Web Service software into requests to the USGS's WaterML service. The response from the USGS service is then routed back through HIS Central's HydroServer to the requesting client. This type of HydroServer is illustrated in Figure 32.

[4] http://his.crwr.utexas.edu/tcoonts/tcoon.asmx?WSDL

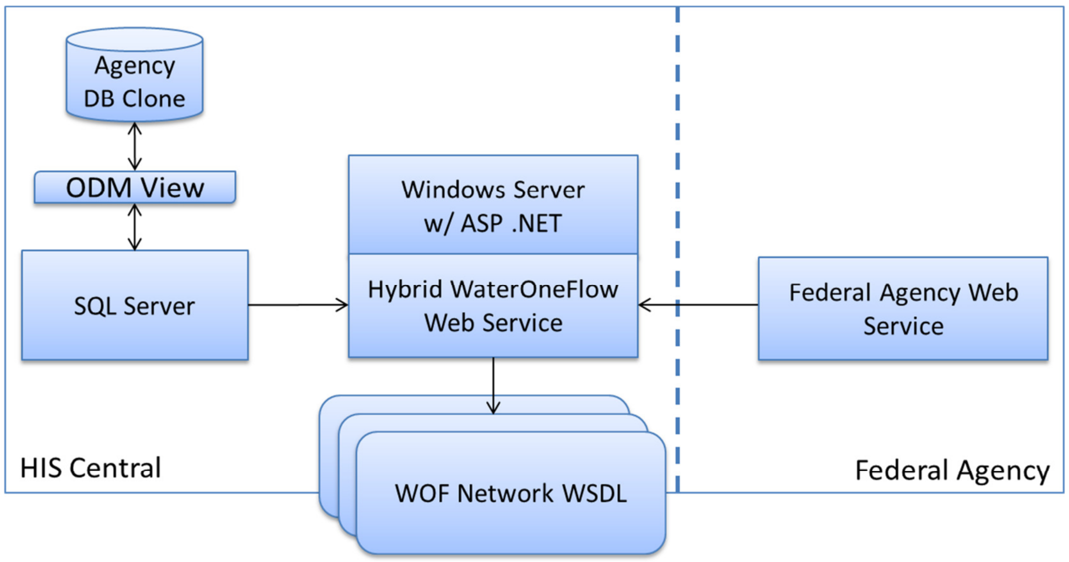

Agency DB Clone
ODM View
SQL Server
Windows Server w/ ASP .NET
Hybrid WaterOneFlow Web Service
Federal Agency Web Service
WOF Network WSDL
HIS Central
Federal Agency

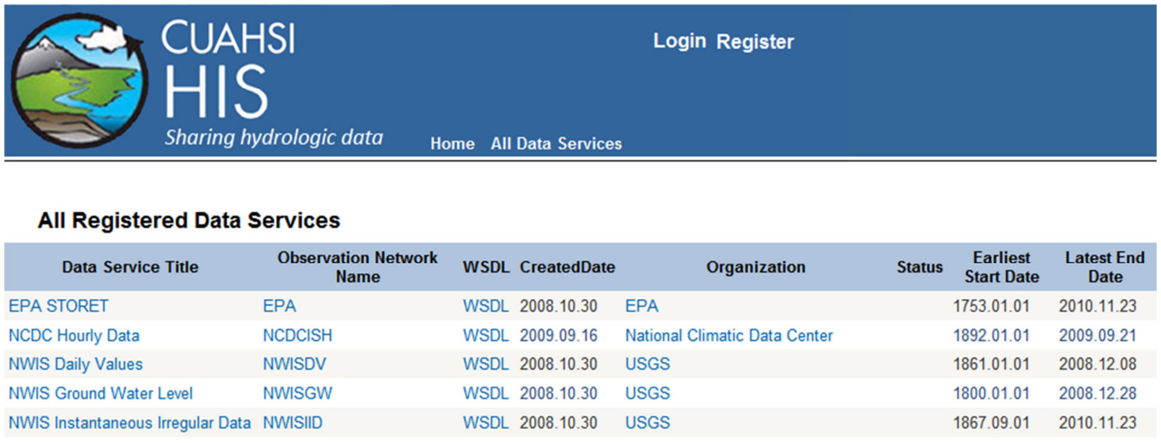
Figure 32: Hybrid HydroServer diagram

3.3.3 HIS Central

HIS Central is the central cataloguing system of HIS. Unlike the HydroServers and client components of the CUAHSI HIS, there is only a single HIS Central within the CUAHSI HIS. HIS Central itself comprises three main components: a registry of WOF services, the Metadata Catalog, and the Hydrologic Concept Ontology

The registry of WOF services is the system through which data providers or managers can register the WSDL addresses of the networks within their HydroServer services. Data providers can create accounts at the HIS Central website[5] and then login to register and manage their service metadata. The HIS Central registry currently has 62 registered WOF services at the time of writing this thesis. A screen capture of the service registry is shown in Figure 33.

[5] http://hiscentral.cuahsi.org/



All Registered Data Services

Data Service Title	Observation Network Name	WSDL	CreatedDate	Organization	Status	Earliest Start Date	Latest End Date
EPA STORET	EPA	WSDL	2008.10.30	EPA		1753.01.01	2010.11.23
NCDC Hourly Data	NCDCISH	WSDL	2009.09.16	National Climatic Data Center		1892.01.01	2009.09.21
NWIS Daily Values	NWISDV	WSDL	2008.10.30	USGS		1861.01.01	2008.12.08
NWIS Ground Water Level	NWISGW	WSDL	2008.10.30	USGS		1800.01.01	2008.12.28
NWIS Instantaneous Irregular Data	NWISIID	WSDL	2008.10.30	USGS		1867.09.01	2010.11.23

Figure 33: HIS Central Service Registry

The HIS Central Metadata Catalog, whose schema is shown in Figure 34, is a database of metadata for each network of each registered HydroServer. The most used table of the Metadata Catalog is the SeriesCatalog table, which has been highlighted by a red box in the figure.

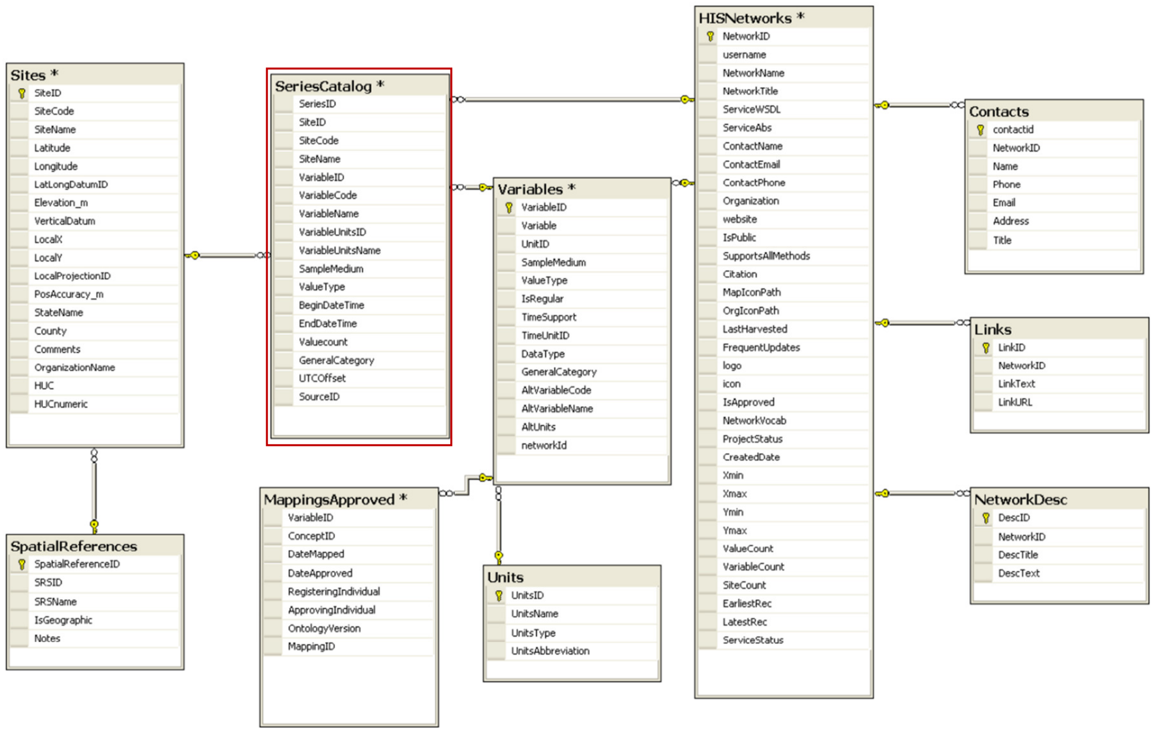


Figure 34: HIS Central Metadata Catalog schema (Whitenack 2010)

HIS Central's Metadata Catalog contains metadata for each time series of each network of each registered HydroServer. The metadata for this catalog is obtained through harvesting procedures. There are two main ways that the harvesting of metadata for cataloging at HIS Central is accomplished. The first way of harvesting metadata is through WOF services from HydroServers. This harvesting is done by a custom program at HIS Central called Web Service Harvester (Whitenack 2010). Approximately once each week, Web Service Harvester connects to registered WOF services. All sites from each network are retrieved using the WOF GetSites method, and for each side, GetSiteInfo is called to retrieve the series catalog. The HIS Central series catalog is updated to include any new or modified series metadata found from these results

(Whitenack 2010). This method is used for almost all of the registered HydroServers, except for the three hybrid federal HydroServers.

The other method of harvesting metadata is through direct database connections to the ODM views on SQL Server-migrated federal database dumps. This harvesting is done by another custom program, the Federal Repository Catalog Harvester, at HIS Central. Rather than running on a defined schedule, this harvester program is only run when a new data dump is received from a federal source and migrated.

To facilitate searching and organization of registered services, the data providers are required to *tag* the variables that their services provide with concepts from the Hydrologic Concept Ontology after metadata harvesting has occurred. This concept tagging mitigates the problem of varied names for what essentially are the same type of observation (e.g., "stream discharge" being referred to as "flow," "runoff," or other similar terms). The Hydrologic Concept Ontology is stored in a database, and is web-accessible for data providers to tag their HydroServer's variables with the appropriate ontological concepts for discovery. The Hydrologic Concept Ontology's tables are shown in Figure 35. Tagging is done through the HydroTagger web application, shown in Figure 36.

