Deployment of geologic map services using GeoSciML-Portrayal

An introduction to a simple feature schema for supporting map services and guidelines for service deployment

A product of the Arizona Geological Survey

3/27/2012

Contents

1 Introduction

The key to interoperable geologic map services is the ability to portray adjacent maps using the same symbolization scheme such that when the maps are displayed, the visual discontinuity at the boundary is minimized. Because of the discrepancies in mapping interpretation and intention, there are commonly differences in the definition of map units between geologic maps produced by different authors or at different times. Resolving such differences is a compilation process that often requires additional field work; this is outside the scope of service deployment. What can be avoided is visual discontinuity due to portrayal schemes that assign different colors to similar units on adjacent maps.

GeoSciML-portrayal addresses this issue by requesting that polygons representing geologic unit outcrops be categorized using identifiers from the CGI Simple Lithology vocabulary and the ICS stratigraphic time scale. The use of a shared map legend based on these categorization schemes provides first order lithologic and age harmonization between maps provided by different services. Map services from OneGeology Europe provide an example of this approach (see http://onegeology-europe.brgm.fr/geoportal/viewer.jsp). The GeoSciML-portrayal schema provides an additional symbol-identification property to enable preservation of the legend units and portrayal from the original source map. This document is designed to facilitate the deployment and maintenance of geoscience information as interoperable, publicly available OGC web services in accordance with USGIN and OneGeology specifications.

1.1 What is USGIN?

The United States Geoscience Information Network (USGIN) is a distributed data-sharing network that uses open-source software and existing World Wide Web infrastructure and browsers. The USGIN initiative is the product of a partnership between the Association of American State Geologists (AASG) and the United States Geological Survey (USGS) created to facilitate discovery of, and access to, geoscience information provided by state and federal geological surveys of the United States.

USGIN is:

- conceptual framework for sharing geoscience data
- a distributed data-sharing network
- a collection of open-source applications, standards, procedures, and protocols for sharing geoscience data
- web-based, distributed, open-source, and interoperable

Anyone can access the data shared via USGIN, and any data that is shared according to USGIN specifications is automatically a part of the network. For more information, see the USGIN website (http://usgin.org/).

1.2 GeoSciML-Portrayal and GeoSciML

GeoSciML-portrayal is a simplified **view** of GeoSciML (Geoscience Markup Language, see Richard and CGI Interoperability Working Group, 2007[1]), which is a specialized XML markup language designed for sharing geoscience data between different computer systems (Figure 1). GeoSciML is based on a complex information model developed to support interchange of a wide variety of detailed geologic information. GeoSciML-portrayal is an XML markup language that implements a small subset of the GeoSciML information model specifically for interoperable map services delivering geologic map unit, contact, and shear displacement structure (fault and ductile shear zone) features. The use of standard vocabularies with this schema enables map portrayal using shared legends to achieve basic visual harmonization of maps, even when these maps are provided by different services.

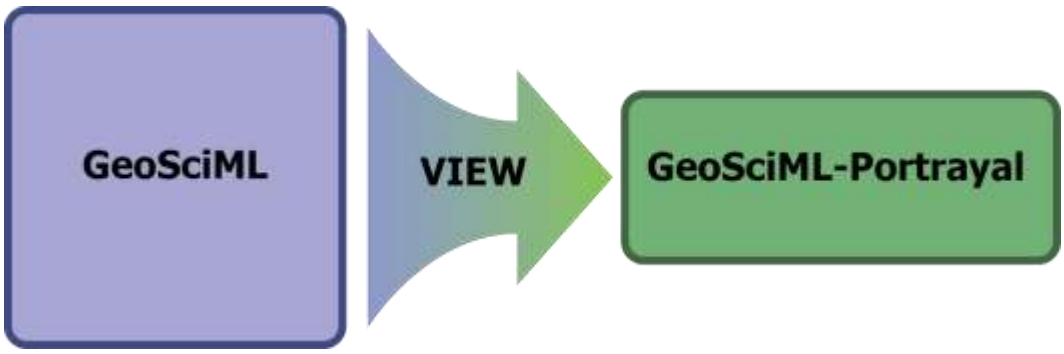

Figure 1: A visual representation of the relationship between GeoSciML and GeoSciML-Portrayal

GeoSciML-portrayal is considered a **view** of the GeoSciML information model. In the context of a database, a **view** is a selection of fields. For example, given a database with twelve fields, an arbitrary grouping of four such fields would constitute a **view** of the database. A more concrete example is a card catalog: if one chose to look *only* at the **Title** field of each card in the catalog, this choice would constitute a discrete **view** of the catalog. In XML markup languages element tags are analogous to database fields. The structure of GeoSciML-portrayal documents is specified by an XML schema. This document describes the GeoSciML-portrayal schema that structures documents written using the GeoSciML-portrayal markup language.

GeoSciML is an XML markup language developed as an OpenGeospatial Consortium (OGC) Geography Markup Language (GML) v3.2 application for encoding a wide variety of geoscientific information (Richard et al., 2007); GeoSciML defines a collection of complex GML features for describing geoscience entities. The GeoSciML-portrayal schema simplifies GeoSciML by merging complex property values into single, human-readable text fields and assigning single, representative identifiers from controlled vocabularies for lithology and age that can be used for

[1] Richard, S. M., and CGI Interoperability Working Group, 2007, GeoSciML – A GML Application for Geoscience Information Interchange, in Soller, D.R., ed., Digital Mapping Techniques '07—Workshop Proceedings: U.S. Geological Survey Open-File Report 2007–1285, p. 47-59. Available at http://pubs.usgs.gov/of/2007/1285/pdf/Richard.pdf (accessed 4/10/2012).

standardized map legends. Use of standard vocabulary enables map portrayal using shared legends for maps provided by different services. In addition the scheme includes text information for human users when browsing a geologic map, a link to a full GeoSciML feature element if available, and a symbol identifier field to enable a user-defined symbolization scheme in each map service. Most WMS implementations simplify deployment of WFS service presenting features using the same schema. Linking the simple-feature WMS and WFS allows clients to acquire basic text geologic feature descriptions that can be used in web-mapping applications to construct custom legends. Linking to full GeoSciML features allows the portrayal scheme to be used in a map browsing and query interface to identify and select features for further processing that can be acquired as highly structured, information-rich, complex GML features.

GeoSciML-portrayal conforms to the level 0 of the Simple Features Profile for GML (OGC 10-100r3 - van den Brink et al., 2011; OGC 06-049). The simple features profile supports only a limited subset of possible GML geometry types that may be used to describe feature geographic location and shape. For the purposes of GeoSciML simple features, these include gml:Point, gml:LineString, gml:Curve, gml:Polygon, gml:Surface, gml:MultiPoint, gml:MultiCurve, gml:MultiSurface and multi-geometry types consisting of collections of these base types. For a useful discussion of GML simple vs. complex features, see this Geoserver documentation page.

GeoSciML-portrayal features are analogous to GeoSciML mapped features with additional text attributes for human consumption, a flatted-relation view of the age, and assignment to a single lithology. The portrayal scheme consists of 'free-text' fields and identifier fields. In robust services the free-text fields will contain well-structured summaries of complex GeoSciML data in a format suitable for reading by the intended users. Identifier fields should contain identifiers for concepts in a controlled vocabulary (for example CGI Simple Lithology) that specify representative thematic properties. Inclusion of these standardized identifiers enables interoperability across services. Ideally these should be URIs that can be dereferenced to obtain machine-processable or human-readable representations of the identified concepts.

In addition, each GeoSciML-portrayal feature includes an (optional) identifier for a specification, which is a resource containing a description of that particular feature. In many cases, the descriptions will be the same for all polygons assigned to the same map unit or classified as the same kind of contact or structure. If more complete information is available, different descriptions may be associated with subsets of features of the same type that are portrayed with the same symbol. In the most extreme case, each feature might have a unique description that captures the full spatial variability of a geologic unit or structure. Following the standard patterns of web architecture, the specification_uri should be dereferenceable to obtain one or more representations of that description. For maximum interoperability, one of these representations should be a GeoSciML-encoded description of the feature, but other encodings might also be available, for example html web pages, other XML schema, or JSON. For those familiar with full GeoSciML v3.0, the specification_uri property is equivalent to the specification association from MappedFeature to GeologicFeature.

Deployment of an interoperable dataset requires first that the interchange format used for interoperability is well understood. The following sections are intended to provide the necessary background understanding of GeoSciML-portrayal, and should be studied carefully. The deployment process consists of determining how the information in the dataset to be published can best be represented using the elements in the GeoSciML-portrayal model, populating the feature collections that will be served, and configuring a WMS server to display the data. To keep costs low and promote adoption, it is possible to perform all of the steps outlined in this cookbook using free-and-open-source software.

1.3 Cookbook Prerequisites

You will need the following:

1. Geologic data in tabular form, for example:
    a. File geodatabases
    b. Microsoft Access databases
    c. Personal geodatabases
    d. PostGIS databases
    e. SDE databases
    f. Shapefiles
2. Server software that implements OGC web services:
    a. GeoServer and accompanying software
    b. ESRI ArcGIS Server (proprietary, but widely used)
3. Access to a server with sufficient permissions to load data and deploy service instances
4. GIS software for testing services; examples include:
    a. ArcGIS
    b. Grass GIS
    c. gvSIG
    d. OpenLayers
    e. PostGIS plugin for PostgreSQL DBMS
    f. Quantum GIS
    g. uDIG

Notes:

- Server software applications capable of providing web services:
    - **ESRI ArcGIS Server** is proprietary server software. It can utilize data in ESRI geodatabase format stored in a variety of relational databases (including PostGIS), in shape files, or in ESRI file geodatabases.
    - **GeoServer** is actively maintained, free-and-open-source server software and can utilize data from PostgreSQL databases with the PostGIS-plugin or in ESRI Shapefiles. . GeoServer can be obtained at the following web locations:
        - http://geoserver.org/
        - http://opengeo.org/technology/GeoServer/

- **PostGIS** is a plugin for **PostgreSQL** database management system (DBMS) that provides geographic information functions.
- **PostgreSQL** DBMS is a free-and-open-source relational database management system
- A database management system is a program that controls the creation, maintenance, and use of a database; numerous examples exist, including:
    - Relational DBMS, such as MRDS, Rel, and Micro DBMS
    - SQL DBMS, such as INGRES, DB2, and Microsoft SQL Server
        - SQL DBMS are so named because they use Standardized Query Language (SQL)
    - NoSQL DBMS, such as memcached, Redis, and CouchDB
- NoSQL DBMS do not use SQL as their primary **query** language (NoSQL means "not only SQL")pgAdmin III is a free, open-source application that provides a user interface to create and edit databases as well as data tables. An introduction to pgAdmin as well as documentation for using the software can be found on their website here.

For information on PostGreSQL and deploying services using multiple geodatabases, click here.

1.4 Cookbook Workflow

Setting up a geologic data web map service involves the following steps:

1. Map your source data from its original schema to the GeoSciML-portrayal schema; this step can be further broken down into the following sub-steps
    a. Map existing content fields to corresponding GeoSciML-portrayal schema elements
    b. Determine controlled vocabulary terms for lithology and age to assign for each map unit
    c. Load data from the original database format to tables or files required by the server software used to deploy the web service
2. Establish symbology for lithostratigraphic polygon portrayal that can be used by the map server.
3. Deploy your web service, create metadata for dataset and service, and, if desired, register service with OneGeology

2 GeoSciML-Portrayal Feature Types

Within the scope of the GeoSciML-portrayal markup language, three feature classes are defined:

- **ContactView**: contains features that represent the mapped traces of boundaries between geologic units
- **ShearDisplacementStructureView**: contains features that represent the mapped trace of any type of fault or shear zone that is treated as a single surface for map portrayal
    - The concept of 'Shear displacement structure' includes all fault types or shear zones along which displacement has occurred, from a simple, single 'planar' brittle or ductile surface to a fault system comprising multiple strands of both brittle and

ductile nature. Because this feature class is constrained to have a linear geometry, representation is limited to shear displacement structures that are considered single surfaces at the scale of portrayal.

- **GeologicUnitView**: contains polygon features that represent the outcrop of a geologic unit mapped on some outcrop surface called the 'map horizon'. Example map horizons include:
    - Earth surface: the most common map horizon for a geologic map
    - Top of basement: bedrock on which a stratigraphic section of interest is deposited. For example, in the Grand Canyon, the Precambrian crystalline rocks in the inner gorge (e.g. Vishnu schist) would be considered the basement.
    - Mission-Pima open-pit mine, 6/20/1990

  Since the map horizon is not specified in each feature's properties, it must be described in the metadata for the feature collection. Overlapping polygons representing outcrops on different horizons (e.g. 'Earth surface' and 'bedrock surface') will be represented as distinct features in different GeoSciML-portrayal services. Each GeoSciML-portrayal service provides geologic unit outcrop polygons, associated contacts between the units, and fault traces on a single map horizon at any particular location in the extent of the feature collection.

For more information about the relationship between GeoSciML and GeoSciML-portrayal, refer back to the GeoSciML-Portrayal section.

2.1 ContactView Features

These features represent linear mapped features that are specified as GeoSciML Contact Features (Figure 2).