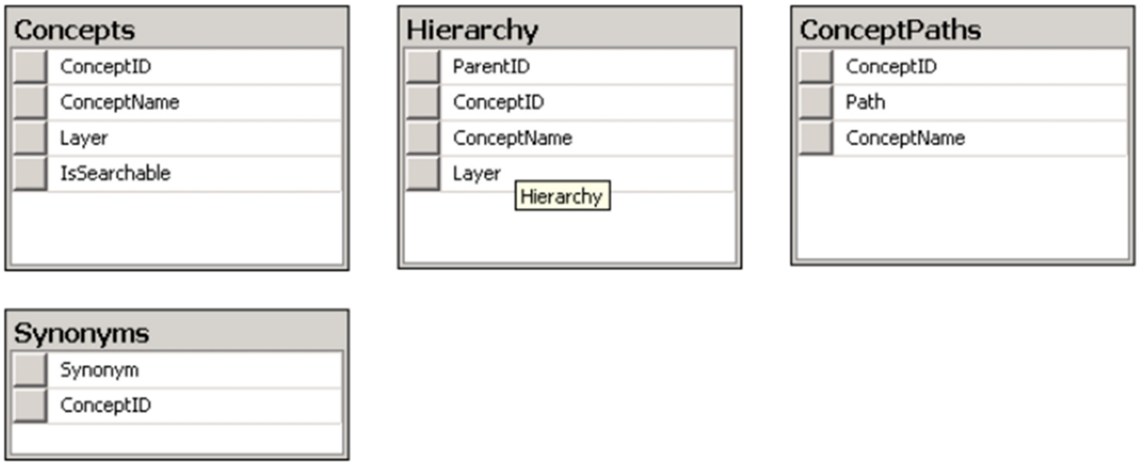
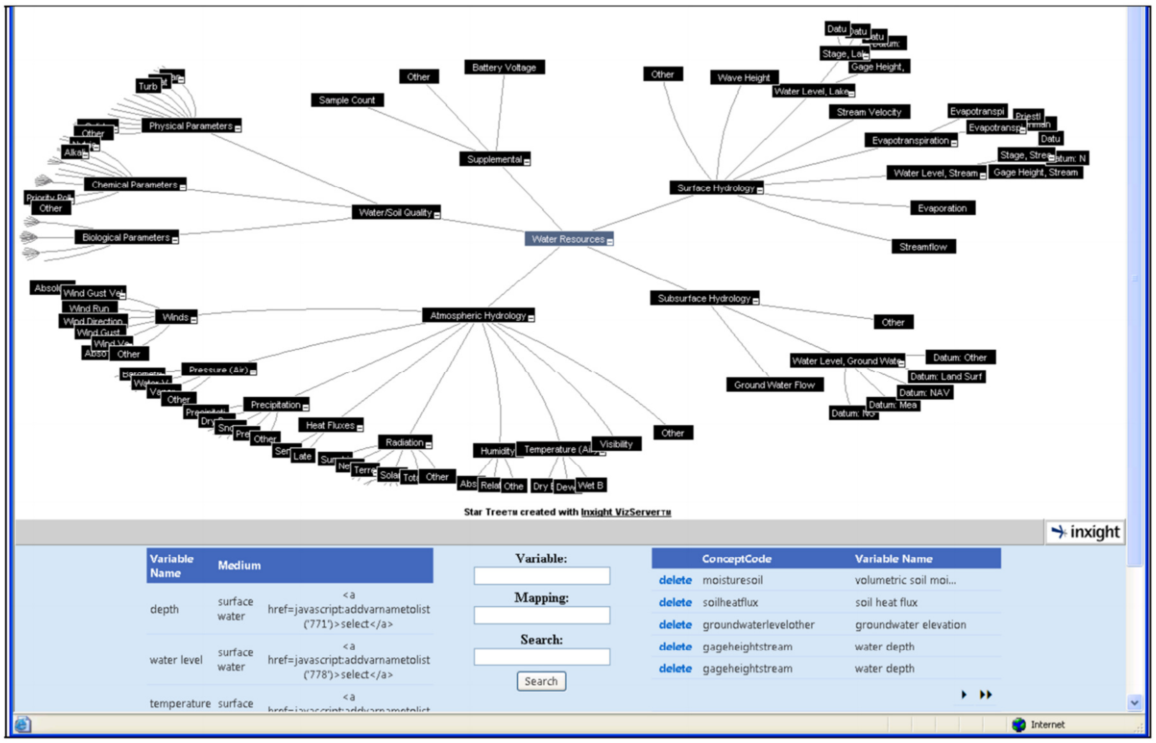


Figure 35: HIS Central Ontology tables (Whitenack 2010)



Variable Name	Medium	
depth	surface water	<a href=javascript:addvarnametolist('771')>select</a>
water level	surface water	<a href=javascript:addvarnametolist('778')>select</a>
temperature	surface	<a

	ConceptCode	Variable Name
delete	moisturesoil	volumetric soil moi...
delete	soilheatflux	soil heat flux
delete	groundwaterlevelother	groundwater elevation
delete	gageheightstream	water depth
delete	gageheightstream	water depth

Figure 36: HydroTagger web interface (Piasecki 2008)

HIS Central exposes several of its capabilities to clients of the HIS through a SOAP web service interface. The methods of the HIS Central web service are different from those in WOF, though some of them, such as GetSeriesCatalogForBox, return responses encoded with WaterML. Others use custom XML responses, which seem to lack documentation. The current list of HIS Central web service methods is:

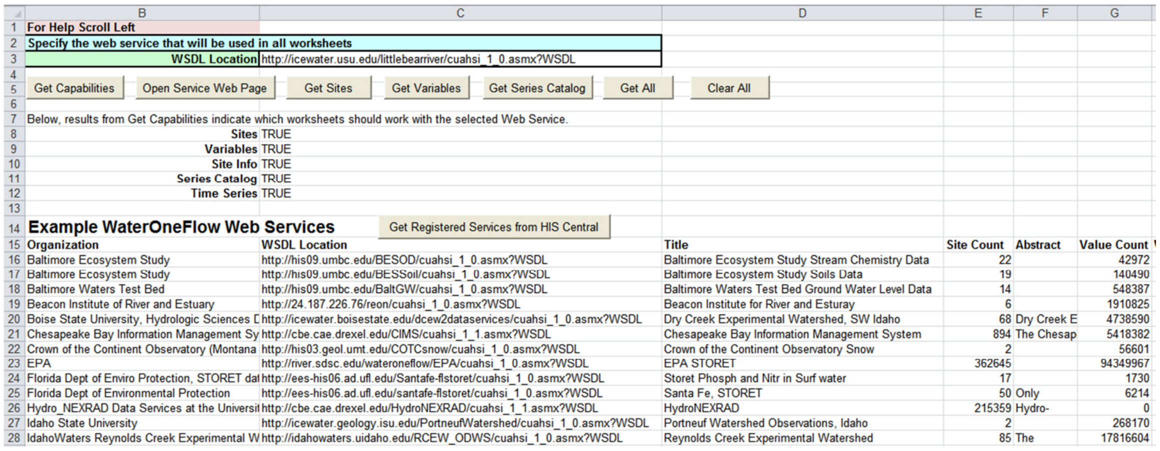
- GetMappedVariables
- GetSearchableConcepts
- GetSeriesCatalogForBox
- GetServicesInBox
- GetSitesInBox
- GetWaterOneFlowServiceInfo
- GetWordList
- getOntologyTree
- getSearchablePaths
- getSeriesCatalogInBoxPaged

Although there are 10 methods (plus some duplicates with slightly different signatures than those listed) in the HIS Central web service, only three are actually used within the current operating models of the HIS, as discussed in section 3.4.

3.3.4 Clients

The third and final component of the HIS architecture is the client applications that consume, analyze, and process hydrologic data. At a minimum, client applications must be able to communicate with WOF services. They need to be able to both make SOAP web service requests and parse the WaterML-formatted responses from HydroServers. To be fully integrated with the HIS architecture, client applications also should be able to make use of the HIS Central SOAP web service methods for searching for series (GetSeriesInBox) and retrieving the hydrologic ontology tree (GetOntologyTree). Clients must also be able to parse the custom XML responses from these services.

There have been two major client applications produced by the HIS project: HydroExcel and HydroDesktop. HydroExcel[6] is an Excel binary spreadsheet highly-customized through the use of embedded VisualBasic macros. HydroExcel requires the installation of a dynamic link library (DLL) called *HydroObjects*[7]. The HydroObjects DLL provides access to functions to interact WaterOneFlow web services. The methods in the HydroObjects DLL are called from the VisualBasic macros in HydroExcel. HydroExcel's interaction in the CUAHSI HIS operating models is described in sections 3.4.2 and 3.4.3 of this thesis.



For Help Scroll Left

Specify the web service that will be used in all worksheets

WSDL Location http://icewater.usu.edu/littlebearriver/cuahsi_1_0.asmx?WSDL

Below, results from Get Capabilities indicate which worksheets should work with the selected Web Service.

Sites	TRUE
Variables	TRUE
Site Info	TRUE
Series Catalog	TRUE
Time Series	TRUE

Example WaterOneFlow Web Services

Organization	WSDL Location	Title	Site Count	Abstract	Value Count
Baltimore Ecosystem Study	http://his09.umbc.edu/BESOD/cuahsi_1_0.asmx?WSDL	Baltimore Ecosystem Study Stream Chemistry Data	22		42972
Baltimore Ecosystem Study	http://his09.umbc.edu/BESSoil/cuahsi_1_0.asmx?WSDL	Baltimore Ecosystem Study Soils Data	19		140490
Baltimore Waters Test Bed	http://his09.umbc.edu/BaltGW/cuahsi_1_0.asmx?WSDL	Baltimore Waters Test Bed Ground Water Level Data	14		548387
Beacon Institute of River and Estuary	http://24.187.226.76/reon/cuahsi_1_0.asmx?WSDL	Beacon Institute for River and Esturay	6		1910825
Boise State University, Hydrologic Sciences D	http://icewater.boisestate.edu/dcew2dataservices/cuahsi_1_0.asmx?WSDL	Dry Creek Experimental Watershed, SW Idaho	68	Dry Creek E	4738590
Chesapeake Bay Information Management Sy	http://cbe.cae.drexel.edu/CIMS/cuahsi_1_1.asmx?WSDL	Chesapeake Bay Information Management System	894	The Chesap	5418382
Crown of the Continent Observatory (Montana	http://his03.geol.umt.edu/COTCsnow/cuahsi_1_0.asmx?WSDL	Crown of the Continent Observatory Snow	2		56601
EPA	http://river.sdsc.edu/wateroneflow/EPA/cuahsi_1_0.asmx?WSDL	EPA STORET	362645		94349967
Florida Dept of Enviro Protection, STORET dat	http://ees-his06.ad.ufl.edu/Santafe-flstoret/cuahsi_1_0.asmx?WSDL	Storet Phosph and Nitr in Surf water	17		1730
Florida Dept of Environmental Protection	http://ees-his06.ad.ufl.edu/santafe-flstoret/cuahsi_1_0.asmx?WSDL	Santa Fe, STORET	50	Only	6214
Hydro_NEXRAD Data Services at the Universit	http://cbe.cae.drexel.edu/HydroNEXRAD/cuahsi_1_1.asmx?WSDL	HydroNEXRAD	215359	Hydro-	0
Idaho State University	http://icewater.geology.isu.edu/PortneufWatershed/cuahsi_1_0.asmx?WSDL	Portneuf Watershed Observations, Idaho	2		268170
IdahoWaters Reynolds Creek Experimental W	http://idahowaters.uidaho.edu/RCEW_ODWS/cuahsi_1_0.asmx?WSDL	Reynolds Creek Experimental Watershed	85	The	17816604

Figure 37: HydroExcel interface

The newest and most robust client application produced by the HIS project is HydroDesktop. HydroDesktop is an open-source .NET desktop application that leverages several other open-source projects to provide a map-based tool for the discovery, management, and analysis of hydrologic data. The geographic information system capabilities of HydroDesktop are provided by the MapWindow6 and DotSpatial

[6] http://his.cuahsi.org/hydroexcel.html
[7] http://his.cuahsi.org/hydroobjects.html

libraries. Rather than using the HydroObjects DLL, HydroDesktop maintains its own codebase for calling WOF and HIS Central web services. HydroDesktop's operation within the current HIS operating models is described in sections 3.4.4 and 3.4.5.