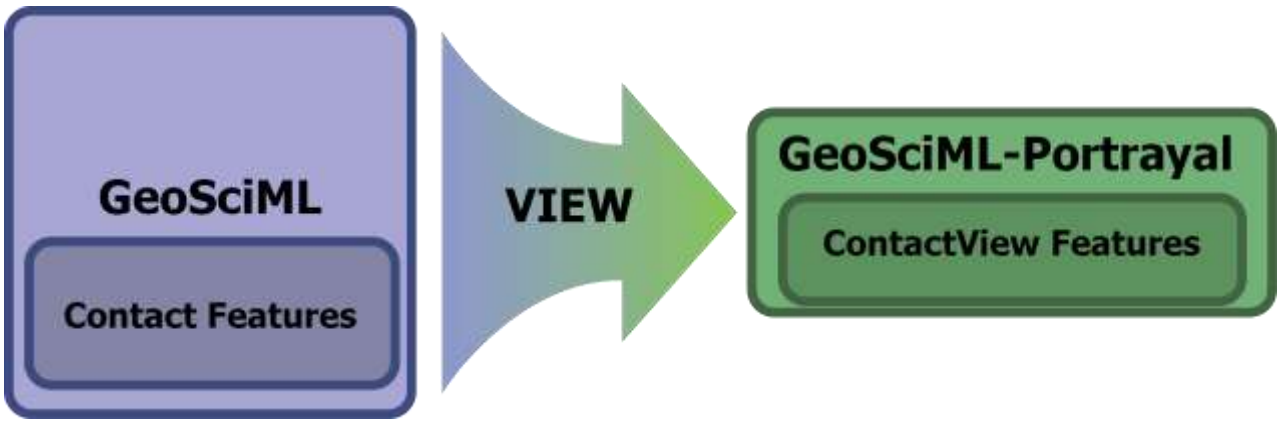

Figure 2: The relationship between GeoSciML Contact Features and GeoSciML ContactView Features

Table 1 describes each of the elements in the **ContactView** feature schema; in this table:

- **Name** is the name of an element that describes a given feature
- **Implementation data type** indicates the XML data type used to implement the element. The xs prefix is an abbreviation for the XML schema namespace:
    - AnyURI: This element will contain an appropriate URI

        - URIs can be used to refer to other resources, such as controlled concepts in published vocabularies
    - String: This element will contain a free-text series of alphanumeric characters
    - Decimal: This element will contain a numeric value

  Note that these values are modified by the **xs:** prefix; the xs: prefix indicates conformity to the [W3C XML Schema]
- **Notes** contains a general description of a given element

Table 1: ContactView feature scheme

Name	Implementation data type	Notes
identifier	xs:AnyURI	Globally unique identifier for the individual feature. Recommended practice is that this identifier be derived from the primary key for the spatial objects in the source data in case information needs to be transferred from the interchange format back to the source database. This identifier is analogous to the identifier for a GeoSciML MappedFeature.
name	xs:string	Display name for the Contact. Examples: 'depositional contact', 'unconformity', 'Martin-Escabrosa contact.'
description	xs:string	Text description of the contact; may be a generic description of a contact type taken from an entry on a geological map legend or a more specific description of the particular contact.
contactType	xs:string	Text label specifying the kind of surface separating two geologic units including primary boundaries such as depositional contacts, all types of unconformities, intrusive contacts, and gradational contacts, as well as faults that separate geologic units. Ideally this would be the preferred label for the concept identified by contactType_uri.
observationMethod	xs:string	Metadata snippet indicating how the spatial extent of the feature was determined. ObservationMethod is a convenience property that provides a quick and dirty approach to observation metadata.
positionalAccuracy	xs:decimal	Quantitative values define the radius of an uncertainty buffer around a mappedFeature (e.g. a positionAccuracy of 100 m for a line feature defines a buffer polygon of total width 200 m centered on the line).
source	xs:string	Text describing feature-specific details and citations to source materials, and if available providing URLs to reference material and publications describing the geologic feature. This could be a short text synopsis of key information that would also be in the metadata record referenced by metadata_uri.
contactType_uri	xs:AnyURI	URI referring to a controlled concept from a vocabulary defining the Contact types. Mandatory property - if no value is provided then a URI referring to a controlled concept explaining why the value is nil must be provided.
specification_uri	xs:AnyURI	URI referring the GeoSciML Contact feature that describes the instance in detail. Mandatory property - if no value is provided then a URI referring to a controlled concept explaining why the value is nil must be provided.
metadata_uri	xs:AnyURI	URI referring to a formal metadata record describing the provenance of data.
genericSymbolizer	xs:string	Identifier for a symbol from standard (locally or community defined) symbolization scheme for portrayal. There should be an accompanying SLD file. available defining the symbol associated with each genericSymbolizer value.

Name	Implementation data type	Notes
shape	GM_Object	Geometry defining the extent of the feature of interest. This is the only element with complex content, and must contain a GML geometry that is valid for the Geography Markup Language (GML) simple features profile (OGC 06-049r1.)

In GeoSciML terms, GeoSciML-portrayal ContactView Features will be an instance of a GeoSciML MappedFeature with key property values from the associated ContactFeature summarized in text (data type xs:string) fields as well as properties suffixed with '_uri' containing URIs referring to other resources (for example, controlled concepts in published vocabularies).

2.2 ShearDisplacementView Features

These features represent linear mapped features that are specified as GeoSciML ShearDisplacementStructure Features (Figure 3).

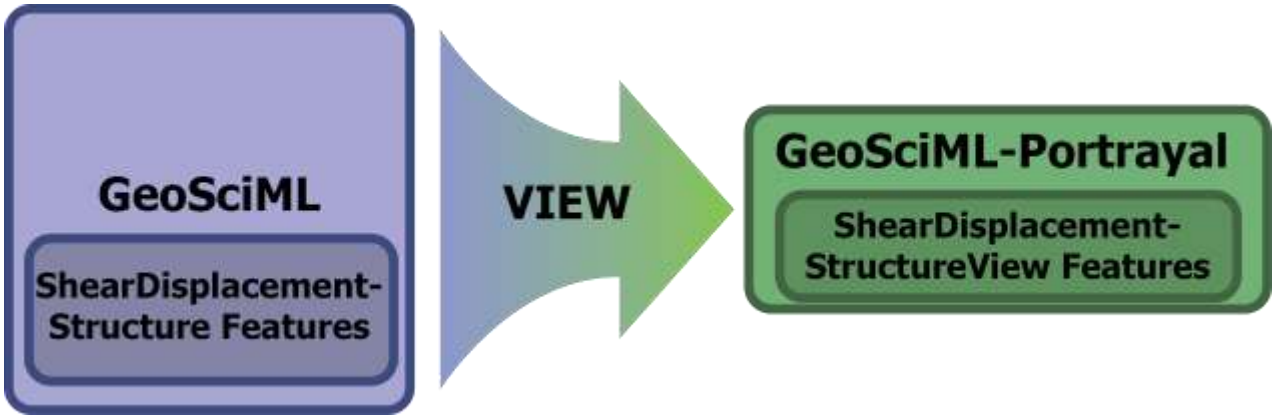

Figure 3: The relationship between GeoSciML ShearDisplacementStructure Features and GeoSciML ShearDisplacementStructureView Features

Table 2 describes each of the elements in the **ShearDisplacementStructure** feature schema; in this table:

- **Name** is the name of an element that describes a given feature
- **Implementation data type** indicates the specifications for a given element:
  - AnyURI: This element will contain an appropriate URI
    - URIs can be used to refer to other resources, such as controlled concepts in published vocabularies
  - String: This element will contain a free-text series of alphanumeric characters
  - Decimal: This element will contain a numeric value

  Note that these values are modified by the **xs:** prefix; the xs: prefix indicates conformity to the W3C XML Schema
- **Notes** contains a general description of a given element

Table 2: Elements in ShearDisplacementStructureView feature scheme

Name	Type	Notes
identifier	xs:AnyURI	Globally unique identifier for the individual feature. Recommended practice is that this identifier be derived from the primary key for the spatial objects in the source data in case information needs to be transferred from the interchange format back to the source database. This identifier is analogous to the identifier for a GeoSciML MappedFeature.
name	xs:string	Display name for the the ShearDisplacementStructure. This may be a generic fault type (e.g. 'thrust fault', 'strike-slip fault') or a particular fault name (e.g. 'Moine thrust', 'San Andreas Fault').
description	xs:string	Text description of the ShearDisplacementStructure, typically taken from an entry on a geological map legend.
faultType	xs:string	Type of ShearDisplacementStructure (as defined in GeoSciML).
movementType	xs:string	Summary of the type of movement (e.g. dip-slip, strike-slip) on the ShearDisplacementStructure.
deformationStyle	xs:string	Description of the style of deformation (e.g. brittle,ductile etc) for the ShearDisplacementStructure.
displacement	xs:string	Text summary of displacement across the ShearDisplacementStructure.
geologicHistory	xs:string	Text (possibly formatted with formal syntax) description of the age of the ShearDisplacementStructure (where age is a sequence of events and may include process and environment information).
observationMethod	xs:string	Metadata snippet indicating how the spatial extent of the feature was determined. ObservationMethod is a convenience property that provides a quick and dirty approach to observation metadata when data are reported using a feature view (as opposed to observation view).
positionalAccuracy	xs:string	Quantitative values which define the radius of an uncertainty buffer around a mappedFeature (eg: a positionAccuracy of 100 m for a line feature defines a buffer polygon of total width 200 m centered on the line).
source	xs:string	Text describing feature-specific details and citations to source materials, and if available providing URLs to reference material and publications describing the geologic feature. This could be a short text synopsis of key information that would also be in the metadata record referenced by metadata_uri.
faultType_uri	xs:AnyURI	URI referring to a controlled concept from a vocabulary defining the fault type (ShearDisplacementStructure). Mandatory property - if no value is provided then a URI referring to a controlled concept explaining why the value is nil must be provided.
movementType_uri	xs:AnyURI	URI referring to a controlled concept from a vocabulary defining the ShearDisplacementStructure movement type. Mandatory property - if no value is provided then a URI referring to a controlled concept explaining why the value is nil must be provided.
deformationStyle_uri	xs:AnyURI	URI referring to a controlled concept from a vocabulary defining the ShearDisplacementStructure deformation style. Mandatory property - if no value is provided then a URI referring to a controlled concept explaining why the value is nil must be provided.
representativeAge_uri	xs:AnyURI	URI referring to a controlled concept specifying the most representative stratigraphic age interval for the GeologicUnit. This will be defined entirely at the discretion of the data provider and may be a single event selected from the geologic feature's geological history or a value summarizing the all or part of the feature's history.

Name	Type	Notes
representativeLowerAge_uri	xs:AnyURI	URI referring to a controlled concept specifying the most representative lower value in a range of stratigraphic age intervals for the GeologicUnit. This will be defined entirely at the discretion of the data provider and may be a single event selected from the geologic feature's geological history or a value summarizing the all or part of the feature's history.
representativeUpperAge_uri	xs:AnyURI	URI referring to a controlled concept specifying the most representative upper value in a range of stratigraphic age intervals for the GeologicUnit. This will be defined entirely at the discretion of the data provider and may be a single event selected from the geologic feature's geological history or a value summarizing the all or part of the feature's history.
specification_uri	xs:AnyURI	URI referring the GeoSciML ShearDisplacementStructure feature that describes the instance in detail. Mandatory property - if no value is provided then a URI referring to a controlled concept explaining why the value is nil must be provided.
metadata_uri	xs:AnyURI	URI referring to a metadata record describing the provenance of data.
genericSymbolizer	xs:string	Identifier for a symbol from standard (locally or community defined) symbolization scheme for portrayal.
shape	GM_Object (GM_curve)	Geometry defining the extent of the feature of interest.

In GeoSciML terms this will be an instance of a MappedFeature with key property values from the associated ShearDisplacementStructure feature summarized in text fields (data type xs:string) as well as fields containing identifiers (URI) for fault type, deformation style, movement type, geologic age, and a formally-encoded (ideally in GeoSciML) specification for interoperability. The latter are the properties suffixed with '_uri' and will contain URIs referring to other resources, for example controlled concepts in published vocabularies.

2.3 GeologicUnitView Features

These features represent polygon mapped features that are specified as GeoSciML GeologicUnit Features (Figure 4), which are used to provide properties associated with a body of material in the earth.