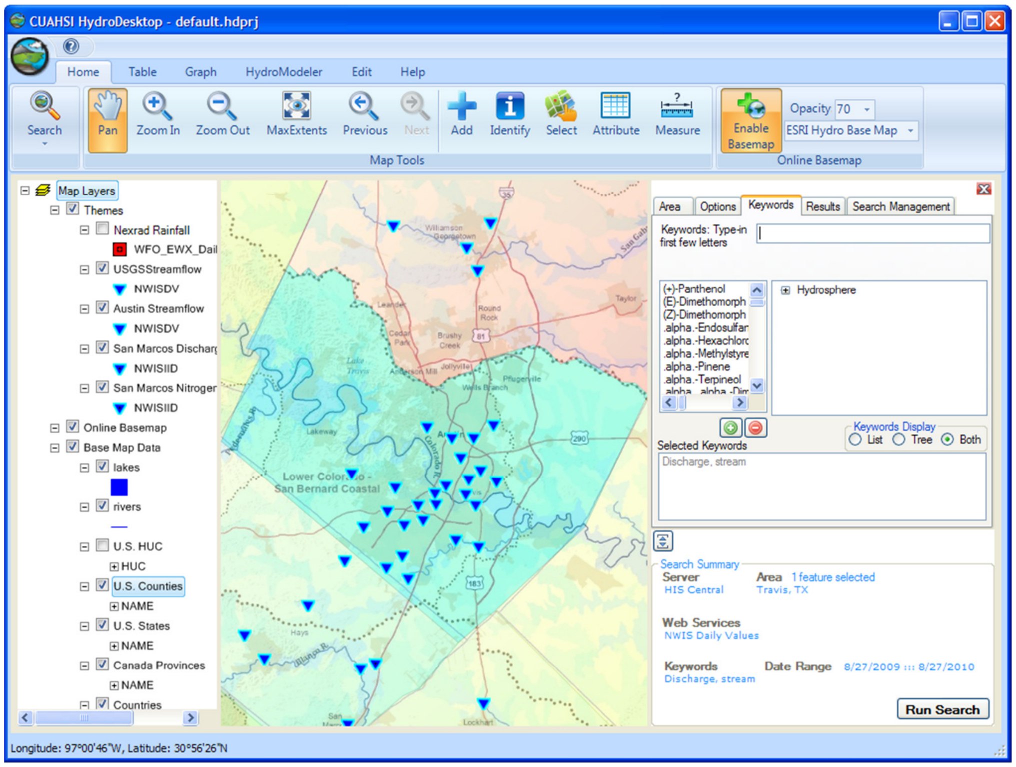


Figure 38: HydroDesktop interface

3.3.5 Summary of Roles and Responsibilities

The following is a list of responsibilities of each of the three components of the current HIS architecture.

- Data providers – HydroServers:
    - Publish hydrologic data on HydroServer using WOF services
    - Register services at HIS Central and request metadata harvest

    - Tag variables found in harvest with concepts from hydrologic ontology using HydroTagger application
- SDSC team – HIS Central:
    - Maintain hydrologic ontology database
    - Maintain series catalog database
    - Maintain HydroTagger application
    - Maintain registration application
    - Maintain harvester programs
    - Maintain HIS Central web services
    - Run metadata harvests from HydroServers
    - Perform metadata harvesting from federal data sources
        - Obtain database dumps
        - Migrate database dumps from native formats to SQL Server format
        - Build custom ODM views over migrated database dumps
        - Harvest metadata
    - Validate harvested metadata
- Client applications:
    - Make SOAP requests and parse SOAP responses (for both WOF and HIS Central services)
    - Parse WaterML and HIS Central service XML
    - Build internal metadata catalogs (in some operating models)

3.4 HIS OPERATING MODELS

3.4.1 Introduction

Although the HIS project has created a prototype SOA with the three aforementioned components working in unison, the ways these components have actually interacted have taken several forms. The conceptual representations of how clients have interacted in the HIS are called *operating models*. The operating models used through the development of the HIS are the: direct client-server, weak central catalog, strong series catalog, and dual-catalog models. This section describes the operating models in terms of the clients that implement them, namely HydroExcel and HydroDesktop.

3.4.2 Direct Client-Server Model

The simplest hydrologic data services operating model from the HIS project is the *direct client-server* model. In this model, a client application interacts only and directly with the HydroServers, without any centralized catalog. Figure 39 illustrates this model applied to the HIS client application *HydroExcel* (versions prior to 1.1.3). HydroExcel (versions prior to 1.1.3) has a hard-coded list of the WOF WSDL file addresses for networks on numerous HydroServers. This list was kept up-to-date by periodically editing the HydroExcel file and reposting the modified version on the HIS website.

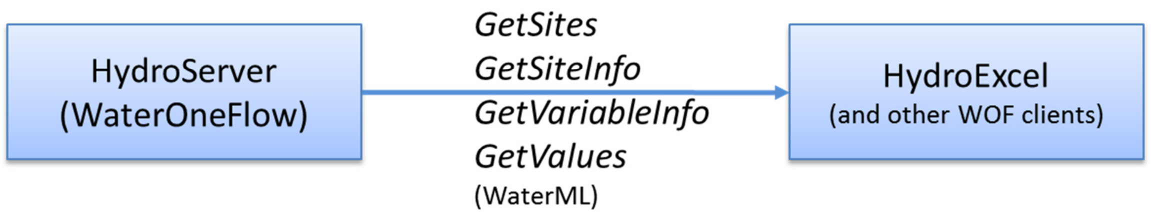


Figure 39: Direct Client-Server Model

In the HydroExcel application of this model, the user can select only a single service from which data and metadata can be obtained at any time. HydroExcel first uses the GetSites method on the selected service to get a list of sites. If the service is WaterOneFlow 1.1-compliant the user can restrict the requested sites (and subsequent series catalog) to a geographic bounding box. For each site returned in the WaterML response, HydroExcel calls GetSiteInfo. The responses from each call to GetSiteInfo are compiled to build an internal series catalog of the time series data available for the chosen WOF service. Once requested by the user, data for a selected series are downloaded via the GetValues method. The user may also request a list of the variables offered through a given service by having HydroExcel invoke the GetVariableInfo method.

3.4.3 Weak Central Catalog Model

In the *weak central catalog* model, a centralized catalog is used to register and retrieve WSDL file addresses of compliant HydroServers. The newest version of HydroExcel, version 1.1.3, follows this model. HydroExcel requests a list of registered WOF services from HIS Central via the GetWaterOneFlowServiceInfo web service method, and is returned an XML file containing this list of WOF WSDL file addresses. This model is illustrated in Figure 40.

The pattern of building an internal series catalog for a single WOF service at a time follows that described in section 3.4.2.

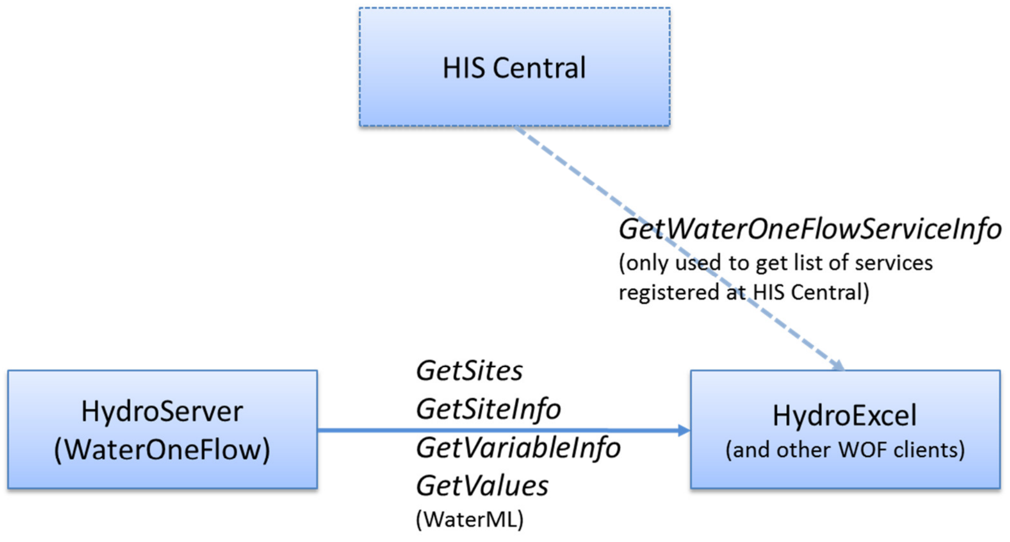

HIS Central
GetWaterOneFlowServiceInfo
(only used to get list of services registered at HIS Central)
HydroServer (WaterOneFlow)
GetSites
GetSiteInfo
GetVariableInfo
GetValues
(WaterML)
HydroExcel
(and other WOF clients)

Figure 40: Weak Central Catalog Model

3.4.4 Strong Series Catalog Model

The *strong series catalog* model builds on the previous model by extending the role of the centralized catalog. In this model, responsibility for building series catalogs is moved from the client application to the central catalog server, which has numerous impacts on the overall flow of information. Figure 41 illustrates this model as it is applied to CUAHSI's HydroDesktop application. HydroDesktop, while still in its development phase, offers numerous improvements over HydroExcel. One improvement in particular is the ability to discover and download hydrologic data series across multiple HydroServers instead of being limited to one server at a time. The technology that enables this capability is the centralized metadata catalog at HIS Central.

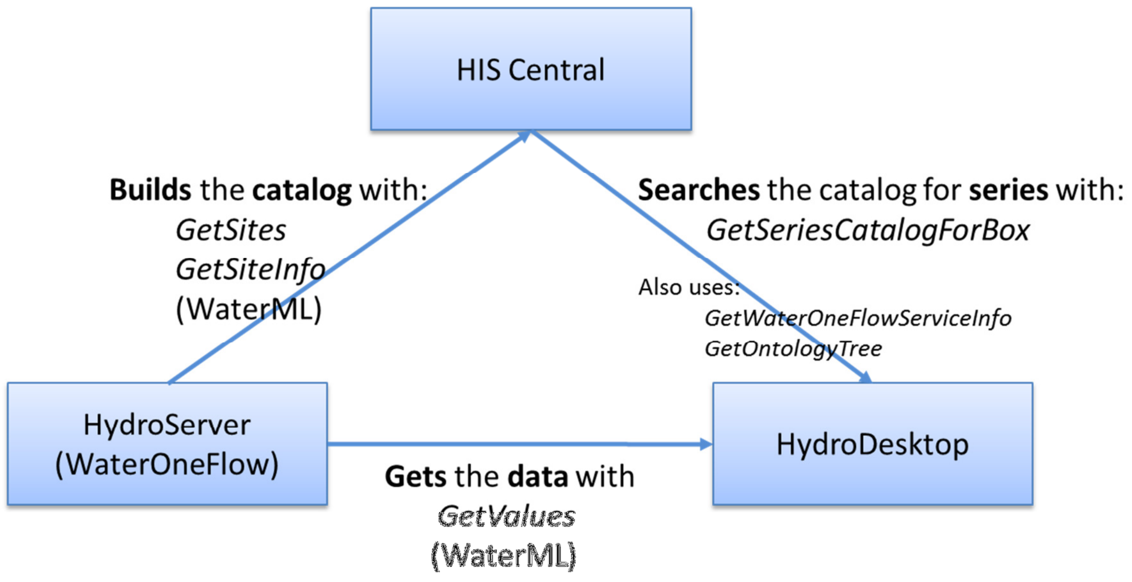

HIS Central
Builds the **catalog** with:
GetSites
GetSiteInfo
(WaterML)
Searches the catalog for **series** with:
GetSeriesCatalogForBox
Also uses:
GetWaterOneFlowServiceInfo
GetOntologyTree
HydroServer (WaterOneFlow)
HydroDesktop
Gets the **data** with
GetValues
(WaterML)

Figure 41: Strong Series Catalog Model

Client applications that follow this model, namely HydroDesktop, can search for series at HIS Central instead of building and maintaining their own internal series catalogs. Prior to searching, HydroDesktop uses HIS Central's GetWaterOneFlowServiceInfo web service method to identify the registered networks at HIS Central. In addition, HIS Central's GetOntologyTree method, which returns an XML document containing the CUAHSI keyword-concept ontology, is used to display the list of searchable terms at HIS Central's series catalog.

HydroDesktop searches HIS Central's series catalog via the GetSeriesCatalogForBox method. This method takes a spatial extent, a concept code, a date range, and a list of identifier numbers indicating which WOF services to search and returns a catalog of matching series in a custom XML format (i.e., not WaterML). A diagram depicting the flow of user actions to select these parameters is shown in Figure 42. Internally, HydroDesktop actually performs multiple calls to this method. The search area (e.g., a state, HUC, county, etc.) is broken into 1°-by-1° boxes, with a separate call to GetSeriesCatalogForBox made for each box. A separate call is also made for each keyword that was selected by the user. Thus, if a user selected two keywords and the selected search area was a 2°-by-2° area, eight calls to the GetSeriesCatalogForBox would be made (4 for the area × 2 for the keywords).