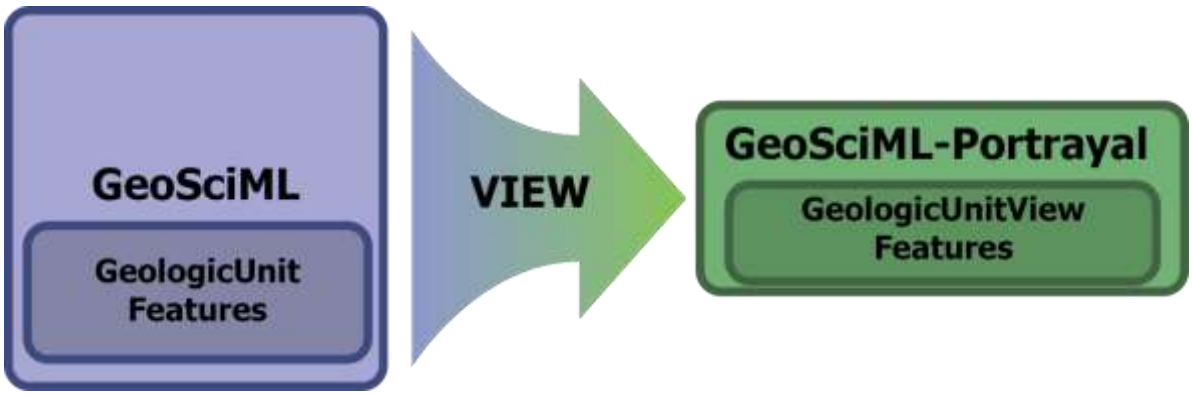

Figure 4: The relationship between GeoSciML GeologicUnit Features and GeoSciML GeologicUnitView Features

Table 3 describes each of the elements that describe **ShearDisplacementStructure Features**; in this table:

- **Name** is the name of an element that describes a given feature
- **Implementation data type** indicates the specifications for a given element:
    - AnyURI: This element will contain an appropriate URI
        - URIs can be used to refer to other resources, such as controlled concepts in published vocabularies
    - String: This element will contain a free-text series of alphanumeric characters
    - Decimal: This element will contain a numeric value

  Note that these values are modified by the **xs:** prefix; the xs: prefix indicates conformity to the W3C XML Schema
- **Notes** contains a general description of a given element

Table 3: Elements in GeologicUnitView feature class

Name	Type	Notes
identifier	xs:AnyURI	Globally unique identifier for the individual feature. Recommended practice is that this identifier be derived from the primary key for the spatial objects in the source data in case information needs to be transferred from the interchange format back to the source database. This identifier is analogous to the identifier for a GeoSciML MappedFeature.
name	xs:string	Display name for the GeologicUnit; this may be populated by a geologic unit name, or more likely, an abbreviation used to label outcrops of the unit in a map display.
description	xs:string	Text description of the GeologicUnit, typically taken from an entry on a geological map legend.
geologicUnitType	xs:string	Type of GeologicUnit (as defined in GeoSciML).
rank	xs:string	Stratigraphic rank of GeologicUnit (as defined in GeoSciML). Examples include formation, member, group, or supergroup.
lithology	xs:string	Text (possibly formatted with formal syntax) description of the GeologicUnit's lithology.
geologicHistory	xs:string	Text (possibly formatted with formal syntax) description of the age of the GeologicUnit (where age is a sequence of events and may include process and environment information).
observationMethod	xs:string	Metadata snippet indicating how the properties of the feature were determined. ObservationMethod is a convenience property that provides a quick and dirty approach to observation metadata. Example values might include 'field observation by author', 'compilation from published maps', 'air photo interpretation'.
positionalAccuracy	xs:string	Quantitative values define the radius of an uncertainty buffer around a mappedFeature (eg: a positionAccuracy of 100 m for a line feature defines a buffer polygon of total width 200 m centered on the line).
source	xs:string	Text describing feature-specific details and citations to source materials, and if available providing URLs to reference material and publications describing the geologic feature. This could be a short text synopsis of key information that would also be in the metadata record referenced by metadata_uri.

Name	Type	Notes
geologicUnitType_uri	xs:AnyURI	URI referring to a controlled concept from a vocabulary defining the GeologicUnit types. Mandatory property - if no value is provided then a URI referring to a controlled concept explaining why the value is nil must be provided.
representativeLithology_uri	xs:AnyURI	URI referring to a controlled concept specifying the characteristic or representative lithology of the unit. This may be a concept that defines the super-type of all lithology values present within a GeologicUnit or a concept defining the lithology of the dominant CompositionPart (as defined in GeoSciML) of the unit. Typically this identifier might be used as the symbol key for a lithologic map portrayal of the geologic unit features.
representativeAge_uri	xs:AnyURI	URI referring to a controlled concept specifying the most representative stratigraphic age interval for the GeologicUnit. This will be defined entirely at the discretion of the data provider and may be a single event selected from the geologic feature's geological history or a value summarizing all or part of the feature's history. Typically, this identifier might be used as a symbol key for a geologic-age-based portrayal of the geologic unit features.
representativeLowerAge_uri	xs:AnyURI	URI referring to a controlled concept specifying the most representative younger value in a range of stratigraphic age intervals for the GeologicUnit. This will be defined entirely at the discretion of the data provider and may be a single event selected from the geologic feature's geological history or a value summarizing all or part of the feature's history.
representativeUpperAge_uri	xs:AnyURI	URI referring to a controlled concept specifying the most representative older value in a range of stratigraphic age intervals for the GeologicUnit. This will be defined entirely at the discretion of the data provider and may be a single event selected from the geologic feature's geological history or a value summarizing all or part of the feature's history.
specification_uri	xs:AnyURI	URI for a complete description of the geologic unit cropping out within the extent of the feature's geometry. Preferred representation is a GeoSciML GeologicUnit feature instance. Mandatory property - if no value is provided then a nil reason URI explaining why the value is nil must be provided.
metadata_uri	xs:AnyURI	URI referring to a metadata record describing the provenance of data.
genericSymbolizer	xs:string	Identifier for a symbol from standard (locally or community defined) symbolization scheme for portrayal.
shape	GM_Object, (GM_polygon)	Geometry defining the extent of the feature of interest.

In GeoSciML terms GeologicUnitView features are instances of MappedFeature with key property values from the associated GeologicUnit feature summarized in text fields (data type xs:string) for human data consumers, with standard identifiers for geologic unit type, representative lithology, and geologic age. The specification_uri identifies a description resource specific to the geologic unit cropping out in the extent of the polygon (or other) geometry of the feature. The specification_uri should dereference to yield a formally-encoded representation of the geologic unit, ideally in GeoSciML for interoperability. Properties populated by identifiers are suffixed with '_uri' and contain URIs referring to other resources, for example controlled concepts in published vocabularies.

3 Deployment Cookbook

Once geologic map data is automated into a structured digital form, publication of the data for access as an Open Geospatial Consortium (OGC) web map service (WMS) or web feature service (WFS) is a three-step process:

1. Extract the data from its source data set
2. Transform the data to an interchange format
    - In the context of this cookbook, the GeoSciML-portrayal schema constitutes the appropriate interchange format
3. Load the transformed data into whatever sort of data store is most convenient for the server implementation of the OGC WMS and WFS

This process is commonly called 'ETL'.

There exists a broad spectrum of approaches to the ETL process. We focus in this tutorial on what we deem to be a common situation in which the source data is in a database feature class (possibly in an ESRI geodatabase, shape file, or PostGIS table) with one record for each geometric feature (line or polygon), each feature has either a link to a description table that contains the required elements for the interchange format, or includes fields specifying required content in each record. Source information for the map data and geologci unit descriptions is also required.

The most difficult part of the ETL process is typically determining what representative lithology, age, younger age and older age categories from a standard vocabulary to assign to each geologic unit. OneGeology conformance proscribes that these vocabularies should be the CGI SimpleLithology vocabulary for lithology and the International Stratigraphic Commission Geologic Time Scale 2009 for age (see vocabulary links at http://resource.geosciml.org/).

3.1 Mapping from source data

The following outline describes the basic ETL workflow:

1. Determine which fields in the source data contain the information that is to be delivered in the interchange format fields. Multiple source fields may be combined into single interchange fields, and a single source field may impact values in multiple interchange fields.
2. Determine what steps are necessary to get the content into the interchange format. This may involve some calculation, such as concatenating text from multiple fields to populate requisite fields in the interchange format. Use of the standard vocabularies for interoperability will likely require mapping vocabulary terms in the source data to identifiers for concepts in the controlled vocabularies for the fields that require URIs (see next section).
3. Set up a query to generate a table with **field names exactly matching the field names in the interchange schema**. In some cases it may be convenient to generate the interchange schema table in several steps, populating subsets of the fields each time. It may also be useful to generate a table with unique combinations of contact, fault, or geologic unit

properties from the source data, and map each combination to corresponding properties in the interchange format. Depending on individual situations, the unique descriptions can be identified using the specification_uri, name, or genericSymbolizer field; identifiers in this field can then be used to join the interchange format properties with individual features.

4. The table of unique descriptions can then be **joined** with the geometry elements to generate the final feature classes for the portrayal view service. ESRI Shapefiles and PostGIS tables are useful representations for the final feature class that the WMS server will use. Shapefiles will truncate field names longer than 10 characters, so in order to use them, the map server must support defining XML element names distinct from data table field names for element names that are longer than 10 characters.
5. Set up the configuration for the WMS server. This procedure will be specific to the particular server implementation that is being used. Procedures for deployment using GeoServer and ArcGIS server are discussed in Appendices A and B respectively.
6. Create metadata for the dataset and its service distribution and load it into a catalog server to make the serviced discoverable.

3.1.1 Interchange formats

This cookbook describes deploying services using GeoSciML-portrayal XML as the information interchange format. Features provided by these services have content structured according to an appropriate GeoSciML-portrayal schema. This content is accessed either through WMS getFeatureInfo requests or through WFS getFeature requests.

The GeoSciML-portrayal feature classes detailed in Section 2 implement three schemas content models:

- Table 1 specifies the content for the GeoSciML-portrayal ContactView schema
- Table 2 specifies the content for the GeoSciML-portrayal ShearDisplacementStructureView
- Table 3 specifies the content for the GeoSciML-portrayal GeologicUnitView

By entering geoscience data into an XML document that implements these content models using the schema at http://schemas.geosciml.org/geosciml-portrayal/1.0/, one is creating an interchange document.

3.1.2 Schema mapping in ESRI ArcGIS

This section provides an example of schema mapping in ESRI ArcGIS, using ArcMap, ArcCatalog, and ArcToolbox. The procedure requires connecting to the spatial data container (file geodatabase, shapefile, or SDE database), creating a new feature class using the geosciml-portrayal schema if necessary, and loading data from the source dataset into the geosciml-portrayal-schema feature class.

1. Open ArcCatalog
2. Locate or create a **spatial database** that will be used to host your web services in accordance with the GeoSciML-portrayal schema

a. Sometimes it's easier to create a new database in accordance with the GeoSciML-portrayal schema; other times, it's better to create a new feature class within an existing database. The conditions needed to make this determination depend on individual context, and beyond the scope of this document.

3. Connect to the desired spatial database under **Database Connections (Figure 5)**
4. In the **Spatial Database Connection** window, specify the location and (if necessary) the

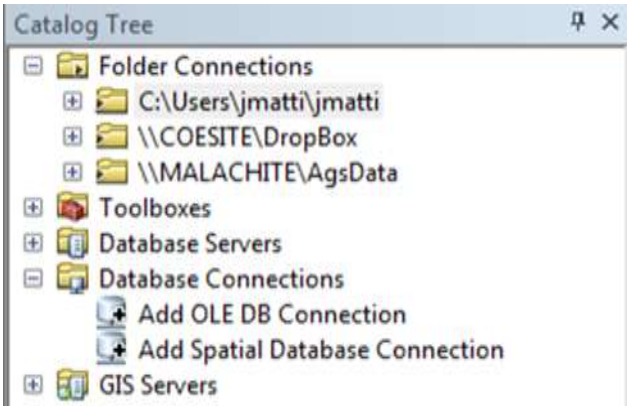

Figure 5: Adding a Spatial Database

credentials needed to access the desired spatial database. Click **OK** when finished.

5. After you have connected to the spatial database you intend to use for your services, right-click the desired database; in the context menu that appears, click **New** > **Feature Class**…

This will bring up the **New Feature Class** dialogue box.

a. In the **New Feature Class** dialogue box, type the name for your feature class (this is *not* the name of the web service). Use the pulldown menu to select the appropriate **Type** of feature geometry for the data you are going to publish. When you are finished filling out the form, click **Next**. Note that if you are loading into a shapefile, field names longer than 10 characters will be truncated; aliases assigned to the fields must contain the complete geosciml-portrayal field name.
b. Choose the **coordinate system** for your feature class.
    i. For the purposes of interoperability, WGS 1984 is the coordinate system required by OneGeology. To select the WGS 1984 coordinate system, navigate to the **Geographic Coordinate System** > **World** directory and click the WGS 1984 coordinate system.

When you have located the desired coordinate system, click **Next**.

c. Specify the resolution for your feature class. The XY tolerance defaults to 0.000000008983153, which is fine for most applications, but may be changed if necessary by entering an appropriate value into the XY Tolerance field.

When finished, click **Next**.

d. Specify the **Configuration keyword** for your feature class.
    i. The default Configuration keyword is fine for most database storage architectures.

When finished, click **Next**.

e. Enter appropriate field headings under the **Field Name** column; specify the parameters for each field by entering appropriate values in the **Data Type** column and in the **Field Properties** section (Figure 6).
    i. To comply with GeoSciML-portrayal, all field names and associated parameters must conform exactly to the GeoSciML-portrayal **schema** that is appropriate for your dataset. This includes capitalization, spacing, underscores, field lengths, and cardinality.

       Appropriate values can be found in Section 4, GeoSciML-Portrayal Feature Types.

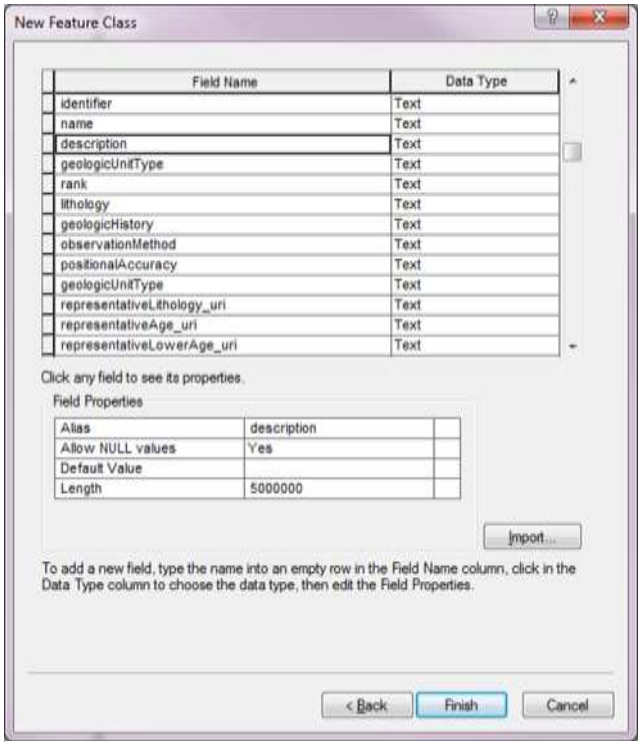

Figure 6: Step 5E - Entering GeoSciML-portrayal schema field names and parameters

When done, click **Finish**.

Your new **Feature Class**, structured by the appropriate **GeoSciML-portrayal schema**, will appear in the spatial database you selected or created above.

Next, you will use **ArcToolbox** to import and map your existing data into the new feature class you just selected or created above.

1. Open ArcCatalog

2. On the menu bar of ArcCatalog, click the **ArcToolbox** button 
3. In ArcToolbox, click Data Management Tools > General > Append
4. In the **Append** window (Figure 7), you need to perform a number of tasks:
    a. Specify the **Input Dataset**. This will be your source data that does not conform to the GeoSciML-portrayal schema
    b. Specify the **Target Dataset**. This will be the Feature Class you created above
    c. Change the default **Schema Type** to **NO_TEST**. This last instruction is very important; it allows you to perform the schema mapping
    d. For each field in the **Field Map (optional)** table, map your **Input Dataset** to the **Target Dataset**:
        i. Right-click a target field in the **Field Map (optional)** table
        ii. In the context menu that appears, click **Add Input Field**
        iii. In the dialog box that appears, select the corresponding field from your **Input Dataset**
    e. Repeat steps **i-iii** for every field in the **Target Dataset**
    f. When finished, click **OK**

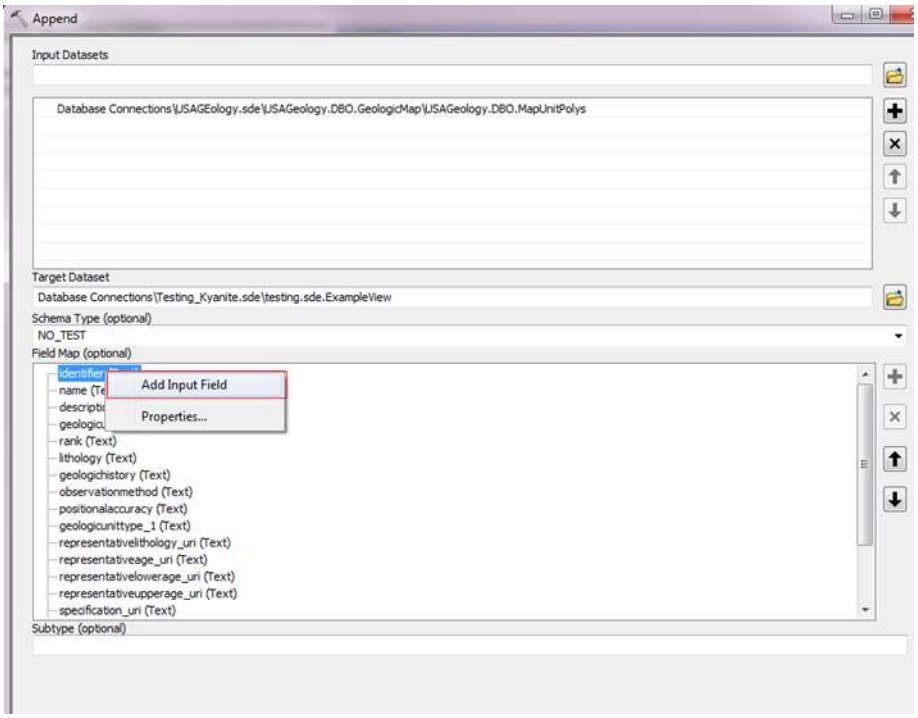

Figure 7: The Append window

5. Repeat Step 4 in this list for each dataset you wish to deploy as a web service.

You are now ready to deploy your services.

3.1.3 Schema mapping in SQL

If you have multiple tables that you need to map into a single schema, it may be easier to write a SQL query to do the mapping. The following example demonstrates a schema mapping query, written in Postgres-flavor SQL (minor modifications may be necessary for other SQL environments). The source schema is the USGS/AASG National Cooperative Geologic Mapping Program NCGMP09 database schema (Haugarud et al., 2010), and the target is a GeoSciML-portrayal GeologicUnitView. This is meant as an illustrative example of the kind of processing that may be required. The level of difficulty in the schema mapping is largely determined by how similar the internal data model schema is to the GeoSciML conceptual model, and how much content the data provider wishes to make available in the portrayal view.

This example SQL query creates a table that has fields conforming to the GeologicUnitView for the geosciml portrayal scheme. The SQL is a PostGIS dialect, so field names that are not all lower case are enclosed in quotes, and constant values are explicitly types (e.g. '::text'). 'mapunitpolys' (mup),' datasources', 'descriptionof-mapunits' (dmu) are tables from the source NCGMP09 database,

Code Example 1: PostGIS SQL query to produce GeoSciML-portrayal view for map unit polygons.

```
1  CREATE TABLE sde.geologicunitview As
2  SELECT
3  mup.objectid as objectid,
4  mup.mup_id AS identifier,
5  dmu.description AS name,
6  mup.notes AS description,
7  'Geologic Unit'::text AS "geologicUnitType",
8  'Not Specified'::text AS rank,
9  polyextattr.lith6name AS lithology,
10 dmu.age AS "geologicHistory",
11 (datasources.source::text || ' '::text) || datasources.notes AS source,
12 'http://…/cgi/geologicunittype/0008'::text AS "geologicUnitType_uri",
13 polyextattr.lithuri AS "representativeLithology_uri",
14 polyextattr.ageuri AS "representativeAge_uri",
15 mapunitages.ageyoungerterm AS "representativeLowerAge_uri",
16 mapunitages.ageolderterm AS "representativeUpperAge_uri",
17 'http://www.opengis.net/def/nil/OGC/0/missing'::text AS specification_uri,
18 'http://catalog.usgin.org/geoportal/…'::text AS metadata_uri,
19 mup.mapunit AS "genericSymbolizer",
20 shape::geometry as shape
21 FROM mapunitpolys AS mup
22 LEFT JOIN polyextattr ON mup.mapunitpolys_id = polyextattr.ownerid
23 LEFT JOIN datasources ON mup.datasourceid = datasources.datasources_id
24 LEFT JOIN descriptionofmapunits as dmu ON mup.mapunit = dmu.mapunit
25 LEFT JOIN mapunitages ON mup.mapunit = mapunitages.mapunit;
```

abbreviated as indicated in the query. 'poly-extattr' and 'mapunitages' are Postgres views that aggregate lithology and age properties correlated with polygons or map units respectively. The source data is a regional dataset that correlates lithology and age categories at the polygon level to document lithologic and age variation within the map units. This information is contained in the 'extendedattributes ' and 'geologicevent' tables in the source dataset (See NGCMP, 2010 or http://ngmdb.usgs.gov/Info/standards/NCGMP09/ for details on the input data structure).

3.1.4 Notes on schema mapping in SQL

Line numbers refer to numbered lines of text in Code example 1.

- Line 1. Result table is placed in ESRI sde schema in PostGIS database.
- Line 3. For ESRI Spatial Database Engine (SDE) to recognize the table as a feature class that may be used to source a service, an integer unique-value field must be present. In this case the field from the source dataset is used because it serves the same purpose there. This field does not need to be set up as an autoincrement field because the resulting table is a read-only view of the data. The ObjectID field will likely be placed at the last column in the table for better interoperability with systems that do not require such a field.
- Line 4. Unique polygon identifiers from the source dataset are carried into the interchange format view.
- Line 7. Source data map units are not categorized into specific unit types (e.g. lithostratigraphic, chronostratigraphic, geophysical, etc.), so this field gets a constant value 'Geologic Unit', which is the preferred label for the corresponding concept in the CGI Geologic Unit Type vocabulary.
- Line 8. Likewise, rank is not assigned for the regional geologic units in the source dataset, so this is a constant that is the preferred label ('Not Specified') for the corresponding concept in the CGI Strati-graphic Rank vocabulary.
- Line 9, 10. Text descriptions of lithology and the age of the unit are derived from corresponding fields in the source data. The lithology description is mapped at the individual polygon level (via the polyextattr Postgres view), note the JOIN in line 22 that uses the .mapunitpolys_id as the foreign key. The age text description is mapped at the map unit level, note the JOIN in line 24 that uses the .mapunit field as the foreign key.
- Line 11. The text for the source field in the portrayal scheme is calculated by concatenating two fields (.source and .notes) from the datasources table in the input dataset.
- Line 12. Since all the map units are classified simply as 'Geologic Unit', this URI is also a constant value from the CGI Geologic Unit Type vocabulary.
- Line 13, 14. Representative lithology and age categories are assigned on a polygon by polygon basis, thus the polyextattr table is the source of these URIs. Note the JOIN in line 22 that uses the .mapunitpolys_id as the foreign key.
- Line 15, 16. Younger and older age bounds are assigned by map unit (not at the individual polygon level), so these fields are sourced from the mapunitages aggregation query in the database; this query uses the extendedattributes table to join map-units with geologicevent records that contain the lower and upper age bounds for the unit.

- Line 17. specification_uri is a link to a more complete description of the geologic unit that crops out in the polygon's extent. This description can be viewed as a resource in the context of web architecture. The intention of the GeoSciML design team was that this URI should dereference to return a full Geo-SciML GeologicUnit element instance. With the use of content negotiation on the web, this description might also have representations as a web page or other structured description (rdf, owl…). In the example instance, a more complete description is not available and an OGC nil URI is used to indicate that the resource is missing (does not exist).
- Line 18. metadata_uri identifies a metadata resource that contains more complete information on the provenance of the information in the feature element. This may be a metadata record that is scoped to the individual feature, or may identify a metadata record describing some collection of polygons that have similar enough provenance to document together. The text in the source field (Line 11) should be a succinct summary of the information in this metadata record pertinent to the containing feature.
- Line 19. The genericSymbolizer field should contain an identifier for the symbol used to display this poloygon in the default portrayal (legend) chosen by the data provider for the gsmlp:GeologicUnitView instance in the containing feature collection. This field can be used to capture the map unit assignment and portrayal color scheme from the original data from which the GeologicUnitView polygon was digitized. The legend may be encoded in an accompanying Styled Layer Descriptor (SLD) file, and if such an SLD exists it should be recorded as a related resource in the metadata record specified by metadata_uri.
- Line 20. The shape field contains a representation of the geometry of the outcrop area described by the GeologicUnitView instance. The content of this field will be managed by the GIS and WMS server, and thus will generally not need to be manipulated by users outside the GIS environment. The shape field may need to be cast into a recognized geometry field in order to get PostGIS and ESRI SDE to recognize the output as a feature class. In our configuration at AZGS, we ended up using this:

  'st_geometryfromtext(ST_AsText(shape)::text, 4326)::geometry as shape'

  More expert PostGIS users may have a better solution for this problem.

3.2 Vocabulary mapping

There are many possible approaches to mapping terms from one vocabulary to another.

One situation in which a standard process can be defined involves a controlled vocabulary used to populate a field in the source data, and that field maps directly to a field in the interchange format. For instance: if the source data contains a 'dominant lithology' field, the information in this field can be used to populate the 'representativeLithology_uri' for a GeologicUnitView feature.

Recommended procedure:

1. Produce a table of the unique values in the source data
2. Add a column to this table for the corresponding element in the target interchange scheme

3. Determine the best matching value from the interchange vocabulary for each term in the source vocabulary

In general: the most specific term from the interchange vocabulary that completely subsumes (encompasses) the meaning of the term in the source vocabulary should be used.

If the source vocabulary has terms that are more specific than the controlled vocabulary, there will be some information lost in this process, but the original source terminology can be preserved in free-text description fields in the interchange document. Sample free-text fields that may be used to preserve source vocabulary information are as follows:

- Lithology
- geologicHistory

Remember, the primary purpose of the controlled vocabulary fields is for data integration and use as search criteria, likely by non-expert users.

In some cases, unique values from a combination of multiple source data fields may be necessary to define mapping to interchange concepts, particularly for the following interchange fields:

- representativeAge_uri
- representativeLithology_uri
- genericSymbolizer

The procedure is the same, but any unique values query will involve more than one source field and multiple-field **joins** will be necessary to construct queries generating the output schema content.

Standard vocabularies are listed in the accompanying Excel Workbook.

3.3 Excel Workbook template

The accompanying Microsoft Excel Workbook provides spreadsheets containing the content models for the description properties associated with geologic contacts, faults or shear zones, and geologic units in appropriate GeoSciML-Portrayal schemas.