The HydroDesktop user can filter this series catalog before finally downloading the time series data. Data are downloaded from their respective HydroServers by using the GetValues method sequentially on each HydroServer for each variable. The final collection of hydrologic observations data collected across services and possibly containing several different variables is known as a *theme*.

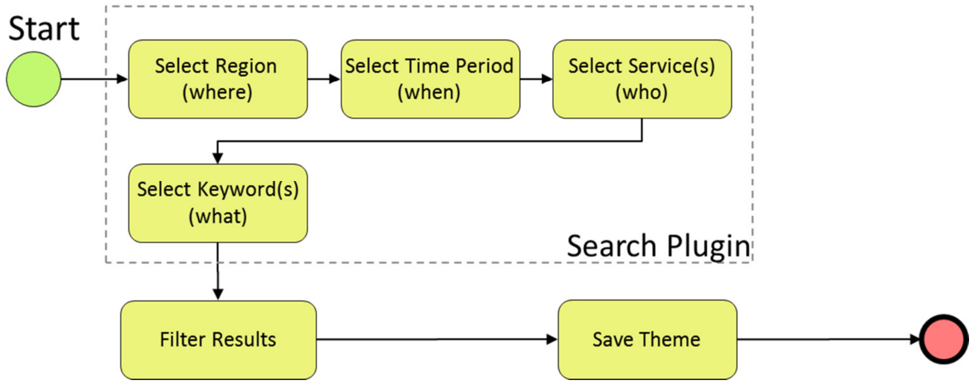


Figure 42: HydroDesktop search process

3.4.5 Dual-Catalog Model

In the *dual-catalog* model, a strong central catalog still exists and is used as described in section 3.4.4. In addition to the central catalog, however, is a local catalog maintained by the client application. Like in the direct client-server model, the local catalog builds its internal catalog through the WOF GetSites and GetSiteInfo methods.

The strength of this model is that both services registered at the central catalog and non-registered servers could have their series searched together.

In the case of HydroDesktop, the local catalog is known as the *metadata cache*. Figure 43 shows the dual-catalog model applied to HydroDesktop. Although full functionality (namely searching across the series catalog of the metadata cache) has not been completely implemented in the version released of HydroDesktop as of this thesis, there are plans to fully implement it in the near future.

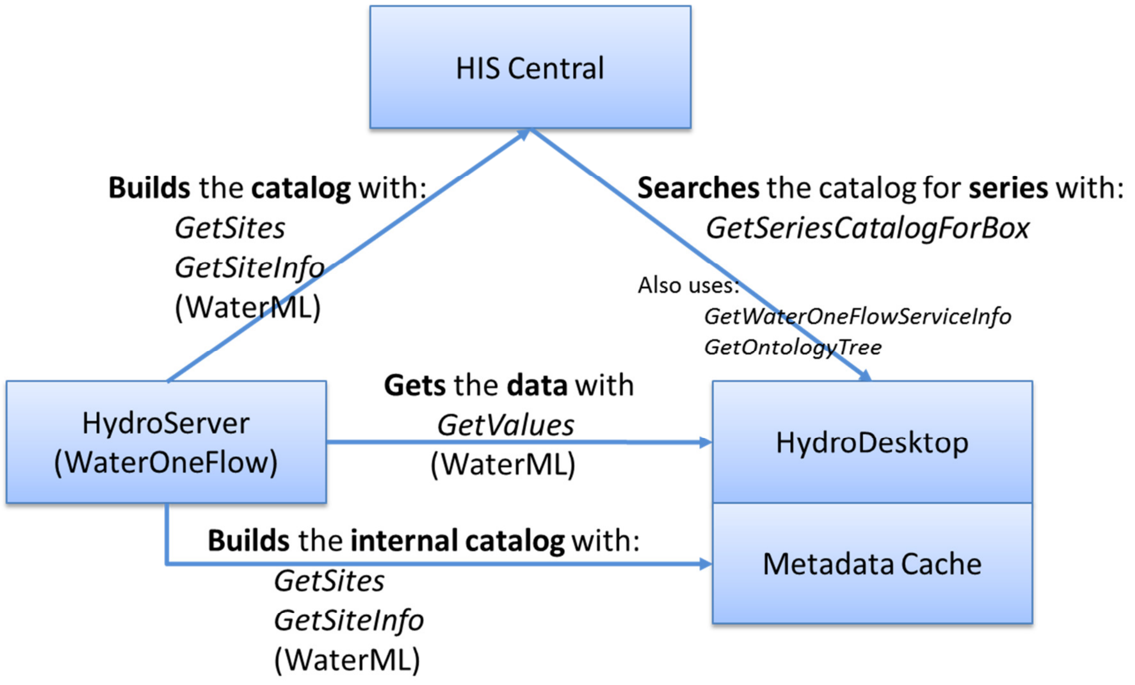


Figure 43: Dual Catalog Model

3.5 Summary and Issues

This Network-Observations Model for the CUAHSI HIS architecture is centered on a strong central metadata catalog maintained at HIS Central. In essence, this has resulted in a centralized metadata, distributed data system.

For off-the-shelf-component HydroServers, if the quantity of series in a network is large, such as with TWDB Groundwater network, metadata harvesting by HIS Central can take several days. While this might be an acceptable approach for services that only need "one-off" harvests of metadata (such as archive services), it is an issue for services that are continuously updated or have often changing series. It also requires constant intervention from the HIS Central team, which is not sustainable in the long run.

The number of methods required to fully implement the WOF standard can be a burden for some data providers, such as the USGS or EPA. These large national providers may not be able to custom-implement the full WOF specification. They presently have their own data models and services, and may lack the financial, personnel, or organizational resources to dedicate to this task. This has led to the database dumping described in section 3.3.3.

The routine of receiving and migrating large database dumps from federal agencies raises both sustainability and data-completeness concerns. Because this process is done infrequently, series metadata might be out-of-date for long periods of time, which results in researchers not being able to search for the data they might need. Another concern is that the federal database schemas sometimes change between dumps, meaning that new work for migrating to SQL Server and creating ODM views over the databases needs to be done. Furthermore, the schemas in these database dumps are sometimes lacking some of the metadata required to fully specify a SeriesCatalog record in the HIS Central Series Catalog. This lack of metadata leads to problems in searching for and understanding the series that are returned to clients like HydroDesktop.

Chapter 4: The Services Stack Model

4.1 Overview

The concerns of sustainability for the Network-Observations Model have led to the desire for an improved HIS architecture. This chapter proposes the *Services Stack Model*, a more decentralized architectural model for the HIS SOA. The proposed model is built-upon OGC standard services and their associated data encoding specifications, including the forthcoming WaterML2.0 specification. By adhering to these international standards, it is hoped that that wider adoption of the CUAHSI HIS, especially by federal data providers, will be possible. By decentralizing metadata services, the approach of the new model will also reduce migration and translation tasks performed by the HIS Central staff, thus creating a more sustainable system.

This chapter first introduces the Thematic Metadata Table format, which plays a critical role in providing series metadata descriptions for the Services Stack Model, as well as forming the logical basis of the model. Next, the proposed architectural model is described in terms of a services stack and the data/metadata organization provided by this stack. Responsibilities of each of the three components of the Services Stack Model are explained and summarized. A simple proof-of-concept application is used to illustrate how clients will function in the proposed architecture.

The proposed Services Stack Model is meant to serve as a starting point for development of the new HIS services architecture. Best practices for putting the concepts espoused in the model into practice will need to be investigated and agreed upon by the CUAHSI-HIS team and its stakeholders to realize a robust system. In support of this, the areas where more analysis is required and where issues may exist are described at the end of this chapter.

4.2 Thematic Metadata Table

The CUAHSI *Metadata Table* (originally called the *Data Cart*) is an observations metadata structure developed by Tim Whiteaker of CRWR and Dean Djokic of ESRI (Whiteaker and Djokic 2010). A Metadata Table contains fields to describe series of data similar to those in the ODM's series catalog. In addition to these fields, however, a Metadata Table also contains "information needed for a client to access each time series described in the cart" (Whiteaker and Djokic 2010, 1). The information to access each series comes in the form of addresses, protocols, and input parameters to the web services from which the actual values data may be downloaded. The inclusion of these service references is a critical factor that separates the Metadata Table from being just a series catalog. The listing and description of all fields, including example values, of the current Metadata Table specification are given in Appendix A. The current specification is based on providing metadata for series available from WOF.

As indicated by its name, a *Thematic* Metadata Table contains metadata about series in a theme. A hydrologic time series is a time-indexed collection of observations about a specific property (i.e., a measured variable) of the hydrologic cycle at a specific location. A *theme* is defined rather loosely as a collection of series that describes a geographic region with respect to some subject. The geographic region of a theme could be a study watershed, a state, the entire United States, or any other geographic area of interest. The subject of a theme is similarly openly-defined, and could be a single variable such as streamflow from a single data provider, or all variables measured by state water agency.

Publication of a Metadata Table as a WFS can be fairly simple using off-the-shelf tools. For example, using ESRI's ArcGIS Desktop, a Metadata Table can be created as a

Shapefile with point features for each series in the Metadata Table. Using ArcGIS Server, this Shapefile can then be published as a WFS with a few relatively quick steps.

4.3 Services Stack Model

The Services Stack Model is structured around a *services stack* comprised of OGC web service specifications, with the concept of the Thematic Metadata Table as a fundamental series metadata descriptor. This three-tiered services stack is illustrated in Figure 44. For each level (Data, Metadata, and Catalog) of the stack an endpoint for the OGC service specification (SOS, WFS, and CSW) that will provide access to that level is shown.

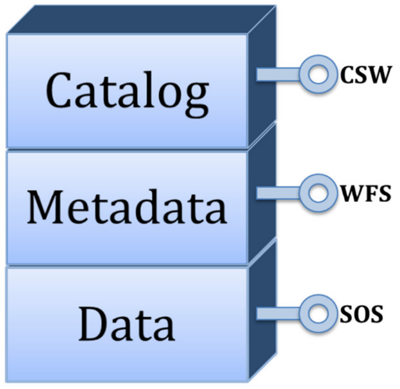


Figure 44: OGC services stack

At the bottom-most level of the stack are data services based on OGC's SOS specification. The data services provide access to hydrologic observations data in WaterML2.0 format. The middle-level of the stack are metadata services using the WFS standard. The metadata provided from this level describe series of observations data available at the preceding level using CUAHSI's Thematic Metadata Table format expressed in GML. The top-most level of the services stack is a catalog service. The catalog service indexes registered metadata services and provides a CSW endpoint to search and access them.

Individual services stacks can be published and maintained by hydrologic data providers or other entities, rather than relying on a single centralized catalog to orchestrate the system. For example, the State of Texas could host a services stack for its hydrologic data. In this example case, the themes available from the WFS metadata service might be organized by agency.

Services stacks can also be joined together, or *federated*, into a larger system, as illustrated in Figure 45. In this larger system, the catalogs of each underlying services stack would have their CSW endpoints registered at a centralized *meta-catalog*. This catalog-of-catalogs can then provide combined searching across all registered services stacks. The loose-coupling of this federated system would provide data providers with the flexibility to manage their own stacks, while still providing a unified search mechanism for data consumers.