These spreadsheets do not include geometry fields that are necessary for an actual GIS feature class for the corresponding ContactView, ShearDisplacementStructureView, and GeologicUnitView features. Rather, they provide a template for compiling the necessary descriptions (combinations of attributes) that can then be **joined** with a GIS feature class to produce the dataset for feature web service deployment.

The specification_uri in each feature class provides a link to structured representation of the geologic feature that is intended to be a GeoSciML feature.

The metadata_uri provides a link to a structured metadata record maintaining complete provenance information for features.

See the notes tab in the workbook for additional information describing the GeoSciML Portrayal feature classes and the work-sheets included in this document.

3.4 Configuring OGC services

GeoSciML-portrayal is intended to support map services for online viewing and exploration of geologic data, and the creation of mash-ups integrating data from different servers and possibly different thematic domains.

The OGC WMS specification includes a 'getFeatureInfo' operation that returns a description of a map feature at a user-identified point on the map. The operation returns a document determined by the server configuration. A variety of different formats may be offered, and specified by a request parameter. For GeoSciML-Portrayal services, an XML document conforming to the GeoSciML-Portrayal XML schema should be offered. Other getFeatureInfo response formats may also be offered.

Server software that implements the OGC Web Map Service will typically also allow parallel deployment of an OGC Web Feature Service with no additional effort. The XML provided by the feature service is the same as that provided in the WMS getFeatureInfo response formats.

The GML simple feature profile requires that each feature include a gml:id attribute. According to the GML specification, this identifier must be an alphanumeric string that has an alphabet letter as its first character. These ID's can be generated from the primary key in the data table used to compose the features for the map service. If data are coming from an ArcGIS environment, the ObjectID required for any ArcGIS feature class can be used to construct the gml:id. If no primary key is present in the feature class for the service, one will have to be added to source the gml:id property. Technically, the gml:id is meant to be unique within the scope of a feature collection from a particular service to enable internal cross referencing within a single WFS response document or to allow queries to retrieve a feature using the gml:id.

Server configuration using GeoServer and ESRI ArcGIS Server is described in Appendices A and B as examples of the workflow necessary to set up a GeoSciML-Portrayal service. Other implementations exist, and the details of the procedure will change as new versions of the server software are released.

4 Styling

Part of the process of deploying a web service includes defining a map legend.

ESRI ArcGIS Server provides a number of tools to define a map legend, but if you are using GeoServer to deploy your data as a web service, you will need an OGC Styled Layer Descriptor (SLD) file to define your map legend.

An SLD is an XML document that defines a mapping from some property of an OGC feature to a symbol to use for portrayal. The correlation of feature property values with symbols used to portray the feature defines a map legend.

SLD files for portrayals using the CGI simple lithology and ICS 2009 time scale URIs are available at the following web location:

http://schemas.usgin.org/schemas/slds/

Aside from those SLDs made available at the above web location, it is possible to use a basic text editor to manually create an SLD. Manually creating an SLD for a geologic map is not a trivial proposition, and instructions for creating an SLD file manually are out of the scope of this document. The GeoServer online documentation provides some guidance for creating SLD's manually. The following section describes software for creating an SLD file based on symbology in an ArcMap project layer.

4.1 How to Create a Styled Layer Descriptor (SLD) using Arc2Earth

Arc2Earth is a plugin for ArcGIS that is available in several different editions. The **Community** edition is free and provides access to the SLD generator. Arc2Earth is available at http://www.arc2earth.com/.

1. Install the Arc2Earth plugin
2. Open ArcMap
3. Open the ArcMap project containing the layer or layers to export as SLDs
4. Select the desired layer in the catalog tree
5. In the Arc2Earth toolbar, click Export > Export Layer Style to SLD
6. Navigate to the appropriate location
7. Click **Export**

After your SLD has been exported, open the .sld in any XML or text editor. Note that the Arc2Earth plug-in adds an outline (stroke) to polygon layers. If you are working with polygon layers, the stroke may need to be tailored or removed manually. Also, it may be necessary to customize the heading for your SLD, or change the schema location.

Note: though .sld files, like any XML document, may be edited by a generic text editor, such as the **Notepad** or **Wordpad** text editors included with Windows operating systems, some text editors are better suited to working with XML documents than others. One example of a free-and-open-source text editor designed with XML support is Notepad++ (be sure to get the XML tools plugin).

Below is a snippet of an SLD created using Arc2Earth (Code Example 2).

Code Example 2: Styling in XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<StyledLayerDescriptor xmlns:xlink='http://www.w3.org/1999/xlink' xmlns:gml='http://www.opengis.net/gml'
xmlns:ogc='http://www.opengis.net/ogc' xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' version='1.0.0'
xsi:schemaLocation='http://www.opengis.net/sld StyledLayerDescriptor.xsd' xmlns='http://www.opengis.net/sld' >
    <NamedLayer>
        <Name><GeologicAge></Name>
        <UserStyle>
            <FeatureTypeStyle>
                <Rule>
                    <Name><Holocene></Name>
                    <Title><Holocene></Title>
                    <ogc:Filter>
                        <ogc:PropertyIsEqualTo>
                            <ogc:PropertyName>representativeage_uri</ogc:PropertyName>
    <ogc:Literal><http://resource.geosciml.org/classifier/ics/ischart/Holocene></ogc:Literal>
                        </ogc:PropertyIsEqualTo>
                    </ogc:Filter>
                    <PolygonSymbolizer>
                        <Fill>
                            <CssParameter name="fill" >#f9f97f</CssParameter>
                            <CssParameter name="fill-opacity" >1</CssParameter>
                        </Fill>
                        <Stroke>
                            <CssParameter name="stroke" >#f0f0f0</CssParameter>
                            <CssParameter name="stroke-width" >0.4</CssParameter>
                            <CssParameter name="stroke-opacity" >1</CssParameter>
                        </Stroke>
                    </PolygonSymbolizer>
                </Rule>
                <Rule>
                    <Name><Quaternary></Name>
                    <Title><Quaternary></Title>
                    <ogc:Filter>
                        <ogc:PropertyIsEqualTo>
                            <ogc:PropertyName>representativeage_uri</ogc:PropertyName>
        <ogc:Literal><http://resource.geosciml.org/classifier/ics/ischart/Quaternary></ogc:Literal>
                        </ogc:PropertyIsEqualTo>
                    </ogc:Filter>
                    <PolygonSymbolizer>
                        <Fill>
                            <CssParameter name="fill" >#f9f97f</CssParameter>
                            <CssParameter name="fill-opacity" >1</CssParameter>
                        </Fill>
                        <Stroke>
                            <CssParameter name="stroke" >#6e6e6e</CssParameter>
                            <CssParameter name="stroke-width" >0</CssParameter>
                            <CssParameter name="stroke-opacity" >1</CssParameter>
                        </Stroke>
                    </PolygonSymbolizer>
                </Rule>
```

5 Deploying Your Web Service

If you have completed the following steps, you are ready to deploy your web service in a GeoServer environment.

1. You meet the prerequisites detailed in Section 1.3, Cookbook Prerequisites
2. You have successfully mapped your data to an appropriate GeoSciML-portrayal schema, as described in Section 3, Deployment Cookbook
3. You have symbolized your data, as described in Section 4, Styling.
4. If you have followed all previous steps in this cookbook, you should now be ready to deploy your GeoSciML-portrayal web service.

The overall process of web service deployment is as follows:

1. Configure server software to provide web services
2. Make your data accessible to the server
3. Configure the desired server software to deploy your data as a web service
4. Test your web service (see Section 6, Testing Your Web Service, for more details)

Details of the actual deployment process depend to a large degree on the individual context; to attempt to account for all possible scenarios is beyond the scope of this cookbook. Instead, this cookbook will provide examples of two common web service deployment patterns:

- Web service deployment in a GeoServer environment with data in a PostGIS database (Appendix A)
- Web service deployment in an ArcGIS Server environment with data in an ESRI file geodatabase (Appendix B)

6 Testing Your Web Service

Web service should be tested by connecting and accessing data using your preferred GIS application. For OGC services the first test the 'getCapabilities' request offered by any service to describe the capabilities of an individual service instance. Further testing requires connecting to the service with client software and accessing data.

6.1 Web Service Requests

All OGC services offer a **GetCapabilities** request that returns an XML document describing the capabilities of the service instance (Figure 8).

To perform a **GetCapabilities** request, paste the capabilities URL in a web browser; this should return a **capabilities document** in your web browser. Different browsers may display the XML response differently. To see the unformatted response document, use the 'view page source' function in your browser, generally accessible via a right-click in the content of the displayed web

page. The **service request** consists of a service endpoint part, and the OGC request part. Here are some sample GetCapabilities requests:

> **http://services.azgs.az.gov/ArcGIS/services/aasggeothermal/ALBoreholeTemperatures/MapServer/WFSServer**?request=GetCapabilities&service=WFS [ArcGIS WMS]
>
> **http://catalog.usgin.org/geoserver/ows**?service=wfs&version=1.1.0&request=GetCapabilities [GeoServer WFS].

In both examples, bolded text constitutes the service endpoint. This has a host name part that identifies a server (e.g. services.azgs.az.gov, catalog.usgin.org), and a local path used by the host to identify the service resource (e.g. '/ArcGIS/services/aasggeothermal/…./WFSServer', '/geoserver/ows'). The text after the '?' constitutes the service request, which consists of a series of key=value pairs, delimited by '&' characters. The service endpoint string will be specific you your particular implementation environment, but the request part will be the same for any service. In order to test your web service, you will need to determine the service endpoint of your web service and append a service request to it.

Once you have accessed the capabilities document of your web service and copied the service endpoint, you can proceed to the Section 7.2.1 or 7.2.2. Sections 7.2.1 and 7.2.2 contain instructions for accessing a web service via GIS applications. Section 7.2.1 will describe the process of accessing a web service via ESRI ArcCatalog for use in the widely used ArcGIS environment. Section 7.2.2 describes how to access an OGC web service using the free-and-open-source desktop GIS application uDig.

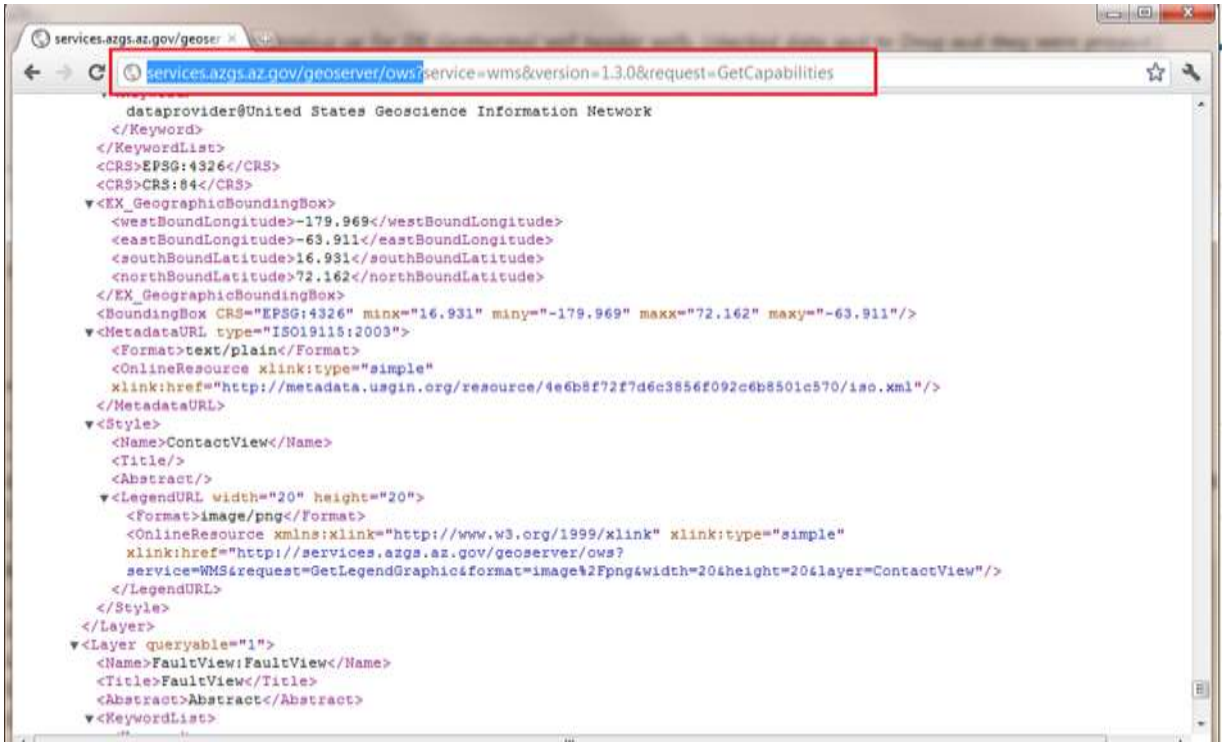

Figure 8: A typical capabilities document returned by a GetCapabilities request in a web

If you are not sure how to access your web service, here is a short list of GIS software that supports web services:

- ArcGIS Desktop
- ArcGIS Explorer
- ArcGIS Online
- Grass GIS
- gvSIG
- OpenLayers
- Quantum GIS
- uDig

6.2 Adding a Web Map Service in ArcCatalog

1. Open ArcCatalog
2. In the Catalog Tree, expand **GIS Servers**
3. Click **Add WMS Server.** See the glossary for more information about WMS.
4. Paste the **service endpoint** (see Section 6, Testing Your Web Service) for your web service into the URL field
5. It may be necessary to navigate back to GeoServer for your URL. On the GeoServer Welcome page, Service Capabilities are listed on the right. Click on the latest version of WMS or WFS, whichever service you wish to use.
6. When redirected, copy the URL (from beginning up to the question mark) and paste into the URL field of Add WMS Server in ArcCatalog:

   http://services.azgs.az.gov/geoserver/ows?service=wms&version=1.3.0&request=GetCapabilities

7. Click Get Layers
8. Click OK
9. Add the desired layer(s) to ArcMap

6.3 Adding a Web Service in uDig

1. Open uDig. The software can be obtained at: http://udig.refractions.net/
2. Create a new map, or open an existing map to which you would like to add the web service
3. On the upper menu bar, click Layer > Add…
4. In the window that appears, click Web Map Server or Web Feature Server, as appropriate; click Next
5. Paste the service endpoint in the URL field; click Next
6. In the Resource Selection window that appears, select all layers you wish to add. When you are done, click Finish

7 Registering with OneGeology and Submitting your Service

To submit your service to OneGeology, you will first need to register your organization on their website by filling out their registration form. If the data provider is able to host their own service, visit this link:

http://www.onegeology.org/technical_progress/data_coordination.cfm

If the data provider would prefer a 'buddy' serve their data, fill out this form:

http://www.onegeology.org/technical_progress/buddy_coordination.cfm

Next, you will need to send an email to onegeology@bgs.ac.uk with the draft URL of the proposed WMS service. Include this information: Name of Geographic Area, Name of Data Provider, Name of Service Provider.

You will be contacted by the OneGeology secretariat with feedback, and confirmation of receipt.

Appendix A: Deploying GeoSciML-Portrayal Web Services in GeoServer

This appendix provides a tutorial of GeoSciML-portrayal web service deployment in a GeoServer environment. This procedure assumes that you have a working GeoServer implementation with

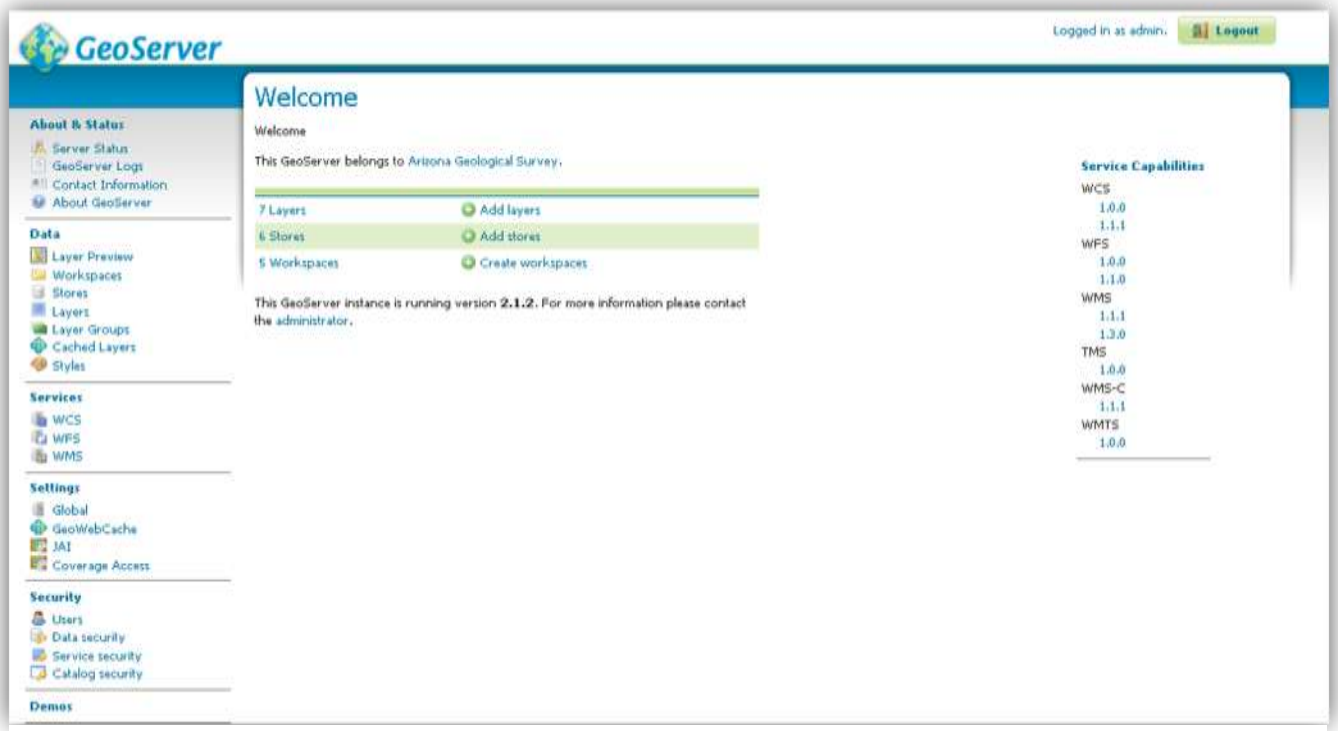

Figure 9: the GeoServer Web Administration Interface

PostGIS, and that you have administrative privileges necessary to create workspaces, stores and layers for GeoServer to work with.

A.1. Logging in

To deploy a web service in the GeoServer environment, your first step is to log in to the **Web Administration Interface** for the GeoServer instance you will be using to deploy your data as a web service (Figure 8). The Web Administration Interface is, as the name suggests, a browser-based user interface capable of administering server software remotely via your Internet connection.

In order for your services to be publicly accessible, configure the host and port of your GeoServer instance in such a way that your GeoServer instance can be accessed remotely. To access the Web Administration Interface of a GeoServer instance, open your web browser and enter the appropriate web address into the navigation bar. GeoServer is usually installed such that the administrative interface can be accessed at a URL with the following address pattern using a web browser:

http://<**host**>:<**port**>/**geoserver/web/**, <**port**> is usually **8080,** and <host> is the name or IP address of the server.

The default account settings for GeoServer are as follows:

Username: admin
Password: geoserver

For security reasons, it is recommended that you change your password to something more secure as soon as possible.

A.2. Service-Level Metadata

Within the GeoServer Web Administration Interface (Figure 8), click Contact Information, under About & Status.

This brings you to a **Contact Information** form (Figure 9) in which you can provide contact information for your GeoServer instance. The information entered here becomes service-level metadata for the web service that is accessed by the OGC GetCapabilities request. Consequently, Contact Information entered here should be as precise and comprehensive as possible.

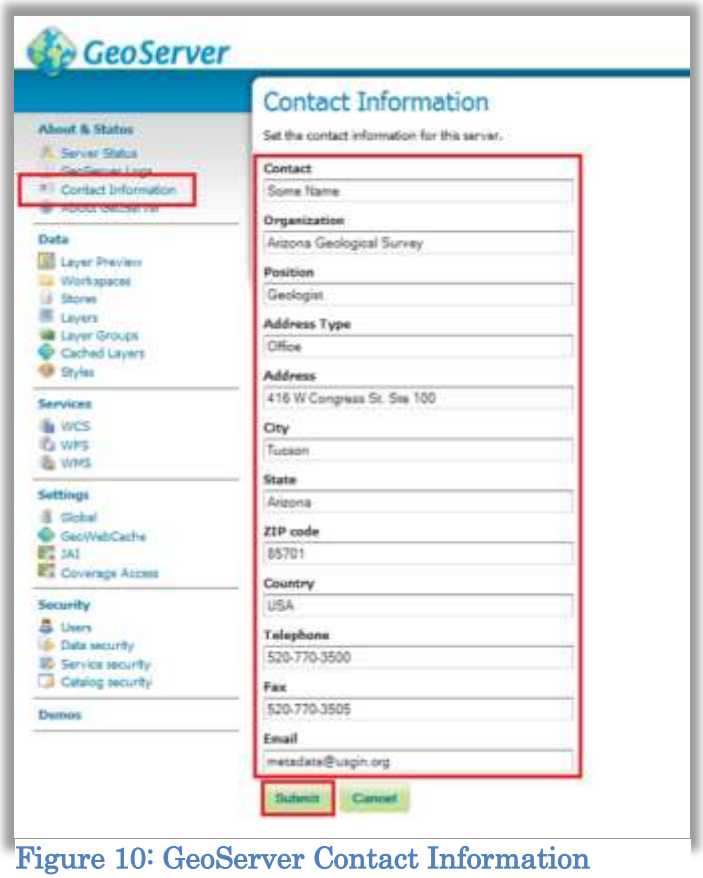

Figure 10: GeoServer Contact Information

If you wish to deploy your service in compliance with OneGeology standards, your server-level metadata will need to meet very specific requirements. See Appendix C.1 for details.

A.3. Creating a Workspace

1 After entering contact information for your GeoServer instance, you will need to create a workspace for your web service. The workspace is the equivalent of the **service name** when deploying a web service in the ArcGIS Server environment.

2 If you wish to deploy your service in compliance with OneGeology standards, your workspace will need to meet very specific requirements. See Appendix C.2 for details.

3 On the left side of the GeoServer **Web Administration Interface**, under **Data**, click **Workspaces**. This will bring you to the **Workspaces** page, wherein you can manage existing workspaces and create new workspaces.

4 Click **Add New Workspace**. This will bring you to the **Edit Workspace** page for your new workspace.

5 Two fields are present on the Edit Workspace page:

6 **Name**: The **service title**; may contain spaces or special characters; usually named according to the following convention:

7 GeographicArea_Theme, where GeographicArea is the overall region containing the map extent of the web service
Theme is an indicator of the kinds of features

8 **Namespace URI**: A URI associated with your project

9 When you are finished, click **Save**.

A.4. Connecting to your PostGIS Database

1 Having created a new workspace, you will now create a **store**, or **database connection**, for your service.
2 On the left side of the GeoServer **Web Administration Interface**, under **Data**, click **Stores**. This will bring you to the **Stores** page. On the **Stores** page, click **Add New Store**.
3 This will bring you to the **New Data Source** page (Figure 10).
4 On the **New Data Source** page, choose **PostGIS** as the data source by clicking **PostGIS**. This will bring you to the **New Vector Data Source** page. Complete the following steps:
5 Select a **Service Title** from the **Workspace** drop down menu. Select the workspace you created in Appendix A.3
6 Type a name for your data store in the **Data Source Name field** (spaces are acceptable here); add a description if desired
7 Make sure that the **Enabled** checkbox is checked
8 Set the **Connection Parameters** for your PostGIS data source; if the PostGIS data source is located on a remote server, you will need to provide the appropriate host, port, database name, user name, and password to access it:
9 **Host**: use "localhost" if the PostGIS data source is on the same machine as your GeoServer instance; more specific host information will be necessary if your PostGIS data source is on a remote server
10 **Port**: default is 5432
11 **Database name**: this information will depend on the PostGIS data source
12 **Schema**: this information will depend on the PostGIS data source
13 **User name**: this information will depend on the PostGIS data source
14 **Password**: this information will depend on the PostGIS data source
15 When finished, click **Save**.

A.5. Adding Layers to a Workspace

Having created a workspace and specified a PostGIS data source for your web service, you will now populate your web service with data layers from your workspace.

On the left side of the GeoServer **Web Administration Interface**, under **Data**, click **Layers**. This will bring up the **Layers** page.

On the **Layers** page, click **Add a new resource**. This will take you to the **New Layer** page.

On the **New Layer** page, use the pulldown menu at the top of the page to select the data source you specified in Appendix A.4. Doing so will populate the **New Layer** page with a list of layers that may be published; click **Publish** to make the associated layer publicly accessible to anyone who connects to your web service.

Note: You may publish the same layer multiple times. To do so, click **Publish again.

After clicking **Publish**, the **Edit Layer** page for the corresponding layer automatically appears (Figure 12). The **Edit Layer** page contains two tabs, **Data** and **Publishing**.

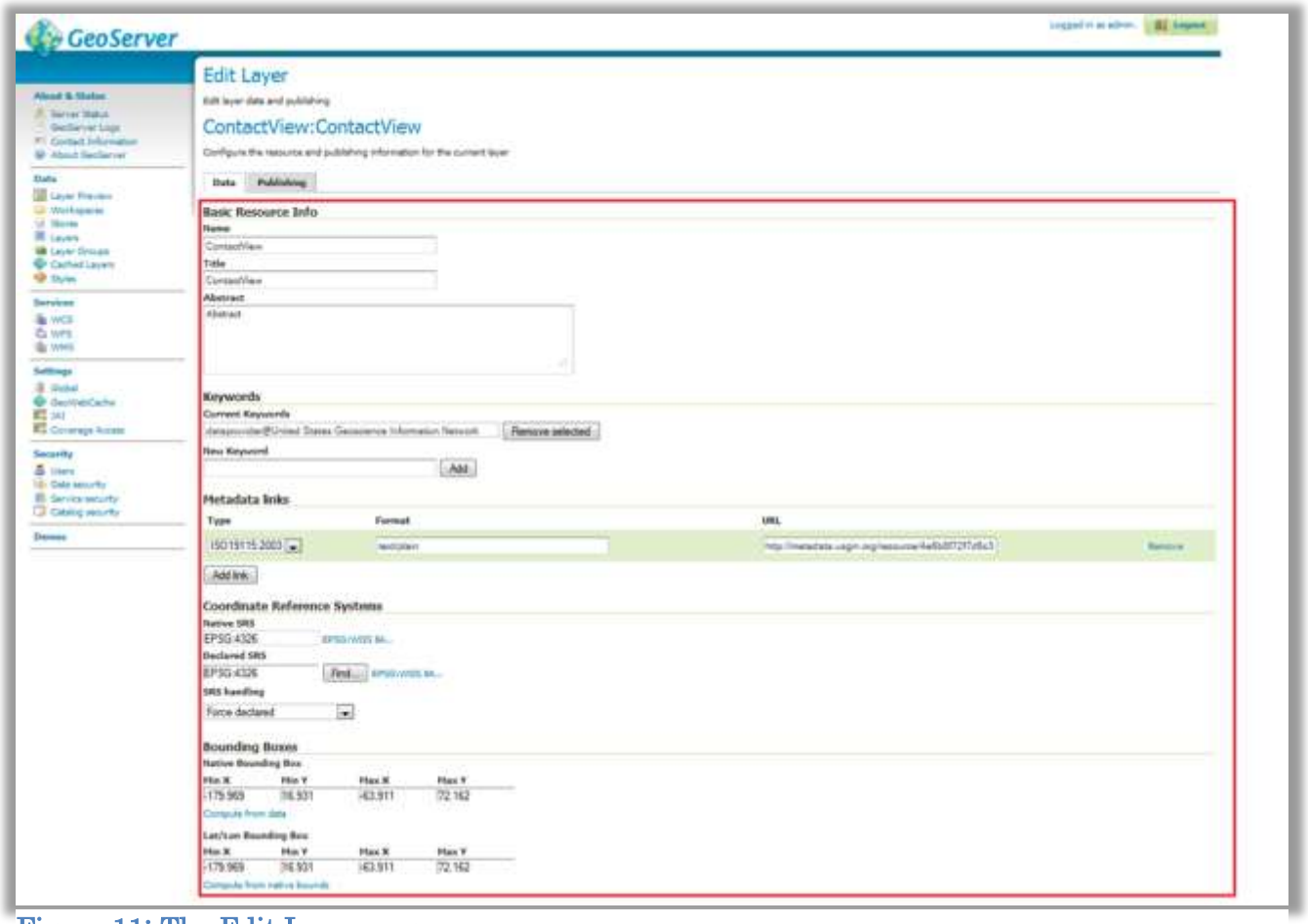

Figure 11: The Edit Layer page

A.4.1. The Data tab of the Edit Layer page

The **Data** tab contains fields within which you may specify the title, abstract, bounding box, spatial reference system, keywords, and metadata links for each layer in your web service. This information will be present within the **Capabilities document** produced by your web service in response to a **GetCapabilities** request, so it is very important to enter this information carefully for each layer in your web service.

If you wish to deploy your service in compliance with OneGeology standards, the information you enter for your layers will need to meet very specific requirements. See Appendix C.3 for details.

It is recommended to enter the bounding boxes for your service manually, as doing so permits you to provide a more useful bounding box for your web service.

**Note: GeoServer occasionally enters a loop of web errors. To fix this, we recommend restarting the service on Apache-Tomcat. It also helps to use Mozilla Firefox or Google Chrome for debugging (rather than Internet Explorer).

A.4.2. The Publishing tab of the Edit Layer page

After populating the fields in the **Data** tab, click the **Publishing** tab.

The **Publishing** tab contains **Layer Style** settings for the corresponding layer of your web service. **Layer Style** settings are dependent on the geometry of the layer (point, line, polygon). See Section 5, Styling, of this document for more details.

Generally, it is faster and more precise to import **Layer Style** settings from an existing style than it is to manually specify values for each field in the **Publishing** tab. To import **Layer Style** settings from an existing style, select the desired style from the **Available Styles** list.

To populate the **Available Styles** list, you might need to import layer styles from an SLD file. For further instructions, see Appendix A.5.3, Importing Layer Styles from an SLD File.