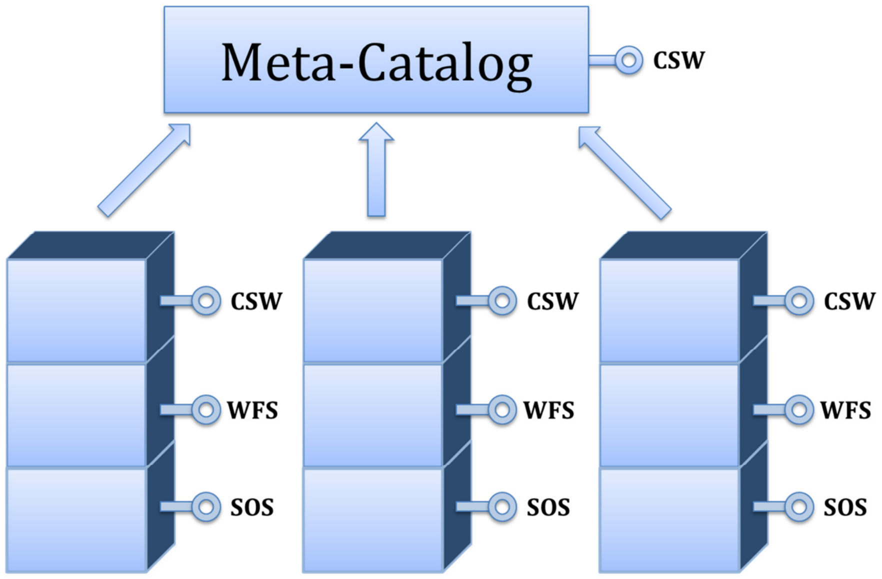


Figure 45: Federated services stacks

4.4 Services Stack Model SOA Components

4.4.1 Overview

The components of a SOA based on the proposed Services Stack Model are similar to those in any SOA and specifically those in the current HIS architectural model. This section provides descriptions of the SOA components and their interactions with each other.

The Services Stack Model has three major components: HydroServers, a Catalog, and Clients. Figure 46 shows a summary of these component interactions organized around the OGC services stack described in the previous section. The inclusion of a federated meta-catalog adds another component to this system and is discussed in section 4.4.3.

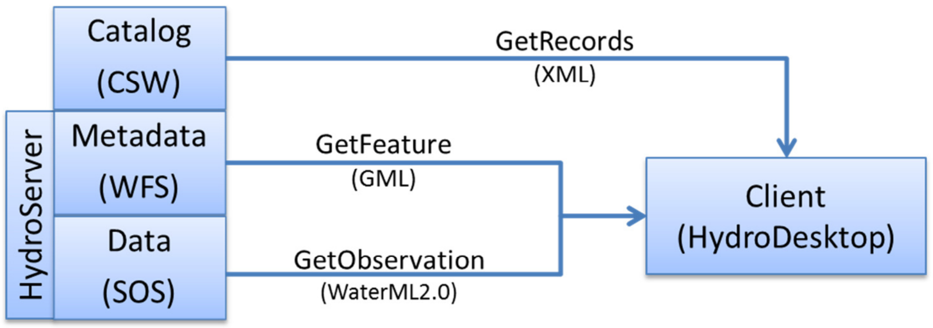


Figure 46: Component interaction within the Services Stack Model

4.4.2 HydroServers

As in the Network-Observations Model, HydroServers in the Services Stack Model provide access to the actual hydrologic observations data. The hydrologic time series data are obtained through a SOS implementation using the GetRecords method, and are encoded in OGC's forthcoming WaterML2.0 specification. Additionally, HydroServers provide series metadata through Thematic Metadata Table WFS services. Each theme at a HydroServer has its own WFS endpoint. This HydroServer concept is depicted in Figure 47.

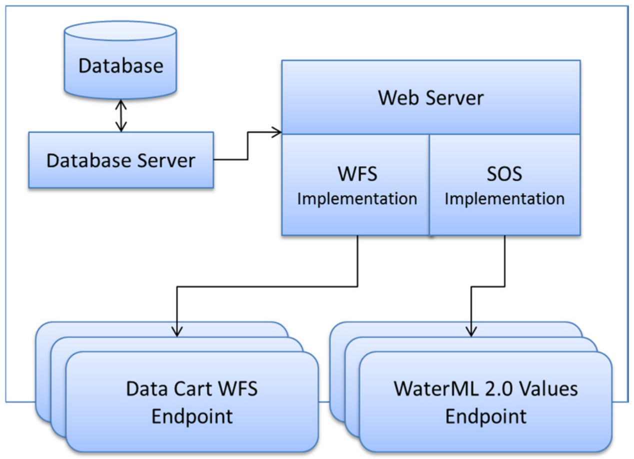


Figure 47: HydroServer in Services Stack Model

Thematic Metadata Table WFS services, as described in Chapter 2, have three primary methods: GetCapabilities, GetFeature, and GetFeatureById. For HydroServers in the proposed architecture, the GetCapabilities method would be used to obtain WFS service-level metadata, such as the title of the Metadata Table feature layer represented in the service. The GetFeature operation would be used to search for matching series in the Metadata Table. This searching is accomplished through the use of OGC Filters, which can be applied spatially and temporally, as well as to any text (such as Concept keyword) or numerical metadata values (such as ValueCount). The GetFeatureById method could be used if a specific series identifier is already known by the client application.

A critical change from the current HIS model for data providers who publish HydroServers is that ontological concept tagging of series will have to happen at the provider level, rather than at the central catalog. The exact implementation of this tagging is not certain, though the hydrologic ontology itself will still need to be centrally-maintained to ensure all providers use the same vocabulary.

4.4.3 Catalogs

Rather than a series catalog, the Services Stack Model uses CSW-compliant *services* catalogs to provide unified searching. The primary CSW method that will be used by client applications is GetRecords. Clients will search for matching Metadata Table WFS services by making GetRecords requests to the service catalog's CSW endpoint. As with the series searching on HydroServers, OGC Filters will be used to match services by spatial and temporal extents, as well as by ontological concept and other criteria.

For this type of Filter-based service searching to be accomplished, data providers will need to register their WFS Metadata Table services at the service catalog. During registration, core service-level metadata (such as Title, Abstract, and Subject) will need to be entered. Additionally, all of the ontological concepts available across the series in the Metadata Table will need to be represented at the service-level.

Several off-the-shelf software options (such as deegree[8], GeoNetwork[9], and ESRI's GeoPortal[10]) for a CSW-compliant server implementation have been identified.

[8] http://www.deegree.org/

[9] http://geonetwork-opensource.org/

[10] http://www.esri.com/software/arcgis/geoportal/index.html

Using an off-the-shelf solution for this piece of the architecture will reduce custom programming requirements.

A meta-catalog that indexes catalogs from individual services stacks adds another layer of interaction to the Services Stack-based SOA model. As discussed in section 4.3, the meta-catalog facilitates searching across other registered catalog services. From the client perspective, this federated searching should occur without modification to the user's activities. Rather, upon receiving a Filtered GetRecords request from the client, the meta-catalog performs similar Filtered GetRecords requests upon the catalogs it indexes, returning a single list of services to the client.

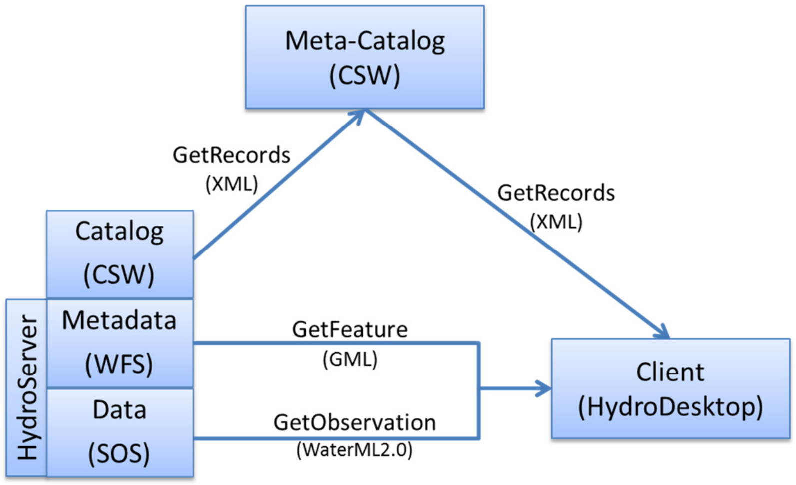


Figure 48: Component interaction with services stack and meta-catalog

4.4.4 Clients

Clients in the Services Stack architecture will have more responsibilities with regard to the discovery of observations data. However, rather than programming against the specialized WOF and HIS Central web services of the current model, client applications can leverage existing libraries for accessing the OGC service specifications.

Several libraries for using these service specifications are listed online at the OGC's web site: http://www.opengeospatial.org/resource/products/byspec.

Current client applications will need to be revised to work with the proposed architecture. This includes being able to query WFS, CSW, and SOS web services, construct OGC Filters, and parse GML and WaterML2.0. Due to the increased number of web service requests inherent with the Services Stack Model over that required in the Network-Observations Model, client applications need to be "intelligent" about restricting search requests or at least warning the user about long response times. Fortunately, the "hits" result type of the WFS GetFeature method, which just returns the number of hits a request would return instead of the actual series, can be used to aid in this restriction. By first making a "hits" request, the user could be warned if a potential result set is too large and could possibly take a long time to retrieve.

4.4.5 Summary of Roles and Responsibilities

The following is a list of responsibilities for each of the three components in the proposed Services Stack architectural model.

- Data providers - HydroServers:
  - Organize hydrologic data series into Thematic Metadata Tables and serve these as WFS in GML
  - Tag each series in Thematic Metadata Tables with its concept from the CUAHSI hydrologic ontology
  - Register Metadata Table WFS services at services catalog
  - Serve the actual hydrologic data series through SOS in WaterML2.0
- Catalogs:

    - Implement CSW endpoint for searching and registering HydroServer Metadata Table WFS endpoints
    - For Meta-Catalog:
        - Implement CSW endpoint for unified searching across registered services
        - Make CSW requests to registered catalogs to obtain their service-level metadata
- Client applications:
    - Make CSW requests to services catalogs and parse XML responses
    - Make WFS requests to HydroServers and parse Metadata Table GML responses
    - Make SOS requests to HydroServers and parse WaterML2.0 responses
    - Construct Filters for desired spatial extents, temporal extents, concepts, and other criteria for searching of hydrologic data

4.5 PROPOSED OPERATING MODEL

Figure 49 is a depiction of the general operating model in the proposed Services Stack SOA. HydroServers are registered at their stack's catalog and provide service-level metadata to the catalog. The client application searches either a selected catalog or searches the unified meta-catalog with filtered CSW GetRecords operations, and gets series metadata from HydroServers via the WFS GetFeature operation. The client application then requests observations data series through SOS GetObservations.

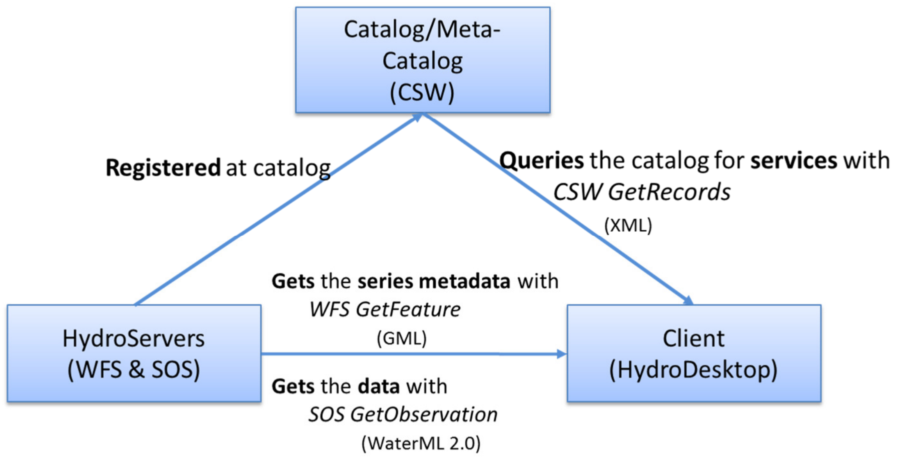

Catalog/Meta-Catalog (CSW)
Registered at catalog
Queries the catalog for **services** with *CSW GetRecords* (XML)
Gets the **series metadata** with *WFS GetFeature* (GML)
HydroServers (WFS & SOS)
Client (HydroDesktop)
Gets the **data** with *SOS GetObservation* (WaterML 2.0)

Figure 49: General operating model in Services Stack SOA

The general operation of the Services Stack Model is described by the following steps for a single search process:

1. For a selected keyword/concept, the client application queries the catalog or meta-catalog CSW endpoint with a CSW GetRecords query. Each request contains Filters specifying the desired current concept, as well as the spatial and temporal extents chosen by the user.
2. The catalog responds to the query with an XML response containing a list of HydroServer Metadata Table WFS service endpoints that have service-level metadata matching the specified parameters.
3. The client application cycles through the list of user-narrowed WFS Metadata Table service addresses, querying each with WFS:GetFeature with the same Filters as used in catalog request, plus any additional desired series-level constraints (such as value count or sample medium).
4. Each HydroServer WFS Metadata Table service responds with a list of GML Features that match the specified Filter parameters. Each Feature is

a series as represented by the Metadata Table specification, and contains a reference to the SOS endpoint from which the actual time series data can be obtained.