When you have populated the fields in the **Publishing** tab, click **Save**.

A.5. Importing Layer Styles from an SLD File

On the left side of the GeoServer **Web Administration Interface**, under **Data**, click **Styles**. This will open the **Styles** page.

If the Style you want is not listed on the **Styles** page, you will need to add it to the list by clicking **Add a New Style**. This brings you to the **Style Editor** page.

On the Style Editor page, you have the choice to copy/paste an SLD or upload a .sld XML document. The **Validate** button may be used to validate the SLD file against a schema before using it. Click **Submit** to add the SLD to the list on the **Styles** page.

A.6. Finishing Up

Repeat Appendix A.5 until all data you wish to publish has been added to the desired workspace. When finished, return to Section 7, Testing Your Web Service.

A.7. GeoServer Troubleshooting

Q: I made a change in the database on my server and now the service is not working.

A: Try clearing the cache and reloading GeoServer on the Server Status page (Figure 22). If that doesn't work, try hard restarting the service through Apache Tomcat.

Q: Can I use a replicated Feature Class to create a service on Geoserver?

A: No.

Q: I set up my services under three separate workspaces. When I connected to the WMS in ArcCatalog, all the layers appeared as one bundle. Is there a way to separate them out so I can add them individually?

A: Yes.

Though setting up your services under different workspaces seems to imply that they can be accessed as discrete services, Geoserver defaults to providing one capabilities document containing the information for all of the services set up on your instance of GeoServer. To access workspaces individually, you will need customize your Get request to specify the desired workspace.

For example: the Arizona Geological Survey runs three services on Geoserver (http://services.azgs.az.gov/geoserver/web/):

- GeologicUnitView
- FaultView
- ContactView

To perform a GetCapabilities request for GeologicUnitView, your GetCapabilities request will appear as follows:

> http://services.azgs.az.gov/geoserver/GeologicUnitView/ows?service=wms&version=1.3.0&request=GetCapabilities

This URL opens the WMS **capabilities document** for the GeologicUnitView workspace only. A generic form of the service endpoint for the request is as follows:

http://[host server]/geoserver/[Workspace Name]/ows

Individual workspaces can be added in ArcGIS, as well.

Q: Is it possible to configure GeoServer so that I do not need to use PostGIS?

A: Try installing the ArcSDE plug-in for GeoServer. To do this, you will need to download the extension from GeoServer's website. Make sure to match the versions of the extension and GeoServer.

Q: All of my data are in Shapefiles. Can I deploy a shapefile as a GeoSciML-portrayal service?

A: The problem you will run into is the truncation of field names that occurs in shapefiles. Ideally you will have a full version of the data in PostGIS. As mentioned in the above document, to be compliant with GeoSciML-portrayal, you will need to make sure there is no truncation in field names; they must be an exact match for the GeoSciML-portrayal schema.

Q: I would like to deploy a service using full GeoSciML v.3.0 schema; where can I go?

A: Visit the GeoSciML website at http://www.geosciml.org/.

Q: Why would I want to create a **WMS** or **WFS** at all?

A: To display your data in a map format. Map format shows: data relative to its geographical surroundings, data relative to other data on the same map, data points relative to other data points in the same dataset (size, quantity, order, etc).

Q: How do I use the ArcCatalog Interoperability Extension?

A: Here is a brief tutorial on connecting to the WFS in ArcCatalog.

1. Open ArcCatalog
2. Enable the Interoperability Extension
    - For more information visit http://www.esri.com/software/arcgis/extensions/datainteroperability/common-questions.html
3. Click Add Interoperability Connection; this will bring up the Interoperability Connection dialog box:
    1. Click the '**…**'button next to **Format** and select the appropriate format
    2. Paste **the service endpoint** for the service you want to add into the appropriate field
    3. Click the **Parameters** button. In the window that appears, click the '**…**' button next to **Table List** and select the tables you wish to load. Confirm that the **Max Features** box is scaled appropriately
        - If you are loading a dataset with 10 million features, the Max Features box must be scaled to ten million. If you are loading a dataset with three (3) data points, the Max Features box can be scaled down quite a bit.
    4. Click **OK** in the Interoperability Connection dialogue box.
4. Depending on the size of the dataset, it may take some time to load. When it has loaded, expand the Connection (you can also rename the connection, which can be helpful).
5. Add your service to ArcMap

Appendix B: Deploying GeoSciML-Portrayal Web Services in ArcGIS Server

This appendix provides a tutorial of GeoSciML-portrayal web service deployment in a GeoServer environment.

B. 1 Connecting to your ArcGIS Server instance

1. Open ArcCatalog
2. In the catalog tree, expand **GIS Servers** by clicking the nearby +
3. Double-click Add ArcGIS Server
4. Click the Manage GIS Services radio button in the **Add ArcGIS Server** window and then click **Next**.
5. Type in the URL for your services and the name of the server, then click **Finish**.

B. 2 Create an ArcMap Project for your web service

1. Open an instance of ArcMap

2. Create a blank map
    a. Note: there is a one-to-one relationship between .mxd files and services: each .mxd file, along with all layers it contains, is equal to one web service
3. Add your source data to the map as a layer
    a. If you wish to deploy your service in compliance with OneGeology standards, your layers will need to meet very specific requirements. See Appendix C.3 for details
4. Save your map as a .mxd file

B. 3 Publishing on ArcGIS Server

1. Open ArcCatalog.
2. Navigate to the database connection you set up in Appendix B.1, Connecting to your ArcGIS Server instance
3. In the catalog tree on the left side of the screen, expand the connection to ArcGIS Server
4. Right-click your server and select **Add New Service**; this opens a window containing fields for **Service Title** and **Service Description; i**f you wish to deploy your service in compliance with OneGeology standards, your **Service Title** will need to meet very specific requirements. See Appendix C.2 for details
    a. Specify the appropriate **Service Title** based on content and map extent
    b. Using the pulldown menu, specify whether you are deploying a Web Map Service or a Web Feature Service
    c. Enter an appropriate description for your service in the **Description** field
    d. Click **Next**
5. Specify the capabilities of your web service
    a. Use the checkboxes to indicate which capabilities your web service will support
        i. OneGeology requires either WMS or WFS capabilities
            1. Specifying WMS or WFS will reveal a form in the **Properties** box; **i**f you wish to deploy your service in compliance with OneGeology standards, this information must be provided in accordance with very specific requirements. See Appendix C for more details.
            2. Alternatively, if you know of a web service capabilities document that is already compliant with OneGeology standards, you may save this capabilities document as an XML file and import it
                a. To import an external XML document, click **Use external capabilities files** and then specify the location of the XML document you will use
                b. To save a capabilities document as an XML document, access the appropriate capabilities document (described in detail in Section 7) in your web browser and click **File > Save Page As...**
    b. Click the Use layer names from the map document checkbox
    c. Click **Next**
    d. Click **Next** again
    e. Click **Finish** to start your service

i. Note: you will need to log in to your service and **Stop** your service in order to edit the capabilities document. Service title cannot be edited after publishing; you will need to create a new service

Proceed to Section 7, Testing Your Web Service.

B. 4 Troubleshooting ArcGIS Server

Q: How do I connect to the WFS in ArcCatalog?

A: Here is a brief tutorial on connecting to the WFS in ArcCatalog.

1. Open ArcCatalog
2. Enable the **Interoperability Extension** in ArcCatalog.
    a. For more information visit http://www.esri.com/software/arcgis/extensions/datainteroperability/common-questions.html
3. Click Add Interoperability Connection
4. A Dialog box appears
    a. Click the '…' button next to the **Format** pulldown menu
        i. Select the appropriate type of web service from the dialog
        ii. Click **OK**
    b. In the **Dataset** field, paste the URL of the capabilities document for the service you wish to add
        i. For more information, see Section 7, Testing Your Web Service
    c. Click the **Parameters** button.
        i. Click the '…' button next to **Table List**
        ii. Select the tables you wish to load
        iii. Confirm that the **Max Features** box is scaled appropriately
            1. If you are loading a dataset with 10 million features, the "Max Features" box must be scaled appropriately to ten million
            2. If you are loading a dataset with three (3) data points, the "Max Features" can be scaled down quite a bit
        iv. Click **OK**
    d. Once the layers have been added under the interoperability connections, you will see something like this in ArcCatalog
    e. Click **OK**
5. Depending on the size of the dataset, it may take some time to load. When it is finished, expand the Connection (you can also rename the connection, which can be helpful).
6. Drag and drop the layer into your ArcMap Project.

Q: All of my data are in Shapefiles. Can I deploy a shapefile as a GeoSciML-portrayal service?

A: The problem you will run into is the truncation of field names that occurs in shapefiles. Ideally you will have a full version of the data in an sde database (or file/personal geodatabase). As mentioned in the above document, to be compliant with GeoSciML-portrayal, you will need to

make sure there is no truncation in field names; they must be an exact match for the GeoSciML-portrayal schema.

Q: I would like to deploy a service using full GeoSciML v.3.0 schema; where can I go?

A: Visit the GeoSciML website

Q: Why would I want to create a WMS or WFS service at all?

A: To display data in a map format. Map format shows: data relative to its geographical surroundings, data relative to other data on the same map, data points relative to other data points in the same dataset (size, quantity, order, etc).

Appendix C: OneGeology Web Service Requirements

If you wish to submit your web service to the OneGeology portal, your web service must comply with OneGeology standards.

This appendix summarizes the service-level and layer-level requirements that must be respected for compliance with OneGeology standards.

C. 1 Service-Level Metadata Requirements

The following section describes OneGeology standards as they relate to service-level metadata.

When you submit your web service to OneGeology, service-level metadata is harvested into the portal in the GetCapabilities response document. The minimum requirements for inclusion with OneGeology (source: http://www.onegeology.org/wmscookbook/2_4_1.html)

Table 4: One-Geology Service-Level Metadata Requirements

Field	Description	Conditions
Version (1.3.0, 1.1.1)	This will be automatically populated by GeoServer	N/A
Title	Required for 2-star service (2-stars); see Section 7.2 for more information	Required for 2 Stars
Abstract	May include data owner organization, goals, scale of data and other information	Required for 2 Stars
Access Constraints	Definition of access constraints, otherwise "none" (2 to 3 Stars)	Required for 2-3 Stars
Keywords	"OneGeology", "MD_DATE@2012-02-02", "MD_LANG@ENG" (3 Stars)	Required for 3 Stars
Data (owner)	Full name of data owner organization (not service provider)	Required for 2

Field	Description	Conditions
provider	(2 Stars)	Stars
Image Format	This will be automatically populated by GeoServer	N/A
On-line Resource	Required by the WMS specification (2 Stars)	Required for 2 Stars
Contact Person	Named Individual or contact within data organization (2 to 3 Stars)	Required for 2-3 Stars
Country	Country of the data owner organization (2 to 3 Stars)	Required for 2-3 Stars
Email Address	Email address of the named contact (2 to 3 Stars)	Required for 2-3 Stars
Fees	Any fees required to use the WMS Service; otherwise "none" (3 Stars)	Required for 3 Stars

C. 2 Service Title and Service Name Requirements

This section describes the OneGeology standards that govern web service **Title** and **Name**.

Service Title: Conforming to the naming conventions for service title is a 2 star requirement with OneGeology. **Service Titles** should contain the following tokens (in order):

[Geographical extent] [Data owner organization] [Language Code] [Scale] [Theme], where:

- **Geographic extent:** a rough approximation of the extent of the portrayal provided by the web service
  - Generally an ISO three letter country code (such as "USA") and, if necessary, a province, such as "USA-KY" for the state of Kentucky
  - See http://en.wikipedia.org/wiki/ISO_3166-1_alpha-3 for ISO three-letter country codes
- **Data owner organization code**: an acronym or abbreviation for the entity that owns the data portrayed by the web service
  - The **data owner organization** *can be*, but is *not necessarily*, the name of the organization responsible for deploying the data as a web service
- **Language**: a two-letter ISO 639-1 code for the language in which the web service is portrayed
  - See http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes for ISO language codes
- **Scale:** expresses the scale of the portrayal as a ratio (X:Y); large-scale portrayals should be abbreviated using SI symbols for million "M" and thousand "k"

    - An example of a valid scale in the service title is "1:1M"
- **Theme**: a short or abbreviated description of the data contained in the layer

Service Name: For version 1.3.0 services, the service name should be 'WMS'; for version 1.1.1, the service name should be 'OGC:WMS'

OneGeology Service Title Examples:

USA USGIN 1:3M Geology
USA-AZ AZGS ES 1:2k Faults

***Note: Language is omitted from the examples above, as they are in the default language English.

C. 3 Layer Naming and Title Conventions

Each layer in a OneGeology-compliant web service should have a **Layer Title** and a **Layer Name**. The **Layer Name** should be identical to the **Layer Title**, using underscores instead of spaces and an abbreviated **Scale** token (see below)

Both **Layer Titles** and **Layer Names** should contain the following tokens (in order):

[Geographical Extent] [Data Owner Organization][Language Code][Scale][Theme], where

- **Geographic extent:** a rough approximation of the extent of the portrayal provided by the web service
    - Generally an ISO three letter country code (such as "USA") and, if necessary, a province, such as "USA-KY" for the state of Kentucky
    - See http://en.wikipedia.org/wiki/ISO_3166-1_alpha-3 for ISO three-letter country codes
- **Data owner organization code**: an acronym or abbreviation for the entity that owns the data portrayed by the web service
    - The **data owner organization** *can be*, but is *not necessarily*, the name of the organization responsible for deploying the data as a web service
- **Language**: a two-letter ISO 639-1 code for the language in which the web service is portrayed
    - See http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes for ISO language codes
- **Scale:** expresses the scale of the portrayal as a ratio (X:Y); large-scale portrayals should be abbreviated using SI symbols for million "M" and thousand "k"
    - An example of a valid scale in the service title is "1:1M"
    - **Layer Names** should abbreviate the **Scale** token by removing the **antecedent** and the **colon**. For example, a 1:23k scale would be abbreviated as 23k
- **Theme**: a short or abbreviated description of the data contained in the layer

Layer Title Examples

GBR BGS 1:625k Bedrock Age
USA USGIN 1:3M Geologic Age

Layer Name Examples

World_25M_Geol_Units
US-KY_KGS_24k_Faults
USA_USGIN_3M_Lithostratigraphy

A sample WMS GetCapabilities document demonstrating Layer Names and Layer Titles

C. 4 Additional Layer-level Metadata Requirements

Aside from **Layer Title** and **Layer Name** (discussed in Section 7.3), OneGeology standards require additional layer-level metadata, including the following:

Abstract: A description of your layer data; should be as comprehensive and interoperable as possible. You may wish to include other metadata in the abstract, such as information about your organization and other data you make available. You may also wish to include a statement on access constraints (source: http://www.onegeology.org/wmscookbook/2_6.html).

Code Example 3: OneGeology-compliant Layer Name and Layer Title in a WMS GetCapabilities document