5. The client application goes through the list of user-narrowed series and queries the SOS endpoints with GetObservations to obtain time series values.
6. The HydroServer SOS instances respond to each time series request with a WaterML2.0-formatted response containing the desired series.

Figure 50 illustrates these steps from the user/client-application activity of "narrowing" result sets. Between each query stage, the user can apply additional constraints to further narrow the list of results passed to the next stage. This narrowing would be accomplished by filtering on additionally desired metadata fields at each level.

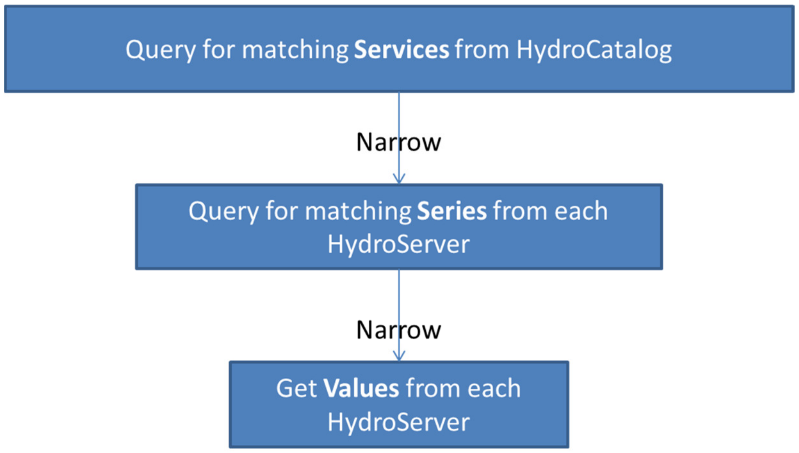
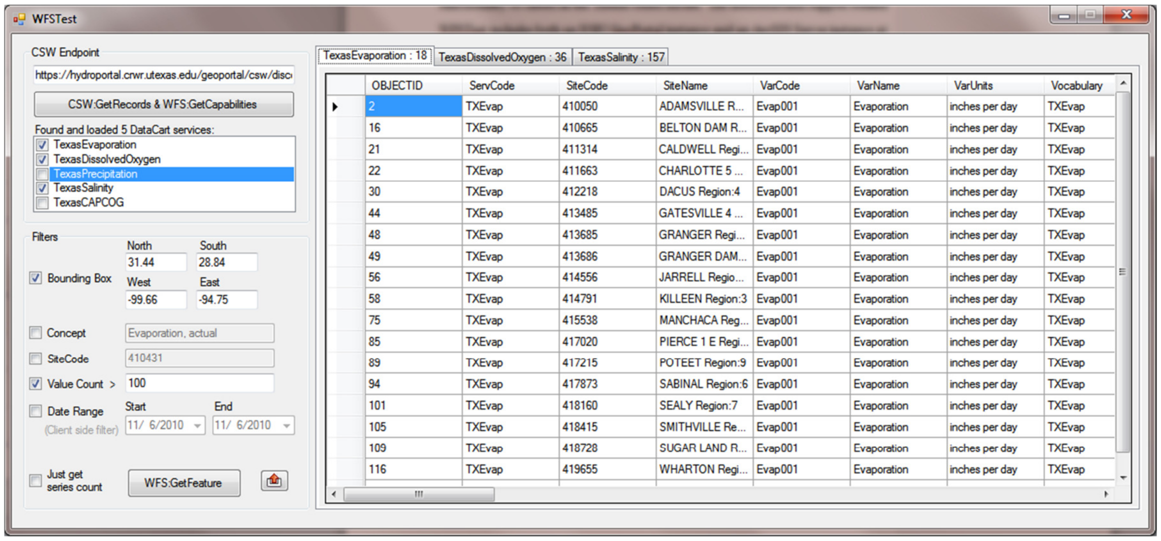

Query for matching Services from HydroCatalog
Narrow
Query for matching Series from each HydroServer
Narrow
Get Values from each HydroServer

Figure 50: "Narrowing" steps with the Services Stack Model

4.6 WFSTest: A Simple Proof-of-Concept Client

WFSTest (see Figure 51) is a proof-of-concept application that implements some of the proposed functionality of clients in the Services Stack Model. The main purpose of writing this application was to provide a demonstration of the main components of the Services Stack Model. The infrastructural support behind WFSTest includes both an ESRI GeoPortal instance and an ArcGIS Server instance at CRWR. The ESRI GeoPortal provides a CSW-compliant endpoint that supports OGC Filters for geographic extent and string matching in metadata fields. Several Metadata Table WFS services published from the CRWR ArcGIS Server instance are registered at the GeoPortal instance, and are identified as Metadata Table services in their Abstract metadata field. Each of these WFS services contain only a single Metadata Table layer.



CSW Endpoint: https://hydroportal.crwr.utexas.edu/geoportal/csw/disc

Found and loaded 5 DataCart services:
- [✓] TexasEvaporation
- [✓] TexasDissolvedOxygen
- [ ] TexasPrecipitation
- [✓] TexasSalinity
- [ ] TexasCAPCOG

Filters

[✓] Bounding Box	North: 31.44	South: 28.84
	West: -99.66	East: -94.75
[ ] Concept	Evaporation, actual	
[ ] SiteCode	410431	
[✓] Value Count >	100	
[ ] Date Range (Client side filter)	Start: 11/ 6/2010	End: 11/ 6/2010
[ ] Just get series count		

TexasEvaporation : 18 | TexasDissolvedOxygen : 36 | TexasSalinity : 157

OBJECTID	ServCode	SiteCode	SiteName	VarCode	VarName	VarUnits	Vocabulary
2	TXEvap	410050	ADAMSVILLE R...	Evap001	Evaporation	inches per day	TXEvap
16	TXEvap	410665	BELTON DAM R...	Evap001	Evaporation	inches per day	TXEvap
21	TXEvap	411314	CALDWELL Regi...	Evap001	Evaporation	inches per day	TXEvap
22	TXEvap	411663	CHARLOTTE 5 ...	Evap001	Evaporation	inches per day	TXEvap
30	TXEvap	412218	DACUS Region:4	Evap001	Evaporation	inches per day	TXEvap
44	TXEvap	413485	GATESVILLE 4 ...	Evap001	Evaporation	inches per day	TXEvap
48	TXEvap	413685	GRANGER Regi...	Evap001	Evaporation	inches per day	TXEvap
49	TXEvap	413686	GRANGER DAM...	Evap001	Evaporation	inches per day	TXEvap
56	TXEvap	414556	JARRELL Regio...	Evap001	Evaporation	inches per day	TXEvap
58	TXEvap	414791	KILLEEN Region:3	Evap001	Evaporation	inches per day	TXEvap
75	TXEvap	415538	MANCHACA Reg...	Evap001	Evaporation	inches per day	TXEvap
85	TXEvap	417020	PIERCE 1 E Regi...	Evap001	Evaporation	inches per day	TXEvap
89	TXEvap	417215	POTEET Region:9	Evap001	Evaporation	inches per day	TXEvap
94	TXEvap	417873	SABINAL Region:6	Evap001	Evaporation	inches per day	TXEvap
101	TXEvap	418160	SEALY Region:7	Evap001	Evaporation	inches per day	TXEvap
105	TXEvap	418415	SMITHVILLE Re...	Evap001	Evaporation	inches per day	TXEvap
109	TXEvap	418728	SUGAR LAND R...	Evap001	Evaporation	inches per day	TXEvap
116	TXEvap	419655	WHARTON Regi...	Evap001	Evaporation	inches per day	TXEvap

Figure 51: WFSTest application interface

WFSTest queries the GeoPortal CSW-endpoint with a GetRecords request. This request includes a Filter to find only those registered services whose Abstract field contains the string "DataCart." The XML containing a list of these services is returned to WFSTest, and the URLs to the matching Metadata Table WFS services are extracted. The GetCapabilities method of each Metadata Table WFS service is called and the titles of each Metadata Table layer are extracted.

WFSTest can construct Filters for geographic extent, concept keyword, site code, and value count. These Filters are used to query each Metadata Table WFS service with the GetFeature operation. The user can select to just receive a count of how many series ("hits") the GetFeature request would contain, or can see the full Metadata Table contents returned from the GetFeature request.

A more detailed description of the use of WFSTest is in Appendix B.

4.7 Issues and Areas for Future Research

The proposed Services Stack architecture and operation are meant to serve as a starting point for exploration and discussion of a new, more sustainable HIS based on international standards. Several possible shortcomings and areas where further research is required are identified in this section. Best practices will need to be determined by the CUAHSI-HIS team to overcome these shortcomings or modify the proposed model to fit technological and other system constraints.

Time extent support for CSW and WFS using Filters is supported in the upcoming 2.0 version of the OGC Filter Encoding specification. However, time extent Filter support in existing CSW-compliant servers appears to be either weak or non-existent. Implementing this capability could be a complex task that takes a long development time. One possible work-around to actual date-time filtering could be to represent date-times as Julian dates. This would allow simple numerical tests, which are already widely supported in OGC Filter implementations, to find if a date-time is between the start and end dates of a series of observations data. If this approach were used, however, client applications would need to be able to work with this somewhat non-standard means of time representation.

The current Metadata Table specification is not a final version, and was designed around the CUAHSI WOF service specification. To be truly effective as a series metadata descriptor format, feedback about the attributes in the specification will need to be collected and revisions made accordingly. Since a main driving force behind redesigning the HIS SOA is to gain acceptance from federal data providers and more accurately describe their hydrologic data series, input from these agencies will be extremely valuable. Fields specific to WOF will either need to be generalized or omitted in light of using OGC standard services. Additionally, the Metadata Table format might need to be made more flexible, with different attributes based on the types of series metadata that can be expected from the different data providers. For example, including drainage area in the USGS Daily Values Metadata Table might be beneficial for searching for streamflow series. Fortunately, using WFS as a publication standard of the Metadata Table metadata will allow for this flexibility, though client applications will also need to be flexible in the metadata they expect.

Due to the increased number of web service requests by clients in the Services Stack Model, latency in response times of this more-distributed architecture could be an issue for users. This issue will have to be overcome by “intelligent” clients, as discussed in section 4.4.4. Clients will also be arguably more complicated than they are in the Network-Observations model due to the number of service specifications and data encoding standards they will need to support. As also previously stated, there is hope that off-the-shelf libraries for communicating with OGC standards would lighten the custom programming required by HIS client application developers.

Chapter 5: Conclusions

5.1 What Have We Learned

The CUAHSI-HIS project has succeeded in bringing together a large volume of hydrologic observations data from data providers across the United States. The providers of these data have included academic and research groups as well as state and national-level agencies. The information system that enables this national synthesis of water data is based on a large-scale prototype service-oriented architecture enabled primarily by CUAHSI's WaterOneFlow web services and WaterML. It is hoped that continued expansion of the sources, amounts, and types of hydrologic data available through this hydrologic information system will lead to increased research and discoveries in the hydrologic sciences and better management of water resources overall.

As with most service-oriented architectures, the current HIS service-oriented architecture, termed the Network-Observations Model, is comprised of three main components: data servers, catalogs, and clients. The data servers of this system are HydroServers, which implement the WaterOneFlow service specification to provide hydrologic data and metadata encoded in WaterML.

HydroServers are registered at HIS Central, where their series metadata are harvested into a central series metadata catalog. Typically, metadata harvesting is accomplished regularly through the GetSites and GetSiteInfo methods of WaterOneFlow. However, in the case of the large federal data providers, database dumps and custom-coded migration scripts are instead used for harvesting. Due to the length of time required for this custom harvesting, series metadata for the federal data providers happens only sporadically. The HIS Central series catalog exposes search capabilities on its central series catalog through a non-standardized web service. This service also provides clients with access to the hydrologic concept ontology that enables semantic mediation of hydrologic data from various data sources.

Clients of the CUAHSI-HIS have followed a number of distinct operating models within the service-oriented architecture. The current version of HydroExcel uses the HIS Central web service only to find a list of registered WaterOneFlow endpoints, and then operates directly with those endpoints to retrieve hydrologic metadata and data. HydroDesktop, on the other hand, is more tightly-integrated with the HIS Central series catalog. In HydroDesktop, the HIS Central services are used to find data series that match desired parameters along the what-when-where axes of the "data-cube." For matching series found from HIS Central, the time series of hydrologic data are then retrieved from corresponding HydroServers using the WaterOneFlow GetValues method. Examination of the client-driven operating models has shown that while HIS Central web service has a number of exposed operations, only the GetSeriesInBox, GetOntologyTree, and GetWaterOneFlowServices methods are used in the current architectural model.

The current system has worked well for smaller, academic and research-based data providers who would likely have not otherwise published their data for online consumption. However, sustainability concerns with the current system, particularly the HIS Central series metadata catalog, have been expressed. These issues include the tedious processes involved in harvesting series catalogs from federal water providers, maintaining custom-coded metadata harvesting programs, and debugging and maintaining the central catalog codebase. Moreover, although WaterOneFlow and WaterML have become standardized through the HIS project, obtaining buy-in to the current architecture from the federal data providers (such as USGS, EPA, and NCDC) has not been as successful as the project would like. Examination of these sustainability issues has led to the conclusion that it is not feasible for CUAHSI to maintain such a large, centralized metadata catalog. In addition, custom-coded solutions should be avoided when possible in favor of off-the-shelf software and standards.

A simpler and more general pattern for hydrologic data sharing through a service-oriented architecture has been proposed. This new model, called the Services Stack Model, is based on existing OGC web service standards and data encodings, including the forthcoming WaterML2.0 specification. The Services Stack Model relies on a *stack* of OGC services to provide catalog, metadata, and data services: CSW, WFS, and SOS, respectively. Another key difference between the proposed architectural model and the current model is that there will no longer be a centralized metadata catalog. Rather, data providers will register their services and service metadata with a CSW-compliant catalog to enable discovery of services. Series metadata, served via WFS in the Thematic Metadata Table format, will be hosted and searched upon at the data provider level.

The proposed services stack also represents a deployable system that could be hosted by data providers or other entities. Catalogs from deployed systems could be brought together into a centralized service "meta-catalog" hosted by CUAHSI's HIS Central team to facilitate searching across them. Off-the-shelf server and library implementations of these OGC standards have been identified and a proof-of-concept application has been built.

5.2 AREAS OF FUTURE RESEARCH

The Service Stack Model comprised of OGC services proposed in this thesis lays the framework for a new direction of the CUAHSI-HIS architecture. This model can be seen as a starting point for a major renovation of the current system to make it sustainable as the current HIS project grant period comes to an end. To that end, much work on a concrete implementation of the model will need to be done. Best practices for the OGC services stack will need to be formulated by the HIS project team and its stakeholders. These best practices should include a finalized specification of the Thematic Metadata Table for series description, determination of a metadata profile for the CSW-compliant services catalog, and how best to provide ontological tagging capabilities for both series metadata and services metadata.

A plan for migration to the proposed architecture will also need to be developed. This plan should aim to minimize service interruptions for clients and their users. Most of the CUAHSI-HIS products, including the ODM-based HydroServer, HydroExcel, and HydroDesktop, will need to be modified to work within the new architecture and utilize its OGC services and data encodings. Existing HydroServers at host data providers will need to be transitioned with the help of CUAHSI staff to publish their data using WFS and SOS. Extensive testing of new and modified code will need to occur to ensure a smooth transition from the current architecture.

The proposed architectural model has already started to show promise in areas of hydrologic data sharing formerly out-of-reach by the HIS. Current research by the CRWR team has indicated that an extended version of Thematic Metadata Table format shows promise for sharing wide-area gridded datasets, such as for climatologic and remote sensing data.

Appendix A: Metadata Table Field Specification

Field Name (Field Type)	**Definition**	**Example**
ServCode **(Text - 50)**	Network prefix for site codes used by the WaterOneFlow service, giving the context within which the site code applies	CCBay
SiteCode **(Text - 50)**	Unique text identifier for a site within a given WaterOneFlow service	H1
SiteName (Text - 255)	Name of a site	Hypoxia_1
VarCode **(Text - 50)**	Unique text identifier for a variable within a given WaterOneFlow service	DOC
VarName (Text - 255)	Name of a variable	Dissolved Oxygen Concentration
VarUnits (Text - 50)	Units of measure for the variable	milligrams per liter
Vocabulary **(Text - 50)**	Vocabulary prefix for variable codes giving the context within which the code applies	CCBay
Ontology (Text – 50)	Unique name for the ontology containing the concept to which the given variable has been mapped	CUAHSI Variable Ontology v1.26
Concept (Text - 50)	Leaf concept keyword from the ontology to which this variable applies	dissolvedOxygen
ValueCount (LongInt)	Number of time series values for the variable at the site for the given time period	270
StartDate **(Date)**	Start date and time for the time period of the variable at the site	5/3/94 8:40 AM
EndDate **(Date)**	End date and time for the time period of the variable at the site	8/31/06 11:26 AM
Latitude **(Double)**	Latitude of the site location in decimal degrees (WGS_1984); for polygons can be *NULL*	27.814
Longitude **(Double)**	Longitude of the site location in decimal degrees (WGS_1984); for polygons can be *NULL*	-97.141

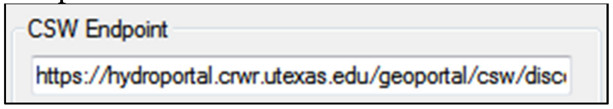
IsRegular (ShortInt)	1 (TRUE) if variable is measured/calculated regularly in time; 0 (FALSE) otherwise	0
TimeUnits (Text - 50)	For regular data, the time step and time units give the length of time between measurements, e.g., 1 day, 6.5 hrs, 1 month	Day
TimeStep (Double)	For regular data, the time step and time units give the length of time between measurements, e.g., 1 day, 6.5 hrs, 1 month	1
DataType (Text - 50)	Type of data	Value, Average, Maximum, Minimum, StandardDeviation
Medium (Text - 50)	Medium in which the variable applies	Surface Water
***MethodID* (Integer)**	Unique ID within a WaterOneFlow service for the method used to measure the variable	1
Method (Text - 255)	Description of the method used to measure the variable	Multiprobe measurement
***QCLevelID* (Integer)**	Unique ID within a WaterOneFlow service for the quality control level of the time series	0
QCLevel (Text - 50)	Description of the quality control level of the time series	Raw Data
***SourceID* (Integer)**	Unique ID within a WaterOneFlow service for the original source of the data	1
SourceName (Text - 255)	Name of the original source of the data	Texas A&M University Corpus Christi
***LocType* (Text – 25)**	Type of service – indicates how the Location parameter of a WaterOneFlow.GetValues call should be formatted	SiteCode LatLongBox LatLongPoint
***ServType* (Text – 10)**	Type of endpoint, REST, SOAP	SOAP
***XLL* (Double)**	For point data, Longitude of the point. For data defined by a lat/lon box, western longitude of the box	-97.141

YLL **(Double)**	For point data, Latitude of the point. For data defined by a lat/lon box, southern latitude of the box	27.814
XUR **(Double)**	For data defined by a lat/lon box, eastern longitude of the box; otherwise can be *NULL*	-93.5
YUR **(Double)**	For data defined by a lat/lon box, northern latitude of the box; otherwise can be *NULL*	30.2
Location (Text - 255)	Properly formatted location parameter to pass to WaterOneFlow.GetValues	CCBay:Hypoxia_1 GEOM:BOX(-97.141 27.814,-93.5 30.2) GEOM:POINT(-97.141 27.814)
Variable (Text - 255)	Properly formatted variable parameter to pass to WaterOneFlow.GetValues	CCBay:DOC NWISDV:00060/DataType=Maximum
ReqsAuth (ShortInt)	Request authorization. 1 (TRUE) if authorization for download is required; 0 (FALSE) otherwise	0
WaterMLURI **(Text - 255)**	URI of WaterOneFlow service WSDL	http://data.com/WoF/ /cuahsi_1_0.asmx?WSDL
WofVersion **(Text - 15)**	Version of the WaterOneFlow service	1.0
WFSURI (Text - 255)	URI of web feature service showing site locations	http://data.com/WFSServer
WMSURI (Text - 255)	URI of web mapping service related to the data	http://data.com/WMSServer
DAccessURI (Text - 255)	URI of Data Access Service, which provides REST querying capabilities for WaterOneFlow, user management, Metadata Table management, and more	http://data.com/DataService

Required fields are in *italics.*

Appendix B: WFSTest Operation

1) **Enter** a Catalogue Services for the Web (CSW) endpoint address in the "CSW Endpoint" text box.



CSW Endpoint

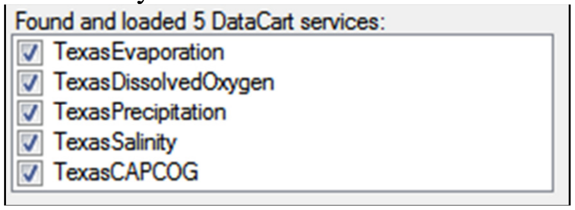
https://hydroportal.crwr.utexas.edu/geoportal/csw/disc

2) The default value in this box is for the ESRI GeoPortal Extension CSW endpoint (`https://hydroportal.crwr.utexas.edu/geoportal/csw/discovery`) at the Center for Research in Water Resources at the University of Texas. It is recommended to use the default value because we have registered and tagged our sample Web Feature Service (WFS) Metadata Table services.
3) Click the "CSW:GetRecords & WFS:GetCapabilities" button. This will send a CSW:GetRecords request to the specified CSW endpoint to find registered services that have the term "DataCart" in their Abstract metadata field.

   For each Metadata Table service found, the WFS endpoint address is extracted and each service's WFS:GetCapabilities method is called. The title for each service along with the title for the FeatureLayer within the service is saved.

   Note: This test client only expects WFS services with a single FeatureLayer.

   Once complete, the number of services found and the title of each service's FeatureLayer are shown:

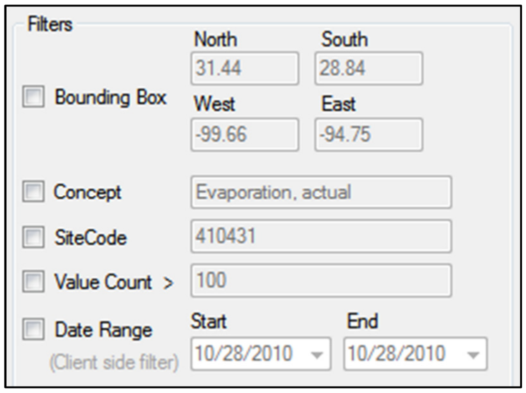


Found and loaded 5 DataCart services:

- ✓ TexasEvaporation
- ✓ TexasDissolvedOxygen
- ✓ TexasPrecipitation
- ✓ TexasSalinity
- ✓ TexasCAPCOG

4) From the “DataCart services” check box list, **select** which services you would like to query with WFS:GetFeature. All of the returned services from step 2 are selected by default.