```
<Layer queryable="1" >
<Name>USA_USGIN_3M-Geologic_Age</Name>
<Title><![CDATA[USA USGIN 1:3M Geologic Age]]></Title>
<Abstract><![CDATA[This layer provides a portrayal in which units are categorized according to the representative geologic age of the unit.]]></Abstract>
<KeywordList>
<Keyword>continent@North America</Keyword>
<Keyword>geographicarea@USA</Keyword>
<Keyword>dataprovider@USGIN</Keyword>
```

Keywords: Tags used to locate and categorize your layer. Required and conditional keywords are described in Table 5 (source: http://www.onegeology.org/wmscookbook/2_6.html):

Table 5: OneGeology Keywords

Keyword Description	Appearance in GetCapabilities Document	Example	Cardinality
Continent	continent@value	continent@North America	Required
Subcontinent	subcontinent@value	subcontinent@India	Conditional
Geographic area (country)	geographicarea@value	geographicarea@USA	Required

State (region or province)	subarea@value	subarea@Arizona	Conditional
Data Provider	dataprovider@value	dataprovider@USGIN	Required
Service Provider	serviceprovider@value	serviceprovider@AZGS	Required

Bounding Box: A rough indicator of the extent of your data; can be computed from data in the GeoServer 'Edit Layer' interface (Figure 17), or customized.

SRS: The spatial reference system used by your web service, indicated by an EPSG code.

Spatial reference systems specify a datum, a reference ellipsoid, and a coordinate system for cartographic representations. Most known spatial reference systems are associated with an EPSG code number; EPSG stands for European Petroleum Survey Group.

The default SRS is WGS 1984 (EPSG:4326). For the purposes of interoperability, your web service must be deployed using EPSG:4326.

MetadataURL: A link to an external ISO-valid metadata document. A **MetadataURL** is required for services deployed on OneGeology. To create an ISO Metadata record using ISO19115:2003, use the USGIN Metadata Wizard.

Readable Legend Graphic: URL(s) for the legend graphic. Readable Legend Graphic URLs should be present in the getCapabilities document by default when an SLD is assigned to a layer. For more information on SLDs, see Section 5, Styling.

8 Glossary

Attribute (GIS)
Within the context of geographic information systems, an attribute describes a feature. For example: attributes of a fault feature might include the latitude and longitude coordinates for each fault, as well as the fault's age, dip, and slip.

Database
A method of storing data. In a database, data is divided up into database records; in turn, database records are divided up into database fields. The advantage of a database is that it can be sorted and searched by field contents.

Though modern databases are usually digital, a physical example of a database is a card catalog in a public library. In a card catalog, data is divided up into individual cards, which are directly analogous to database records. Each card (record) in the catalog corresponds with and describes a book. The information about each book is divided up into fields: title; author; subject; publication

date; etc. Digital databases can be in tabular format (that is, a table) in which rows represent individual records and columns constitute fields; or they can be viewed record-by-record.

Database Field
A subdivision of a database record in which a specific type of data is entered. Using the analogy of a card catalog in a public library: if the card catalog is directly analogous to a database, and if cards in the catalog are directly analogous to database records, then the different subdivisions of information found in each card in the catalog (title, author, publishing date, etc.) are all database fields.

Database fields are functionally similar to markup language elements.

Database Record
A subdivision of a database. Using the analogy of a card catalog in a public library, each record in a database is analogous to an index card in the card catalog; each card (record) corresponds with and describes an individual book. Database records are further subdivided into fields, which contain specific kinds of data (title, author, etc).

Element
Elements are logical document components found in markup languages such as XML and HTML. Elements simultaneously define the structure and content of a document; they are constituted by two markup language tags and the content between the two. For example:
<tag>content</tag>

The two tags and their content form an element. In a more concrete example, HTML uses the tag to demarcate text that should be Italicized. So, an Italicized element of an HTML document would appear as follows:

Italicized content

In a web browser, this element would produce the following result:

Italicized content

For a more detailed overview of elements, see the USGIN XML Tutorial.

Markup language elements are functionally similar to database fields.

Feature
The word *feature* may indicate a *geologic* feature, such as a fault, formation, or dike. Alternatively, a feature can be a GIS representation of a real-world resource. GIS features do not always correspond with geologic features because GIS software can be used to represent anthropogenic objects such as buildings, roads, or canals. The definition of the term feature therefore depends on its context: it can be used to mean either a geologic feature or a GIS feature.

GIS features are often described by attributes.

Interchange Format
An interchange format is an application-neutral format used to exchange data between systems. For example, if one needed to transfer data between an Oracle database and an Access database, an interchange format could be used to transfer this data from one database to the other.

Interoperability
The capability to communicate, execute programs, or transfer data in a manner that requires the user to have little or no specialized knowledge. Interoperability is based on a shared data schema and the use of standard vocabulary terms.

Markup Language
Markup languages, such as HTML and XML, use elements to annotate text in such a way that the structure of the document is distinguishable from the contents of the document. Consequently, markup language documents are very good for storing data, since the distinction between structure and content is transparent and immediately discernible.

Metadata
Literally, data about data. Metadata is used to organize, categorize, browse, and discover data resources. For more information, see the USGIN Metadata Tutorial.

Schema
A schema is the structural framework for a data model or collection of attributes that describe something. For practical purposes, schemas *structure* documents by dictating where and how data from a dataset should be entered into a document.

Uniform Resource Identifier (URI)
Uniquely identifies a resource. See the USGIN URI Tutorial for more information about URIs.

Web Service
Under the client-server relationship, a web service is a protocol for requesting data from a server; data requests and responses made in accordance with this protocol are standardized. In other words: by using standardized data requests and responses, client software can make requests for data regardless of server configuration.

Matters are complicated slightly by the fact that "hosting data as a service" is considered synonymous with "hosting data in accordance with a web service protocol." Likewise, data hosted in accordance with a web service protocol is often referred to as a web service.

The Open Geospatial Consortium has produced several different flavors of web service that are relevant to geographic information systems, including:

Web Feature Services (WFS)
Web feature services provide information as collections of features, each having associated georeferenced geometry and attributes. These can be thought of as a collection of database records, each of which has a geometry field that locates the described feature, and a collection of properties

(the columns in the table) that describe the entity of interest at that location. For example a feature service for rivers might provide features that have linear geometry representing river segments, and properties like average flow rate, width, and depth for the segment. Feature services are most useful for client applications that need to process information about individual geographically located entities using properties of the entities. A client application that assists a user to navigate by showing them a road map would use a map service, but a client application that calculated the most efficient route between points in a city utilizing real-time traffic information would require a representation of the streets as features.

Web Map Services (WMS)

Web map services provide a method to acquire a georeferenced map image for some geographic bounding box. Clients send a 'getMap' request to a server specifying the geographic area of the bounding box, and the map that is desired, and the server responds by returning an image file (typical tif, jpg or png) for the requested area. Of course this assumes that the server actually offers that map for the requested bounding box extent. A variety of other parameters allow control over image size, map projection, and other details. The service may be based on a source shapefile or feature class, or a georeferenced raster image. The service offers another request to 'getFeatureInfo' at a point location, which returns information according to the service provider's choices;the response is not standardized. Map services are most useful for client applications designed for browsing maps and visually exploring geographic relationships between information displayed in superimposed map layers.

XML

XML stands for Extensible Markup Language. XML documents are often used as interchange formats for database records, owing to the ability for XML elements to emulate database fields. To use XML documents as an interchange format, USGIN defines XML schemas in which each XML document corresponds with a specific database record and each element corresponds with data entered in a database field. In these documents, tags define the database fields.

For example: well_name is a field within the WellHeader content model. Any known oil and gas well names go in the well_name field. An XML schema for expressing database records within the WellHeader content model might appear as follows:

```
<content_model_name>WellHeader</content_model_name>

<database_record>

 <well_name>Union Oil Company of California-South 7 Veysey</well_name>
```

In this example, the Union Oil Company of California-South 7 Veysey well name constitutes the content delineated by the <well_name> tag.

For a more detailed overview of XML, see the USGIN XML Tutorial.