5) From the “Filter” section, **choose** which filters you would like to apply to the WFS:GetFeature request.



Filters

Bounding Box: North 31.44, South 28.84, West -99.66, East -94.75

Concept: Evaporation, actual

SiteCode: 410431

Value Count > 100

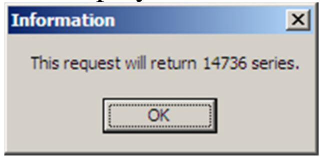
Date Range (Client side filter): Start 10/28/2010, End 10/28/2010

   - “Bounding Box” will constrain the geographic extent.
   - “Concept” will constrain the features by their Concept attribute. This field accepts wildcard characters (* and _), but it is case-sensitive.
   - “SiteCode” will constrain features by their SiteCode attribute.
   - “Value Count” will constrain the results to only those that have ValueCount attributes greater than the specified number.
   - “Date Range” will constrain results to those whose StartDate and EndDate attributes fall within the specified range.

     Note: Date Range filtering is done on the client side. This means features are first retrieved from the service and then filtered by their StartDate and EndDate. The ESRI GeoPortal does not implement OGC Filters for temporal extents.

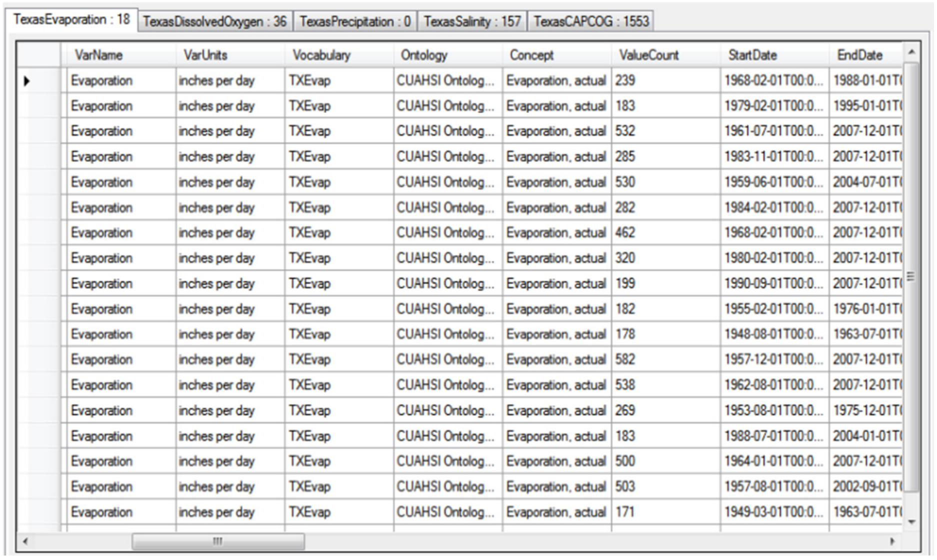
6) **Click** the “WFS:GetFeature” button to query (with the selected filters applied) each of the selected WFS Metadata Table services for their features.

   - If the “Just get series count” checkbox is checked, a message box displaying the number of features that could be retrieved with the request and selected filters will display. The "hits" result type is a native feature of WFS services.



This request will return 14736 series.

- If the "Just get hits" checkbox is **not checked**, a tab page for each selected service will appear to the right side of the application, and each tab page will contain a grid of the returned features for the associated service. Each row is a returned feature, and the columns are the Metadata Table attributes.



VarName	VarUnits	Vocabulary	Ontology	Concept	ValueCount	StartDate	EndDate
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	239	1968-02-01T00:0...	1988-01-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	183	1979-02-01T00:0...	1995-01-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	532	1961-07-01T00:0...	2007-12-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	285	1983-11-01T00:0...	2007-12-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	530	1959-06-01T00:0...	2004-07-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	282	1984-02-01T00:0...	2007-12-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	462	1968-02-01T00:0...	2007-12-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	320	1980-02-01T00:0...	2007-12-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	199	1990-09-01T00:0...	2007-12-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	182	1955-02-01T00:0...	1976-01-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	178	1948-08-01T00:0...	1963-07-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	582	1957-12-01T00:0...	2007-12-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	538	1962-08-01T00:0...	2007-12-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	269	1953-08-01T00:0...	1975-12-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	183	1988-07-01T00:0...	2004-01-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	500	1964-01-01T00:0...	2007-12-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	503	1957-08-01T00:0...	2002-09-01T
Evaporation	inches per day	TXEvap	CUAHSI Ontolog...	Evaporation, actual	171	1949-03-01T00:0...	1963-07-01T

7) To export the features in the currently selected tab, **click** the 📤 button. This will open a Save File Dialog and save the features to a comma-separated values (.csv) file at the specified location.

Appendix C: List of Acronyms

CQL	Common Query Language
CRS	Coordinate Reference System
CRWR	Center for Research in Water Resources
CSW	Catalogue Services for the Web
CUAHSI	Consortium of Universities for the Advancement of Hydrologic Science, Inc.
EPA	Environmental Protection Agency
FES	Filter Encoding Standard
FGDC	Federal Geospatial Data Committee
GIS	Geographic Information System
GML	Geographic Markup Language
HDWG	Hydrology Domain Working Group
HIS	Hydrologic Information System
HTTP	Hypertext Transfer Protocol
KVP	Key-Value Pair
NCDC	National Climatic Data Center
NWIS	National Water Information System
O&M	Observations & Measurements
ODM	Observations Data Model
OGC	Open Geospatial Consortium
REST	REpresentation State Transfer
SDSC	San Diego Supercomputer Center
SOA	Services-Oriented Architecture
SOAP	(formerly) Simple Object Access Protocol
SOS	Sensor Observation Service
SRS	Spatial Reference System

STORET	STOrage and RETrieval
TCEQ	Texas Commission on Environmental Quality
USGS	United States Geological Survey
WaterML	Water Markup Language
WCS	Web Coverage Service
WFS	Web Feature Service
WMO	World Meteorological Organization
WMS	Web Map Service
WOF	WaterOneFlow
XML	eXtensible Markup Language

References

CUAHSI. *CUAHSI Mission* . 2010. http://www.cuahsi.org/mission.html (accessed October 20, 2010).

CUAHSI-HIS. *What is the CUAHSI Hydrologic Information System (CUAHSI-HIS)?* 2010. http://his.cuahsi.org/project.html (accessed October 20, 2010).

Endrei, Mark, et al. *Patterns: Service-Oriented Architecture and Web Services.* Durham: IBM, 2004.

Fielding, Roy Thomas. *Architectural styles and the design of network-based software architectures.* Doctoral Dissertation, Irvine: University of California, 2000, 180.

Fierro, Jr., Pedro. *The Water Encyclopedia: Hydrologic Data and Internet Resources.* Hoboken: CRC, 2007.

Horsburgh, Jeffery S., et al. "An integrated system for publishing environmental observations data." *Environmental Modelling & Software* 24 (2009): 879-888.

Josuttis, Nicolai M. *SOA in Practice: The Art of Distributed System Design.* Sebastopol: O'Reilly Media, 2007.

Langefors, Börje. *Theoretical Analysis of Information Systems.* 4th Edition. Philadelphia: Auerbach Publishers, 1973.

Maidment, David R. (ed.). *CUAHSI Hydrologic Information System: 2009 Status Report.* Consortium of Universities for the Advancement of Hydrologic Science, Inc., 2009.

Marble, Duane F. "Geographic Information Systems: an Overview." *Proceedings, Pecora 9 Conference.* Sioux Falls, 1984. 18-24.

Nickul, Duane (ed.). *Service Oriented Architecture (SOA) and Specialized Messaging Patterns.* Technical White Paper, San Jose: Adobe, 2007.

OGC. *FGDC CSDGM Application Profile for CSW 2.0.* October 29, 2006. http://portal.opengeospatial.org/files/?artifact_id=16936 (accessed November 10, 2010).

—. *OGC Standards Officially Endorsed by Federal Geographic Data Committee.* October 12, 2010. http://www.opengeospatial.org/pressroom/pressreleases/1294 (accessed November 18, 2010).

—. *OpenGIS (R) Catalogue Services Implementation Specification.* February 23, 2007. http://portal.opengeospatial.org/files/?artifact_id=20555 (accessed October 31, 2010).

—. *OpenGIS (R) Filter Encoding Implementation Specification.* May 3, 2005. http://portal.opengeospatial.org/files/?artifact_id=8340 (accessed November 10, 2010).

—. *OpenGIS (R) Web Map Server Implementation Specification.* March 15, 2006. http://portal.opengeospatial.org/files/?artifact_id=14416 (accessed October 31, 2010).

—. *Sensor Observation Service.* October 26, 2007. http://portal.opengeospatial.org/files/?artifact_id=26667 (accessed October 31, 2010).

—. *Web Feature Service Implementation Specification.* May 3, 2005. http://portal.opengeospatial.org/files/?artifact_id=8339 (accessed October 31, 2010).

Pautasso, Cesare, Olaf Zimmerman, and Frank Leymann. "Restful web services vs. "big"' web services: making the right architectural decision." *WWW '08: Proceedings of the 17th international conference on World Wide Web.* Beijing: ACM, 2008. 805-814.

Piasecki, Michael. *HydroTagger Functional Description Document (Version 1.0).* CUAHSI, 2008.

Rosen, Michael, Boris Lublinsky, and Kevin T. Smith. *Applied SOA: Servce-Oriented Architecture and Design Strategies.* Hoboken: Wiley & Sons, 2008.

Tarboton, David G., David Maidment, Ilya Zaslavsky, Daniel P. Ames, Jon Goodall, and Jeffery S. Horsburgh. "CUAHSI Hydrologic Information System 2010 Status Report." 2010.

Tarboton, David G., Jeffery S. Horsburgh, and David R. Maidment. *CUAHSI Community Observations Data Model (ODM).* Design Specifications, CUAHSI, 2007.

Tarboton, David G., Jeffery S. Horsburgh, and David R. Maidment. *CUAHSI Community Observations Data Model (ODM) Version 1.0.* Design Specifications, CUAHSI, 2007.

W3C. *Hypertext Transfer Protocol -- HTTP/1.1.* 1999. http://www.w3.org/Protocols/rfc2616/rfc2616.html (accessed November 10, 2010).

—. *SOAP Version 1.2.* April 27, 2007. http://www.w3.org/TR/soap12-part1/ (accessed November 11, 2010).

—. *Web Services Architecture.* February 11, 2004. http://www.w3.org/TR/ws-arch/#whatis (accessed October 14, 2010).

Whiteaker, Tim. *CUAHSI WaterOneFlow Workbook.* Austin: CUAHSI, 2010.

Whiteaker, Tim. "The CUAHSI Hydrologic Information System." Presentation, 2010.

Whiteaker, Tim, and Dean Djokic. "Data Cart Specification." 2010.

Whitenack, Tom. *CUAHSI HIS Central 1.2*. CUAHSI, 2010.

Zaslavsky, Ilya, David Valentine, and Tim Whiteaker. *CUAHSI WaterML.* Discussion Paper, Open Geospatial Consortium, 2007.

Vita

James Adam Seppi was born in Maryland to parents Eleanor and Bruno J. on February 11, 1984. He attended primary school in Bowie, MD and graduated with honors from the Science and Technology program at Eleanor Roosevelt High School in Greenbelt, MD. James attended the University of Maryland where he graduated with degrees in Computer Engineering (B.S.) and Chinese (B.A.) in 2007. While at the University of Maryland, James became a member of the Tau Beta Pi, Eta Kappa Nu, and Omicron Delta Kappa honor societies. James worked as an analyst consultant at Accenture in Austin, TX before beginning his graduate studies in Environmental and Water Resources Engineering at the University of Texas at Austin in January of 2009. After graduation, James and his wife Taylor plan on traveling and volunteering abroad.

Permanent email: james.seppi@gmail.com

This thesis was typed by the author.